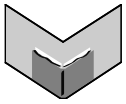
The Journal of Machine Learning Research
Volume 15
Print-Archive Edition

Pages 2773–4143



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) 2773-2832 Submitted 2/13; Revised 3/14; Published 8/14

Tensor Decompositions for Learning Latent Variable Models

Animashree Anandkumar A.ANANDKUMAR@UCI.EDU
Electrical Engineering and Computer Science
University of California, Irvine
2200 Engineering Hall
Irvine, CA 92697

Rong Ge RONGGE@MICROSOFT.COM
Microsoft Research
One Memorial Drive
Cambridge, MA 02142

Daniel Hsu DJHSU@CS.COLUMBIA.EDU
Department of Computer Science
Columbia University
1214 Amsterdam Avenue, #0401
New York, NY 10027

Sham M. Kakade SKAKADE@MICROSOFT.COM
Microsoft Research
One Memorial Drive
Cambridge, MA 02142

Matus Telgarsky MTELGARS@CS.UCSD.EDU
Department of Statistics
Rutgers University
110 Frelinghuysen Road
Piscataway, NJ 08854

Editor: Benjamin Recht

Abstract

This work considers a computationally and statistically efficient parameter estimation method for a wide class of latent variable models—including Gaussian mixture models, hidden Markov models, and latent Dirichlet allocation—which exploits a certain tensor structure in their low-order observable moments (typically, of second- and third-order). Specifically, parameter estimation is reduced to the problem of extracting a certain (orthogonal) decomposition of a symmetric tensor derived from the moments; this decomposition can be viewed as a natural generalization of the singular value decomposition for matrices. Although tensor decompositions are generally intractable to compute, the decomposition of these specially structured tensors can be efficiently obtained by a variety of approaches, including power iterations and maximization approaches (similar to the case of matrices). A detailed analysis of a robust tensor power method is provided, establishing an analogue of Wedin's perturbation theorem for the singular vectors of matrices. This implies a robust and computationally tractable estimation approach for several popular latent variable models.

Keywords: latent variable models, tensor decompositions, mixture models, topic models, method of moments, power method

1. Introduction

The method of moments is a classical parameter estimation technique (Pearson, 1894) from statistics which has proved invaluable in a number of application domains. The basic paradigm is simple and intuitive: (i) compute certain statistics of the data—often empirical moments such as means and correlations—and (ii) find model parameters that give rise to (nearly) the same corresponding population quantities. In a number of cases, the method of moments leads to consistent estimators which can be efficiently computed; this is especially relevant in the context of latent variable models, where standard maximum likelihood approaches are typically computationally prohibitive, and heuristic methods can be unreliable and difficult to validate with high-dimensional data. Furthermore, the method of moments can be viewed as complementary to the maximum likelihood approach; simply taking a single step of Newton-Raphson on the likelihood function starting from the moment based estimator (Le Cam, 1986) often leads to the best of both worlds: a computationally efficient estimator that is (asymptotically) statistically optimal.

The primary difficulty in learning latent variable models is that the latent (hidden) state of the data is not directly observed; rather only observed variables correlated with the hidden state are observed. As such, it is not evident the method of moments should fare any better than maximum likelihood in terms of computational performance: matching the model parameters to the observed moments may involve solving computationally intractable systems of multivariate polynomial equations. Fortunately, for many classes of latent variable models, there is rich structure in low-order moments (typically second- and third-order) which allow for this inverse moment problem to be solved efficiently (Cattell, 1944; Cardoso, 1991; Chang, 1996; Mossel and Roch, 2006; Hsu et al., 2012b; Anandkumar et al., 2012c,a; Hsu and Kakade, 2013). What is more is that these decomposition problems are often amenable to simple and efficient iterative methods, such as gradient descent and the power iteration method.

1.1 Contributions

In this work, we observe that a number of important and well-studied latent variable models—including Gaussian mixture models, hidden Markov models, and Latent Dirichlet allocation—share a certain structure in their low-order moments, and this permits certain tensor decomposition approaches to parameter estimation. In particular, this decomposition can be viewed as a natural generalization of the singular value decomposition for matrices.

While much of this (or similar) structure was implicit in several previous works (Chang, 1996; Mossel and Roch, 2006; Hsu et al., 2012b; Anandkumar et al., 2012c,a; Hsu and Kakade, 2013), here we make the decomposition explicit under a unified framework. Specifically, we express the observable moments as sums of rank-one terms, and reduce the parameter estimation task to the problem of extracting a symmetric orthogonal decomposition of a symmetric tensor derived from these observable moments. The problem can then be solved by a variety of approaches, including fixed-point and variational methods.

One approach for obtaining the orthogonal decomposition is the tensor power method of Lathauwer et al. (2000, Remark 3). We provide a convergence analysis of this method for orthogonally decomposable symmetric tensors, as well as a detailed perturbation analysis for a robust (and a computationally tractable) variant (Theorem 5.1). This perturbation analysis can be viewed as an analogue of Wedin's perturbation theorem for singular vectors of matrices (Wedin, 1972), providing a bound on the error of the recovered decomposition in terms of the operator norm of the tensor perturbation. This analysis is subtle in at least two ways. First, unlike for matrices (where every matrix has a singular value decomposition), an orthogonal decomposition need not exist for the perturbed tensor. Our robust variant uses random restarts and deflation to extract an approximate decomposition in a computationally tractable manner. Second, the analysis of the deflation steps is non-trivial; a naïve argument would entail error accumulation in each deflation step, which we show can in fact be avoided. When this method is applied for parameter estimation in latent variable models previously discussed, improved sample complexity bounds (over previous work) can be obtained using this perturbation analysis.

Finally, we also address computational issues that arise when applying the tensor decomposition approaches to estimating latent variable models. Specifically, we show that the basic operations of simple iterative approaches (such as the tensor power method) can be efficiently executed in time linear in the dimension of the observations and the size of the training data. For instance, in a topic modeling application, the proposed methods require time linear in the number of words in the vocabulary and in the number of non-zero entries of the term-document matrix. The combination of this computational efficiency and the robustness of the tensor decomposition techniques makes the overall framework a promising approach to parameter estimation for latent variable models.

1.2 Related Work

The connection between tensor decompositions and latent variable models has a long history across many scientific and mathematical disciplines. We review some of the key works that are most closely related to ours.

1.2.1 Tensor Decompositions

The role of tensor decompositions in the context of latent variable models dates back to early uses in psychometrics (Cattell, 1944). These ideas later gained popularity in chemometrics, and more recently in numerous science and engineering disciplines, including neuroscience, phylogenetics, signal processing, data mining, and computer vision. A thorough survey of these techniques and applications is given by Kolda and Bader (2009). Below, we discuss a few specific connections to two applications in machine learning and statistics, independent component analysis and latent variable models (between which there is also significant overlap).

Tensor decompositions have been used in signal processing and computational neuroscience for blind source separation and independent component analysis (ICA) (Comon and Jutten, 2010). Here, statistically independent non-Gaussian sources are linearly mixed in the observed signal, and the goal is to recover the mixing matrix (and ultimately, the original source signals). A typical solution is to locate projections of the observed signals that

correspond to local extrema of the so-called "contrast functions" which distinguish Gaussian variables from non-Gaussian variables. This method can be effectively implemented using fast descent algorithms (Hyvarinen, 1999). When using the excess kurtosis (*i.e.*, fourth-order cumulant) as the contrast function, this method reduces to a generalization of the power method for symmetric tensors (Lathauwer et al., 2000; Zhang and Golub, 2001; Kofidis and Regalia, 2002). This case is particularly important, since all local extrema of the kurtosis objective correspond to the true sources (under the assumed statistical model) (Delfosse and Loubaton, 1995); the descent methods can therefore be rigorously analyzed, and their computational and statistical complexity can be bounded (Frieze et al., 1996; Nguyen and Regev, 2009; Arora et al., 2012b).

Higher-order tensor decompositions have also been used to develop estimators for commonly used mixture models, hidden Markov models, and other related latent variable models, often using the the algebraic procedure of R. Jennrich (as reported in the article of Harshman, 1970), which is based on a simultaneous diagonalization of different ways of flattening a tensor to matrices. Jennrich's procedure was employed for parameter estimation of discrete Markov models by Chang (1996) via pair-wise and triple-wise probability tables; and it was later used for other latent variable models such as hidden Markov models (HMMs), latent trees, Gaussian mixture models, and topic models such as latent Dirichlet allocation (LDA) by many others (Mossel and Roch, 2006; Hsu et al., 2012b; Anandkumar et al., 2012c,a; Hsu and Kakade, 2013). In these contexts, it is often also possible to establish strong identifiability results, without giving an explicit estimators, by invoking the non-constructive identifiability argument of Kruskal (1977)—see the article by Allman et al. (2009) for several examples.

Related simultaneous diagonalization approaches have also been used for blind source separation and ICA (as discussed above), and a number of efficient algorithms have been developed for this problem (Bunse-Gerstner et al., 1993; Cardoso and Souloumiac, 1993; Cardoso, 1994; Cardoso and Comon, 1996; Corless et al., 1997; Ziehe et al., 2004). A rather different technique that uses tensor flattening and matrix eigenvalue decomposition has been developed by Cardoso (1991) and later by De Lathauwer et al. (2007). A significant advantage of this technique is that it can be used to estimate overcomplete mixtures, where the number of sources is larger than the observed dimension.

The relevance of tensor analysis to latent variable modeling has been long recognized in the field of algebraic statistics (Pachter and Sturmfels, 2005), and many works characterize the algebraic varieties corresponding to the moments of various classes of latent variable models (Drton et al., 2007; Sturmfels and Zwiernik, 2013). These works typically do not address computational or finite sample issues, but rather are concerned with basic questions of identifiability.

The specific tensor structure considered in the present work is the symmetric orthogonal decomposition. This decomposition expresses a tensor as a linear combination of simple tensor forms; each form is the tensor product of a vector (*i.e.*, a rank-1 tensor), and the collection of vectors form an orthonormal basis. An important property of tensors with such decompositions is that they have eigenvectors corresponding to these basis vectors. Although the concepts of eigenvalues and eigenvectors of tensors is generally significantly more complicated than their matrix counterpart—both algebraically (Qi, 2005; Cartwright and Sturmfels, 2013; Lim, 2005) and computationally (Hillar and Lim, 2013; Kofidis and

Regalia, 2002)—the special symmetric orthogonal structure we consider permits simple algorithms to efficiently and stably recover the desired decomposition. In particular, a generalization of the matrix power method to symmetric tensors, introduced by Lathauwer et al. (2000, Remark 3) and analyzed by Kofidis and Regalia (2002), provides such a decomposition. This is in fact implied by the characterization of Zhang and Golub (2001), which shows that iteratively obtaining the best rank-1 approximation of such orthogonally decomposable tensors also yields the exact decomposition. We note that in general, obtaining such approximations for general (symmetric) tensors is NP-hard (Hillar and Lim, 2013).

1.2.2 Latent Variable Models

This work focuses on the particular application of tensor decomposition methods to estimating latent variable models, a significant departure from many previous approaches in the machine learning and statistics literature. By far the most popular heuristic for parameter estimation for such models is the Expectation-Maximization (EM) algorithm (Dempster et al., 1977; Redner and Walker, 1984). Although EM has a number of merits, it may suffer from slow convergence and poor quality local optima (Redner and Walker, 1984), requiring practitioners to employ many additional heuristics to obtain good solutions. For some models such as latent trees (Roch, 2006) and topic models (Arora et al., 2012a), maximum likelihood estimation is NP-hard, which suggests that other estimation approaches may be more attractive. More recently, algorithms from theoretical computer science and machine learning have addressed computational and sample complexity issues related to estimating certain latent variable models such as Gaussian mixture models and HMMs (Dasgupta, 1999; Arora and Kannan, 2005; Dasgupta and Schulman, 2007; Vempala and Wang, 2004; Kannan et al., 2008; Achlioptas and McSherry, 2005; Chaudhuri and Rao, 2008; Brubaker and Vempala, 2008; Kalai et al., 2010; Belkin and Sinha, 2010; Moitra and Valiant, 2010; Hsu and Kakade, 2013; Chang, 1996; Mossel and Roch, 2006; Hsu et al., 2012b; Anandkumar et al., 2012c; Arora et al., 2012a; Anandkumar et al., 2012a). See the works by Anandkumar et al. (2012c) and Hsu and Kakade (2013) for a discussion of these methods, together with the computational and statistical hardness barriers that they face. The present work reviews a broad range of latent variables where a mild non-degeneracy condition implies the symmetric orthogonal decomposition structure in the tensors of low-order observable moments.

Notably, another class of methods, based on subspace identification (Overschee and Moor, 1996) and observable operator models/multiplicity automata (Schützenberger, 1961; Jaeger, 2000; Littman et al., 2001), have been proposed for a number of latent variable models. These methods were successfully developed for HMMs by Hsu et al. (2012b), and subsequently generalized and extended for a number of related sequential and tree Markov models models (Siddiqi et al., 2010; Bailly, 2011; Boots et al., 2010; Parikh et al., 2011; Rodu et al., 2013; Balle et al., 2012; Balle and Mohri, 2012), as well as certain classes of parse tree models (Luque et al., 2012; Cohen et al., 2012; Dhillon et al., 2012). These methods use low-order moments to learn an "operator" representation of the distribution, which can be used for density estimation and belief state updates. While finite sample bounds can be given to establish the learnability of these models (Hsu et al., 2012b), the algorithms do not

actually give parameter estimates (*e.g.*, of the emission or transition matrices in the case of HMMs).

1.3 Organization

The rest of the paper is organized as follows. Section 2 reviews some basic definitions of tensors. Section 3 provides examples of a number of latent variable models which, after appropriate manipulations of their low order moments, share a certain natural tensor structure. Section 4 reduces the problem of parameter estimation to that of extracting a certain (symmetric orthogonal) decomposition of a tensor. We then provide a detailed analysis of a robust tensor power method and establish an analogue of Wedin's perturbation theorem for the singular vectors of matrices. The discussion in Section 6 addresses a number of practical concerns that arise when dealing with moment matrices and tensors.

2. Preliminaries

We introduce some tensor notations borrowed from Lim (2005). A real *p-th order tensor* $A \in \bigotimes_{i=1}^{p} \mathbb{R}^{n_i}$ is a member of the tensor product of Euclidean spaces $\mathbb{R}^{n_i}$, $i \in [p]$. We generally restrict to the case where $n_1 = n_2 = \cdots = n_p = n$, and simply write $A \in \bigotimes^p \mathbb{R}^n$. For a vector $v \in \mathbb{R}^n$, we use $v^{\otimes p} := v \otimes v \otimes \cdots \otimes v \in \bigotimes^p \mathbb{R}^n$ to denote its p-th tensor power. As is the case for vectors (where $p = 1$) and matrices (where $p = 2$), we may identify a p-th order tensor with the p-way array of real numbers $[A_{i_1,i_2,\ldots,i_p} : i_1, i_2, \ldots, i_p \in [n]]$, where $A_{i_1,i_2,\ldots,i_p}$ is the $(i_1, i_2, \ldots, i_p)$-th coordinate of A (with respect to a canonical basis).

We can consider A to be a multilinear map in the following sense: for a set of matrices $\{V_i \in \mathbb{R}^{n \times m_i} : i \in [p]\}$, the $(i_1, i_2, \ldots, i_p)$-th entry in the p-way array representation of $A(V_1, V_2, \ldots, V_p) \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_p}$ is

$$[A(V_1, V_2, \ldots, V_p)]_{i_1,i_2,\ldots,i_p} := \sum_{j_1,j_2,\ldots,j_p \in [n]} A_{j_1,j_2,\ldots,j_p} \, [V_1]_{j_1,i_1} \, [V_2]_{j_2,i_2} \, \cdots \, [V_p]_{j_p,i_p}.$$

Note that if A is a matrix ($p = 2$), then

$$A(V_1, V_2) = V_1^\top A V_2.$$

Similarly, for a matrix A and vector $v \in \mathbb{R}^n$, we can express Av as

$$A(I, v) = Av \in \mathbb{R}^n,$$

where I is the $n \times n$ identity matrix. As a final example of this notation, observe

$$A(e_{i_1}, e_{i_2}, \ldots, e_{i_p}) = A_{i_1,i_2,\ldots,i_p},$$

where $\{e_1, e_2, \ldots, e_n\}$ is the canonical basis for $\mathbb{R}^n$.

Most tensors $A \in \bigotimes^p \mathbb{R}^n$ considered in this work will be *symmetric* (sometimes called *supersymmetric*), which means that their p-way array representations are invariant to permutations of the array indices: *i.e.*, for all indices $i_1, i_2, \ldots, i_p \in [n]$, $A_{i_1,i_2,\ldots,i_p} = A_{i_{\pi(1)},i_{\pi(2)},\ldots,i_{\pi(p)}}$ for any permutation π on $[p]$. It can be checked that this reduces to the usual definition of a symmetric matrix for $p = 2$.

The *rank* of a p-th order tensor $A \in \bigotimes^p \mathbb{R}^n$ is the smallest non-negative integer k such that $A = \sum_{j=1}^k u_{1,j} \otimes u_{2,j} \otimes \cdots \otimes u_{p,j}$ for some $u_{i,j} \in \mathbb{R}^n, i \in [p], j \in [k]$, and the *symmetric rank* of a symmetric p-th order tensor A is the smallest non-negative integer k such that $A = \sum_{j=1}^k u_j^{\otimes p}$ for some $u_j \in \mathbb{R}^n, j \in [k]$.[1] The notion of rank readily reduces to the usual definition of matrix rank when $p = 2$, as revealed by the singular value decomposition. Similarly, for symmetric matrices, the symmetric rank is equivalent to the matrix rank as given by the spectral theorem. A decomposition into such rank-one terms is known as a *canonical polyadic decomposition* (Hitchcock, 1927a,b).

The notion of tensor (symmetric) rank is considerably more delicate than matrix (symmetric) rank. For instance, it is not clear *a priori* that the symmetric rank of a tensor should even be finite (Comon et al., 2008). In addition, removal of the best rank-1 approximation of a (general) tensor may increase the tensor rank of the residual (Stegeman and Comon, 2010).

Throughout, we use $\|v\| = (\sum_i v_i^2)^{1/2}$ to denote the Euclidean norm of a vector v, and $\|M\|$ to denote the spectral (operator) norm of a matrix. We also use $\|T\|$ to denote the operator norm of a tensor, which we define later.

3. Tensor Structure in Latent Variable Models

In this section, we give several examples of latent variable models whose low-order moments can be written as symmetric tensors of low symmetric rank; some of these examples can be deduced using the techniques developed in the text by McCullagh (1987). The basic form is demonstrated in Theorem 3.1 for the first example, and the general pattern will emerge from subsequent examples.

3.1 Exchangeable Single Topic Models

We first consider a simple bag-of-words model for documents in which the words in the document are assumed to be *exchangeable*. Recall that a collection of random variables $x_1, x_2, \ldots, x_\ell$ are exchangeable if their joint probability distribution is invariant to permutation of the indices. The well-known De Finetti's theorem (Austin, 2008) implies that such exchangeable models can be viewed as mixture models in which there is a latent variable h such that $x_1, x_2, \ldots, x_\ell$ are conditionally i.i.d. given h (see Figure 1(a) for the corresponding graphical model) and the conditional distributions are identical at all the nodes.

In our simplified topic model for documents, the latent variable h is interpreted as the (sole) topic of a given document, and it is assumed to take only a finite number of distinct values. Let k be the number of distinct topics in the corpus, d be the number of distinct words in the vocabulary, and $\ell \geq 3$ be the number of words in each document. The generative process for a document is as follows: the document's topic is drawn according to the discrete distribution specified by the probability vector $w := (w_1, w_2, \ldots, w_k) \in \Delta^{k-1}$. This is modeled as a discrete random variable h such that

$$\Pr[h = j] = w_j, \quad j \in [k].$$

1. For even p, the definition is slightly different (Comon et al., 2008).

Given the topic h, the document's ℓ words are drawn independently according to the discrete distribution specified by the probability vector $\mu_h \in \Delta^{d-1}$. It will be convenient to represent the ℓ words in the document by d-dimensional random *vectors* $x_1, x_2, \ldots, x_\ell \in \mathbb{R}^d$. Specifically, we set

$$x_t = e_i \quad \text{if and only if} \quad \text{the } t\text{-th word in the document is } i, \quad t \in [\ell],$$

where $e_1, e_2, \ldots e_d$ is the standard coordinate basis for $\mathbb{R}^d$.

One advantage of this encoding of words is that the (cross) moments of these random vectors correspond to joint probabilities over words. For instance, observe that

$$\begin{aligned}\mathbb{E}[x_1 \otimes x_2] &= \sum_{1 \le i,j \le d} \Pr[x_1 = e_i, x_2 = e_j]\ e_i \otimes e_j \\ &= \sum_{1 \le i,j \le d} \Pr[\text{1st word} = i, \text{2nd word} = j]\ e_i \otimes e_j,\end{aligned}$$

so the (i,j)-the entry of the matrix $\mathbb{E}[x_1 \otimes x_2]$ is $\Pr[\text{1st word} = i, \text{2nd word} = j]$. More generally, the $(i_1, i_2, \ldots, i_\ell)$-th entry in the tensor $\mathbb{E}[x_1 \otimes x_2 \otimes \cdots \otimes x_\ell]$ is $\Pr[\text{1st word} = i_1, \text{2nd word} = i_2, \ldots, \ell\text{-th word} = i_\ell]$. This means that estimating cross moments, say, of $x_1 \otimes x_2 \otimes x_3$, is the same as estimating joint probabilities of the first three words over all documents. (Recall that we assume that each document has at least three words.)

The second advantage of the vector encoding of words is that the conditional expectation of x_t given $h = j$ is simply μ_j, the vector of word probabilities for topic j:

$$\mathbb{E}[x_t | h = j] \ = \ \sum_{i=1}^{d} \Pr[t\text{-th word} = i | h = j]\ e_i \ = \ \sum_{i=1}^{d} [\mu_j]_i\ e_i \ = \ \mu_j, \quad j \in [k]$$

(where $[\mu_j]_i$ is the i-th entry in the vector μ_j). Because the words are conditionally independent given the topic, we can use this same property with conditional cross moments, say, of x_1 and x_2:

$$\mathbb{E}[x_1 \otimes x_2 | h = j] \ = \ \mathbb{E}[x_1 | h = j] \otimes \mathbb{E}[x_2 | h = j] \ = \ \mu_j \otimes \mu_j, \quad j \in [k].$$

This and similar calculations lead one to the following theorem.

Theorem 3.1 (Anandkumar et al., 2012c) *If*

$$\begin{aligned} M_2 &:= \mathbb{E}[x_1 \otimes x_2] \\ M_3 &:= \mathbb{E}[x_1 \otimes x_2 \otimes x_3], \end{aligned}$$

then

$$\begin{aligned} M_2 &= \sum_{i=1}^{k} w_i\ \mu_i \otimes \mu_i \\ M_3 &= \sum_{i=1}^{k} w_i\ \mu_i \otimes \mu_i \otimes \mu_i. \end{aligned}$$

As we will see in Section 4.3, the structure of M_2 and M_3 revealed in Theorem 3.1 implies that the topic vectors $\mu_1, \mu_2, \ldots, \mu_k$ can be estimated by computing a certain symmetric tensor decomposition. Moreover, due to exchangeability, all triples (resp., pairs) of words in a document—and not just the first three (resp., two) words—can be used in forming M_3 (resp., M_2); see Section 6.1.

3.2 Beyond Raw Moments

In the single topic model above, the raw (cross) moments of the observed words directly yield the desired symmetric tensor structure. In some other models, the raw moments do not explicitly have this form. Here, we show that the desired tensor structure can be found through various manipulations of different moments.

3.2.1 Spherical Gaussian Mixtures: Common Covariance

We now consider a mixture of k Gaussian distributions with spherical covariances. We start with the simpler case where all of the covariances are identical; this probabilistic model is closely related to the (non-probabilistic) k-means clustering problem (MacQueen, 1967).

Let $w_i \in (0,1)$ be the probability of choosing component $i \in [k]$, $\{\mu_1, \mu_2, \ldots, \mu_k\} \subset \mathbb{R}^d$ be the component mean vectors, and $\sigma^2 I$ be the common covariance matrix. An observation in this model is given by

$$x := \mu_h + z,$$

where h is the discrete random variable with $\Pr[h = i] = w_i$ for $i \in [k]$ (similar to the exchangeable single topic model), and $z \sim \mathcal{N}(0, \sigma^2 I)$ is an independent multivariate Gaussian random vector in $\mathbb{R}^d$ with zero mean and spherical covariance $\sigma^2 I$.

The Gaussian mixture model differs from the exchangeable single topic model in the way observations are generated. In the single topic model, we observe multiple draws (words in a particular document) $x_1, x_2, \ldots, x_\ell$ given the same fixed h (the topic of the document). In contrast, for the Gaussian mixture model, every realization of x corresponds to a different realization of h.

Theorem 3.2 (Hsu and Kakade, 2013) *Assume $d \geq k$. The variance σ^2 is the smallest eigenvalue of the covariance matrix $\mathbb{E}[x \otimes x] - \mathbb{E}[x] \otimes \mathbb{E}[x]$. Furthermore, if*

$$\begin{aligned} M_2 &:= \mathbb{E}[x \otimes x] - \sigma^2 I \\ M_3 &:= \mathbb{E}[x \otimes x \otimes x] - \sigma^2 \sum_{i=1}^{d} \big(\mathbb{E}[x] \otimes e_i \otimes e_i + e_i \otimes \mathbb{E}[x] \otimes e_i + e_i \otimes e_i \otimes \mathbb{E}[x]\big), \end{aligned}$$

then

$$\begin{aligned} M_2 &= \sum_{i=1}^{k} w_i \ \mu_i \otimes \mu_i \\ M_3 &= \sum_{i=1}^{k} w_i \ \mu_i \otimes \mu_i \otimes \mu_i. \end{aligned}$$

3.2.2 Spherical Gaussian Mixtures: Differing Covariances

The general case is where each component may have a *different* spherical covariance. An observation in this model is again $x = \mu_h + z$, but now $z \in \mathbb{R}^d$ is a random vector whose conditional distribution given $h = i$ (for some $i \in [k]$) is a multivariate Gaussian $\mathcal{N}(0, \sigma_i^2 I)$ with zero mean and spherical covariance $\sigma_i^2 I$.

Theorem 3.3 (Hsu and Kakade, 2013) *Assume $d \geq k$. The average variance $\bar{\sigma}^2 := \sum_{i=1}^k w_i \sigma_i^2$ is the smallest eigenvalue of the covariance matrix $\mathbb{E}[x \otimes x] - \mathbb{E}[x] \otimes \mathbb{E}[x]$. Let v be any unit norm eigenvector corresponding to the eigenvalue $\bar{\sigma}^2$. If*

$$\begin{aligned} M_1 &:= \mathbb{E}[x(v^\top (x - \mathbb{E}[x]))^2] \\ M_2 &:= \mathbb{E}[x \otimes x] - \bar{\sigma}^2 I \\ M_3 &:= \mathbb{E}[x \otimes x \otimes x] - \sum_{i=1}^d \big(M_1 \otimes e_i \otimes e_i + e_i \otimes M_1 \otimes e_i + e_i \otimes e_i \otimes M_1\big), \end{aligned}$$

then

$$\begin{aligned} M_2 &= \sum_{i=1}^k w_i \ \mu_i \otimes \mu_i \\ M_3 &= \sum_{i=1}^k w_i \ \mu_i \otimes \mu_i \otimes \mu_i. \end{aligned}$$

As shown by Hsu and Kakade (2013), $M_1 = \sum_{i=1}^k w_i \sigma_i^2 \mu_i$. Note that for the common covariance case, where $\sigma_i^2 = \sigma^2$, we have that $M_1 = \sigma^2 \mathbb{E}[x]$ (*cf.* Theorem 3.2).

3.2.3 Independent Component Analysis (ICA)

The standard model for ICA (Comon, 1994; Cardoso and Comon, 1996; Hyvärinen and Oja, 2000; Comon and Jutten, 2010), in which independent signals are linearly mixed and corrupted with Gaussian noise before being observed, is specified as follows. Let $h \in \mathbb{R}^k$ be a latent random *vector* with independent coordinates, $A \in \mathbb{R}^{d \times k}$ the mixing matrix, and z be a multivariate Gaussian random vector. The random vectors h and z are assumed to be independent. The observed random vector is

$$x := Ah + z.$$

Let μ_i denote the i-th column of the mixing matrix A.

Theorem 3.4 (Comon and Jutten, 2010) *Define*

$$M_4 := \mathbb{E}[x \otimes x \otimes x \otimes x] - T$$

where T is the fourth-order tensor with

$$\begin{aligned} [T]_{i_1,i_2,i_3,i_4} := \mathbb{E}[x_{i_1} x_{i_2}]\mathbb{E}[x_{i_3} x_{i_4}] + \mathbb{E}[x_{i_1} x_{i_3}]\mathbb{E}[x_{i_2} x_{i_4}] \\ + \mathbb{E}[x_{i_1} x_{i_4}]\mathbb{E}[x_{i_2} x_{i_3}], \quad 1 \leq i_1, i_2, i_3, i_4 \leq k \end{aligned}$$

*(*i.e., *T is the fourth derivative tensor of the function $v \mapsto 8^{-1}\mathbb{E}[(v^\top x)^2]^2$, so M_4 is the fourth cumulant tensor). Let $\kappa_i := \mathbb{E}[h_i^4] - 3$ for each $i \in [k]$. Then*

$$M_4 \quad = \quad \sum_{i=1}^{k} \kappa_i \ \mu_i \otimes \mu_i \otimes \mu_i \otimes \mu_i.$$

Note that κ_i corresponds to the excess kurtosis, a measure of non-Gaussianity as $\kappa_i = 0$ if h_i is a standard normal random variable. Furthermore, note that A is not identifiable if h is a multivariate Gaussian.

We may derive forms similar to that of M_2 and M_3 from Theorem 3.1 using M_4 by observing that

$$M_4(I, I, u, v) = \sum_{i=1}^{k} \kappa_i (\mu_i^\top u)(\mu_i^\top v) \ \mu_i \otimes \mu_i,$$

$$M_4(I, I, I, v) = \sum_{i=1}^{k} \kappa_i (\mu_i^\top v) \ \mu_i \otimes \mu_i \otimes \mu_i$$

for any vectors $u, v \in \mathbb{R}^d$.

3.2.4 Latent Dirichlet Allocation (LDA)

An increasingly popular class of latent variable models are *mixed membership models*, where each datum may belong to several different latent classes simultaneously. LDA is one such model for the case of document modeling; here, each document corresponds to a mixture over topics (as opposed to just a single topic). The distribution over such topic mixtures is a Dirichlet distribution $\mathrm{Dir}(\alpha)$ with parameter vector $\alpha \in \mathbb{R}^k_{++}$ with strictly positive entries; its density over the probability simplex $\Delta^{k-1} := \{v \in \mathbb{R}^k : v_i \in [0, 1] \forall i \in [k], \ \sum_{i=1}^k v_i = 1\}$ is given by

$$p_\alpha(h) = \frac{\Gamma(\alpha_0)}{\prod_{i=1}^k \Gamma(\alpha_i)} \prod_{i=1}^{k} h_i^{\alpha_i - 1}, \quad h \in \Delta^{k-1}$$

where

$$\alpha_0 := \alpha_1 + \alpha_2 + \cdots + \alpha_k.$$

As before, the k topics are specified by probability vectors $\mu_1, \mu_2, \ldots, \mu_k \in \Delta^{d-1}$. To generate a document, we first draw the topic mixture $h = (h_1, h_2, \ldots, h_k) \sim \mathrm{Dir}(\alpha)$, and then conditioned on h, we draw ℓ words $x_1, x_2, \ldots, x_\ell$ independently from the discrete distribution specified by the probability vector $\sum_{i=1}^k h_i \mu_i$ (*i.e.*, for each x_t, we independently sample a topic j according to h and then sample x_t according to μ_j). Again, we encode a word x_t by setting $x_t = e_i$ iff the t-th word in the document is i.

The parameter α_0 (the sum of the "pseudo-counts") characterizes the concentration of the distribution. As $\alpha_0 \to 0$, the distribution degenerates to a single topic model (*i.e.*, the limiting density has, with probability 1, exactly one entry of h being 1 and the rest are 0). At the other extreme, if $\alpha = (c, c, \ldots, c)$ for some scalar $c > 0$, then as $\alpha_0 = ck \to \infty$, the distribution of h becomes peaked around the uniform vector $(1/k, 1/k, \ldots, 1/k)$ (furthermore, the distribution behaves like a product distribution). We are typically interested in

(a) Multi-view models (b) Hidden Markov model

Figure 1: Examples of latent variable models.

the case where α_0 is small (*e.g.*, a constant independent of k), whereupon h typically has only a few large entries. This corresponds to the setting where the documents are mainly comprised of just a few topics.

Theorem 3.5 (Anandkumar et al., 2012a) *Define*

$$\begin{aligned}
M_1 &:= \mathbb{E}[x_1] \\
M_2 &:= \mathbb{E}[x_1 \otimes x_2] - \frac{\alpha_0}{\alpha_0 + 1} M_1 \otimes M_1 \\
M_3 &:= \mathbb{E}[x_1 \otimes x_2 \otimes x_3] \\
&\quad - \frac{\alpha_0}{\alpha_0 + 2} \Big(\mathbb{E}[x_1 \otimes x_2 \otimes M_1] + \mathbb{E}[x_1 \otimes M_1 \otimes x_2] + \mathbb{E}[M_1 \otimes x_1 \otimes x_2] \Big) \\
&\quad + \frac{2\alpha_0^2}{(\alpha_0 + 2)(\alpha_0 + 1)} M_1 \otimes M_1 \otimes M_1.
\end{aligned}$$

Then

$$\begin{aligned}
M_2 &= \sum_{i=1}^{k} \frac{\alpha_i}{(\alpha_0 + 1)\alpha_0} \, \mu_i \otimes \mu_i \\
M_3 &= \sum_{i=1}^{k} \frac{2\alpha_i}{(\alpha_0 + 2)(\alpha_0 + 1)\alpha_0} \, \mu_i \otimes \mu_i \otimes \mu_i.
\end{aligned}$$

Note that α_0 needs to be known to form M_2 and M_3 from the raw moments. This, however, is a much weaker than assuming that the entire distribution of h is known (*i.e.*, knowledge of the whole parameter vector α).

3.3 Multi-View Models

Multi-view models (also sometimes called naïve Bayes models) are a special class of Bayesian networks in which observed variables $x_1, x_2, \ldots, x_\ell$ are conditionally independent given a latent variable h. This is similar to the exchangeable single topic model, but here we do not require the conditional distributions of the $x_t, t \in [\ell]$ to be identical. Techniques developed for this class can be used to handle a number of widely used models including hidden Markov models (Mossel and Roch, 2006; Anandkumar et al., 2012c), phylogenetic tree models (Chang, 1996; Mossel and Roch, 2006), certain tree mixtures (Anandkumar et al., 2012b), and certain probabilistic grammar models (Hsu et al., 2012a).

As before, we let $h \in [k]$ be a discrete random variable with $\Pr[h = j] = w_j$ for all $j \in [k]$. Now consider random vectors $x_1 \in \mathbb{R}^{d_1}$, $x_2 \in \mathbb{R}^{d_2}$, and $x_3 \in \mathbb{R}^{d_3}$ which are conditionally independent given h, and

$$\mathbb{E}[x_t|h = j] = \mu_{t,j}, \quad j \in [k],\ t \in \{1, 2, 3\}$$

where the $\mu_{t,j} \in \mathbb{R}^{d_t}$ are the conditional means of the x_t given $h = j$. Thus, we allow the observations $x_1, x_2, \ldots, x_\ell$ to be random vectors, parameterized only by their conditional means. Importantly, these conditional distributions may be discrete, continuous, or even a mix of both.

We first note the form for the raw (cross) moments.

Proposition 3.1 *We have that:*

$$\begin{aligned}
\mathbb{E}[x_t \otimes x_{t'}] &= \sum_{i=1}^{k} w_i\ \mu_{t,i} \otimes \mu_{t',i}, \quad \{t, t'\} \subset \{1, 2, 3\}, t \neq t' \\
\mathbb{E}[x_1 \otimes x_2 \otimes x_3] &= \sum_{i=1}^{k} w_i\ \mu_{1,i} \otimes \mu_{2,i} \otimes \mu_{3,i}.
\end{aligned}$$

The cross moments do not possess a symmetric tensor form when the conditional distributions are different. Nevertheless, the moments can be "symmetrized" via a simple linear transformation of x_1 and x_2 (roughly speaking, this relates x_1 and x_2 to x_3); this leads to an expression from which the conditional means of x_3 (*i.e.*, $\mu_{3,1}, \mu_{3,2}, \ldots, \mu_{3,k}$) can be recovered. For simplicity, we assume $d_1 = d_2 = d_3 = k$; the general case (with $d_t \geq k$) is easily handled using low-rank singular value decompositions.

Theorem 3.6 (Anandkumar et al., 2012a) *Assume that $\{\mu_{v,1}, \mu_{v,2}, \ldots, \mu_{v,k}\}$ are linearly independent for each $v \in \{1, 2, 3\}$. Define*

$$\begin{aligned}
\tilde{x}_1 &:= \mathbb{E}[x_3 \otimes x_2]\mathbb{E}[x_1 \otimes x_2]^{-1} x_1 \\
\tilde{x}_2 &:= \mathbb{E}[x_3 \otimes x_1]\mathbb{E}[x_2 \otimes x_1]^{-1} x_2 \\
M_2 &:= \mathbb{E}[\tilde{x}_1 \otimes \tilde{x}_2] \\
M_3 &:= \mathbb{E}[\tilde{x}_1 \otimes \tilde{x}_2 \otimes x_3].
\end{aligned}$$

Then

$$\begin{aligned}
M_2 &= \sum_{i=1}^{k} w_i\ \mu_{3,i} \otimes \mu_{3,i} \\
M_3 &= \sum_{i=1}^{k} w_i\ \mu_{3,i} \otimes \mu_{3,i} \otimes \mu_{3,i}.
\end{aligned}$$

We now discuss three examples (taken mostly from Anandkumar et al., 2012c) where the above observations can be applied. The first two concern mixtures of product distributions, and the last one is the time-homogeneous hidden Markov model.

3.3.1 Mixtures of Axis-Aligned Gaussians and Other Product Distributions

The first example is a mixture of k product distributions in $\mathbb{R}^n$ under a mild incoherence assumption (Anandkumar et al., 2012c). Here, we allow each of the k component distributions to have a different product distribution (*e.g.*, Gaussian distribution with an axis-aligned covariance matrix), but require the matrix of component means $A := [\mu_1|\mu_2|\cdots|\mu_k] \in \mathbb{R}^{n\times k}$ to satisfy a certain (very mild) incoherence condition. The role of the incoherence condition is explained below.

For a mixture of product distributions, any partitioning of the dimensions $[n]$ into three groups creates three (possibly asymmetric) "views" which are conditionally independent once the mixture component is selected. However, recall that Theorem 3.6 requires that for each view, the k conditional means be linearly independent. In general, this may not be achievable; consider, for instance, the case $\mu_i = e_i$ for each $i \in [k]$. Such cases, where the component means are very aligned with the coordinate basis, are precluded by the incoherence condition.

Define $\text{coherence}(A) := \max_{i\in[n]}\{e_i^\top \Pi_A e_i\}$ to be the largest diagonal entry of the orthogonal projector to the range of A, and assume A has rank k. The coherence lies between k/n and 1; it is largest when the range of A is spanned by the coordinate axes, and it is k/n when the range is spanned by a subset of the Hadamard basis of cardinality k. The incoherence condition requires, for some $\varepsilon, \delta \in (0,1)$, $\text{coherence}(A) \le (\varepsilon^2/6)/\ln(3k/\delta)$. Essentially, this condition ensures that the non-degeneracy of the component means is not isolated in just a few of the n dimensions. Operationally, it implies the following.

Proposition 3.2 (Anandkumar et al., 2012c) *Assume A has rank k, and*

$$\text{coherence}(A) \le \frac{\varepsilon^2/6}{\ln(3k/\delta)}$$

for some $\varepsilon, \delta \in (0,1)$. With probability at least $1-\delta$, a random partitioning of the dimensions $[n]$ into three groups (for each $i \in [n]$, independently pick $t \in \{1,2,3\}$ uniformly at random and put i in group t) has the following property. For each $t \in \{1,2,3\}$ and $j \in [k]$, let $\mu_{t,j}$ be the entries of μ_j put into group t, and let $A_t := [\mu_{t,1}|\mu_{t,2}|\cdots|\mu_{t,k}]$. Then for each $t \in \{1,2,3\}$, A_t has full column rank, and the k-th largest singular value of A_t is at least $\sqrt{(1-\varepsilon)/3}$ times that of A.

Therefore, three asymmetric views can be created by randomly partitioning the observed random vector x into x_1, x_2, and x_3, such that the resulting component means for each view satisfy the conditions of Theorem 3.6.

3.3.2 Spherical Gaussian Mixtures, Revisited

Consider again the case of spherical Gaussian mixtures (*cf.* Section 3.2). As we shall see in Section 4.3, the previous techniques (based on Theorem 3.2 and Theorem 3.3) lead to estimation procedures when the dimension of x is k or greater (and when the k component means are linearly independent). We now show that when the dimension is slightly larger, say greater than $3k$, a different (and simpler) technique based on the multi-view structure can be used to extract the relevant structure.

We again use a randomized reduction. Specifically, we create three views by (i) applying a random rotation to x, and then (ii) partitioning $x \in \mathbb{R}^n$ into three views $\tilde{x}_1, \tilde{x}_2, \tilde{x}_3 \in \mathbb{R}^d$ for $d := n/3$. By the rotational invariance of the multivariate Gaussian distribution, the distribution of x after random rotation is still a mixture of spherical Gaussians (*i.e.*, a mixture of product distributions), and thus $\tilde{x}_1, \tilde{x}_2, \tilde{x}_3$ are conditionally independent given h. What remains to be checked is that, for each view $t \in \{1, 2, 3\}$, the matrix of conditional means of $\tilde{x}_t$ for each view has full column rank. This is true with probability 1 as long as the matrix of conditional means $A := [\mu_1 | \mu_2 | \cdots | \mu_k] \in \mathbb{R}^{n \times k}$ has rank k and $n \geq 3k$. To see this, observe that a random rotation in $\mathbb{R}^n$ followed by a restriction to d coordinates is simply a random projection from $\mathbb{R}^n$ to $\mathbb{R}^d$, and that a random projection of a linear subspace of dimension k to $\mathbb{R}^d$ is almost surely injective as long as $d \geq k$. Applying this observation to the range of A implies the following.

Proposition 3.3 (Hsu and Kakade, 2013) *Assume A has rank k and that $n \geq 3k$. Let $R \in \mathbb{R}^{n \times n}$ be chosen uniformly at random among all orthogonal $n \times n$ matrices, and set $\tilde{x} := Rx \in \mathbb{R}^n$ and $\tilde{A} := RA = [R\mu_1 | R\mu_2 | \cdots | R\mu_k] \in \mathbb{R}^{n \times k}$. Partition $[n]$ into three groups of sizes d_1, d_2, d_3 with $d_t \geq k$ for each $t \in \{1, 2, 3\}$. Furthermore, for each t, define $\tilde{x}_t \in \mathbb{R}^{d_t}$ (respectively, $\tilde{A}_t \in \mathbb{R}^{d_t \times k}$) to be the subvector of $\tilde{x}$ (resp., submatrix of $\tilde{A}$) obtained by selecting the d_t entries (resp., rows) in the t-th group. Then $\tilde{x}_1, \tilde{x}_2, \tilde{x}_3$ are conditionally independent given h; $\mathbb{E}[\tilde{x}_t | h = j] = \tilde{A}_t e_j$ for each $j \in [k]$ and $t \in \{1, 2, 3\}$; and with probability 1, the matrices $\tilde{A}_1, \tilde{A}_2, \tilde{A}_3$ have full column rank.*

It is possible to obtain a quantitative bound on the k-th largest singular value of each A_t in terms of the k-th largest singular value of A (analogous to Proposition 3.2). One avenue is to show that a random rotation in fact causes $\tilde{A}$ to have low coherence, after which we can apply Proposition 3.2. With this approach, it is sufficient to require $n = O(k \log k)$ (for constant ε and δ), which results in the k-th largest singular value of each A_t being a constant fraction of the k-th largest singular value of A. We conjecture that, in fact, $n \geq c \cdot k$ for some $c > 3$ suffices.

3.3.3 Hidden Markov Models

Our last example is the time-homogeneous HMM for sequences of vector-valued observations $x_1, x_2, \ldots \in \mathbb{R}^d$. Consider a Markov chain of discrete hidden states $y_1 \to y_2 \to y_3 \to \cdots$ over k possible states $[k]$; given a state y_t at time t, the observation x_t at time t (a random vector taking values in $\mathbb{R}^d$) is independent of all other observations and hidden states. See Figure 1(b).

Let $\pi \in \Delta^{k-1}$ be the initial state distribution (*i.e.*, the distribution of y_1), and $T \in \mathbb{R}^{k \times k}$ be the stochastic transition matrix for the hidden state Markov chain: for all times t,

$$\Pr[y_{t+1} = i | y_t = j] = T_{i,j}, \quad i, j \in [k].$$

Finally, let $O \in \mathbb{R}^{d \times k}$ be the matrix whose j-th column is the conditional expectation of x_t given $y_t = j$: for all times t,

$$\mathbb{E}[x_t | y_t = j] = O e_j, \quad j \in [k].$$

Proposition 3.4 (Anandkumar et al., 2012c) *Define $h := y_2$, where y_2 is the second hidden state in the Markov chain. Then*

- *x_1, x_2, x_3 are conditionally independent given h;*
- *the distribution of h is given by the vector $w := T\pi \in \Delta^{k-1}$;*
- *for all $j \in [k]$,*

$$\begin{aligned}\mathbb{E}[x_1|h=j] &= O\,\mathrm{diag}(\pi)T^\top \mathrm{diag}(w)^{-1}e_j \\ \mathbb{E}[x_2|h=j] &= Oe_j \\ \mathbb{E}[x_3|h=j] &= OTe_j.\end{aligned}$$

Note the matrix of conditional means of x_t has full column rank, for each $t \in \{1,2,3\}$, provided that: (i) O has full column rank, (ii) T is invertible, and (iii) π and $T\pi$ have positive entries.

4. Orthogonal Tensor Decompositions

We now show how recovering the μ_i's in our aforementioned problems reduces to the problem of finding a certain orthogonal tensor decomposition of a symmetric tensor. We start by reviewing the spectral decomposition of symmetric matrices, and then discuss a generalization to the higher-order tensor case. Finally, we show how orthogonal tensor decompositions can be used for estimating the latent variable models from the previous section.

4.1 Review: The Matrix Case

We first build intuition by reviewing the matrix setting, where the desired decomposition is the eigendecomposition of a symmetric rank-k matrix $M = V\Lambda V^\top$, where $V = [v_1|v_2|\cdots|v_k] \in \mathbb{R}^{n\times k}$ is the matrix with orthonormal eigenvectors as columns, and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_k) \in \mathbb{R}^{k\times k}$ is diagonal matrix of non-zero eigenvalues. In other words,

$$M \;=\; \sum_{i=1}^{k} \lambda_i \; v_i v_i^\top = \sum_{i=1}^{k} \lambda_i \; v_i^{\otimes 2}. \tag{1}$$

Such a decomposition is guaranteed to exist for every symmetric matrix.

Recovery of the v_i's and λ_i's can be viewed at least two ways. First, each v_i is fixed under the mapping $u \mapsto Mu$, up to a scaling factor λ_i:

$$Mv_i = \sum_{j=1}^{k} \lambda_j (v_j^\top v_i) v_j = \lambda_i v_i$$

as $v_j^\top v_i = 0$ for all $j \neq i$ by orthogonality. The v_i's are not necessarily the only such fixed points. For instance, with the multiplicity $\lambda_1 = \lambda_2 = \lambda$, then any linear combination of v_1 and v_2 is similarly fixed under M. However, in this case, the decomposition in (1) is not unique, as $\lambda_1 v_1 v_1^\top + \lambda_2 v_2 v_2^\top$ is equal to $\lambda(u_1 u_1^\top + u_2 u_2^\top)$ for any pair of orthonormal vectors,

u_1 and u_2 spanning the same subspace as v_1 and v_2. Nevertheless, the decomposition is unique when $\lambda_1, \lambda_2, \ldots, \lambda_k$ are distinct, whereupon the v_j's are the only directions fixed under $u \mapsto Mu$ up to non-trivial scaling.

The second view of recovery is via the variational characterization of the eigenvalues. Assume $\lambda_1 > \lambda_2 > \cdots > \lambda_k$; the case of repeated eigenvalues again leads to similar non-uniqueness as discussed above. Then the *Rayleigh quotient*

$$u \mapsto \frac{u^\top M u}{u^\top u}$$

is maximized over non-zero vectors by v_1. Furthermore, for any $s \in [k]$, the maximizer of the Rayleigh quotient, subject to being orthogonal to $v_1, v_2, \ldots, v_{s-1}$, is v_s. Another way of obtaining this second statement is to consider the *deflated* Rayleigh quotient

$$u \mapsto \frac{u^\top \left(M - \sum_{j=1}^{s-1} \lambda_j v_j v_j^\top \right) u}{u^\top u}$$

and observe that v_s is the maximizer.

Efficient algorithms for finding these matrix decompositions are well studied (Golub and van Loan, 1996, Section 8.2.3), and iterative power methods are one effective class of algorithms.

We remark that in our multilinear tensor notation, we may write the maps $u \mapsto Mu$ and $u \mapsto u^\top M u / \|u\|_2^2$ as

$$u \mapsto Mu \;\equiv\; u \mapsto M(I, u), \tag{2}$$

$$u \mapsto \frac{u^\top M u}{u^\top u} \;\equiv\; u \mapsto \frac{M(u, u)}{u^\top u}. \tag{3}$$

4.2 The Tensor Case

Decomposing general tensors is a delicate issue; tensors may not even have unique decompositions. Fortunately, the orthogonal tensors that arise in the aforementioned models have a structure which permits a unique decomposition under a mild non-degeneracy condition. We focus our attention to the case $p = 3$, *i.e.*, a third order tensor; the ideas extend to general p with minor modifications.

An *orthogonal decomposition* of a symmetric tensor $T \in \bigotimes^3 \mathbb{R}^n$ is a collection of orthonormal (unit) vectors $\{v_1, v_2, \ldots, v_k\}$ together with corresponding positive scalars $\lambda_i > 0$ such that

$$T = \sum_{i=1}^{k} \lambda_i v_i^{\otimes 3}. \tag{4}$$

Note that since we are focusing on odd-order tensors ($p = 3$), we have added the requirement that the λ_i be positive. This convention can be followed without loss of generality since $-\lambda_i v_i^{\otimes p} = \lambda_i (-v_i)^{\otimes p}$ whenever p is odd. Also, it should be noted that orthogonal decompositions do not necessarily exist for every symmetric tensor.

In analogy to the matrix setting, we consider two ways to view this decomposition: a fixed-point characterization and a variational characterization. Related characterizations based on optimal rank-1 approximations are given by Zhang and Golub (2001).

4.2.1 Fixed-Point Characterization

For a tensor T, consider the vector-valued map

$$u \mapsto T(I, u, u) \tag{5}$$

which is the third-order generalization of (2). This can be explicitly written as

$$T(I, u, u) = \sum_{i=1}^{d} \sum_{1 \le j,l \le d} T_{i,j,l}(e_j^\top u)(e_l^\top u)e_i.$$

Observe that (5) is *not* a linear map, which is a key difference compared to the matrix case.

An eigenvector u for a matrix M satisfies $M(I, u) = \lambda u$, for some scalar λ. We say a unit vector $u \in \mathbb{R}^n$ is an *eigenvector* of T, with corresponding *eigenvalue* $\lambda \in \mathbb{R}$, if

$$T(I, u, u) = \lambda u.$$

(To simplify the discussion, we assume throughout that eigenvectors have unit norm; otherwise, for scaling reasons, we replace the above equation with $T(I, u, u) = \lambda \|u\| u$.) This concept was originally introduced by Lim (2005) and Qi (2005). For orthogonally decomposable tensors $T = \sum_{i=1}^{k} \lambda_i v_i^{\otimes 3}$,

$$T(I, u, u) = \sum_{i=1}^{k} \lambda_i (u^\top v_i)^2 v_i \ .$$

By the orthogonality of the v_i, it is clear that $T(I, v_i, v_i) = \lambda_i v_i$ for all $i \in [k]$. Therefore each (v_i, λ_i) is an eigenvector/eigenvalue pair.

There are a number of subtle differences compared to the matrix case that arise as a result of the non-linearity of (5). First, even with the multiplicity $\lambda_1 = \lambda_2 = \lambda$, a linear combination $u := c_1 v_1 + c_2 v_2$ may *not* be an eigenvector. In particular,

$$T(I, u, u) = \lambda_1 c_1^2 v_1 + \lambda_2 c_2^2 v_2 = \lambda(c_1^2 v_1 + c_2^2 v_2)$$

may not be a multiple of $c_1 v_1 + c_2 v_2$. This indicates that the issue of repeated eigenvalues does not have the same status as in the matrix case. Second, even if all the eigenvalues are distinct, it turns out that the v_i's are not the only eigenvectors. For example, set $u := (1/\lambda_1)v_1 + (1/\lambda_2)v_2$. Then,

$$T(I, u, u) = \lambda_1 (1/\lambda_1)^2 v_1 + \lambda_2 (1/\lambda_2)^2 v_2 = u,$$

so $u/\|u\|$ is an eigenvector. More generally, for any subset $S \subseteq [k]$, the vector

$$\sum_{i \in S} \frac{1}{\lambda_i} \cdot v_i$$

is (proportional to) an eigenvector.

As we now see, these additional eigenvectors can be viewed as spurious. We say a unit vector u is a *robust eigenvector* of T if there exists an $\epsilon > 0$ such that for all $\theta \in \{u' \in \mathbb{R}^n : \|u' - u\| \leq \epsilon\}$, repeated iteration of the map

$$\bar{\theta} \mapsto \frac{T(I, \bar{\theta}, \bar{\theta})}{\|T(I, \bar{\theta}, \bar{\theta})\|}, \tag{6}$$

starting from θ converges to u. Note that the map (6) rescales the output to have unit Euclidean norm. Robust eigenvectors are also called attracting fixed points of (6) (see, *e.g.*, Kolda and Mayo, 2011).

The following theorem implies that if T has an orthogonal decomposition as given in (4), then the set of robust eigenvectors of T are precisely the set $\{v_1, v_2, \ldots v_k\}$, implying that the orthogonal decomposition is unique. (For even order tensors, the uniqueness is true up to sign-flips of the v_i.)

Theorem 4.1 *Let T have an orthogonal decomposition as given in* (4)*.*

1. *The set of $\theta \in \mathbb{R}^n$ which do not converge to some v_i under repeated iteration of* (6) *has measure zero.*

2. *The set of robust eigenvectors of T is equal to $\{v_1, v_2, \ldots, v_k\}$.*

The proof of Theorem 4.1 is given in Appendix A.1, and follows readily from simple orthogonality considerations. Note that *every* v_i in the orthogonal tensor decomposition is robust, whereas for a symmetric matrix M, for almost all initial points, the map $\bar{\theta} \mapsto \frac{M\bar{\theta}}{\|M\bar{\theta}\|}$ converges only to an eigenvector corresponding to the largest magnitude eigenvalue. Also, since the tensor order is odd, the signs of the robust eigenvectors are fixed, as each $-v_i$ is mapped to v_i under (6).

4.2.2 Variational Characterization

We now discuss a variational characterization of the orthogonal decomposition. The *generalized Rayleigh quotient* (Zhang and Golub, 2001) for a third-order tensor is

$$u \mapsto \frac{T(u, u, u)}{(u^\top u)^{3/2}},$$

which can be compared to (3). For an orthogonally decomposable tensor, the following theorem shows that a non-zero vector $u \in \mathbb{R}^n$ is an *isolated local maximizer* (Nocedal and Wright, 1999) of the generalized Rayleigh quotient if and only if $u = v_i$ for some $i \in [k]$.

Theorem 4.2 *Let T have an orthogonal decomposition as given in* (4)*, and consider the optimization problem*

$$\max_{u \in \mathbb{R}^n} T(u, u, u) \ \textit{s.t.} \ \|u\| \leq 1.$$

1. *The stationary points are eigenvectors of T.*

2. *A stationary point u is an isolated local maximizer if and only if $u = v_i$ for some $i \in [k]$.*

The proof of Theorem 4.2 is given in Appendix A.2. It is similar to local optimality analysis for ICA methods using fourth-order cumulants (*e.g.*, Delfosse and Loubaton, 1995; Frieze et al., 1996).

Again, we see similar distinctions to the matrix case. In the matrix case, the only local maximizers of the Rayleigh quotient are the eigenvectors with the largest eigenvalue (and these maximizers take on the globally optimal value). For the case of orthogonal tensor forms, the robust eigenvectors are precisely the isolated local maximizers.

An important implication of the two characterizations is that, for orthogonally decomposable tensors T, (i) the local maximizers of the objective function $u \mapsto T(u,u,u)/(u^\top u)^{3/2}$ correspond precisely to the vectors v_i in the decomposition, and (ii) these local maximizers can be reliably identified using a simple fixed-point iteration (*i.e.*, the tensor analogue of the matrix power method). Moreover, a second-derivative test based on $T(I,I,u)$ can be employed to test for local optimality and rule out other stationary points.

4.3 Estimation via Orthogonal Tensor Decompositions

We now demonstrate how the moment tensors obtained for various latent variable models in Section 3 can be reduced to an orthogonal form. For concreteness, we take the specific form from the exchangeable single topic model (Theorem 3.1):

$$\begin{aligned} M_2 &= \sum_{i=1}^{k} w_i \ \mu_i \otimes \mu_i, \\ M_3 &= \sum_{i=1}^{k} w_i \ \mu_i \otimes \mu_i \otimes \mu_i. \end{aligned}$$

(The more general case allows the weights w_i in M_2 to differ in M_3, but for simplicity we keep them the same in the following discussion.) We now show how to reduce these forms to an orthogonally decomposable tensor from which the w_i and μ_i can be recovered. See Appendix D for a discussion as to how previous approaches (Mossel and Roch, 2006; Anandkumar et al., 2012c,a; Hsu and Kakade, 2013) achieved this decomposition through a certain simultaneous diagonalization method.

Throughout, we assume the following non-degeneracy condition.

Condition 4.1 (Non-degeneracy) *The vectors $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^d$ are linearly independent, and the scalars $w_1, w_2, \ldots, w_k > 0$ are strictly positive.*

Observe that Condition 4.1 implies that $M_2 \succeq 0$ is positive semidefinite and has rank k. This is often a mild condition in applications. When this condition is not met, learning is conjectured to be generally hard for both computational (Mossel and Roch, 2006) and information-theoretic reasons (Moitra and Valiant, 2010). As discussed by Hsu et al. (2012b) and Hsu and Kakade (2013), when the non-degeneracy condition does not hold, it is often possible to combine multiple observations using tensor products to increase the rank of the relevant matrices. Indeed, this observation has been rigorously formulated in very recent works of Bhaskara et al. (2014) and Anderson et al. (2014) using the framework of smoothed analysis (Spielman and Teng, 2009).

4.3.1 The Reduction

First, let $W \in \mathbb{R}^{d \times k}$ be a linear transformation such that

$$M_2(W, W) \;=\; W^\top M_2 W \;=\; I$$

where I is the $k \times k$ identity matrix (*i.e.*, W whitens M_2). Since $M_2 \succeq 0$, we may for concreteness take $W := UD^{-1/2}$, where $U \in \mathbb{R}^{d \times k}$ is the matrix of orthonormal eigenvectors of M_2, and $D \in \mathbb{R}^{k \times k}$ is the diagonal matrix of positive eigenvalues of M_2. Let

$$\tilde{\mu}_i := \sqrt{w_i}\; W^\top \mu_i.$$

Observe that

$$M_2(W, W) \;=\; \sum_{i=1}^{k} W^\top (\sqrt{w_i}\mu_i)(\sqrt{w_i}\mu_i)^\top W \;=\; \sum_{i=1}^{k} \tilde{\mu}_i \tilde{\mu}_i^\top \;=\; I,$$

so the $\tilde{\mu}_i \in \mathbb{R}^k$ are orthonormal vectors.

Now define $\widetilde{M}_3 := M_3(W, W, W) \in \mathbb{R}^{k \times k \times k}$, so that

$$\widetilde{M}_3 \;=\; \sum_{i=1}^{k} w_i \; (W^\top \mu_i)^{\otimes 3} \;=\; \sum_{i=1}^{k} \frac{1}{\sqrt{w_i}} \; \tilde{\mu}_i^{\otimes 3}.$$

As the following theorem shows, the orthogonal decomposition of $\widetilde{M}_3$ can be obtained by identifying its robust eigenvectors, upon which the original parameters w_i and μ_i can be recovered. For simplicity, we only state the result in terms of robust eigenvector/eigenvalue pairs; one may also easily state everything in variational form using Theorem 4.2.

Theorem 4.3 *Assume Condition 4.1 and take $\widetilde{M}_3$ as defined above.*

1. *The set of robust eigenvectors of $\widetilde{M}_3$ is equal to $\{\tilde{\mu}_1, \tilde{\mu}_2, \ldots, \tilde{\mu}_k\}$.*
2. *The eigenvalue corresponding to the robust eigenvector $\tilde{\mu}_i$ of $\widetilde{M}_3$ is equal to $1/\sqrt{w_i}$, for all $i \in [k]$.*
3. *If $B \in \mathbb{R}^{d \times k}$ is the Moore-Penrose pseudoinverse of $W^\top$, and (v, λ) is a robust eigenvector/eigenvalue pair of $\widetilde{M}_3$, then $\lambda B v = \mu_i$ for some $i \in [k]$.*

The theorem follows by combining the above discussion with the robust eigenvector characterization of Theorem 4.1. Recall that we have taken as convention that eigenvectors have unit norm, so the μ_i are exactly determined from the robust eigenvector/eigenvalue pairs of $\widetilde{M}_3$ (together with the pseudoinverse of $W^\top$); in particular, the scale of each μ_i is correctly identified (along with the corresponding w_i). Relative to previous works on moment-based estimators for latent variable models (*e.g.*, Anandkumar et al., 2012c,a; Hsu and Kakade, 2013), Theorem 4.3 emphasizes the role of the special tensor structure, which in turn makes transparent the applicability of methods for orthogonal tensor decomposition.

4.3.2 Local Maximizers of (Cross Moment) Skewness

The variational characterization provides an interesting perspective on the robust eigenvectors for these latent variable models. Consider the exchangeable single topic models (Theorem 3.1), and the objective function

$$u \mapsto \frac{\mathbb{E}[(x_1^\top u)(x_2^\top u)(x_3^\top u)]}{\mathbb{E}[(x_1^\top u)(x_2^\top u)]^{3/2}} = \frac{M_3(u,u,u)}{M_2(u,u)^{3/2}}.$$

In this case, every local maximizer u^* satisfies $M_2(I, u^*) = \sqrt{w_i}\mu_i$ for some $i \in [k]$. The objective function can be interpreted as the (cross moment) skewness of the random vectors x_1, x_2, x_3 along direction u.

5. Tensor Power Method

In this section, we consider the tensor power method of Lathauwer et al. (2000, Remark 3) for orthogonal tensor decomposition. We first state a simple convergence analysis for an orthogonally decomposable tensor T.

When only an approximation $\hat{T}$ to an orthogonally decomposable tensor T is available (*e.g.*, when empirical moments are used to estimate population moments), an orthogonal decomposition need not exist for this perturbed tensor (unlike for the case of matrices), and a more robust approach is required to extract the approximate decomposition. Here, we propose such a variant in Algorithm 1 and provide a detailed perturbation analysis. We note that alternative approaches such as simultaneous diagonalization can also be employed (see Appendix D).

5.1 Convergence Analysis for Orthogonally Decomposable Tensors

The following lemma establishes the quadratic convergence of the tensor power method—*i.e.*, repeated iteration of (6)—for extracting a single component of the orthogonal decomposition. Note that the initial vector θ_0 determines which robust eigenvector will be the convergent point. Computation of subsequent eigenvectors can be computed with deflation, *i.e.*, by subtracting appropriate terms from T.

Lemma 5.1 *Let $T \in \bigotimes^3 \mathbb{R}^n$ have an orthogonal decomposition as given in (4). For a vector $\theta_0 \in \mathbb{R}^n$, suppose that the set of numbers $|\lambda_1 v_1^\top \theta_0|, |\lambda_2 v_2^\top \theta_0|, \ldots, |\lambda_k v_k^\top \theta_0|$ has a unique largest element. Without loss of generality, say $|\lambda_1 v_1^\top \theta_0|$ is this largest value and $|\lambda_2 v_2^\top \theta_0|$ is the second largest value. For $t = 1, 2, \ldots$, let*

$$\theta_t \;:=\; \frac{T(I, \theta_{t-1}, \theta_{t-1})}{\|T(I, \theta_{t-1}, \theta_{t-1})\|}.$$

Then

$$\|v_1 - \theta_t\|^2 \leq \left(2\lambda_1^2 \sum_{i=2}^{k} \lambda_i^{-2}\right) \cdot \left|\frac{\lambda_2 v_2^\top \theta_0}{\lambda_1 v_1^\top \theta_0}\right|^{2^{t+1}}.$$

That is, repeated iteration of (6) starting from θ_0 converges to v_1 at a quadratic rate.

To obtain all eigenvectors, we may simply proceed iteratively using deflation, executing the power method on $T - \sum_j \lambda_j v_j^{\otimes 3}$ after having obtained robust eigenvector / eigenvalue pairs $\{(v_j, \lambda_j)\}$.

Proof Let $\overline{\theta}_0, \overline{\theta}_1, \overline{\theta}_2, \ldots$ be the sequence given by $\overline{\theta}_0 := \theta_0$ and $\overline{\theta}_t := T(I, \theta_{t-1}, \theta_{t-1})$ for $t \geq 1$. Let $c_i := v_i^\top \theta_0$ for all $i \in [k]$. It is easy to check that (i) $\theta_t = \overline{\theta}_t / \|\overline{\theta}_t\|$, and (ii) $\overline{\theta}_t = \sum_{i=1}^k \lambda_i^{2^t-1} c_i^{2^t} v_i$. (Indeed, $\overline{\theta}_{t+1} = \sum_{i=1}^k \lambda_i (v_i^\top \overline{\theta}_t)^2 v_i = \sum_{i=1}^k \lambda_i (\lambda_i^{2^t-1} c_i^{2^t})^2 v_i = \sum_{i=1}^k \lambda_i^{2^{t+1}-1} c_i^{2^{t+1}} v_i$.) Then

$$1 - (v_1^\top \theta_t)^2 = 1 - \frac{(v_1^\top \overline{\theta}_t)^2}{\|\overline{\theta}_t\|^2} = 1 - \frac{\lambda_1^{2^{t+1}-2} c_1^{2^{t+1}}}{\sum_{i=1}^k \lambda_i^{2^{t+1}-2} c_i^{2^{t+1}}} \leq \frac{\sum_{i=2}^k \lambda_i^{2^{t+1}-2} c_i^{2^{t+1}}}{\sum_{i=1}^k \lambda_i^{2^{t+1}-2} c_i^{2^{t+1}}}$$
$$\leq \lambda_1^2 \sum_{i=2}^k \lambda_i^{-2} \cdot \left| \frac{\lambda_2 c_2}{\lambda_1 c_1} \right|^{2^{t+1}}.$$

Since $\lambda_1 > 0$, we have $v_1^\top \theta_t > 0$ and hence $\|v_1 - \theta_t\|^2 = 2(1 - v_1^\top \theta_t) \leq 2(1 - (v_1^\top \theta_t)^2)$ as required. ■

5.2 Perturbation Analysis of a Robust Tensor Power Method

Now we consider the case where we have an approximation $\hat{T}$ to an orthogonally decomposable tensor T. Here, a more robust approach is required to extract an approximate decomposition. We propose such an algorithm in Algorithm 1, and provide a detailed perturbation analysis. For simplicity, we assume the tensor $\hat{T}$ is of size $k \times k \times k$ as per the reduction from Section 4.3. In some applications, it may be preferable to work directly with a $n \times n \times n$ tensor of rank $k \leq n$ (as in Lemma 5.1); our results apply in that setting with little modification.

Algorithm 1 Robust tensor power method

input symmetric tensor $\tilde{T} \in \mathbb{R}^{k \times k \times k}$, number of iterations L, N.

output the estimated eigenvector/eigenvalue pair; the deflated tensor.

1: **for** $\tau = 1$ to L **do**
2: Draw $\theta_0^{(\tau)}$ uniformly at random from the unit sphere in $\mathbb{R}^k$.
3: **for** $t = 1$ to N **do**
4: Compute power iteration update

$$\theta_t^{(\tau)} := \frac{\tilde{T}(I, \theta_{t-1}^{(\tau)}, \theta_{t-1}^{(\tau)})}{\|\tilde{T}(I, \theta_{t-1}^{(\tau)}, \theta_{t-1}^{(\tau)})\|} \tag{7}$$

5: **end for**
6: **end for**
7: Let $\tau^* := \arg\max_{\tau \in [L]} \{\tilde{T}(\theta_N^{(\tau)}, \theta_N^{(\tau)}, \theta_N^{(\tau)})\}$.
8: Do N power iteration updates (7) starting from $\theta_N^{(\tau^*)}$ to obtain $\hat{\theta}$, and set $\hat{\lambda} := \tilde{T}(\hat{\theta}, \hat{\theta}, \hat{\theta})$.
9: **return** the estimated eigenvector/eigenvalue pair $(\hat{\theta}, \hat{\lambda})$; the deflated tensor $\tilde{T} - \hat{\lambda}\, \hat{\theta}^{\otimes 3}$.

Assume that the symmetric tensor $T \in \mathbb{R}^{k\times k\times k}$ is orthogonally decomposable, and that $\hat{T} = T + E$, where the perturbation $E \in \mathbb{R}^{k\times k\times k}$ is a symmetric tensor with small operator norm:

$$\|E\| := \sup_{\|\theta\|=1} |E(\theta,\theta,\theta)|.$$

In our latent variable model applications, $\hat{T}$ is the tensor formed by using empirical moments, while T is the orthogonally decomposable tensor derived from the population moments for the given model. In the context of parameter estimation (as in Section 4.3), E must account for any error amplification throughout the reduction, such as in the whitening step (see, *e.g.*, Hsu and Kakade, 2013, for such an analysis).

The following theorem is similar to Wedin's perturbation theorem for singular vectors of matrices (Wedin, 1972) in that it bounds the error of the (approximate) decomposition returned by Algorithm 1 on input $\hat{T}$ in terms of the size of the perturbation, provided that the perturbation is small enough.

Theorem 5.1 *Let $\hat{T} = T + E \in \mathbb{R}^{k\times k\times k}$, where T is a symmetric tensor with orthogonal decomposition $T = \sum_{i=1}^{k} \lambda_i v_i^{\otimes 3}$ where each $\lambda_i > 0$, $\{v_1, v_2, \ldots, v_k\}$ is an orthonormal basis, and E is a symmetric tensor with operator norm $\|E\| \le \epsilon$. Define $\lambda_{\min} := \min\{\lambda_i : i \in [k]\}$, and $\lambda_{\max} := \max\{\lambda_i : i \in [k]\}$. There exists universal constants $C_1, C_2, C_3 > 0$ such that the following holds. Pick any $\eta \in (0,1)$, and suppose*

$$\epsilon \le C_1 \cdot \frac{\lambda_{\min}}{k}, \qquad N \ge C_2 \cdot \left(\log(k) + \log\log\Big(\frac{\lambda_{\max}}{\epsilon}\Big)\right),$$

and

$$\sqrt{\frac{\ln(L/\log_2(k/\eta))}{\ln(k)}} \cdot \left(1 - \frac{\ln(\ln(L/\log_2(k/\eta))) + C_3}{4\ln(L/\log_2(k/\eta))} - \sqrt{\frac{\ln(8)}{\ln(L/\log_2(k/\eta))}}\right) \ge 1.02\left(1 + \sqrt{\frac{\ln(4)}{\ln(k)}}\right).$$

(Note that the condition on L holds with $L = \mathrm{poly}(k)\log(1/\eta)$.) Suppose that Algorithm 1 is iteratively called k times, where the input tensor is $\hat{T}$ in the first call, and in each subsequent call, the input tensor is the deflated tensor returned by the previous call. Let $(\hat{v}_1, \hat{\lambda}_1), (\hat{v}_2, \hat{\lambda}_2), \ldots, (\hat{v}_k, \hat{\lambda}_k)$ be the sequence of estimated eigenvector/eigenvalue pairs returned in these k calls. With probability at least $1 - \eta$, there exists a permutation π on $[k]$ such that

$$\|v_{\pi(j)} - \hat{v}_j\| \le 8\epsilon/\lambda_{\pi(j)}, \qquad |\lambda_{\pi(j)} - \hat{\lambda}_j| \le 5\epsilon, \quad \forall j \in [k],$$

and

$$\left\| T - \sum_{j=1}^{k} \hat{\lambda}_j \hat{v}_j^{\otimes 3} \right\| \le 55\epsilon.$$

The proof of Theorem 5.1 is given in Appendix B.

One important difference from Wedin's theorem is that this is an algorithm dependent perturbation analysis, specific to Algorithm 1 (since the perturbed tensor need not have an

orthogonal decomposition). Furthermore, note that Algorithm 1 uses multiple restarts to ensure (approximate) convergence—the intuition is that by restarting at multiple points, we eventually start at a point in which the initial contraction towards some eigenvector dominates the error E in our tensor. The proof shows that we find such a point with high probability within $L = \text{poly}(k)$ trials. It should be noted that for large k, the required bound on L is very close to linear in k.

We note that it is also possible to design a variant of Algorithm 1 that instead uses a stopping criterion to determine if an iterate has (almost) converged to an eigenvector. For instance, if $\tilde{T}(\theta,\theta,\theta) > \max\{\|\tilde{T}\|_F/\sqrt{2r}, \|\tilde{T}(I,I,\theta)\|_F/1.05\}$, where $\|\tilde{T}\|_F$ is the tensor Frobenius norm (vectorized Euclidean norm), and r is the expected rank of the unperturbed tensor ($r = k - \#$ of deflation steps), then it can be shown that θ must be close to one of the eigenvectors, provided that the perturbation is small enough. Using such a stopping criterion can reduce the number of random restarts when a good initial point is found early on. See Appendix C for details.

In general, it is possible, when run on a general symmetric tensor (*e.g.*, $\hat{T}$), for the tensor power method to exhibit oscillatory behavior (Kofidis and Regalia, 2002, Example 1). This is not in conflict with Theorem 5.1, which effectively bounds the amplitude of these oscillations; in particular, if $\hat{T} = T + E$ is a tensor built from empirical moments, the error term E (and thus the amplitude of the oscillations) can be driven down by drawing more samples. The practical value of addressing these oscillations and perhaps stabilizing the algorithm is an interesting direction for future research (Kolda and Mayo, 2011).

A final consideration is that for specific applications, it may be possible to use domain knowledge to choose better initialization points. For instance, in the topic modeling applications (*cf.* Section 3.1), the eigenvectors are related to the topic word distributions, and many documents may be primarily composed of words from just single topic. Therefore, good initialization points can be derived from these single-topic documents themselves, as these points would already be close to one of the eigenvectors.

6. Discussion

In this section, we discuss some practical and application-oriented issues related to the tensor decomposition approach to learning latent variable models.

6.1 Practical Implementation Considerations

A number of practical concerns arise when dealing with moment matrices and tensors. Below, we address two issues that are especially pertinent to topic modeling applications (Anandkumar et al., 2012c,a) or other settings where the observations are sparse.

6.1.1 Efficient Moment Representation for Exchangeable Models

In an exchangeable bag-of-words model, it is assumed that the words $x_1, x_2, \ldots, x_\ell$ in a document are conditionally i.i.d. given the topic h. This allows one to estimate p-th order moments using just p words per document. The estimators obtained via Theorem 3.1 (single topic model) and Theorem 3.5 (LDA) use only up to third-order moments, which suggests that each document only needs to have three words.

In practice, one should use all of the words in a document for efficient estimation of the moments. One way to do this is to average over all $\binom{\ell}{3} \cdot 3!$ ordered triples of words in a document of length ℓ. At first blush, this seems computationally expensive (when ℓ is large), but as it turns out, the averaging can be done implicitly, as shown by Zou et al. (2013). Let $c \in \mathbb{R}^d$ be the word count vector for a document of length ℓ, so c_i is the number of occurrences of word i in the document, and $\sum_{i=1}^d c_i = \ell$. Note that c is a sufficient statistic for the document. Then, the contribution of this document to the empirical third-order moment tensor is given by

$$\frac{1}{\binom{\ell}{3}} \cdot \frac{1}{3!} \cdot \Bigg(c \otimes c \otimes c + 2 \sum_{i=1}^d c_i \ (e_i \otimes e_i \otimes e_i) \\ - \sum_{i=1}^d \sum_{j=1}^d c_i c_j \ (e_i \otimes e_i \otimes e_j) - \sum_{i=1}^d \sum_{j=1}^d c_i c_j \ (e_i \otimes e_j \otimes e_i) - \sum_{i=1}^d \sum_{j=1}^d c_i c_j \ (e_i \otimes e_j \otimes e_j) \Bigg). \quad (8)$$

It can be checked that this quantity is equal to

$$\frac{1}{\binom{\ell}{3}} \cdot \frac{1}{3!} \cdot \sum_{\text{ordered word triple } (x,y,z)} e_x \otimes e_y \otimes e_z$$

where the sum is over all ordered word triples in the document. A similar expression is easily derived for the contribution of the document to the empirical second-order moment matrix:

$$\frac{1}{\binom{\ell}{2}} \cdot \frac{1}{2!} \cdot \Big(c \otimes c - \mathrm{diag}(c) \Big). \quad (9)$$

Note that the word count vector c is generally a sparse vector, so this representation allows for efficient multiplication by the moment matrices and tensors in time linear in the size of the document corpus (*i.e.*, the number of non-zero entries in the term-document matrix).

6.1.2 Dimensionality Reduction

Another serious concern regarding the use of tensor forms of moments is the need to operate on multidimensional arrays with $\Omega(d^3)$ values (it is typically not exactly d^3 due to symmetry). When d is large (*e.g.*, when it is the size of the vocabulary in natural language applications), even storing a third-order tensor in memory can be prohibitive. Sparsity is one factor that alleviates this problem. Another approach is to use efficient linear dimensionality reduction. When this is combined with efficient techniques for matrix and tensor multiplication that avoid explicitly constructing the moment matrices and tensors (such as the procedure described above), it is possible to avoid any computational scaling more than linear in the dimension d and the training sample size.

Consider for concreteness the tensor decomposition approach for the exchangeable single topic model as discussed in Section 4.3. Using recent techniques for randomized linear algebra computations (*e.g.*, Halko et al., 2011), it is possible to efficiently approximate the whitening matrix $W \in \mathbb{R}^{d \times k}$ from the second-moment matrix $M_2 \in \mathbb{R}^{d \times d}$. To do this, one first multiplies M_2 by a random matrix $R \in \mathbb{R}^{d \times k'}$ for some $k' \geq k$, and then computes the top k singular vectors of the product $M_2 R$. This provides a basis $U \in \mathbb{R}^{d \times k}$ whose span

is approximately the range of M_2. From here, an approximate SVD of $U^\top M_2 U$ is used to compute the approximate whitening matrix W. Note that both matrix products $M_2 R$ and $U^\top M_2 U$ may be performed via implicit access to M_2 by exploiting (9), so that M_2 need not be explicitly formed. With the whitening matrix W in hand, the third-moment tensor $\widetilde{M}_3 = M_3(W, W, W) \in \mathbb{R}^{k \times k \times k}$ can be implicitly computed via (8). For instance, the core computation in the tensor power method $\theta' := \widetilde{M}_3(I, \theta, \theta)$ is performed by (i) computing $\eta := W\theta$, (ii) computing $\eta' := M_3(I, \eta, \eta)$, and finally (iii) computing $\theta' := W^\top \eta'$. Using the fact that M_3 is an empirical third-order moment tensor, these steps can be computed with $O(dk + N)$ operations, where N is the number of non-zero entries in the term-document matrix (Zou et al., 2013).

6.2 Computational Complexity

It is interesting to consider the computational complexity of the tensor power method in the dense setting where $T \in \mathbb{R}^{k \times k \times k}$ is orthogonally decomposable but otherwise unstructured. Each iteration requires $O(k^3)$ operations, and assuming at most $k^{1+\delta}$ random restarts for extracting each eigenvector (for some small $\delta > 0$) and $O(\log(k) + \log\log(1/\epsilon))$ iterations per restart, the total running time is $O(k^{5+\delta}(\log(k) + \log\log(1/\epsilon)))$ to extract all k eigenvectors and eigenvalues.

An alternative approach to extracting the orthogonal decomposition of T is to reorganize T into a matrix $M \in \mathbb{R}^{k \times k^2}$ by flattening two of the dimensions into one. In this case, if $T = \sum_{i=1}^k \lambda_i v_i^{\otimes 3}$, then $M = \sum_{i=1}^k \lambda_i v_i \otimes \mathrm{vec}(v_i \otimes v_i)$. This reveals the singular value decomposition of M (assuming the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_k$ are distinct), and therefore can be computed with $O(k^4)$ operations. Therefore it seems that the tensor power method is less efficient than a pure matrix-based approach via singular value decomposition. However, it should be noted that this matrix-based approach fails to recover the decomposition when eigenvalues are repeated, and can be unstable when the gap between eigenvalues is small—see Appendix D for more discussion.

It is worth noting that the running times differ by roughly a factor of $\Theta(k^{1+\delta})$, which can be accounted for by the random restarts. This gap can potentially be alleviated or removed by using a more clever method for initialization. Moreover, using special structure in the problem (as discussed above) can also improve the running time of the tensor power method.

6.3 Sample Complexity Bounds

Previous work on using linear algebraic methods for estimating latent variable models crucially rely on matrix perturbation analysis for deriving sample complexity bounds (Mossel and Roch, 2006; Hsu et al., 2012b; Anandkumar et al., 2012c,a; Hsu and Kakade, 2013). The learning algorithms in these works are plug-in estimators that use empirical moments in place of the population moments, and then follow algebraic manipulations that result in the desired parameter estimates. As long as these manipulations can tolerate small perturbations of the population moments, a sample complexity bound can be obtained by exploiting the convergence of the empirical moments to the population moments via the law of large numbers. As discussed in Appendix D, these approaches do not directly lead to practical

algorithms due to a certain amplification of the error (a polynomial factor of k, which is observed in practice).

Using the perturbation analysis for the tensor power method, improved sample complexity bounds can be obtained for all of the examples discussed in Section 3. The underlying analysis remains the same as in previous works (*e.g.*, Anandkumar et al., 2012a; Hsu and Kakade, 2013), the main difference being the accuracy of the orthogonal tensor decomposition obtained via the tensor power method. Relative to the previously cited works, the sample complexity bound will be considerably improved in its dependence on the rank parameter k, as Theorem 5.1 implies that the tensor estimation error (*e.g.*, error in estimating $\widetilde{M}_3$ from Section 4.3) is not amplified by any factor explicitly depending on k (there is a requirement that the error be smaller than some factor depending on k, but this only contributes to a lower-order term in the sample complexity bound). See Appendix D for further discussion regarding the stability of the techniques from these previous works.

6.4 Other Perspectives

The tensor power method is simply one approach for extracting the orthogonal decomposition needed in parameter estimation. The characterizations from Section 4.2 suggest that a number of fixed point and variational techniques may be possible (and Appendix D provides yet another perspective based on simultaneous diagonalization). One important consideration is that the model is often misspecified, and therefore approaches with more robust guarantees (*e.g.*, for convergence) are desirable. Our own experience with the tensor power method (as applied to exchangeable topic modeling) is that while model misspecification does indeed affect convergence, the results can be very reasonable even after just a dozen or so iterations (Anandkumar et al., 2012a). Nevertheless, robustness is likely more important in other applications, and thus the stabilization approaches (Kofidis and Regalia, 2002; Regalia and Kofidis, 2003; Erdogan, 2009; Kolda and Mayo, 2011) may be advantageous.

Acknowledgments

We thank Boaz Barak, Dean Foster, Jon Kelner, and Greg Valiant for helpful discussions. We are also grateful to Hanzhang Hu, Drew Bagnell, and Martial Hebert for alerting us of an issue with Theorem 4.2 and suggesting a simple fix. This work was completed while DH was a postdoctoral researcher at Microsoft Research New England, and partly while AA, RG, and MT were visiting the same lab. AA is supported in part by the NSF Award CCF-1219234, AFOSR Award FA9550-10-1-0310 and the ARO Award W911NF-12-1-0404.

Appendix A. Fixed-Point and Variational Characterizations of Orthogonal Tensor Decompositions

We give detailed proofs of Theorems 4.1 and 4.2 in this section for completeness.

A.1 Proof of Theorem 4.1

Theorem A.1 *Let T have an orthogonal decomposition as given in* (4).

1. *The set of $\theta \in \mathbb{R}^n$ which do not converge to some v_i under repeated iteration of* (6) *has measure zero.*

2. *The set of robust eigenvectors of T is $\{v_1, v_2, \ldots, v_k\}$.*

Proof For a random choice of $\theta \in \mathbb{R}^n$ (under any distribution absolutely continuous with respect to Lebesgue measure), the values $|\lambda_1 v_1^\top \theta|, |\lambda_2 v_2^\top \theta|, \ldots, |\lambda_k v_k^\top \theta|$ will be distinct with probability 1. Therefore, there exists a unique largest value, say $|\lambda_i v_i^\top \theta|$ for some $i \in [k]$, and by Lemma 5.1, we have convergence to v_i under repeated iteration of (6). Thus the first claim holds.

We now prove the second claim. First, we show that every v_i is a robust eigenvector. Pick any $i \in [k]$, and note that for a sufficiently small ball around v_i, we have that for all θ in this ball, $\lambda_i v_i^\top \theta$ is strictly greater than $\lambda_j v_j^\top \theta$ for $j \in [k] \setminus \{i\}$. Thus by Lemma 5.1, v_i is a robust eigenvector. Now we show that the v_i are the only robust eigenvectors. Suppose there exists some robust eigenvector u not equal to v_i for any $i \in [k]$. Then there exists a positive measure set around u such that all points in this set converge to u under repeated iteration of (6). This contradicts the first claim. ■

A.2 Proof of Theorem 4.2

Theorem A.2 *Let T have an orthogonal decomposition as given in* (4)*, and consider the optimization problem*

$$\max_{u \in \mathbb{R}^n} T(u, u, u) \text{ s.t. } \|u\| \leq 1.$$

1. *The stationary points are eigenvectors of T.*

2. *A stationary point u is an isolated local maximizer if and only if $u = v_i$ for some $i \in [k]$.*

Proof Consider the Lagrangian form of the corresponding constrained maximization problem over unit vectors $u \in \mathbb{R}^n$:

$$\mathcal{L}(u, \lambda) := T(u, u, u) - \frac{3}{2}\lambda(u^\top u - 1).$$

Since

$$\nabla_u \mathcal{L}(u, \lambda) = \nabla_u \Big(\sum_{i=1}^k \lambda_i (v_i^\top u)^3 - \frac{3}{2}\lambda(u^\top u - 1) \Big) = 3\Big(T(I, u, u) - \lambda u \Big),$$

the stationary points $u \in \mathbb{R}^n$ (with $\|u\| \leq 1$) satisfy

$$T(I, u, u) = \lambda u$$

for some $\lambda \in \mathbb{R}$, *i.e.*, (u, λ) is an eigenvector/eigenvalue pair of T.

Now we characterize the isolated local maximizers. Observe that if $u \neq 0$ and $T(I, u, u) = \lambda u$ for $\lambda < 0$, then $T(u, u, u) < 0$. Therefore $u' = (1 - \delta)u$ for any $\delta \in (0, 1)$ satisfies $T(u', u', u') = (1 - \delta)^3 T(u, u, u) > T(u, u, u)$. So such a u cannot be a local maximizer.

Moreover, if $\|u\| < 1$ and $T(I, u, u) = \lambda u$ for $\lambda > 0$, then $u' = (1 + \delta)u$ for a small enough $\delta \in (0, 1)$ satisfies $\|u'\| \leq 1$ and $T(u', u', u') = (1 + \delta)^3 T(u, u, u) > T(u, u, u)$. Therefore a local maximizer must have $T(I, u, u) = \lambda u$ for some $\lambda \geq 0$, and $\|u\| = 1$ whenever $\lambda > 0$.

Extend $\{v_1, v_2, \ldots, v_k\}$ to an orthonormal basis $\{v_1, v_2, \ldots, v_n\}$ of $\mathbb{R}^n$. Now pick any stationary point $u = \sum_{i=1}^n c_i v_i$ with $\lambda := T(u, u, u) = u^\top T(I, u, u)$. Then

$$\lambda_i c_i^2 = \lambda_i (u^\top v_i)^2 = v_i^\top T(I, u, u) = \lambda v_i^\top u = \lambda c_i \geq 0, \quad i \in [k],$$

and thus

$$\nabla_u^2 \mathcal{L}(u, \lambda) = 6 \sum_{i=1}^k \lambda_i c_i \; v_i v_i^\top - 3\lambda I = 3\lambda \left(2 \sum_{i \in \Omega} v_i v_i^\top - I \right)$$

where $\Omega := \{i \in [k] : c_i \neq 0\}$. This implies that for any unit vector $w \in \mathbb{R}^n$,

$$w^\top \nabla_u^2 \mathcal{L}(u, \lambda) w = 3\lambda \left(2 \sum_{i \in \Omega} (v_i^\top w)^2 - 1 \right).$$

The point u is an isolated local maximum if the above quantity is strictly negative for all unit vectors w orthogonal to u. We now consider three cases depending on the cardinality of Ω and the sign of λ.

- Case 1: $|\Omega| = 1$ and $\lambda > 0$. This means $u = v_i$ for some $i \in [k]$ (as $u = -v_i$ implies $\lambda = -\lambda_i < 0$). In this case,

  $$w^\top \nabla_u^2 \mathcal{L}(u, \lambda) w = 3\lambda_i (2(v_i^\top w)^2 - 1) = -3\lambda_i < 0$$

  for all $w \in \mathbb{R}^n$ satisfying $(u^\top w)^2 = (v_i^\top w)^2 = 0$. Hence u is an isolated local maximizer.

- Case 2: $|\Omega| \geq 2$ and $\lambda > 0$. Since $|\Omega| \geq 2$, we may pick a strict non-empty subset $S \subsetneq \Omega$ and set

  $$w := \frac{1}{Z} \left(\frac{1}{Z_S} \sum_{i \in S} c_i v_i - \frac{1}{Z_{S^c}} \sum_{i \in \Omega \setminus S} c_i v_i \right)$$

  where $Z_S := \sum_{i \in S} c_i^2$, $Z_{S^c} := \sum_{i \in \Omega \setminus S} c_i^2$, and $Z := \sqrt{1/Z_S + 1/Z_{S^c}}$. It is easy to check that $\|w\|^2 = \sum_{i \in \Omega} (v_i^\top w)^2 = 1$ and $u^\top w = 0$. Consider any open neighborhood U of u, and pick $\delta > 0$ small enough so that $\tilde{u} := \sqrt{1 - \delta^2} u + \delta w$ is contained in U. Set $u_0 := \sqrt{1 - \delta^2} u$. By Taylor's theorem, there exists $\epsilon \in [0, \delta]$ such that, for

$\bar{u} := u_0 + \epsilon w$, we have

$$\begin{aligned}
T(\tilde{u},\tilde{u},\tilde{u}) &= T(u_0,u_0,u_0) + \nabla_u T(u,u,u)^\top (\tilde{u}-u_0)\Big|_{u=u_0} \\
&\quad + \frac{1}{2}(\tilde{u}-u_0)^\top \nabla_u^2 T(u,u,u)(\tilde{u}-u_0)\Big|_{u=\bar{u}} \\
&= (1-\delta^2)^{3/2}\lambda + \delta(1-\delta^2)\lambda u^\top w + \frac{1}{2}\delta^2 w^\top \nabla_u^2 T(u,u,u) w\Big|_{u=\bar{u}} \\
&= (1-\delta^2)^{3/2}\lambda + 0 + 3\delta^2 \sum_{i=1}^k \lambda_i (v_i^\top (u_0+\epsilon w))(v_i^\top w)^2 \\
&= (1-\delta^2)^{3/2}\lambda + 3\delta^2\sqrt{1-\delta^2}\sum_{i=1}^k \lambda_i c_i (v_i^\top w)^2 + 3\delta^2\epsilon \sum_{i=1}^k \lambda_i (v_i^\top w)^3 \\
&= (1-\delta^2)^{3/2}\lambda + 3\delta^2\sqrt{1-\delta^2}\lambda\sum_{i\in\Omega} (v_i^\top w)^2 + 3\delta^2\epsilon \sum_{i=1}^k \lambda_i (v_i^\top w)^3 \\
&= (1-\delta^2)^{3/2}\lambda + 3\delta^2\sqrt{1-\delta^2}\lambda + 3\delta^2\epsilon \sum_{i=1}^k \lambda_i (v_i^\top w)^3 \\
&= \left(1 - \frac{3}{2}\delta^2 + O(\delta^4)\right)\lambda + 3\delta^2\sqrt{1-\delta^2}\lambda + 3\delta^2\epsilon\sum_{i=1}^k \lambda_i (v_i^\top w)^3.
\end{aligned}$$

Since $\epsilon \le \delta$, for small enough δ, the RHS is strictly greater than λ. This implies that u is not an isolated local maximizer.

- Case 3: $|\Omega| = 0$ or $\lambda = 0$. Note that if $|\Omega| = 0$, then $\lambda = 0$, so we just consider $\lambda = 0$. Consider any open neighborhood U of u, and pick $j \in [n]$ and $\delta > 0$ small enough so that $\tilde{u} := \sqrt{1-\delta^2}u + \delta v_j$ is contained in U. Then

$$T(\tilde{u},\tilde{u},\tilde{u}) = (1-\delta^2)^{3/2}T(u,u,u) + 3\lambda_j(1-\delta^2)\delta c_j^2 + 3\lambda_i\sqrt{1-\delta^2}\delta^2 c_j + \delta^3 > 0 = \lambda$$

for sufficiently small δ. Thus u is not an isolated local maximizer.

From these exhaustive cases, we conclude that a stationary point u is an isolated local maximizer if and only if $u = v_i$ for some $i \in [k]$. ■

We are grateful to Hanzhang Hu, Drew Bagnell, and Martial Hebert for alerting us of an issue with our original statement of Theorem 4.2 and its proof, and for suggesting a simple fix. The original statement used the optimization constraint $\|u\| = 1$ (rather than $\|u\| \le 1$), but the characterization of the decomposition with this constraint is then only given by isolated local maximizers u with the additional constraint that $T(u,u,u) > 0$—that is, there can be isolated local maximizers with $T(u,u,u) \le 0$ that are not vectors in the decomposition. The suggested fix of Hu, Bagnell, and Herbert is to relax to $\|u\| \le 1$, which eliminates isolated local maximizers with $T(u,u,u) \le 0$; this way, the characterization of the decomposition is simply the isolated local maximizers under the relaxed constraint.

Appendix B. Analysis of Robust Power Method

In this section, we prove Theorem 5.1. The proof is structured as follows. In Appendix B.1, we show that with high probability, at least one out of L random vectors will be a good initializer for the tensor power iterations. An initializer is good if its projection onto an eigenvector is noticeably larger than its projection onto other eigenvectors. We then analyze in Appendix B.2 the convergence behavior of the tensor power iterations. Relative to the proof of Lemma 5.1, this analysis is complicated by the tensor perturbation. We show that there is an initial slow convergence phase (linear rate rather than quadratic), but as soon as the projection of the iterate onto an eigenvector is large enough, it enters the quadratic convergence regime until the perturbation dominates. Finally, we show how errors accrue due to deflation in Appendix B.3, which is rather subtle and different from deflation with matrix eigendecompositions. This is because when some initial set of eigenvectors and eigenvalues are accurately recovered, the additional errors due to deflation are effectively only lower-order terms. These three pieces are assembled in Appendix B.4 to complete the proof of Theorem 5.1.

B.1 Initialization

Consider a set of non-negative numbers $\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_k \geq 0$. For $\gamma \in (0,1)$, we say a unit vector $\theta_0 \in \mathbb{R}^k$ is γ-*separated* relative to $i^* \in [k]$ if

$$\tilde{\lambda}_{i^*} |\theta_{i^*,0}| - \max_{i \in [k] \setminus \{i^*\}} \tilde{\lambda}_i |\theta_{i,0}| \geq \gamma \tilde{\lambda}_i |\theta_{i^*,0}|$$

(the dependence on $\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_k$ is implicit).

The following lemma shows that for any constant γ, with probability at least $1 - \eta$, at least one out of $\mathrm{poly}(k)\log(1/\eta)$ i.i.d. random vectors (uniformly distributed over the unit sphere S^{k-1}) is γ-separated relative to $\arg\max_{i \in [k]} \tilde{\lambda}_i$. (For small enough γ and large enough k, the polynomial is close to linear in k.)

Lemma B.1 *There exists an absolute constant $c > 0$ such that if positive integer $L \geq 2$ satisfies*

$$\sqrt{\frac{\ln(L)}{\ln(k)}} \cdot \left(1 - \frac{\ln(\ln(L)) + c}{4\ln(L)} - \sqrt{\frac{\ln(8)}{\ln(L)}}\right) \geq \frac{1}{1-\gamma} \cdot \left(1 + \sqrt{\frac{\ln(4)}{\ln(k)}}\right), \tag{10}$$

the following holds. With probability at least $1/2$ over the choice of L i.i.d. random vectors drawn uniformly distributed over the unit sphere S^{k-1} in $\mathbb{R}^k$, at least one of the vectors is γ-separated relative to $\arg\max_{i \in [k]} \tilde{\lambda}_i$. Moreover, with the same c, L, and for any $\eta \in (0,1)$, with probability at least $1 - \eta$ over $L \cdot \log_2(1/\eta)$ i.i.d. uniform random unit vectors, at least one of the vectors is γ-separated.

Proof Without loss of generality, assume $\arg\max_{i \in [k]} \tilde{\lambda}_i = 1$. Consider a random matrix $Z \in \mathbb{R}^{k \times L}$ whose entries are independent $\mathcal{N}(0,1)$ random variables; we take the j-th column of Z to be comprised of the random variables used for the j-th random vector (before normalization). Specifically, for the j-th random vector,

$$\theta_{i,0} := \frac{Z_{i,j}}{\sqrt{\sum_{i'=1}^k Z_{i',j}^2}}, \quad i \in [n].$$

It suffices to show that with probability at least $1/2$, there is a column $j^* \in [L]$ such that

$$|Z_{1,j^*}| \geq \frac{1}{1-\gamma} \max_{i \in [k] \setminus \{1\}} |Z_{i,j^*}|.$$

Since $\max_{j \in [L]} |Z_{1,j}|$ is a 1-Lipschitz function of L independent $\mathcal{N}(0,1)$ random variables, it follows that

$$\Pr\left[\left|\max_{j \in [L]} |Z_{1,j}| - \text{median}\left[\max_{j \in [L]} |Z_{1,j}|\right]\right| > \sqrt{2\ln(8)}\right] \leq 1/4.$$

Moreover,

$$\text{median}\left[\max_{j \in [L]} |Z_{1,j}|\right] \geq \text{median}\left[\max_{j \in [L]} Z_{1,j}\right] =: m.$$

Observe that the cumulative distribution function of $\max_{j \in [L]} Z_{1,j}$ is given by $F(z) = \Phi(z)^L$, where Φ is the standard Gaussian CDF. Since $F(m) = 1/2$, it follows that $m = \Phi^{-1}(2^{-1/L})$. It can be checked that

$$\Phi^{-1}(2^{-1/L}) \geq \sqrt{2\ln(L)} - \frac{\ln(\ln(L)) + c}{2\sqrt{2\ln(L)}}$$

for some absolute constant $c > 0$. Also, let $j^* := \arg\max_{j \in [L]} |Z_{1,j}|$.

Now for each $j \in [L]$, let $|Z_{2:k,j}| := \max\{|Z_{2,j}|, |Z_{3,j}|, \ldots, |Z_{k,j}|\}$. Again, since $|Z_{2:k,j}|$ is a 1-Lipschitz function of $k-1$ independent $\mathcal{N}(0,1)$ random variables, it follows that

$$\Pr\left[|Z_{2:k,j}| > \mathbb{E}\left[|Z_{2:k,j}|\right] + \sqrt{2\ln(4)}\right] \leq 1/4.$$

Moreover, by a standard argument,

$$\mathbb{E}\left[|Z_{2:k,j}|\right] \leq \sqrt{2\ln(k)}.$$

Since $|Z_{2:k,j}|$ is independent of $|Z_{1,j}|$ for all $j \in [L]$, it follows that the previous two displayed inequalities also hold with j replaced by j^*.

Therefore we conclude with a union bound that with probability at least $1/2$,

$$|Z_{1,j^*}| \geq \sqrt{2\ln(L)} - \frac{\ln(\ln(L)) + c}{2\sqrt{2\ln(L)}} - \sqrt{2\ln(8)} \quad \text{and} \quad |Z_{2:k,j^*}| \leq \sqrt{2\ln(k)} + \sqrt{2\ln(4)}.$$

Since L satisfies (10) by assumption, in this event, the j^*-th random vector is γ-separated. ■

B.2 Tensor Power Iterations

Recall the update rule used in the power method. Let $\theta_t = \sum_{i=1}^k \theta_{i,t} v_i \in \mathbb{R}^k$ be the unit vector at time t. Then

$$\theta_{t+1} = \sum_{i=1}^k \theta_{i,t+1} v_i := \tilde{T}(I, \theta_t, \theta_t) / \|\tilde{T}(I, \theta_t, \theta_t)\|.$$

In this subsection, we assume that $\tilde{T}$ has the form

$$\tilde{T} = \sum_{i=1}^{k} \tilde{\lambda}_i v_i^{\otimes 3} + \tilde{E} \tag{11}$$

where $\{v_1, v_2, \ldots, v_k\}$ is an orthonormal basis, and, without loss of generality,

$$\tilde{\lambda}_1 |\theta_{1,t}| = \max_{i \in [k]} \tilde{\lambda}_i |\theta_{i,t}| > 0.$$

Also, define

$$\tilde{\lambda}_{\min} := \min\{\tilde{\lambda}_i : i \in [k],\ \tilde{\lambda}_i > 0\}, \quad \tilde{\lambda}_{\max} := \max\{\tilde{\lambda}_i : i \in [k]\}.$$

We further assume the error $\tilde{E}$ is a symmetric tensor such that, for some constant $p > 1$,

$$\|\tilde{E}(I, u, u)\| \le \tilde{\epsilon}, \quad \forall u \in S^{k-1}; \tag{12}$$

$$\|\tilde{E}(I, u, u)\| \le \tilde{\epsilon}/p, \quad \forall u \in S^{k-1} \text{ s.t. } (u^\top v_1)^2 \ge 1 - (3\tilde{\epsilon}/\tilde{\lambda}_1)^2. \tag{13}$$

In the next two propositions (Propositions B.1 and B.2) and next two lemmas (Lemmas B.2 and B.3), we analyze the power method iterations using $\tilde{T}$ at some arbitrary iterate θ_t using only the property (12) of $\tilde{E}$. But throughout, the quantity $\tilde{\epsilon}$ can be replaced by $\tilde{\epsilon}/p$ if θ_t satisfies $(\theta_t^\top v_1)^2 \ge 1 - (3\tilde{\epsilon}/\tilde{\lambda}_1)^2$ as per property (13).

Define

$$\begin{aligned} R_\tau &:= \left(\frac{\theta_{1,\tau}^2}{1 - \theta_{1,\tau}^2}\right)^{1/2}, \qquad r_{i,\tau} := \frac{\tilde{\lambda}_1 \theta_{1,\tau}}{\tilde{\lambda}_i |\theta_{i,\tau}|}, \\ \gamma_\tau &:= 1 - \frac{1}{\min_{i \ne 1} |r_{i,\tau}|}, \quad \delta_\tau := \frac{\tilde{\epsilon}}{\tilde{\lambda}_1 \theta_{1,\tau}^2}, \quad \kappa := \frac{\tilde{\lambda}_{\max}}{\tilde{\lambda}_1} \end{aligned} \tag{14}$$

for $\tau \in \{t, t+1\}$.

Proposition B.1

$$\min_{i \ne 1} |r_{i,t}| \ge \frac{R_t}{\kappa}, \qquad \gamma_t \ge 1 - \frac{\kappa}{R_t}, \qquad \theta_{1,t}^2 = \frac{R_t^2}{1 + R_t^2}.$$

Proposition B.2

$$r_{i,t+1} \ge r_{i,t}^2 \cdot \frac{1 - \delta_t}{1 + \kappa \delta_t r_{i,t}^2} = \frac{1 - \delta_t}{\frac{1}{r_{i,t}^2} + \kappa \delta_t}, \quad i \in [k], \tag{15}$$

$$R_{t+1} \ge R_t \cdot \frac{1 - \delta_t}{1 - \gamma_t + \delta_t R_t} \ge \frac{1 - \delta_t}{\frac{\kappa}{R_t^2} + \delta_t}. \tag{16}$$

Proof Let $\check{\theta}_{t+1} := \tilde{T}(I, \theta_t, \theta_t)$, so $\theta_{t+1} = \check{\theta}_{t+1}/\|\check{\theta}_{t+1}\|$. Since $\check{\theta}_{i,t+1} = \tilde{T}(v_i, \theta_t, \theta_t) = T(v_i, \theta_t, \theta_t) + E(v_i, \theta_t, \theta_t)$, we have

$$\check{\theta}_{i,t+1} = \tilde{\lambda}_i \theta_{i,t}^2 + E(v_i, \theta_t, \theta_t), \quad i \in [k].$$

Using the triangle inequality and the fact $\|E(v_i, \theta_t, \theta_t)\| \le \tilde{\epsilon}$, we have

$$\check{\theta}_{i,t+1} \ge \tilde{\lambda}_i \theta_{i,t}^2 - \tilde{\epsilon} \ge |\theta_{i,t}| \cdot \Big(\tilde{\lambda}_i |\theta_{i,t}| - \tilde{\epsilon}/|\theta_{i,t}| \Big) \tag{17}$$

and

$$|\check{\theta}_{i,t+1}| \le |\tilde{\lambda}_i \theta_{i,t}^2| + \tilde{\epsilon} \le |\theta_{i,t}| \cdot \Big(\tilde{\lambda}_i |\theta_{i,t}| + \tilde{\epsilon}/|\theta_{i,t}| \Big) \tag{18}$$

for all $i \in [k]$. Combining (17) and (18) gives

$$r_{i,t+1} = \frac{\tilde{\lambda}_1 \theta_{1,t+1}}{\tilde{\lambda}_i |\theta_{i,t+1}|} = \frac{\tilde{\lambda}_1 \check{\theta}_{1,t+1}}{\tilde{\lambda}_i |\check{\theta}_{i,t+1}|} \ge r_{i,t}^2 \cdot \frac{1 - \delta_t}{1 + \frac{\tilde{\epsilon}}{\tilde{\lambda}_i \theta_{i,t}^2}} = r_{i,t}^2 \cdot \frac{1 - \delta_t}{1 + (\tilde{\lambda}_i / \tilde{\lambda}_1) \delta_t r_{i,t}^2} \ge r_{i,t}^2 \cdot \frac{1 - \delta_t}{1 + \kappa \delta_t r_{i,t}^2}.$$

Moreover, by the triangle inequality and Hölder's inequality,

$$\begin{aligned}
\Big(\sum_{i=2}^n [\check{\theta}_{i,t+1}]^2 \Big)^{1/2} &= \Big(\sum_{i=2}^n \Big(\tilde{\lambda}_i \theta_{i,t}^2 + E(v_i, \theta_t, \theta_t) \Big)^2 \Big)^{1/2} \\
&\le \Big(\sum_{i=2}^n \tilde{\lambda}_i^2 \theta_{i,t}^4 \Big)^{1/2} + \Big(\sum_{i=2}^n E(v_i, \theta_t, \theta_t)^2 \Big)^{1/2} \\
&\le \max_{i \ne 1} \tilde{\lambda}_i |\theta_{i,t}| \Big(\sum_{i=2}^n \theta_{i,t}^2 \Big)^{1/2} + \tilde{\epsilon} \\
&= (1 - \theta_{1,t}^2)^{1/2} \cdot \Big(\max_{i \ne 1} \tilde{\lambda}_i |\theta_{i,t}| + \tilde{\epsilon}/(1 - \theta_{1,t}^2)^{1/2} \Big).
\end{aligned} \tag{19}$$

Combining (17) and (19) gives

$$\frac{|\theta_{1,t+1}|}{(1 - \theta_{1,t+1}^2)^{1/2}} = \frac{|\check{\theta}_{1,t+1}|}{\Big(\sum_{i=2}^n [\check{\theta}_{i,t+1}]^2 \Big)^{1/2}} \ge \frac{|\theta_{1,t}|}{(1 - \theta_{1,t}^2)^{1/2}} \cdot \frac{\tilde{\lambda}_1 |\theta_{1,t}| - \tilde{\epsilon}/|\theta_{1,t}|}{\max_{i \ne 1} \tilde{\lambda}_i |\theta_{i,t}| + \tilde{\epsilon}/(1 - \theta_{1,t}^2)^{1/2}}.$$

In terms of R_{t+1}, R_t, γ_t, and δ_t, this reads

$$R_{t+1} \ge \frac{1 - \delta_t}{(1 - \gamma_t) \Big(\frac{1 - \theta_{1,t}^2}{\theta_{1,t}^2} \Big)^{1/2} + \delta_t} = R_t \cdot \frac{1 - \delta_t}{1 - \gamma_t + \delta_t R_t} = \frac{1 - \delta_t}{\frac{1 - \gamma_t}{R_t} + \delta_t} \ge \frac{1 - \delta_t}{\frac{\kappa}{R_t^2} + \delta_t}$$

where the last inequality follows from Proposition B.1. ■

Lemma B.2 *Fix any $\rho > 1$. Assume*

$$0 \le \delta_t < \min\Big\{ \frac{1}{2(1 + 2\kappa\rho^2)}, \ \frac{1 - 1/\rho}{1 + \kappa\rho} \Big\}$$

and $\gamma_t > 2(1 + 2\kappa\rho^2)\delta_t$.

1. *If $r_{i,t}^2 \le 2\rho^2$, then $r_{i,t+1} \ge |r_{i,t}| \big(1 + \frac{\gamma_t}{2} \big)$.*

2. *If* $\rho^2 < r_{i,t}^2$, *then* $r_{i,t+1} \geq \min\{r_{i,t}^2/\rho,\ \frac{1-\delta_t-1/\rho}{\kappa\delta_t}\}$.

3. $\gamma_{t+1} \geq \min\{\gamma_t, 1-1/\rho\}$.

4. *If* $\min_{i\neq 1} r_{i,t}^2 > (\rho(1-\delta_t)-1)/(\kappa\delta_t)$, *then* $R_{t+1} > \frac{1-\delta_t-1/\rho}{\kappa\delta_t} \cdot \frac{\tilde{\lambda}_{\min}}{\tilde{\lambda}_1} \cdot \frac{1}{\sqrt{k}}$.

5. *If* $R_t \leq 1+2\kappa\rho^2$, *then* $R_{t+1} \geq R_t\big(1+\frac{\gamma_t}{3}\big)$, $\theta_{1,t+1}^2 \geq \theta_{1,t}^2$, *and* $\delta_{t+1} \leq \delta_t$.

Proof Consider two (overlapping) cases depending on $r_{i,t}^2$.

- Case 1: $r_{i,t}^2 \leq 2\rho^2$. By (15) from Proposition B.2,

$$r_{i,t+1} \geq r_{i,t}^2 \cdot \frac{1-\delta_t}{1+\kappa\delta_t r_{i,t}^2} \geq |r_{i,t}| \cdot \frac{1}{1-\gamma_t} \cdot \frac{1-\delta_t}{1+2\kappa\rho^2\delta_t} \geq |r_{i,t}|\Big(1+\frac{\gamma_t}{2}\Big)$$

  where the last inequality uses the assumption $\gamma_t > 2(1+2\kappa\rho^2)\delta_t$. This proves the first claim.

- Case 2: $\rho^2 < r_{i,t}^2$. We split into two sub-cases. Suppose $r_{i,t}^2 \leq (\rho(1-\delta_t)-1)/(\kappa\delta_t)$. Then, by (15),

$$r_{i,t+1} \geq r_{i,t}^2 \cdot \frac{1-\delta_t}{1+\kappa\delta_t r_{i,t}^2} \geq r_{i,t}^2 \cdot \frac{1-\delta_t}{1+\kappa\delta_t \frac{\rho(1-\delta_t)-1}{\kappa\delta_t}} = \frac{r_{i,t}^2}{\rho}.$$

  Now suppose instead $r_{i,t}^2 > (\rho(1-\delta_t)-1)/(\kappa\delta_t)$. Then

$$r_{i,t+1} \geq \frac{1-\delta_t}{\frac{\kappa\delta_t}{\rho(1-\delta_t)-1}+\kappa\delta_t} = \frac{1-\delta_t-1/\rho}{\kappa\delta_t}. \tag{20}$$

Observe that if $\min_{i\neq 1} r_{i,t}^2 \leq (\rho(1-\delta_t)-1)/(\kappa\delta_t)$, then $r_{i,t+1} \geq |r_{i,t}|$ for all $i \in [k]$, and hence $\gamma_{t+1} \geq \gamma_t$. Otherwise we have $\gamma_{t+1} > 1 - \frac{\kappa\delta_t}{1-\delta_t-1/\rho} > 1-1/\rho$. This proves the third claim.

If $\min_{i\neq 1} r_{i,t}^2 > (\rho(1-\delta_t)-1)/(\kappa\delta_t)$, then we may apply the inequality (20) from the second sub-case of Case 2 above to get

$$R_{t+1} = \frac{1}{\Big(\sum_{i\neq 1}(\tilde{\lambda}_1/\tilde{\lambda}_i)^2/r_{i,t+1}^2\Big)^{1/2}} > \left(\frac{1-\delta_t-1/\rho}{\kappa\delta_t}\right) \cdot \frac{\tilde{\lambda}_{\min}}{\tilde{\lambda}_1} \cdot \frac{1}{\sqrt{k}}.$$

This proves the fourth claim.

Finally, for the last claim, if $R_t \leq 1+2\kappa\rho^2$, then by (16) from Proposition B.2 and the assumption $\gamma_t > 2(1+2\kappa\rho^2)\delta_t$,

$$R_{t+1} \geq R_t \cdot \frac{1-\delta_t}{1-\gamma_t+\delta_t R_t} \geq R_t \cdot \frac{1-\frac{\gamma_t}{2(1+2\kappa\rho^2)}}{1-\gamma_t/2} \geq R_t\left(1+\gamma_t \cdot \frac{\kappa\rho^2}{1+2\kappa\rho^2}\right) \geq R_t\Big(1+\frac{\gamma_t}{3}\Big).$$

This in turn implies that $\theta_{1,t+1}^2 \geq \theta_{1,t}^2$ via Proposition B.1, and thus $\delta_{t+1} \leq \delta_t$. ■

Lemma B.3 *Assume $0 \le \delta_t < 1/2$ and $\gamma_t > 0$. Pick any $\beta > \alpha > 0$ such that*

$$\frac{\alpha}{(1+\alpha)(1+\alpha^2)} \ge \frac{\tilde{\epsilon}}{\gamma_t \tilde{\lambda}_1}, \quad \frac{\alpha}{2(1+\alpha)(1+\beta^2)} \ge \frac{\tilde{\epsilon}}{\tilde{\lambda}_1}.$$

1. *If $R_t \ge 1/\alpha$, then $R_{t+1} \ge 1/\alpha$.*
2. *If $1/\alpha > R_t \ge 1/\beta$, then $R_{t+1} \ge \min\{R_t^2/(2\kappa),\ 1/\alpha\}$.*

Proof Observe that for any $c > 0$,

$$R_t \ge \frac{1}{c} \quad \Leftrightarrow \quad \theta_{1,t}^2 \ge \frac{1}{1+c^2} \quad \Leftrightarrow \quad \delta_t \le \frac{(1+c^2)\tilde{\epsilon}}{\tilde{\lambda}_1}. \tag{21}$$

Now consider the following cases depending on R_t.

- Case 1: $R_t \ge 1/\alpha$. In this case, we have

  $$\delta_t \le \frac{(1+\alpha^2)\tilde{\epsilon}}{\tilde{\lambda}_1} \le \frac{\alpha\gamma_t}{1+\alpha}$$

  by (21) (with $c = \alpha$) and the condition on α. Combining this with (16) from Proposition B.2 gives

  $$R_{t+1} \ge \frac{1-\delta_t}{\frac{1-\gamma_t}{R_t} + \delta_t} \ge \frac{1 - \frac{\alpha\gamma_t}{1+\alpha}}{(1-\gamma_t)\alpha + \frac{\alpha\gamma_t}{1+\alpha}} = \frac{1}{\alpha}.$$

- Case 2: $1/\beta \le R_t < 1/\alpha$. In this case, we have

  $$\delta_t \le \frac{(1+\beta^2)\tilde{\epsilon}}{\tilde{\lambda}_1} \le \frac{\alpha}{2(1+\alpha)}$$

  by (21) (with $c = \beta$) and the conditions on α and β. If $\delta_t \ge 1/(2 + R_t^2/\kappa)$, then (16) implies

  $$R_{t+1} \ge \frac{1-\delta_t}{\frac{\kappa}{R_t^2} + \delta_t} \ge \frac{1-2\delta_t}{2\delta_t} \ge \frac{1 - \frac{\alpha}{1+\alpha}}{\frac{\alpha}{1+\alpha}} = \frac{1}{\alpha}.$$

  If instead $\delta_t < 1/(2 + R_t^2/\kappa)$, then (16) implies

  $$R_{t+1} \ge \frac{1-\delta_t}{\frac{\kappa}{R_t^2} + \delta_t} > \frac{1 - \frac{1}{2+R_t^2/\kappa}}{\frac{\kappa}{R_t^2} + \frac{1}{2+R_t^2/\kappa}} = \frac{R_t^2}{2\kappa}.$$

■

B.2.1 Approximate Recovery of a Single Eigenvector

We now state the main result regarding the approximate recovery of a single eigenvector using the tensor power method on $\tilde{T}$. Here, we exploit the special properties of the error $\tilde{E}$—both (12) and (13).

Lemma B.4 *There exists a universal constant $C > 0$ such that the following holds. Let $i^* := \arg\max_{i\in[k]} \tilde{\lambda}_i|\theta_{i,0}|$. If*

$$\tilde{\epsilon} < \frac{\gamma_0}{2(1+8\kappa)} \cdot \tilde{\lambda}_{\min} \cdot \theta_{i^*,0}^2 \quad \text{and} \quad N \geq C \cdot \left(\frac{\log(k\kappa)}{\gamma_0} + \log\log \frac{p\tilde{\lambda}_{i^*}}{\tilde{\epsilon}}\right),$$

then after $t \geq N$ iterations of the tensor power method on tensor $\tilde{T}$ as defined in (11) and satisfying (12) and (13), the final vector θ_t satisfies

$$\theta_{i^*,t} \geq \sqrt{1 - \left(\frac{3\tilde{\epsilon}}{p\tilde{\lambda}_{i^*}}\right)^2}, \quad \|\theta_t - v_{i^*}\| \leq \frac{4\tilde{\epsilon}}{p\tilde{\lambda}_{i^*}}, \quad |\tilde{T}(\theta_t,\theta_t,\theta_t) - \tilde{\lambda}_{i^*}| \leq \left(27\kappa\Big(\frac{\tilde{\epsilon}}{p\lambda_{i^*}}\Big)^2 + 2\right)\frac{\tilde{\epsilon}}{p}.$$

Proof Assume without loss of generality that $i^* = 1$. We consider three phases: (i) iterations before the first time t such that $R_t > 1 + 2\kappa\rho^2 = 1 + 8\kappa$ (using $\rho := 2$), (ii) the subsequent iterations before the first time t such that $R_t \geq 1/\alpha$ (where α will be defined below), and finally (iii) the remaining iterations.

We begin by analyzing the first phase, *i.e.*, the iterates in $T_1 := \{t \geq 0 : R_t \leq 1+2\kappa\rho^2 = 1 + 8\kappa\}$. Observe that the condition on $\tilde{\epsilon}$ implies

$$\delta_0 = \frac{\tilde{\epsilon}}{\tilde{\lambda}_1\theta_{1,0}^2} < \frac{\gamma_0}{2(1+8\kappa)} \cdot \frac{\tilde{\lambda}_{\min}}{\tilde{\lambda}_1} \leq \min\left\{\frac{\gamma_0}{2(1+2\kappa\rho^2)}, \frac{1-1/\rho}{2(1+2\kappa\rho^2)}\right\},$$

and hence the preconditions on δ_t and γ_t of Lemma B.2 hold for $t = 0$. For all $t \in T_1$ satisfying the preconditions, Lemma B.2 implies that $\delta_{t+1} \leq \delta_t$ and $\gamma_{t+1} \geq \min\{\gamma_t, 1-1/\rho\}$, so the next iteration also satisfies the preconditions. Hence by induction, the preconditions hold for all iterations in T_1. Moreover, for all $i \in [k]$, we have

$$|r_{i,0}| \geq \frac{1}{1-\gamma_0};$$

and while $t \in T_1$: (i) $|r_{i,t}|$ increases at a linear rate while $r_{i,t}^2 \leq 2\rho^2$, and (ii) $|r_{i,t}|$ increases at a quadratic rate while $\rho^2 \leq r_{i,t}^2 \leq \frac{1-\delta_t-1/\rho}{\kappa\delta_t}$. (The specific rates are given, respectively, in Lemma B.2, claims 1 and 2.) Since $\frac{1-\delta_t-1/\rho}{\kappa\delta_t} \leq \frac{\tilde{\lambda}_1}{2\kappa\tilde{\epsilon}}$, it follows that $\min_{i\neq 1} r_{i,t}^2 \leq \frac{1-\delta_t-1/\rho}{\kappa\delta_t}$ for at most

$$\frac{2}{\gamma_0}\ln\left(\frac{\sqrt{2\rho^2}}{\frac{1}{1-\gamma_0}}\right) + \ln\left(\frac{\ln\frac{\tilde{\lambda}_1}{2\kappa\tilde{\epsilon}}}{\ln\sqrt{2}}\right) = O\left(\frac{1}{\gamma_0} + \log\log\frac{\tilde{\lambda}_1}{\tilde{\epsilon}}\right) \tag{22}$$

iterations in T_1. As soon as $\min_{i\neq 1} r_{i,t}^2 > \frac{1-\delta_t-1/\rho}{\kappa\delta_t}$, we have that in the next iteration,

$$R_{t+1} > \frac{1-\delta_t-1/\rho}{\kappa\delta_t} \cdot \frac{\tilde{\lambda}_{\min}}{\tilde{\lambda}_1} \cdot \frac{1}{\sqrt{k}} \geq \frac{7}{\sqrt{k}};$$

and all the while R_t is growing at a linear rate (given in Lemma B.2, claim 5). Therefore, there are at most an additional

$$1 + \frac{3}{\gamma_0} \ln\left(\frac{1+8\kappa}{7/\sqrt{k}}\right) = O\left(\frac{\log(k\kappa)}{\gamma_0}\right) \tag{23}$$

iterations in T_1 over that counted in (22). Therefore, by combining the counts in (22) and (23), we have that the number of iterations in the first phase satisfies

$$|T_1| = O\left(\log\log\frac{\tilde{\lambda}_1}{\tilde{\epsilon}} + \frac{\log(k\kappa)}{\gamma_0}\right).$$

We now analyze the second phase, *i.e.*, the iterates in $T_2 := \{t \geq 0 : t \notin T_1,\ R_t < 1/\alpha\}$. Define

$$\alpha := \frac{3\tilde{\epsilon}}{\tilde{\lambda}_1}, \quad \beta := \frac{1}{1+2\kappa\rho^2} = \frac{1}{1+8\kappa}.$$

Note that for the initial iteration $t' := \min T_2$, we have that $R_{t'} \geq 1 + 2\kappa\rho^2 = 1 + 8\kappa = 1/\beta$, and by Proposition B.1, $\gamma_{t'} \geq 1 - \kappa/(1+8\kappa) > 7/8$. It can be checked that δ_t, γ_t, α, and β satisfy the preconditions of Lemma B.3 for this initial iteration t'. For all $t \in T_2$ satisfying these preconditions, Lemma B.3 implies that $R_{t+1} \geq \min\{R_t, 1/\alpha\}$, $\theta^2_{1,t+1} \geq \min\{\theta^2_{1,t}, 1/(1+\alpha^2)\}$ (via Proposition B.1), $\delta_{t+1} \leq \max\{\delta_t, (1+\alpha)^2\tilde{\epsilon}/\tilde{\lambda}_1\}$ (using the definition of δ_t), and $\gamma_{t+1} \geq \min\{\gamma_t, 1-\alpha\kappa\}$ (via Proposition B.1). Hence the next iteration $t+1$ also satisfies the preconditions, and by induction, so do all iterations in T_2. To bound the number of iterations in T_2, observe that R_t increases at a quadratic rate until $R_t \geq 1/\alpha$, so

$$|T_2| \leq \ln\left(\frac{\ln(1/\alpha)}{\ln((1/\beta)/(2\kappa))}\right) < \ln\left(\frac{\ln\frac{\tilde{\lambda}_1}{3\tilde{\epsilon}}}{\ln 4}\right) = O\left(\log\log\frac{\tilde{\lambda}_1}{\tilde{\epsilon}}\right). \tag{24}$$

Therefore the total number of iterations before $R_t \geq 1/\alpha$ is

$$O\left(\frac{\log(k\kappa)}{\gamma_0} + \log\log\frac{\tilde{\lambda}_1}{\tilde{\epsilon}}\right).$$

After $R_{t''} \geq 1/\alpha$ (for $t'' := \max(T_1 \cup T_2) + 1$), we have

$$\theta^2_{1,t''} \geq \frac{1/\alpha^2}{1+1/\alpha^2} \geq 1 - \alpha^2 \geq 1 - \left(\frac{3\tilde{\epsilon}}{\tilde{\lambda}_1}\right)^2.$$

Therefore, the vector $\theta_{t''}$ satisfies the condition for property (13) of $\tilde{E}$ to hold. Now we apply Lemma B.3 using $\tilde{\epsilon}/p$ in place of $\tilde{\epsilon}$, including in the definition of δ_t (which we call $\overline{\delta}_t$):

$$\overline{\delta}_t := \frac{\tilde{\epsilon}}{p\tilde{\lambda}_1\theta^2_{1,t}};$$

we also replace α and β with $\overline{\alpha}$ and $\overline{\beta}$, which we set to

$$\overline{\alpha} := \frac{3\tilde{\epsilon}}{p\tilde{\lambda}_1}, \quad \overline{\beta} := \frac{3\tilde{\epsilon}}{\tilde{\lambda}_1}.$$

It can be checked that $\overline{\delta}_{t''} \in (0, 1/2)$, $\gamma_{t''} \geq 1 - 3\tilde{\epsilon}\kappa/\lambda_1 > 0$,

$$\frac{\overline{\alpha}}{(1+\overline{\alpha})(1+\overline{\alpha}^2)} \geq \frac{\tilde{\epsilon}}{p(1-3\tilde{\epsilon}\kappa/\lambda_1)\tilde{\lambda}_1} \geq \frac{\tilde{\epsilon}}{p\gamma_{t''}\tilde{\lambda}_1}, \quad \frac{\overline{\alpha}}{2(1+\overline{\alpha})(1+\overline{\beta}^2)} \geq \frac{\tilde{\epsilon}}{p\tilde{\lambda}_1}.$$

Therefore, the preconditions of Lemma B.3 are satisfied for the initial iteration t'' in this final phase, and by the same arguments as before, the preconditions hold for all subsequent iterations $t \geq t''$. Initially, we have $R_{t''} \geq 1/\alpha \geq 1/\overline{\beta}$, and by Lemma B.3, we have that R_t increases at a quadratic rate in this final phase until $R_t \geq 1/\overline{\alpha}$. So the number of iterations before $R_t \geq 1/\overline{\alpha}$ can be bounded as

$$\ln\left(\frac{\ln(1/\overline{\alpha})}{\ln((1/\overline{\beta})/(2\kappa))}\right) = \ln\left(\frac{\ln \frac{p\tilde{\lambda}_1}{3\tilde{\epsilon}}}{\ln\left(\frac{\lambda_1}{3\tilde{\epsilon}} \cdot \frac{1}{2\kappa}\right)}\right) \leq \ln\ln\frac{p\tilde{\lambda}_1}{3\tilde{\epsilon}} = O\left(\log\log\frac{p\tilde{\lambda}_1}{\tilde{\epsilon}}\right).$$

Once $R_t \geq 1/\overline{\alpha}$, we have

$$\theta_{1,t}^2 \geq 1 - \left(\frac{3\tilde{\epsilon}}{p\tilde{\lambda}_1}\right)^2.$$

Since $\mathrm{sign}(\theta_{1,t}) = r_{1,t} \geq r_{1,t-1}^2 \cdot (1-\overline{\delta}_{t-1})/(1+\kappa\overline{\delta}_{t-1}r_{1,t-1}^2) = (1-\overline{\delta}_{t-1})/(1+\kappa\overline{\delta}_{t-1}) > 0$ by Proposition B.2, we have $\theta_{1,t} > 0$. Therefore we can conclude that

$$\|\theta_t - v_1\| = \sqrt{2(1-\theta_{1,t})} \leq \sqrt{2\left(1 - \sqrt{1-(3\tilde{\epsilon}/(p\tilde{\lambda}_1))^2}\right)} \leq 4\tilde{\epsilon}/(p\tilde{\lambda}_1).$$

Finally,

$$\begin{aligned}
|\tilde{T}(\theta_t,\theta_t,\theta_t) - \tilde{\lambda}_1| &= \left|\tilde{\lambda}_1(\theta_{1,t}^3 - 1) + \sum_{i=2}^k \tilde{\lambda}_i\theta_{i,t}^3 + \tilde{E}(\theta_t,\theta_t,\theta_t)\right| \\
&\leq \tilde{\lambda}_1|\theta_{1,t}^3 - 1| + \sum_{i=2}^k \tilde{\lambda}_i|\theta_{i,t}|\theta_{i,t}^2 + \|\tilde{E}(I,\theta_t,\theta_t)\| \\
&\leq \tilde{\lambda}_1\big(1 - \theta_{1,t} + |\theta_{1,t}(1-\theta_{1,t}^2)|\big) + \max_{i\neq 1}\tilde{\lambda}_i|\theta_{i,t}|\sum_{i=2}^k \theta_{i,t}^2 + \|\tilde{E}(I,\theta_t,\theta_t)\| \\
&\leq \tilde{\lambda}_1\big(1 - \theta_{1,t} + |\theta_{1,t}(1-\theta_{1,t}^2)|\big) + \max_{i\neq 1}\tilde{\lambda}_i\sqrt{1-\theta_{1,t}^2}\sum_{i=2}^k \theta_{i,t}^2 + \|\tilde{E}(I,\theta_t,\theta_t)\| \\
&= \tilde{\lambda}_1\big(1 - \theta_{1,t} + |\theta_{1,t}(1-\theta_{1,t}^2)|\big) + \max_{i\neq 1}\tilde{\lambda}_i(1-\theta_{1,t}^2)^{3/2} + \|\tilde{E}(I,\theta_t,\theta_t)\| \\
&\leq \tilde{\lambda}_1 \cdot 3\left(\frac{3\tilde{\epsilon}}{p\tilde{\lambda}_1}\right)^2 + \kappa\tilde{\lambda}_1 \cdot \left(\frac{3\tilde{\epsilon}}{p\tilde{\lambda}_1}\right)^3 + \frac{\tilde{\epsilon}}{p} \\
&\leq \frac{(27\kappa\cdot(\tilde{\epsilon}/p\tilde{\lambda}_1)^2 + 2)\tilde{\epsilon}}{p}.
\end{aligned}$$

■

B.3 Deflation

Lemma B.5 *Fix some $\tilde{\epsilon} \geq 0$. Let $\{v_1, v_2, \ldots, v_k\}$ be an orthonormal basis for $\mathbb{R}^k$, and $\lambda_1, \lambda_2, \ldots, \lambda_k \geq 0$ with $\lambda_{\min} := \min_{i \in [k]} \lambda_i$. Also, let $\{\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_k\}$ be a set of unit vectors in $\mathbb{R}^k$ (not necessarily orthogonal), $\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_k \geq 0$ be non-negative scalars, and define*

$$\mathcal{E}_i := \lambda_i v_i^{\otimes 3} - \hat{\lambda}_i \hat{v}_i^{\otimes 3}, \quad i \in [k].$$

Pick any $t \in [k]$. If

$$\begin{aligned} |\hat{\lambda}_i - \lambda_i| &\leq \tilde{\epsilon}, \\ \|\hat{v}_i - v_i\| &\leq \min\{\sqrt{2},\ 2\tilde{\epsilon}/\lambda_i\} \end{aligned}$$

for all $i \in [t]$, then for any unit vector $u \in S^{k-1}$,

$$\begin{aligned} \left\| \sum_{i=1}^{t} \mathcal{E}_i(I, u, u) \right\|_2^2 \leq \Bigg(& 4(5 + 11\tilde{\epsilon}/\lambda_{\min})^2 + 128(1 + \tilde{\epsilon}/\lambda_{\min})^2 (\tilde{\epsilon}/\lambda_{\min})^2 \Bigg) \tilde{\epsilon}^2 \sum_{i=1}^{t} (u^\top v_i)^2 \\ & + 64(1 + \tilde{\epsilon}/\lambda_{\min})^2 \tilde{\epsilon}^2 \sum_{i=1}^{t} (\tilde{\epsilon}/\lambda_i)^2 + 2048(1 + \tilde{\epsilon}/\lambda_{\min})^2 \tilde{\epsilon}^2 \left(\sum_{i=1}^{t} (\tilde{\epsilon}/\lambda_i)^3 \right)^2. \end{aligned}$$

In particular, for any $\Delta \in (0, 1)$, there exists a constant $\Delta' > 0$ (depending only on Δ) such that $\tilde{\epsilon} \leq \Delta' \lambda_{\min}/\sqrt{k}$ implies

$$\left\| \sum_{i=1}^{t} \mathcal{E}_i(I, u, u) \right\|_2^2 \leq \left(\Delta + 100 \sum_{i=1}^{t} (u^\top v_i)^2 \right) \tilde{\epsilon}^2.$$

Proof For any unit vector u and $i \in [t]$, the error term

$$\mathcal{E}_i(I, u, u) = \lambda_i (u^\top v_i)^2 v_i - \hat{\lambda}_i (u^\top \hat{v}_i)^2 \hat{v}_i$$

lives in $\mathrm{span}\{v_i, \hat{v}_i\}$; this space is the same as $\mathrm{span}\{v_i, \hat{v}_i^\perp\}$, where

$$\hat{v}_i^\perp := \hat{v}_i - (v_i^\top \hat{v}_i) v_i$$

is the projection of $\hat{v}_i$ onto the subspace orthogonal to v_i. Since $\|\hat{v}_i - v_i\|^2 = 2(1 - v_i^\top \hat{v}_i)$, it follows that

$$c_i := v_i^\top \hat{v}_i = 1 - \|\hat{v}_i - v_i\|^2/2 \geq 0$$

(the inequality follows from the assumption $\|\hat{v}_i - v_i\| \leq \sqrt{2}$, which in turn implies $0 \leq c_i \leq 1$). By the Pythagorean theorem and the above inequality for c_i,

$$\|\hat{v}_i^\perp\|^2 = 1 - c_i^2 \leq \|\hat{v}_i - v_i\|^2.$$

Later, we will also need the following bound, which is easily derived from the above inequalities and the triangle inequality:

$$|1 - c_i^3| = |1 - c_i + c_i(1 - c_i^2)| \leq 1 - c_i + |c_i(1 - c_i^2)| \leq 1.5\|\hat{v}_i - v_i\|^2.$$

We now express $\mathcal{E}_i(I,u,u)$ in terms of the coordinate system defined by v_i and $\hat{v}_i^\perp$, depicted below. Define

$$a_i := u^\top v_i \quad \text{and} \quad b_i := u^\top\big(\hat{v}_i^\perp/\|\hat{v}_i^\perp\|\big).$$

(Note that the part of u living in $\operatorname{span}\{v_i,\hat{v}_i^\perp\}^\perp$ is irrelevant for analyzing $\mathcal{E}_i(I,u,u)$.) We have

$$\begin{aligned}
\mathcal{E}_i(I,u,u) &= \lambda_i(u^\top v_i)^2 v_i - \hat{\lambda}_i(u^\top\hat{v}_i)^2\hat{v}_i \\
&= \lambda_i a_i^2 v_i - \hat{\lambda}_i\big(a_i c_i + \|\hat{v}_i^\perp\| b_i\big)^2\big(c_i v_i + \hat{v}_i^\perp\big) \\
&= \lambda_i a_i^2 v_i - \hat{\lambda}_i\big(a_i^2 c_i^2 + 2\|\hat{v}_i^\perp\| a_i b_i c_i + \|\hat{v}_i^\perp\|^2 b_i^2\big)c_i v_i - \hat{\lambda}_i\big(a_i c_i + \|\hat{v}_i^\perp\| b_i\big)^2\hat{v}_i^\perp \\
&= \underbrace{\Big((\lambda_i - \hat{\lambda}_i c_i^3)a_i^2 - 2\hat{\lambda}_i\|\hat{v}_i^\perp\| a_i b_i c_i^2 - \hat{\lambda}_i\|\hat{v}_i^\perp\|^2 b_i^2 c_i\Big)}_{=:A_i} v_i - \underbrace{\hat{\lambda}_i\|\hat{v}_i^\perp\|\big(a_i c_i + \|\hat{v}_i^\perp\| b_i\big)^2}_{=:B_i}\big(\hat{v}_i^\perp/\|\hat{v}_i^\perp\|\big) \\
&= A_i v_i - B_i\big(\hat{v}_i^\perp/\|\hat{v}_i^\perp\|\big).
\end{aligned}$$

The overall error can also be expressed in terms of the A_i and B_i:

$$\begin{aligned}
\Big\|\sum_{i=1}^t \mathcal{E}_i(I,u,u)\Big\|_2^2 &= \Big\|\sum_{i=1}^t A_i v_i - \sum_{i=1}^t B_i(\hat{v}_i^\perp/\|\hat{v}_i^\perp\|)\Big\|_2^2 \\
&\le 2\Big\|\sum_{i=1}^t A_i v_i\Big\|^2 + 2\Big\|\sum_{i=1}^t B_i(\hat{v}_i^\perp/\|\hat{v}_i^\perp\|)\Big\|_2^2 \\
&\le 2\sum_{i=1}^t A_i^2 + 2\Big(\sum_{i=1}^t |B_i|\Big)^2 \qquad (25)
\end{aligned}$$

where the first inequality uses the fact $(x+y)^2 \le 2(x^2+y^2)$ and the triangle inequality, and the second inequality uses the orthonormality of the v_i and the triangle inequality.

It remains to bound A_i^2 and $|B_i|$ in terms of $|a_i|$, λ_i, and $\tilde{\epsilon}$. The first term, A_i^2, can be bounded using the triangle inequality and the various bounds on $|\lambda_i - \hat{\lambda}_i|$, $\|\hat{v}_i - v_i\|$, $\|\hat{v}_i^\perp\|$, and c_i:

$$\begin{aligned}
|A_i| &\le (|\lambda_i - \hat{\lambda}_i|c_i^3 + \lambda_i|c_i^3 - 1|)a_i^2 + 2(\lambda_i + |\lambda_i - \hat{\lambda}_i|)\|\hat{v}_i^\perp\||a_i b_i|c_i^2 + (\lambda_i + |\lambda_i - \hat{\lambda}_i|)\|\hat{v}_i^\perp\|^2 b_i^2 c_i \\
&\le (|\lambda_i - \hat{\lambda}_i| + 1.5\lambda_i\|\hat{v}_i - v_i\|^2 + 2(\lambda_i + |\lambda_i - \hat{\lambda}_i|)\|\hat{v}_i - v_i\|)|a_i| + (\lambda_i + |\lambda_i - \hat{\lambda}_i|)\|\hat{v}_i - v_i\|^2 \\
&\le (\tilde{\epsilon} + 7\tilde{\epsilon}^2/\lambda_i + 4\tilde{\epsilon} + 4\tilde{\epsilon}^2/\lambda_i)|a_i| + 4\tilde{\epsilon}^2/\lambda_i + \tilde{\epsilon}^3/\lambda_i^2 \\
&= (5 + 11\tilde{\epsilon}/\lambda_i)\tilde{\epsilon}|a_i| + 4(1 + \tilde{\epsilon}/\lambda_i)\tilde{\epsilon}^2/\lambda_i,
\end{aligned}$$

and therefore (via $(x+y)^2 \le 2(x^2+y^2)$)

$$A_i^2 \le 2(5 + 11\tilde{\epsilon}/\lambda_i)^2\tilde{\epsilon}^2 a_i^2 + 32(1 + \tilde{\epsilon}/\lambda_i)^2\tilde{\epsilon}^4/\lambda_i^2.$$

The second term, $|B_i|$, is bounded similarly:

$$\begin{aligned}
|B_i| &\le 2(\lambda_i + |\lambda_i - \hat{\lambda}_i|)\|\hat{v}_i^\perp\|^2(a_i^2 + \|\hat{v}_i^\perp\|^2) \\
&\le 2(\lambda_i + |\lambda_i - \hat{\lambda}_i|)\|\hat{v}_i - v_i\|^2(a_i^2 + \|\hat{v}_i - v_i\|^2) \\
&\le 8(1 + \tilde{\epsilon}/\lambda_i)(\tilde{\epsilon}^2/\lambda_i)a_i^2 + 32(1 + \tilde{\epsilon}/\lambda_i)\tilde{\epsilon}^4/\lambda_i^3.
\end{aligned}$$

Therefore, using the inequality from (25) and again $(x+y)^2 \le 2(x^2+y^2)$,

$$\begin{aligned}
\left\|\sum_{i=1}^t \mathcal{E}_i(I,u,u)\right\|_2^2 &\le 2\sum_{i=1}^t A_i^2 + 2\Big(\sum_{i=1}^t |B_i|\Big)^2 \\
&\le 4(5+11\tilde{\epsilon}/\lambda_{\min})^2\tilde{\epsilon}^2\sum_{i=1}^t a_i^2 + 64(1+\tilde{\epsilon}/\lambda_{\min})^2\tilde{\epsilon}^2\sum_{i=1}^t(\tilde{\epsilon}/\lambda_i)^2 \\
&\quad + 2\bigg(8(1+\tilde{\epsilon}/\lambda_{\min})(\tilde{\epsilon}^2/\lambda_{\min})\sum_{i=1}^t a_i^2 + 32(1+\tilde{\epsilon}/\lambda_{\min})\tilde{\epsilon}\sum_{i=1}^t(\tilde{\epsilon}/\lambda_i)^3\bigg)^2 \\
&\le 4(5+11\tilde{\epsilon}/\lambda_{\min})^2\tilde{\epsilon}^2\sum_{i=1}^t a_i^2 + 64(1+\tilde{\epsilon}/\lambda_{\min})^2\tilde{\epsilon}^2\sum_{i=1}^t(\tilde{\epsilon}/\lambda_i)^2 \\
&\quad + 128(1+\tilde{\epsilon}/\lambda_{\min})^2(\tilde{\epsilon}/\lambda_{\min})^2\tilde{\epsilon}^2\sum_{i=1}^t a_i^2 \\
&\quad + 2048(1+\tilde{\epsilon}/\lambda_{\min})^2\tilde{\epsilon}^2\bigg(\sum_{i=1}^t(\tilde{\epsilon}/\lambda_i)^3\bigg)^2 \\
&= \Big(4(5+11\tilde{\epsilon}/\lambda_{\min})^2 + 128(1+\tilde{\epsilon}/\lambda_{\min})^2(\tilde{\epsilon}/\lambda_{\min})^2\Big)\tilde{\epsilon}^2\sum_{i=1}^t a_i^2 \\
&\quad + 64(1+\tilde{\epsilon}/\lambda_{\min})^2\tilde{\epsilon}^2\sum_{i=1}^t(\tilde{\epsilon}/\lambda_i)^2 + 2048(1+\tilde{\epsilon}/\lambda_{\min})^2\tilde{\epsilon}^2\bigg(\sum_{i=1}^t(\tilde{\epsilon}/\lambda_i)^3\bigg)^2.
\end{aligned}$$

■

B.4 Proof of the Main Theorem

Theorem B.1 *Let $\hat{T} = T + E \in \mathbb{R}^{k\times k\times k}$, where T is a symmetric tensor with orthogonal decomposition $T = \sum_{i=1}^k \lambda_i v_i^{\otimes 3}$ where each $\lambda_i > 0$, $\{v_1, v_2, \ldots, v_k\}$ is an orthonormal basis, and E has operator norm $\epsilon := \|E\|$. Define $\lambda_{\min} := \min\{\lambda_i : i \in [k]\}$, and $\lambda_{\max} := \max\{\lambda_i : i \in [k]\}$. There exists universal constants $C_1, C_2, C_3 > 0$ such that the following holds. Pick any $\eta \in (0,1)$, and suppose*

$$\epsilon \le C_1 \cdot \frac{\lambda_{\min}}{k}, \qquad N \ge C_2 \cdot \bigg(\log(k) + \log\log\Big(\frac{\lambda_{\max}}{\epsilon}\Big)\bigg),$$

and

$$\sqrt{\frac{\ln(L/\log_2(k/\eta))}{\ln(k)}} \cdot \left(1 - \frac{\ln(\ln(L/\log_2(k/\eta))) + C_3}{4\ln(L/\log_2(k/\eta))} - \sqrt{\frac{\ln(8)}{\ln(L/\log_2(k/\eta))}}\right) \ge 1.02\left(1 + \sqrt{\frac{\ln(4)}{\ln(k)}}\right).$$

(Note that the condition on L holds with $L = \mathrm{poly}(k)\log(1/\eta)$.) Suppose that Algorithm 1 is iteratively called k times, where the input tensor is $\hat{T}$ in the first call, and in each subsequent call, the input tensor is the deflated tensor returned by the previous call. Let $(\hat{v}_1, \hat{\lambda}_1), (\hat{v}_2, \hat{\lambda}_2), \ldots, (\hat{v}_k, \hat{\lambda}_k)$ be the sequence of estimated eigenvector/eigenvalue pairs returned in these k calls. With probability at least $1 - \eta$, there exists a permutation π on $[k]$ such that

$$\|v_{\pi(j)} - \hat{v}_j\| \leq 8\epsilon/\lambda_{\pi(j)}, \qquad |\lambda_{\pi(j)} - \hat{\lambda}_j| \leq 5\epsilon, \quad \forall j \in [k],$$

and

$$\left\| T - \sum_{j=1}^{k} \hat{\lambda}_j \hat{v}_j^{\otimes 3} \right\| \leq 55\epsilon.$$

Proof We prove by induction that for each $i \in [k]$ (corresponding to the i-th call to Algorithm 1), with probability at least $1 - i\eta/k$, there exists a permutation π on $[k]$ such that the following assertions hold.

1. For all $j \leq i$, $\|v_{\pi(j)} - \hat{v}_j\| \leq 8\epsilon/\lambda_{\pi(j)}$ and $|\lambda_{\pi(j)} - \hat{\lambda}_j| \leq 12\epsilon$.
2. The error tensor

$$\tilde{E}_{i+1} := \left(\hat{T} - \sum_{j \leq i} \hat{\lambda}_j \hat{v}_j^{\otimes 3} \right) - \sum_{j \geq i+1} \lambda_{\pi(j)} v_{\pi(j)}^{\otimes 3} = E + \sum_{j \leq i} \left(\lambda_{\pi(j)} v_{\pi(j)}^{\otimes 3} - \hat{\lambda}_j \hat{v}_j^{\otimes 3} \right)$$

   satisfies

$$\|\tilde{E}_{i+1}(I, u, u)\| \leq 56\epsilon, \quad \forall u \in S^{k-1}; \tag{26}$$

$$\|\tilde{E}_{i+1}(I, u, u)\| \leq 2\epsilon, \quad \forall u \in S^{k-1} \text{ s.t. } \exists j \geq i+1 \centerdot (u^\top v_{\pi(j)})^2 \geq 1 - (168\epsilon/\lambda_{\pi(j)})^2. \tag{27}$$

We actually take $i = 0$ as the base case, so we can ignore the first assertion, and just observe that for $i = 0$,

$$\tilde{E}_1 = \hat{T} - \sum_{j=1}^{k} \lambda_i v_i^{\otimes 3} = E.$$

We have $\|\tilde{E}_1\| = \|E\| = \epsilon$, and therefore the second assertion holds.

Now fix some $i \in [k]$, and assume as the inductive hypothesis that, with probability at least $1 - (i-1)\eta/k$, there exists a permutation π such that two assertions above hold for $i - 1$ (call this Event_{i-1}). The i-th call to Algorithm 1 takes as input

$$\tilde{T}_i := \hat{T} - \sum_{j \leq i-1} \hat{\lambda}_j \hat{v}_j^{\otimes 3},$$

which is intended to be an approximation to

$$T_i := \sum_{j \geq i} \lambda_{\pi(j)} v_{\pi(j)}^{\otimes 3}.$$

Observe that

$$\tilde{T}_i - T_i = \tilde{E}_i,$$

which satisfies the second assertion in the inductive hypothesis. We may write $T_i = \sum_{l=1}^{k} \tilde{\lambda}_l v_l^{\otimes 3}$ where $\tilde{\lambda}_l = \lambda_l$ whenever $\pi^{-1}(l) \geq i$, and $\tilde{\lambda}_l = 0$ whenever $\pi^{-1}(l) \leq i-1$. This form is used when referring to $\tilde{T}$ or the $\tilde{\lambda}_i$ in preceding lemmas (in particular, Lemma B.1 and Lemma B.4).

By Lemma B.1, with conditional probability at least $1 - \eta/k$ given Event_{i-1}, at least one of $\theta_0^{(\tau)}$ for $\tau \in [L]$ is γ-separated relative to $\pi(j_{\max})$, where $j_{\max} := \arg\max_{j \geq i} \lambda_{\pi(j)}$, (for $\gamma = 0.01$; call this Event'_i; note that the application of Lemma B.1 determines C_3). Therefore $\Pr[\mathsf{Event}_{i-1} \cap \mathsf{Event}'_i] = \Pr[\mathsf{Event}'_i | \mathsf{Event}_{i-1}] \Pr[\mathsf{Event}_{i-1}] \geq (1 - \eta/k)(1 - (i-1)\eta/k) \geq 1 - i\eta/k$. It remains to show that $\mathsf{Event}_{i-1} \cap \mathsf{Event}'_i \subseteq \mathsf{Event}_i$; so henceforth we condition on $\mathsf{Event}_{i-1} \cap \mathsf{Event}'_i$.

Set

$$C_1 := \min\left\{(56 \cdot 9 \cdot 102)^{-1}, (100 \cdot 168)^{-1}, \Delta' \text{ from Lemma B.5 with } \Delta = 1/50\right\}. \tag{28}$$

For all $\tau \in [L]$ such that $\theta_0^{(\tau)}$ is γ-separated relative to $\pi(j_{\max})$, we have (i) $|\theta^{(\tau)}_{j_{\max},0}| \geq 1/\sqrt{k}$, and (ii) that by Lemma B.4 (using $\tilde{\epsilon}/p := 2\epsilon$, $\kappa := 1$, and $i^* := \pi(j_{\max})$, and providing C_2),

$$|\tilde{T}_i(\theta_N^{(\tau)}, \theta_N^{(\tau)}, \theta_N^{(\tau)}) - \lambda_{\pi(j_{\max})}| \leq 5\epsilon$$

(notice by definition that $\gamma \geq 1/100$ implies $\gamma_0 \geq 1 - /(1+\gamma) \geq 1/101$, thus it follows from the bounds on the other quantities that $\tilde{\epsilon} = 2p\epsilon \leq 56 C_1 \cdot \frac{\lambda_{\min}}{k} < \frac{\gamma_0}{2(1+8\kappa)} \cdot \tilde{\lambda}_{\min} \cdot \theta^2_{i^*,0}$ as necessary). Therefore $\theta_N := \theta_N^{(\tau^*)}$ must satisfy

$$\tilde{T}_i(\theta_N, \theta_N, \theta_N) = \max_{\tau \in [L]} \tilde{T}_i(\theta_N^{(\tau)}, \theta_N^{(\tau)}, \theta_N^{(\tau)}) \geq \max_{j \geq i} \lambda_{\pi(j)} - 5\epsilon = \lambda_{\pi(j_{\max})} - 5\epsilon.$$

On the other hand, by the triangle inequality,

$$\begin{aligned}
\tilde{T}_i(\theta_N, \theta_N, \theta_N) &\leq \sum_{j \geq i} \lambda_{\pi(j)} \theta^3_{\pi(j),N} + |\tilde{E}_i(\theta_N, \theta_N, \theta_N)| \\
&\leq \sum_{j \geq i} \lambda_{\pi(j)} |\theta_{\pi(j),N}| \theta^2_{\pi(j),N} + 56\epsilon \\
&\leq \lambda_{\pi(j^*)} |\theta_{\pi(j^*),N}| + 56\epsilon
\end{aligned}$$

where $j^* := \arg\max_{j \geq i} \lambda_{\pi(j)} |\theta_{\pi(j),N}|$. Therefore

$$\lambda_{\pi(j^*)} |\theta_{\pi(j^*),N}| \geq \lambda_{\pi(j_{\max})} - 5\epsilon - 56\epsilon \geq \frac{4}{5} \lambda_{\pi(j_{\max})}.$$

Squaring both sides and using the fact that $\theta^2_{\pi(j^*),N} + \theta^2_{\pi(j),N} \leq 1$ for any $j \neq j^*$,

$$\begin{aligned}
\left(\lambda_{\pi(j^*)} \theta_{\pi(j^*),N}\right)^2 &\geq \frac{16}{25} \left(\lambda_{\pi(j_{\max})} \theta_{\pi(j^*),N}\right)^2 + \frac{16}{25} \left(\lambda_{\pi(j_{\max})} \theta_{\pi(j),N}\right)^2 \\
&\geq \frac{16}{25} \left(\lambda_{\pi(j^*)} \theta_{\pi(j^*),N}\right)^2 + \frac{16}{25} \left(\lambda_{\pi(j)} \theta_{\pi(j),N}\right)^2
\end{aligned}$$

which in turn implies

$$\lambda_{\pi(j)}|\theta_{\pi(j),N}| \le \frac{3}{4}\lambda_{\pi(j^*)}|\theta_{\pi(j^*),N}|, \quad j \neq j^*.$$

This means that θ_N is $(1/4)$-separated relative to $\pi(j^*)$. Also, observe that

$$|\theta_{\pi(j^*),N}| \ge \frac{4}{5} \cdot \frac{\lambda_{\pi(j_{\max})}}{\lambda_{\pi(j^*)}} \ge \frac{4}{5}, \quad \frac{\lambda_{\pi(j_{\max})}}{\lambda_{\pi(j^*)}} \le \frac{5}{4}.$$

Therefore by Lemma B.4 (using $\tilde{\epsilon}/p := 2\epsilon$, $\gamma := 1/4$, and $\kappa := 5/4$), executing another N power iterations starting from θ_N gives a vector $\hat{\theta}$ that satisfies

$$\|\hat{\theta} - v_{\pi(j^*)}\| \le \frac{8\epsilon}{\lambda_{\pi(j^*)}}, \qquad |\hat{\lambda} - \lambda_{\pi(j^*)}| \le 5\epsilon.$$

Since $\hat{v}_i = \hat{\theta}$ and $\hat{\lambda}_i = \hat{\lambda}$, the first assertion of the inductive hypothesis is satisfied, as we can modify the permutation π by swapping $\pi(i)$ and $\pi(j^*)$ without affecting the values of $\{\pi(j) : j \le i-1\}$ (recall $j^* \ge i$).

We now argue that $\tilde{E}_{i+1}$ has the required properties to complete the inductive step. By Lemma B.5 (using $\tilde{\epsilon} := 5\epsilon$ and $\Delta := 1/50$, the latter providing one upper bound on C_1 as per (28)), we have for any unit vector $u \in S^{k-1}$,

$$\left\| \left(\sum_{j \le i} \left(\lambda_{\pi(j)} v_{\pi(j)}^{\otimes 3} - \hat{\lambda}_j \hat{v}_j^{\otimes 3} \right) \right)(I, u, u) \right\| \le \left(1/50 + 100 \sum_{j=1}^{i} (u^\top v_{\pi(j)})^2 \right)^{1/2} 5\epsilon \le 55\epsilon. \quad (29)$$

Therefore by the triangle inequality,

$$\|\tilde{E}_{i+1}(I, u, u)\| \le \|E(I, u, u)\| + \left\| \left(\sum_{j \le i} \left(\lambda_{\pi(j)} v_{\pi(j)}^{\otimes 3} - \hat{\lambda}_j \hat{v}_j^{\otimes 3} \right) \right)(I, u, u) \right\| \le 56\epsilon.$$

Thus the bound (26) holds.

To prove that (27) holds, pick any unit vector $u \in S^{k-1}$ such that there exists $j' \ge i+1$ with $(u^\top v_{\pi(j')})^2 \ge 1 - (168\epsilon/\lambda_{\pi(j')})^2$. We have, via the second bound on C_1 in (28) and the corresponding assumed bound $\epsilon \le C_1 \cdot \frac{\lambda_{\min}}{k}$,

$$100 \sum_{j=1}^{i} (u^\top v_{\pi(j)})^2 \le 100 \left(1 - (u^\top v_{\pi(j')})^2 \right) \le 100 \left(\frac{168\epsilon}{\lambda_{\pi(j')}} \right)^2 \le \frac{1}{50},$$

and therefore

$$\left(1/50 + 100 \sum_{j=1}^{i} (u^\top v_{\pi(j)})^2 \right)^{1/2} 5\epsilon \le (1/50 + 1/50)^{1/2} 5\epsilon \le \epsilon.$$

By the triangle inequality, we have $\|\tilde{E}_{i+1}(I, u, u)\| \le 2\epsilon$. Therefore (27) holds, so the second assertion of the inductive hypothesis holds. Thus $\mathsf{Event}_{i-1} \cap \mathsf{Event}'_i \subseteq \mathsf{Event}_i$, and $\Pr[\mathsf{Event}_i] \ge \Pr[\mathsf{Event}_{i-1} \cap \mathsf{Event}'_i] \ge 1 - i\eta/k$. We conclude that by the induction principle,

there exists a permutation π such that two assertions hold for $i = k$, with probability at least $1 - \eta$.

From the last induction step ($i = k$), it is also clear from (29) that $\|T - \sum_{j=1}^{k} \hat{\lambda}_j \hat{v}_j^{\otimes 3}\| \leq 55\epsilon$ (in $\mathsf{Event}_{k-1} \cap \mathsf{Event}'_k$). This completes the proof of the theorem. ■

Appendix C. Variant of Robust Power Method that uses a Stopping Condition

In this section we analyze a variant of Algorithm 1 that uses a stopping condition. The variant is described in Algorithm 2. The key difference is that the inner for-loop is repeated until a stopping condition is satisfied (rather than explicitly L times). The stopping condition ensures that the power iteration is converging to an eigenvector, and it will be satisfied within poly(k) random restarts with high probability. The condition depends on one new quantity, r, which should be set to $r := k - \#$ deflation steps so far (*i.e.*, the first call to Algorithm 2 uses $r = k$, the second call uses $r = k - 1$, and so on).

Algorithm 2 Robust tensor power method with stopping condition

input symmetric tensor $\tilde{T} \in \mathbb{R}^{k \times k \times k}$, number of iterations N, expected rank r.
output the estimated eigenvector/eigenvalue pair; the deflated tensor.

1: **repeat**
2: Draw θ_0 uniformly at random from the unit sphere in $\mathbb{R}^k$.
3: **for** $t = 1$ to N **do**
4: Compute power iteration update

$$\theta_t := \frac{\tilde{T}(I, \theta_{t-1}, \theta_{t-1})}{\|\tilde{T}(I, \theta_{t-1}, \theta_{t-1})\|} \tag{30}$$

5: **end for**
6: **until** the following stopping condition is satisfied:

$$|\tilde{T}(\theta_N, \theta_N, \theta_N)| \geq \max\left\{ \frac{1}{2\sqrt{r}} \|\tilde{T}\|_F,\ \frac{1}{1.05} \|\tilde{T}(I, I, \theta_N)\|_F \right\}.$$

7: Do N power iteration updates (30) starting from θ_N to obtain $\hat{\theta}$, and set $\hat{\lambda} := \tilde{T}(\hat{\theta}, \hat{\theta}, \hat{\theta})$.
8: **return** the estimated eigenvector/eigenvalue pair $(\hat{\theta}, \hat{\lambda})$; the deflated tensor $\tilde{T} - \hat{\lambda}\, \hat{\theta}^{\otimes 3}$.

C.1 Stopping Condition Analysis

For a matrix A, we use $\|A\|_F := (\sum_{i,j} A_{i,j}^2)^{1/2}$ to denote its Frobenius norm. For a third-order tensor A, we use $\|A\|_F := (\sum_i \|A(I, I, e_i)\|_F^2)^{1/2} = (\sum_i \|A(I, I, v_i)\|_F^2)^{1/2}$.

Define $\tilde{T}$ as before in (11):

$$\tilde{T} := \sum_{i=1}^{k} \tilde{\lambda}_i v_i^{\otimes 3} + \tilde{E}.$$

We assume $\tilde{E}$ is a symmetric tensor such that, for some constant $p > 1$,

$$\begin{aligned}
&\|\tilde{E}(I, u, u)\| \le \tilde{\epsilon}, \quad \forall u \in S^{k-1};\\
&\|\tilde{E}(I, u, u)\| \le \tilde{\epsilon}/p, \quad \forall u \in S^{k-1} \text{ s.t. } (u^\top v_1)^2 \ge 1 - (3\tilde{\epsilon}/\tilde{\lambda}_1)^2;\\
&\|\tilde{E}\|_F \le \tilde{\epsilon}_F.
\end{aligned}$$

Assume that not all $\tilde{\lambda}_i$ are zero, and define

$$\begin{aligned}
\tilde{\lambda}_{\min} &:= \min\{\tilde{\lambda}_i : i \in [k],\ \tilde{\lambda}_i > 0\}, & \tilde{\lambda}_{\max} &:= \max\{\tilde{\lambda}_i : i \in [k]\},\\
\ell &:= |\{i \in [k] : \tilde{\lambda}_i > 0\}|, & \tilde{\lambda}_{\text{avg}} &:= \left(\frac{1}{\ell}\sum_{i=1}^{k} \tilde{\lambda}_i^2\right)^{1/2}.
\end{aligned}$$

We show in Lemma C.1 that if the stopping condition is satisfied by a vector θ, then it must be close to an eigenvector of $\tilde{T}$. Then in Lemma C.2, we show that the stopping condition is satisfied by θ_N when θ_0 is a good starting point (as per the conditions of Lemma B.4).

Lemma C.1 *Fix any vector $\theta = \sum_{i=1}^{k} \theta_i v_i$, and let $i^* := \arg\max_{i\in[k]} \tilde{\lambda}_i|\theta_i|$. Assume that $\ell \ge 1$ and that for some $\alpha \in (0, 1/20)$ and $\beta \ge 2\alpha/\sqrt{k}$,*

$$\tilde{\epsilon} \le \alpha \cdot \frac{\tilde{\lambda}_{\min}}{\sqrt{k}}, \quad \tilde{\epsilon}_F \le \sqrt{\ell}\Big(\frac{1}{2} - \frac{\alpha}{\beta\sqrt{k}}\Big) \cdot \tilde{\lambda}_{\text{avg}}.$$

If the stopping condition

$$|\tilde{T}(\theta, \theta, \theta)| \ge \max\left\{\frac{\beta}{\sqrt{\ell}}\|\tilde{T}\|_F,\ \frac{1}{1+\alpha}\|\tilde{T}(I, I, \theta)\|_F\right\} \tag{31}$$

holds, then

1. *$\tilde{\lambda}_{i^*} \ge \beta\tilde{\lambda}_{\text{avg}}/2$ and $\tilde{\lambda}_{i^*}|\theta_{i^*}| > 0$;*
2. *$\max_{i\ne i^*} \tilde{\lambda}_i|\theta_i| \le \sqrt{7\alpha} \cdot \tilde{\lambda}_{i^*}|\theta_{i^*}|$;*
3. *$\theta_{i^*} \ge 1 - 2\alpha$.*

Proof Without loss of generality, assume $i^* = 1$. First, we claim that $\tilde{\lambda}_1|\theta_1| > 0$. By the triangle inequality,

$$|\tilde{T}(\theta, \theta, \theta)| \le \sum_{i=1}^{k} \tilde{\lambda}_i\theta_i^3 + |\tilde{E}(\theta, \theta, \theta)| \le \sum_{i=1}^{k} \tilde{\lambda}_i|\theta_i|\theta_i^2 + \tilde{\epsilon} \le \tilde{\lambda}_1|\theta_1| + \tilde{\epsilon}.$$

Moreover,

$$\begin{aligned}
\|\tilde{T}\|_F &\geq \left\|\sum_{i=1}^k \tilde{\lambda}_i v_i^{\otimes 3}\right\|_F - \|\tilde{E}\|_F \\
&= \left(\sum_{j=1}^k \left\|\sum_{i=1}^k \tilde{\lambda}_i v_i v_i^\top (v_i^\top v_j)\right\|_F^2\right)^{1/2} - \|\tilde{E}\|_F \\
&= \left(\sum_{j=1}^k \left\|\tilde{\lambda}_j v_j v_j^\top\right\|_F^2\right)^{1/2} - \|\tilde{E}\|_F \\
&= \left(\sum_{j=1}^k \tilde{\lambda}_j^2\right)^{1/2} - \|\tilde{E}\|_F \\
&\geq \sqrt{\ell}\tilde{\lambda}_{\text{avg}} - \tilde{\epsilon}_F.
\end{aligned}$$

By assumption, $|\tilde{T}(\theta,\theta,\theta)| \geq (\beta/\sqrt{\ell})\|\tilde{T}\|_F$, so

$$\tilde{\lambda}_1|\theta_1| \geq \beta\tilde{\lambda}_{\text{avg}} - \frac{\beta}{\sqrt{\ell}}\tilde{\epsilon}_F - \tilde{\epsilon} \geq \beta\tilde{\lambda}_{\text{avg}} - \beta\Big(\frac{1}{2} - \frac{\alpha}{\beta\sqrt{k}}\Big)\tilde{\lambda}_{\text{avg}} - \frac{\alpha}{\sqrt{k}}\tilde{\lambda}_{\min} \geq \frac{\beta}{2}\tilde{\lambda}_{\text{avg}}$$

where the second inequality follows from the assumptions on $\tilde{\epsilon}$ and $\tilde{\epsilon}_F$. Since $\beta > 0$, $\tilde{\lambda}_{\text{avg}} > 0$, and $|\theta_1| \leq 1$, it follows that

$$\tilde{\lambda}_1 \geq \frac{\beta}{2}\tilde{\lambda}_{\text{avg}}, \quad \tilde{\lambda}_1|\theta_1| > 0.$$

This proves the first claim.

Now we prove the second claim. Define $\tilde{M} := \tilde{T}(I,I,\theta) = \sum_{i=1}^k \tilde{\lambda}_i\theta_i v_i v_i^\top + \tilde{E}(I,I,\theta)$ (a symmetric $k \times k$ matrix), and consider its eigenvalue decomposition

$$\tilde{M} = \sum_{i=1}^k \phi_i u_i u_i^\top$$

where, without loss of generality, $|\phi_1| \geq |\phi_2| \geq \cdots \geq |\phi_k|$ and $\{u_1, u_2, \ldots, u_k\}$ is an orthonormal basis. Let $M := \sum_{i=1}^k \tilde{\lambda}_i\theta_i v_i v_i^\top$, so $\tilde{M} = M + \tilde{E}(I,I,\theta)$. Note that the $\tilde{\lambda}_i|\theta_i|$ and $|\phi_i|$ are the singular values of M and $\tilde{M}$, respectively. We now show that the assumption on $|\tilde{T}(\theta,\theta,\theta)|$ implies that almost all of the energy in M is contained in its top singular component.

By Weyl's theorem,

$$|\phi_1| \leq \tilde{\lambda}_1|\theta_1| + \|\tilde{M} - M\| \leq \tilde{\lambda}_1|\theta_1| + \tilde{\epsilon}.$$

Next, observe that the assumption $\|\tilde{T}(I,I,\theta)\|_F \leq (1+\alpha)\tilde{T}(\theta,\theta,\theta)$ is equivalent to $(1+\alpha)\theta^\top\tilde{M}\theta \geq \|\tilde{M}\|_F$. Therefore, using the fact that $|\phi_1| = \max_{u \in S^{k-1}} |u^\top\tilde{M}u|$, the triangle

inequality, and the fact $\|A\|_F \le \sqrt{k}\|A\|$ for any matrix $A \in \mathbb{R}^{k\times k}$,

$$\begin{aligned}(1+\alpha)|\phi_1| \ge (1+\alpha)\theta^\top \tilde{M}\theta &\ge \|\tilde{M}\|_F \\ &\ge \left\|\sum_{i=1}^k \tilde{\lambda}_i\theta_i v_i v_i^\top\right\|_F - \left\|\tilde{E}(I,I,\theta)\right\|_F \\ &\ge \left(\sum_{i=1}^k \tilde{\lambda}_i^2\theta_i^2\right)^{1/2} - \sqrt{k}\|\tilde{E}(I,I,\theta)\| \\ &\ge \left(\sum_{i=1}^k \tilde{\lambda}_i^2\theta_i^2\right)^{1/2} - \sqrt{k}\tilde{\epsilon}.\end{aligned} \tag{32}$$

Combining these bounds on $|\phi_1|$ gives

$$\tilde{\lambda}_1|\theta_1| + \tilde{\epsilon} \ge \frac{1}{1+\alpha}\left[\left(\sum_{i=1}^k \tilde{\lambda}_i^2\theta_i^2\right)^{1/2} - \sqrt{k}\tilde{\epsilon}\right]. \tag{33}$$

The assumption $\tilde{\epsilon} \le \alpha\tilde{\lambda}_{\min}/\sqrt{k}$ implies that

$$\sqrt{k}\tilde{\epsilon} \le \alpha\tilde{\lambda}_{\min} \le \alpha\left(\sum_{i=1}^k \tilde{\lambda}_i^2\theta_i^2\right)^{1/2}.$$

Moreover, since $\tilde{\lambda}_1|\theta_1| > 0$ (by the first claim) and $\tilde{\lambda}_1|\theta_1| = \max_{i\in[k]} \tilde{\lambda}_i|\theta_i|$, it follows that

$$\tilde{\lambda}_1|\theta_1| \ge \tilde{\lambda}_{\min}\max_{i\in[k]}|\theta_i| \ge \frac{\tilde{\lambda}_{\min}}{\sqrt{k}}, \tag{34}$$

so we also have

$$\tilde{\epsilon} \le \alpha\tilde{\lambda}_1|\theta_1|.$$

Applying these bounds on $\tilde{\epsilon}$ to (33), we obtain

$$\tilde{\lambda}_1|\theta_1| \ge \frac{1-\alpha}{(1+\alpha)^2}\left(\sum_{i=1}^k \tilde{\lambda}_i^2\theta_i^2\right)^{1/2} \ge \frac{1-\alpha}{(1+\alpha)^2}\left(\tilde{\lambda}_1^2\theta_1^2 + \max_{i\ne 1}\tilde{\lambda}_i^2\theta_i^2\right)^{1/2}$$

which in turn implies (for $\alpha \in (0, 1/20)$)

$$\max_{i\ne 1}\tilde{\lambda}_i^2\theta_i^2 \le \left(\frac{(1+\alpha)^4}{(1-\alpha)^2} - 1\right)\cdot\tilde{\lambda}_1^2\theta_1^2 \le 7\alpha\cdot\tilde{\lambda}_1^2\theta_1^2.$$

Therefore $\max_{i\ne 1}\tilde{\lambda}_i|\theta_i| \le \sqrt{7\alpha}\cdot\tilde{\lambda}_1|\theta_1|$, proving the second claim.

Now we prove the final claim. This is done by (i) showing that θ has a large projection onto u_1, (ii) using an SVD perturbation argument to show that $\pm u_1$ is close to v_1, and (iii) concluding that θ has a large projection onto v_1.

We begin by showing that $(u_1^\top\theta)^2$ is large. Observe that from (32), we have $(1+\alpha)^2\phi_1^2 \ge \|\tilde{M}\|_F^2 \ge \phi_1^2 + \max_{i\ne 1}\phi_i^2$, and therefore

$$\max_{i\ne 1}|\phi_i| \le \sqrt{2\alpha+\alpha^2}\cdot|\phi_1|.$$

Moreover, by the triangle inequality,

$$\begin{aligned}|\theta^\top \tilde{M}\theta| &\le \sum_{i=1}^k |\phi_i|(u_i^\top\theta)^2 \\ &\le |\phi_1|(u_1^\top\theta)^2 + \max_{i\neq 1}|\phi_i|\big(1-(u_1^\top\theta)^2\big) \\ &= (u_1^\top\theta)^2\big(|\phi_1| - \max_{i\neq 1}|\phi_i|\big) + \max_{i\neq 1}|\phi_i|.\end{aligned}$$

Using (32) once more, we have $|\theta^\top\tilde{M}\theta| \ge \|\tilde{M}\|_F/(1+\alpha) \ge |\phi_1|/(1+\alpha)$, so

$$(u_1^\top\theta)^2 \ge \frac{\frac{1}{1+\alpha} - \max_{i\neq 1}\frac{|\phi_i|}{|\phi_1|}}{1-\max_{i\neq 1}\frac{|\phi_i|}{|\phi_1|}} = 1 - \frac{\alpha}{(1+\alpha)\Big(1-\max_{i\neq 1}\frac{|\phi_i|}{|\phi_1|}\Big)} \le 1 - \frac{\alpha}{(1+\alpha)(1-\sqrt{2\alpha+\alpha^2})}.$$

Now we show that $(u_1^\top v_1)^2$ is also large. By the second claim, the assumption on $\tilde{\epsilon}$, and (34),

$$\tilde{\lambda}_1|\theta_1| - \max_{i\neq 1}\tilde{\lambda}_i|\theta_i| > (1-\sqrt{7\alpha})\cdot\tilde{\lambda}_1|\theta_1| \ge (1-\sqrt{7\alpha})\cdot\tilde{\lambda}_{\min}/\sqrt{k}.$$

Combining this with Weyl's theorem gives

$$|\phi_1| - \max_{i\neq 1}\tilde{\lambda}_i|\theta_i| \ge \tilde{\lambda}_1|\theta_1| - \tilde{\epsilon} - \max_{i\neq 1}\tilde{\lambda}_i|\theta_i| \ge (1-(\alpha+\sqrt{7\alpha}))\cdot\tilde{\lambda}_{\min}/\sqrt{k},$$

so we may apply Wedin's theorem to obtain

$$(u_1^\top v_1)^2 \ge 1 - \left(\frac{\|\tilde{E}(I,I,\theta)\|}{|\phi_1| - \max_{i\neq 1}\tilde{\lambda}_i|\theta_i|}\right)^2 \ge 1 - \left(\frac{\alpha}{1-(\alpha+\sqrt{7\alpha})}\right)^2.$$

It remains to show that $\theta_1 = v_1^\top\theta$ is large. Indeed, by the triangle inequality, Cauchy-Schwarz, and the above inequalities on $(u_1^\top v_1)^2$ and $(u_1^\top\theta)^2$,

$$\begin{aligned}|v_1^\top\theta| &= \left|\sum_{i=1}^k (u_i^\top v_1)(u_i^\top\theta)\right| \\ &\ge |u_1^\top v_1||u_1^\top\theta| - \sum_{i=2}^k |u_i^\top v_1||u_i^\top\theta| \\ &\ge |u_1^\top v_1||u_1^\top\theta| - \left(\sum_{i=2}^k (u_i^\top v_1)^2\right)^{1/2}\left(\sum_{i=2}^k (u_i^\top\theta)^2\right)^{1/2} \\ &= |u_1^\top v_1||u_1^\top\theta| - \left(\Big(1-(u_i^\top v_1)^2\Big)\Big(1-(u_i^\top\theta)^2\Big)\right)^{1/2} \\ &\ge \left(\left(1-\frac{\alpha}{(1+\alpha)(1-\sqrt{2\alpha+\alpha^2})}\right)\left(1-\left(\frac{\alpha}{1-(\alpha+\sqrt{7\alpha})}\right)^2\right)\right)^{1/2} \\ &\quad - \left(\frac{\alpha}{(1+\alpha)(1-\sqrt{2\alpha+\alpha^2})}\cdot\left(\frac{\alpha}{1-(\alpha+\sqrt{7\alpha})}\right)^2\right)^{1/2} \\ &\ge 1-2\alpha\end{aligned}$$

for $\alpha \in (0, 1/20)$. Moreover, by assumption we have $\tilde{T}(\theta, \theta, \theta) \geq 0$, and

$$\begin{aligned}
\tilde{T}(\theta,\theta,\theta) &= \sum_{i=1}^{k} \tilde{\lambda}_i \theta_i^3 + \tilde{E}(\theta,\theta,\theta) \\
&= \tilde{\lambda}_1 \theta_1^3 + \sum_{i=2}^{k} \tilde{\lambda}_i \theta_i^3 + \tilde{E}(\theta,\theta,\theta) \\
&\leq \tilde{\lambda}_1 \theta_1^3 + \max_{i \neq 1} \tilde{\lambda}_i |\theta_i| \sum_{i=2}^{k} \theta_i^2 + \tilde{\epsilon} \\
&\leq \tilde{\lambda}_1 \theta_1^3 + \sqrt{7\alpha} \tilde{\lambda}_1 |\theta_1| (1 - \theta_1^2) + \tilde{\epsilon} \quad \text{(by the second claim)} \\
&\leq \tilde{\lambda}_1 |\theta_1|^3 \Big(\operatorname{sign}(\theta_1) + \frac{\sqrt{7\alpha}}{(1-2\alpha)^2} - \sqrt{7\alpha} + \frac{\alpha}{(1-2\alpha)^3} \Big) \quad \text{(since } |\theta_1| \geq 1 - 2\alpha\text{)} \\
&< \tilde{\lambda}_1 |\theta_1|^3 \Big(\operatorname{sign}(\theta_1) + 1 \Big)
\end{aligned}$$

so $\operatorname{sign}(\theta_1) > -1$, meaning $\theta_1 > 0$. Therefore $\theta_1 = |\theta_1| \geq 1 - 2\alpha$. This proves the final claim. ■

Lemma C.2 *Fix $\alpha, \beta \in (0, 1)$. Assume $\tilde{\lambda}_{i^*} = \max_{i \in [k]} \tilde{\lambda}_i$ and*

$$\tilde{\epsilon} \leq \min\left\{ \frac{\alpha}{5\sqrt{k} + 7}, \frac{1-\beta}{7} \right\} \cdot \tilde{\lambda}_{i^*}, \quad \tilde{\epsilon}_F \leq \sqrt{\ell} \cdot \frac{1-\beta}{2\beta} \cdot \tilde{\lambda}_{i^*}.$$

To the conclusion of Lemma B.4, it can be added that the stopping condition (31) *is satisfied by $\theta = \theta_t$.*

Proof Without loss of generality, assume $i^* = 1$. By the triangle inequality and Cauchy-Schwarz,

$$\begin{aligned}
\|\tilde{T}(I, I, \theta_t)\|_F &\leq \tilde{\lambda}_1 |\theta_{1,t}| + \sum_{i \neq 1} \lambda_i |\theta_{i,t}| + \|\tilde{E}(I, I, \theta_t)\|_F \leq \tilde{\lambda}_1 |\theta_{1,t}| + \tilde{\lambda}_1 \sqrt{k} \Big(\sum_{i \neq 1} \theta_{i,t}^2 \Big)^{1/2} + \sqrt{k} \tilde{\epsilon} \\
&\leq \tilde{\lambda}_1 |\theta_{1,t}| + \frac{3\sqrt{k}\tilde{\epsilon}}{p} + \sqrt{k}\tilde{\epsilon}.
\end{aligned}$$

where the last step uses the fact that $\theta_{1,t}^2 \geq 1 - (3\tilde{\epsilon}/(p\tilde{\lambda}_1))^2$. Moreover,

$$\tilde{T}(\theta_t, \theta_t, \theta_t) \geq \tilde{\lambda}_1 - \left(27 \Big(\frac{\tilde{\epsilon}}{p\tilde{\lambda}_1} \Big)^2 + 2 \right) \frac{\tilde{\epsilon}}{p}.$$

Combining these two inequalities with the assumption on $\tilde{\epsilon}$ implies that

$$\tilde{T}(\theta_t, \theta_t, \theta_t) \geq \frac{1}{1+\alpha} \|\tilde{T}(I, I, \theta_t)\|_F.$$

Using the definition of the tensor Frobenius norm, we have

$$\frac{1}{\sqrt{\ell}}\|\tilde{T}\|_F \leq \frac{1}{\sqrt{\ell}}\left\|\sum_{i=1}^{k}\tilde{\lambda}_i v_i^{\otimes 3}\right\|_F + \frac{1}{\sqrt{\ell}}\|\tilde{E}\|_F = \tilde{\lambda}_{\text{avg}} + \frac{1}{\sqrt{\ell}}\|\tilde{E}\|_F \leq \tilde{\lambda}_{\text{avg}} + \frac{1}{\sqrt{\ell}}\tilde{\epsilon}_F.$$

Combining this with the above inequality implies

$$\tilde{T}(I, I, \theta_t) \geq \frac{\beta}{\sqrt{\ell}}\|\tilde{T}\|_F.$$

Therefore the stopping condition (31) is satisfied. ■

C.2 Sketch of Analysis of Algorithm 2

The analysis of Algorithm 2 is very similar to the proof of Theorem 5.1 for Algorithm 1, so here we just sketch the essential differences.

First, the guarantee afforded to Algorithm 2 is somewhat different than Theorem 5.1. Specifically, it is of the following form: (i) under appropriate conditions, upon termination, the algorithm returns an accurate decomposition, and (ii) the algorithm terminates after poly(k) random restarts with high probability.

The conditions on ϵ and N are the same (but for possibly different universal constants C_1, C_2). In Lemma C.1 and Lemma C.2, there is reference to a condition on the Frobenius norm of E, but we may use the inequality $\|E\|_F \leq k\|E\| \leq k\epsilon$ so that the condition is subsumed by the ϵ condition.

Now we outline the differences relative to the proof of Theorem 5.1. The basic structure of the induction argument is the same. In the induction step, we argue that (i) if the stopping condition is satisfied, then by Lemma C.1 (with $\alpha = 0.05$ and $\beta = 1/2$), we have a vector θ_N such that, for some $j^* \geq i$,

1. $\lambda_{\pi(j^*)} \geq \lambda_{\pi(j_{\max})}/(4\sqrt{k})$;
2. θ_N is $(1/4)$-separated relative to $\pi(j^*)$;
3. $\theta_{\pi(j^*),N} \geq 4/5$;

and (ii) the stopping condition is satisfied within poly(k) random restarts (via Lemma B.1 and Lemma C.2) with high probability. We now invoke Lemma B.4 to argue that executing another N power iterations starting from θ_N gives a vector $\hat{\theta}$ that satisfies

$$\|\hat{\theta} - v_{\pi(j^*)}\| \leq \frac{8\epsilon}{\lambda_{\pi(j^*)}}, \qquad |\hat{\lambda} - \lambda_{\pi(j^*)}| \leq 5\epsilon.$$

The main difference here, relative to the proof of Theorem 5.1, is that we use $\kappa := 4\sqrt{k}$ (rather than $\kappa = O(1)$), but this ultimately leads to the same guarantee after taking into consideration the condition $\epsilon \leq C_1\lambda_{\min}/k$. The remainder of the analysis is essentially the same as the proof of Theorem 5.1.

Appendix D. Simultaneous Diagonalization for Tensor Decomposition

As discussed in the introduction, another standard approach to certain tensor decomposition problems is to simultaneously diagonalize a collection of similar matrices obtained from the given tensor. We now examine this approach in the context of our latent variable models, where

$$\begin{aligned} M_2 &= \sum_{i=1}^{k} w_i \ \mu_i \otimes \mu_i \\ M_3 &= \sum_{i=1}^{k} w_i \ \mu_i \otimes \mu_i \otimes \mu_i. \end{aligned}$$

Let $V := [\mu_1 | \mu_2 | \cdots | \mu_k]$ and $D(\eta) := \operatorname{diag}(\mu_1^\top \eta, \mu_2^\top \eta, \ldots, \mu_k^\top \eta)$, so

$$\begin{aligned} M_2 &= V \operatorname{diag}(w_1, w_2, \ldots w_k) V^\top \\ M_3(I, I, \eta) &= V \operatorname{diag}(w_1, w_2, \ldots w_k) D(\eta) V^\top \end{aligned}$$

Thus, the problem of determining the μ_i can be cast as a simultaneous diagonalization problem: find a matrix X such that $X^\top M_2 X$ and $X^\top M_3(I, I, \eta) X$ (for all η) are diagonal. It is easy to see that if the μ_i are linearly independent, then the solution $X^\top = V^\dagger$ is unique up to permutation and rescaling of the columns.

With exact moments, a simple approach is as follows. Assume for simplicity that $d = k$, and define

$$M(\eta) := M_3(I, I, \eta) M_2^{-1} = V D(\eta) V^{-1}.$$

Observe that if the diagonal entries of $D(\eta)$ are distinct, then the eigenvectors of $M(\eta)$ are the columns of V (up to permutation and scaling). This criterion is satisfied almost surely when η is chosen randomly from a continuous distribution over $\mathbb{R}^k$.

The above technique (or some variant thereof) was previously used to give the efficient learnability results, where the computational and sample complexity bounds were polynomial in relevant parameters of the problem, including the rank parameter k (Mossel and Roch, 2006; Anandkumar et al., 2012c,a; Hsu and Kakade, 2013). However, the specific polynomial dependence on k was rather large due to the need for the diagonal entries of $D(\eta)$ to be well-separated. This is because with finite samples, $M(\eta)$ is only known up to some perturbation, and thus the sample complexity bound depends inversely in (some polynomial of) the separation of the diagonal entries of $D(\eta)$. With η drawn uniformly at random from the unit sphere in $\mathbb{R}^k$, the separation was only guaranteed to be roughly $1/k^{2.5}$ (Anandkumar et al., 2012c) (while this may be a loose estimate, the instability is observed in practice). In contrast, using the tensor power method to approximately recover V (and hence the model parameters μ_i and w_i) requires only a mild, *lower-order* dependence on k.

It should be noted, however, that the use of a single random choice of η is quite restrictive, and it is easy to see that a simultaneous diagonalization of $M(\eta)$ for several choices of η can be beneficial. While the uniqueness of the eigendecomposition of $M(\eta)$ is only guaranteed when the diagonal entries of $D(\eta)$ are distinct, the *simultaneous diagonalization* of $M(\eta^{(1)}), M(\eta^{(2)}), \ldots, M(\eta^{(m)})$ for vectors $\eta^{(1)}, \eta^{(2)}, \ldots, \eta^{(m)}$ is unique as long as the

columns of

$$\begin{bmatrix} \mu_1^\top \eta^{(1)} & \mu_2^\top \eta^{(1)} & \cdots & \mu_k^\top \eta^{(1)} \\ \mu_1^\top \eta^{(2)} & \mu_2^\top \eta^{(2)} & \cdots & \mu_k^\top \eta^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_1^\top \eta^{(m)} & \mu_2^\top \eta^{(m)} & \cdots & \mu_k^\top \eta^{(m)} \end{bmatrix}$$

are distinct (*i.e.*, for each pair of column indices i, j, there exists a row index r such that the (r, i)-th and (r, j)-th entries are distinct). This is a much weaker requirement for uniqueness, and therefore may translate to an improved perturbation analysis. In fact, using the techniques discussed in Section 4.3, we may even reduce the problem to an orthogonal simultaneous diagonalization, which may be easier to obtain. Furthermore, a number of robust numerical methods for (approximately) simultaneously diagonalizing collections of matrices have been proposed and used successfully in the literature (*e.g.*, Bunse-Gerstner et al., 1993; Cardoso and Souloumiac, 1993; Cardoso, 1994; Cardoso and Comon, 1996; Ziehe et al., 2004). Another alternative and a more stable approach compared to full diagonalization is a Schur-like method which finds a unitary matrix U which simultaneously triangularizes the respective matrices (Corless et al., 1997). It is an interesting open question whether these techniques can yield similar improved learnability results and also enjoy the attractive computational properties of the tensor power method.

References

D. Achlioptas and F. McSherry. On spectral learning of mixtures of distributions. In *Eighteenth Annual Conference on Learning Theory*, pages 458–469, 2005.

E. S. Allman, C. Matias, and J. A. Rhodes. Identifiability of parameters in latent structure models with many observed variables. *The Annals of Statistics*, 37(6A):3099–3132, 2009.

A. Anandkumar, D. P. Foster, D. Hsu, S. M. Kakade, and Y.-K. Liu. A spectral algorithm for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems 25*, 2012a.

A. Anandkumar, D. Hsu, F. Huang, and S. M. Kakade. Learning mixtures of tree graphical models. In *Advances in Neural Information Processing Systems 25*, 2012b.

A. Anandkumar, D. Hsu, and S. M. Kakade. A method of moments for mixture models and hidden Markov models. In *Twenty-Fifth Annual Conference on Learning Theory*, volume 23, pages 33.1–33.34, 2012c.

J. Anderson, M. Belkin, N. Goyal, L. Rademacher, and J. Voss. The more, the merrier: the blessing of dimensionality for learning large Gaussian mixtures. In *Twenty-Seventh Annual Conference on Learning Theory*, 2014.

S. Arora and R. Kannan. Learning mixtures of separated nonspherical Gaussians. *The Annals of Applied Probability*, 15(1A):69–92, 2005.

S. Arora, R. Ge, and A. Moitra. Learning topic models — going beyond SVD. In *Fifty-Third IEEE Annual Symposium on Foundations of Computer Science*, pages 1–10, 2012a.

S. Arora, R. Ge, A. Moitra, and S. Sachdeva. Provable ICA with unknown Gaussian noise, and implications for Gaussian mixtures and autoencoders. In *Advances in Neural Information Processing Systems 25*, 2012b.

T. Austin. On exchangeable random variables and the statistics of large graphs and hypergraphs. *Probab. Survey*, 5:80–145, 2008.

R. Bailly. Quadratic weighted automata: Spectral algorithm and likelihood maximization. *Journal of Machine Learning Research*, 2011.

B. Balle and M. Mohri. Spectral learning of general weighted automata via constrained matrix completion. In *Advances in Neural Information Processing Systems 25*, 2012.

B. Balle, A. Quattoni, and X. Carreras. Local loss optimization in operator models: A new insight into spectral learning. In *Twenty-Ninth International Conference on Machine Learning*, 2012.

M. Belkin and K. Sinha. Polynomial learning of distribution families. In *Fifty-First Annual IEEE Symposium on Foundations of Computer Science*, pages 103–112, 2010.

A. Bhaskara, M. Charikar, A. Moitra, and A. Vijayaraghavan. Smoothed analysis of tensor decompositions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, 2014.

B. Boots, S. M. Siddiqi, and G. J. Gordon. Closing the learning-planning loop with predictive state representations. In *Proceedings of the Robotics Science and Systems Conference*, 2010.

S. C. Brubaker and S. Vempala. Isotropic PCA and affine-invariant clustering. In *Forty-Ninth Annual IEEE Symposium on Foundations of Computer Science*, 2008.

A. Bunse-Gerstner, R. Byers, and V. Mehrmann. Numerical methods for simultaneous diagonalization. *SIAM Journal on Matrix Analysis and Applications*, 14(4):927–949, 1993.

J.-F. Cardoso. Super-symmetric decomposition of the fourth-order cumulant tensor. blind identification of more sources than sensors. In *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, pages 3109–3112. IEEE, 1991.

J.-F. Cardoso. Perturbation of joint diagonalizers. Technical Report 94D027, Signal Department, Télécom Paris, 1994.

J.-F. Cardoso and P. Comon. Independent component analysis, a survey of some algebraic methods. In *IEEE International Symposium on Circuits and Systems*, pages 93–96, 1996.

J.-F. Cardoso and A. Souloumiac. Blind beamforming for non Gaussian signals. *IEE Proceedings-F*, 140(6):362–370, 1993.

D. Cartwright and B. Sturmfels. The number of eigenvalues of a tensor. *Linear Algebra Appl.*, 438(2):942–952, 2013.

R. B. Cattell. Parallel proportional profiles and other principles for determining the choice of factors by rotation. *Psychometrika*, 9(4):267–283, 1944.

J. T. Chang. Full reconstruction of Markov models on evolutionary trees: Identifiability and consistency. *Mathematical Biosciences*, 137:51–73, 1996.

K. Chaudhuri and S. Rao. Learning mixtures of product distributions using correlations and independence. In *Twenty-First Annual Conference on Learning Theory*, pages 9–20, 2008.

S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. Spectral learning of latent-variable PCFGs. In *Fiftieth Annual Meeting of the Association for Computational Linguistics*, 2012.

P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3): 287–314, 1994.

P. Comon and C. Jutten. *Handbook of Blind Source Separation: Independent Component Analysis and Applications*. Academic Press. Elsevier, 2010.

P. Comon, G. Golub, L.-H. Lim, and B. Mourrain. Symmetric tensors and symmetric tensor rank. *SIAM Journal on Matrix Analysis Appl.*, 30(3):1254–1279, 2008.

R. M. Corless, P. M. Gianni, and B. M. Trager. A reordered Schur factorization method for zero-dimensional polynomial systems with multiple roots. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, pages 133–140. ACM, 1997.

S. Dasgupta. Learning mixtures of Gaussians. In *Fortieth Annual IEEE Symposium on Foundations of Computer Science*, pages 634–644, 1999.

S. Dasgupta and L. Schulman. A probabilistic analysis of EM for mixtures of separated, spherical Gaussians. *Journal of Machine Learning Research*, 8(Feb):203–226, 2007.

L. De Lathauwer, J. Castaing, and J.-F. Cardoso. Fourth-order cumulant-based blind identification of underdetermined mixtures. *Signal Processing, IEEE Transactions on*, 55(6):2965–2973, 2007.

N. Delfosse and P. Loubaton. Adaptive blind separation of independent sources: a deflation approach. *Signal processing*, 45(1):59–83, 1995.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. Ser. B*, 39:1–38, 1977.

P. Dhillon, J. Rodu, M. Collins, D. P. Foster, and L. Ungar. Spectral dependency parsing with latent variables. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012.

M. Drton, B. Sturmfels, and S. Sullivant. Algebraic factor analysis: tetrads, pentads and beyond. *Probability Theory and Related Fields*, 138(3):463–493, 2007.

A. T. Erdogan. On the convergence of ICA algorithms with symmetric orthogonalization. *IEEE Transactions on Signal Processing*, 57:2209–2221, 2009.

A. M. Frieze, M. Jerrum, and R. Kannan. Learning linear transformations. In *Thirty-Seventh Annual Symposium on Foundations of Computer Science*, pages 359–368, 1996.

G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2), 2011.

R. Harshman. Foundations of the PARAFAC procedure: model and conditions for an 'explanatory' multi-mode factor analysis. Technical report, UCLA Working Papers in Phonetics, 1970.

C. J. Hillar and L.-H. Lim. Most tensor problems are NP-hard. *J. ACM*, 60(6):45:1–45:39, November 2013. ISSN 0004-5411. doi: 10.1145/2512329.

F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6:164–189, 1927a.

F. L. Hitchcock. Multiple invariants and generalized rank of a p-way matrix or tensor. *Journal of Mathematics and Physics*, 7:39–79, 1927b.

D. Hsu and S. M. Kakade. Learning mixtures of spherical Gaussians: moment methods and spectral decompositions. In *Fourth Innovations in Theoretical Computer Science*, 2013.

D. Hsu, S. M. Kakade, and P. Liang. Identifiability and unmixing of latent parse trees. In *Advances in Neural Information Processing Systems 25*, 2012a.

D. Hsu, S. M. Kakade, and T. Zhang. A spectral algorithm for learning hidden Markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480, 2012b.

A. Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *Neural Networks, IEEE Transactions on*, 10(3):626–634, 1999.

A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4–5):411–430, 2000.

H. Jaeger. Observable operator models for discrete stochastic time series. *Neural Comput.*, 12(6), 2000.

A. T. Kalai, A. Moitra, and G. Valiant. Efficiently learning mixtures of two Gaussians. In *Forty-second ACM Symposium on Theory of Computing*, pages 553–562, 2010.

R. Kannan, H. Salmasian, and S. Vempala. The spectral method for general mixture models. *SIAM Journal on Computing*, 38(3):1141–1156, 2008.

E. Kofidis and P. A. Regalia. On the best rank-1 approximation of higher-order supersymmetric tensors. *SIAM Journal on Matrix Analysis and Applications*, 23(3):863–884, 2002.

T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51 (3):455, 2009.

T. G. Kolda and J. R. Mayo. Shifted power method for computing tensor eigenpairs. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1095–1124, October 2011.

J. B. Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra and Appl.*, 18(2): 95–138, 1977.

L. D. Lathauwer, B. D. Moor, and J. Vandewalle. On the best rank-1 and rank-$(R_1, R_2, ..., R_n)$ approximation and applications of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, 2000.

L. Le Cam. *Asymptotic Methods in Statistical Decision Theory*. Springer, 1986.

L.-H. Lim. Singular values and eigenvalues of tensors: a variational approach. *Proceedings of the IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, 1:129–132, 2005.

M. Littman, R. Sutton, and S. Singh. Predictive representations of state. In *Advances in Neural Information Processing Systems 14*, pages 1555–1561, 2001.

F. M. Luque, A. Quattoni, B. Balle, and X. Carreras. Spectral learning for non-deterministic dependency parsing. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2012.

J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

P. McCullagh. *Tensor Methods in Statistics*. Chapman and Hall, 1987.

A. Moitra and G. Valiant. Settling the polynomial learnability of mixtures of Gaussians. In *Fifty-First Annual IEEE Symposium on Foundations of Computer Science*, pages 93–102, 2010.

E. Mossel and S. Roch. Learning nonsingular phylogenies and hidden Markov models. *Annals of Applied Probability*, 16(2):583–614, 2006.

P. Q. Nguyen and O. Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *Journal of Cryptology*, 22(2):139–160, 2009.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

P. V. Overschee and B. D. Moor. *Subspace Identification of Linear Systems*. Kluwer Academic Publishers, 1996.

L. Pachter and B. Sturmfels. *Algebraic Statistics for Computational Biology*, volume 13. Cambridge University Press, 2005.

A. Parikh, L. Song, and E. P. Xing. A spectral algorithm for latent tree graphical models. In *Twenty-Eighth International Conference on Machine Learning*, 2011.

K. Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society, London, A.*, page 71, 1894.

L. Qi. Eigenvalues of a real supersymmetric tensor. *Journal of Symbolic Computation*, 40 (6):1302–1324, 2005.

R. A. Redner and H. F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–239, 1984.

P. A. Regalia and E. Kofidis. Monotonic convergence of fixed-point algorithms for ICA. *IEEE Transactions on Neural Networks*, 14:943–949, 2003.

S. Roch. A short proof that phylogenetic tree reconstruction by maximum likelihood is hard. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 3(1), 2006.

J. Rodu, D. P. Foster, W. Wu, and L. H. Ungar. Using regression for spectral estimation of HMMs. In *Statistical Language and Speech Processing*, pages 212–223, 2013.

M. P. Schützenberger. On the definition of a family of automata. *Inf. Control*, 4:245–270, 1961.

S. M. Siddiqi, B. Boots, and G. J. Gordon. Reduced-rank hidden Markov models. In *Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.

D. A. Spielman and S. H. Teng. Smoothed analysis: An attempt to explain the behavior of algorithms in practice. *Communications of the ACM*, pages 76–84, 2009.

A. Stegeman and P. Comon. Subtracting a best rank-1 approximation may increase tensor rank. *Linear Algebra and Its Applications*, 433:1276–1300, 2010.

B. Sturmfels and P. Zwiernik. Binary cumulant varieties. *Ann. Comb.*, (17):229–250, 2013.

S. Vempala and G. Wang. A spectral algorithm for learning mixtures models. *Journal of Computer and System Sciences*, 68(4):841–860, 2004.

P. Wedin. Perturbation bounds in connection with singular value decomposition. *BIT Numerical Mathematics*, 12(1):99–111, 1972.

T. Zhang and G. Golub. Rank-one approximation to high order tensors. *SIAM Journal on Matrix Analysis and Applications*, 23:534–550, 2001.

A. Ziehe, P. Laskov, G. Nolte, and K. R. Müller. A fast algorithm for joint diagonalization with non-orthogonal transformations and its application to blind source separation. *Journal of Machine Learning Research*, 5:777–800, 2004.

J. Zou, D. Hsu, D. Parkes, and R. P. Adams. Contrastive learning using spectral methods. In *Advances in Neural Information Processing Systems 26*, 2013.

Journal of Machine Learning Research 15 (2014) 2833-2868 Submitted 4/13; Revised 4/14; Published 10/14

Bayesian Entropy Estimation for Countable Discrete Distributions

Evan Archer* EARCHER@UTEXAS.EDU
Center for Perceptual Systems
The University of Texas at Austin, Austin, TX 78712, USA

Max Planck Institute for Biological Cybernetics
Spemannstrasse 41
72076 Tübingen, Germany

Il Memming Park* MEMMING@AUSTIN.UTEXAS.EDU
Center for Perceptual Systems
The University of Texas at Austin, Austin, TX 78712, USA

Jonathan W. Pillow PILLOW@UTEXAS.EDU
Department of Psychology, Section of Neurobiology,
Division of Statistics and Scientific Computation, and Center for Perceptual Systems
The University of Texas at Austin, Austin, TX 78712, USA

Editor: Lawrence Carin

Abstract

We consider the problem of estimating Shannon's entropy H from discrete data, in cases where the number of possible symbols is unknown or even countably infinite. The Pitman-Yor process, a generalization of Dirichlet process, provides a tractable prior distribution over the space of countably infinite discrete distributions, and has found major applications in Bayesian non-parametric statistics and machine learning. Here we show that it provides a natural family of priors for Bayesian entropy estimation, due to the fact that moments of the induced posterior distribution over H can be computed analytically. We derive formulas for the posterior mean (Bayes' least squares estimate) and variance under Dirichlet and Pitman-Yor process priors. Moreover, we show that a fixed Dirichlet or Pitman-Yor process prior implies a narrow prior distribution over H, meaning the prior strongly determines the entropy estimate in the under-sampled regime. We derive a family of continuous measures for mixing Pitman-Yor processes to produce an approximately flat prior over H. We show that the resulting "Pitman-Yor Mixture" (PYM) entropy estimator is consistent for a large class of distributions. Finally, we explore the theoretical properties of the resulting estimator, and show that it performs well both in simulation and in application to real data.

Keywords: entropy, information theory, Bayesian estimation, Bayesian nonparametrics, Dirichlet process, Pitman–Yor process, neural coding

1. Introduction

Shannon's discrete entropy appears as an important quantity in many fields, from probability theory to engineering, ecology, and neuroscience. While entropy may be best known for its role in information theory, the practical problem of estimating entropy from samples arises in many applied settings. For example, entropy provides an important tool for quantifying the information carried by neural signals, and there is an extensive literature in neuroscience devoted to estimating the entropy of neural spike trains (Strong et al., 1998; Barbieri et al., 2004; Shlens et al., 2007; Rolls et al., 1999;

∗. EA and IP contributed equally.

Knudson and Pillow, 2013). Entropy is also used for estimating dependency structure and inferring causal relations in statistics and machine learning (Chow and Liu, 1968; Schindler et al., 2007), as well as in molecular biology (Hausser and Strimmer, 2009). Entropy also arises in the study of complexity and dynamics in physics (Letellier, 2006), and as a measure of diversity in ecology (Chao and Shen, 2003) and genetics (Farach et al., 1995).

In these settings, researchers are confronted with data arising from an unknown discrete distribution, and seek to estimate its entropy. One reason for estimating the entropy, as opposed to estimating the full distribution, is that it may be infeasible to collect enough data to estimate the full distribution reliably. The problem is not just that we may not have enough data to estimate the probability of an event accurately. In the so-called "undersampled regime" we may not even observe all events that have non-zero probability. In general, estimating a distribution in this setting is a hopeless endeavor. Estimating the entropy, by contrast, is much easier. In fact, in many cases, entropy can be accurately estimated with fewer samples than the number of distinct

Nonetheless, entropy estimation remains a difficult problem. There is no unbiased estimator for entropy, and the maximum likelihood estimator is severely biased for small data sets (Paninski, 2003). Many previous studies have taken a frequentist approach and focused on methods for computing and reducing this bias (Miller, 1955; Panzeri and Treves, 1996; Strong et al., 1998; Paninski, 2003; Grassberger, 2008). Here, we instead take a Bayesian approach to entropy estimation, building upon an approach introduced by Nemenman and colleagues (Nemenman et al., 2002). Our basic strategy is to place a prior over the space of discrete probability distributions and then perform inference using the induced posterior distribution over entropy. Figure 1 shows a graphical model illustrating the dependencies between the basic quantities of interest.

When there are few samples relative to the total number of symbols, entropy estimation is especially difficult. We refer to this informally as the "under-sampled" regime. In this regime, it is common for many symbols with non-zero probability to remain unobserved, and often we can only bound or estimate the *support* of the distribution (i.e., the number of symbols with non-zero probability). Previous Bayesian approaches to entropy estimation (Nemenman et al., 2002) required *a priori* knowledge of the support. Here we overcome this limitation by formulating a prior over the space of countably-infinite discrete distributions. As we will show, the resulting estimator is consistent even when the support of the true distribution is finite.

Our approach relies on Pitman-Yor process (PYP), a two-parameter generalization of the Dirichlet process (DP) (Pitman and Yor, 1997; Ishwaran and James, 2003; Goldwater et al., 2006), which provides a prior distribution over the space of countably infinite discrete distributions. The PYP provides an attractive family of priors in this setting because: (1) the induced posterior distribution over entropy given data has analytically tractable moments; and (2) distributions sampled from a PYP can exhibit power-law tails, a feature commonly observed in data from social, biological and physical systems (Zipf, 1949; Dudok de Wit, 1999; Newman, 2005).

However, we show that a PYP prior with fixed hyperparameters imposes a narrow prior distribution over entropy, leading to severe bias and overly narrow posterior credible intervals given a small data set. Our approach, inspired by Nemenman and colleagues (Nemenman et al., 2002), is to introduce a family of mixing measures over Pitman-Yor processes such that the resulting Pitman-Yor Mixture (PYM) prior provides an approximately non-informative (i.e., flat) prior over entropy.

The remainder of the paper is organized as follows. In Section 2, we introduce the entropy estimation problem and review prior work. In Section 3, we introduce the Dirichlet and Pitman-Yor processes and discuss key mathematical properties relating to entropy. In Section 4, we introduce a novel entropy estimator based on PYM priors and derive several of its theoretical properties. In Section 5, we compare various estimators with applications to data.

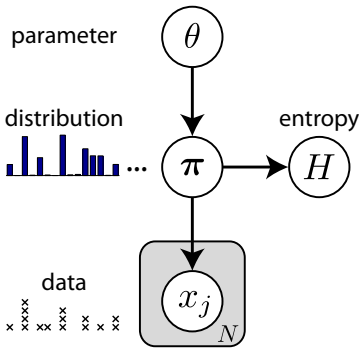


Figure 1: Graphical model illustrating the ingredients for Bayesian entropy estimation. Arrows indicate conditional dependencies between variables, and the gray "plate" denotes multiple copies of a random variable (with the number of copies N indicated at bottom). For entropy estimation, the joint probability distribution over entropy H, data $\mathbf{x} = \{x_j\}$, discrete distribution $\boldsymbol{\pi} = \{\pi_i\}$, and parameter θ factorizes as: $p(H, \mathbf{x}, \boldsymbol{\pi}, \theta) = p(H|\boldsymbol{\pi})p(\mathbf{x}|\boldsymbol{\pi})p(\boldsymbol{\pi}|\theta)p(\theta)$. Entropy is a deterministic function of $\boldsymbol{\pi}$, so $p(H|\boldsymbol{\pi}) = \delta(H - \sum_i \pi_i \log \pi_i)$. The Bayes least squares estimator corresponds to the posterior mean: $\mathbb{E}[H|\mathbf{x}] = \iint p(H|\boldsymbol{\pi})p(\boldsymbol{\pi}, \theta|\mathbf{x}) d\boldsymbol{\pi}\, d\theta$.

2. Entropy Estimation

Consider samples $\mathbf{x} := \{x_j\}_{j=1}^N$ drawn *iid* from an unknown discrete distribution $\boldsymbol{\pi} := \{\pi_i\}_{i=1}^{\mathcal{A}}$, $p(x_j = i) = \pi_i$, on a finite or (countably) infinite alphabet $\mathcal{X}$ with cardinality $\mathcal{A}$. We wish to estimate the entropy of $\boldsymbol{\pi}$

$$H(\boldsymbol{\pi}) = -\sum_{i=1}^{\mathcal{A}} \pi_i \log \pi_i. \tag{1}$$

We are interested in the so-called "under-sampled regime," $N \ll \mathcal{A}$, where many of the symbols remain unobserved. We will see that a naive approach to entropy estimation in this regime results in severely biased estimators and briefly review approaches for correcting this bias. We then consider Bayesian techniques for entropy estimation in general before introducing the Nemenman–Shafee–Bialek (NSB) method upon which the remainder of the article will build.

2.1 Plugin Estimator and Bias-Correction Methods

Perhaps the most straightforward entropy estimation technique is to estimate the distribution $\boldsymbol{\pi}$ and then use the plugin formula (1) to evaluate its entropy. The empirical distribution $\hat{\boldsymbol{\pi}} = (\hat{\pi}_1, \ldots, \hat{\pi}_{\mathcal{A}})$ is computed by normalizing the observed counts $\mathbf{n} := (n_1, \ldots, n_{\mathcal{A}})$ of each symbol

$$\hat{\pi}_k = n_k/N, \qquad n_k = \sum_{i=1}^{N} \mathbf{1}_{\{x_i = k\}}, \tag{2}$$

for each $k \in \mathcal{X}$. Plugging this estimate for $\boldsymbol{\pi}$ into (1), we obtain the so-called "plugin" estimator

$$\hat{H}_{\text{plugin}} = -\sum \hat{\pi}_i \log \hat{\pi}_i, \tag{3}$$

which is also the maximum-likelihood estimator under categorical (or multinomial) likelihood.

Despite its simplicity and desirable asymptotic properties, $\hat{H}_{\text{plugin}}$ exhibits substantial negative bias in the under-sampled regime. There exists a large literature on methods for removing this bias,

much of which considers the setting in which $\mathcal{A}$ is known and finite. One popular and well-studied method involves taking a series expansion of the bias (Miller, 1955; Treves and Panzeri, 1995; Panzeri and Treves, 1996; Grassberger, 2008) and then subtracting it from the plugin estimate. Other recent proposals include minimizing an upper bound over a class of linear estimators (Paninski, 2003), and a James-Stein estimator (Hausser and Strimmer, 2009). Recently, Wolpert and colleagues have considered entropy estimation in the case of unknown alphabet size (Wolpert and DeDeo, 2013). In that paper, the authors infer entropy under a (finite) Dirichlet prior, but treat the alphabet size itself as a random variable that can be either inferred from the data or integrated out.

Other recent work considers countably infinite alphabets. The coverage-adjusted estimator (CAE) (Chao and Shen, 2003; Vu et al., 2007) addresses bias by combining the Horvitz-Thompson estimator with a nonparametric estimate of the proportion of total probability mass (the "coverage") accounted for by the observed data $\mathbf{x}$. In a similar spirit, Zhang proposed an estimator based on the Good-Turing estimate of population size (Zhang, 2012).

2.2 Bayesian Entropy Estimation

The Bayesian approach to entropy estimation involves formulating a prior over distributions $\boldsymbol{\pi}$, and then turning the crank of Bayesian inference to infer H using the posterior distribution. Bayes' least squares (BLS) estimators take the form

$$\hat{H}(\mathbf{x}) = \mathbb{E}[H|\mathbf{x}] = \int H(\boldsymbol{\pi}) p(H|\boldsymbol{\pi}) p(\boldsymbol{\pi}|\mathbf{x}) \, \mathrm{d}\boldsymbol{\pi},$$

where $p(\boldsymbol{\pi}|\mathbf{x})$ is the posterior over $\boldsymbol{\pi}$ under some prior $p(\boldsymbol{\pi})$ and discrete likelihood $p(\mathbf{x}|\boldsymbol{\pi})$, and

$$p(H|\boldsymbol{\pi}) = \delta(H + \sum_i \pi_i \log \pi_i),$$

since H is deterministically related to $\boldsymbol{\pi}$. To the extent that $p(\boldsymbol{\pi})$ expresses our true prior uncertainty over the unknown distribution that generated the data, this estimate is optimal (in a least-squares sense), and the corresponding credible intervals capture our uncertainty about H given the data.

For distributions with known finite alphabet size $\mathcal{A}$, the Dirichlet distribution provides an obvious choice of prior due to its conjugacy with the categorical distribution. It takes the form

$$p_{Dir}(\boldsymbol{\pi}) \propto \prod_{i=1}^{\mathcal{A}} \pi_i^{a-1},$$

for $\boldsymbol{\pi}$ on the $\mathcal{A}$-dimensional simplex ($\pi_i \geq 1$, $\sum \pi_i = 1$), where $a > 0$ is a "concentration" parameter (Hutter, 2002). Many previously proposed estimators can be viewed as Bayesian under a Dirichlet prior with particular fixed choice of a (Hausser and Strimmer, 2009).

2.3 Nemenman-Shafee-Bialek (NSB) Estimator

In a seminal paper, Nemenman and colleagues showed that for finite distributions with known $\mathcal{A}$, Dirichlet priors with fixed a impose a narrow prior distribution over entropy (Nemenman et al., 2002). In the under-sampled regime, Bayesian estimates based on such highly informative priors are essentially determined by the value of a. Moreover, they have undesirably narrow posterior credible intervals, reflecting narrow prior uncertainty rather than strong evidence from the data. (These estimators generally give incorrect answers with high confidence!). To address this problem, Nemenman and colleagues suggested a mixture-of-Dirichlets prior

$$p(\boldsymbol{\pi}) = \int p_{\mathrm{Dir}}(\boldsymbol{\pi}|a) p(a) \, \mathrm{d}a, \tag{4}$$

where $p_{\mathrm{Dir}}(\boldsymbol{\pi}|a)$ denotes a $\mathrm{Dir}(a)$ prior on $\boldsymbol{\pi}$, and $p(a)$ denotes a set of mixing weights, given by

$$p(a) \propto \frac{d}{da}\mathbb{E}[H|a] = \mathcal{A}\psi_1(\mathcal{A}a+1) - \psi_1(a+1), \tag{5}$$

where $\mathbb{E}[H|a]$ denotes the expected value of H under a $\mathrm{Dir}(a)$ prior, and $\psi_1(\cdot)$ denotes the tri-gamma function. To the extent that $p(H|a)$ resembles a delta function, (4) and (5) imply a uniform prior for H on $[0, \log \mathcal{A}]$. The BLS estimator under the NSB prior can be written

$$\begin{aligned}\hat{H}_{nsb} = \mathbb{E}[H|\mathbf{x}] &= \iint H(\boldsymbol{\pi})p(\boldsymbol{\pi}|\mathbf{x},a)\,p(a|\mathbf{x})\,\mathrm{d}\boldsymbol{\pi}\,\mathrm{d}a \\ &= \int \mathbb{E}[H|\mathbf{x},a]\frac{p(\mathbf{x}|a)p(a)}{p(\mathbf{x})}\,\mathrm{d}a,\end{aligned}$$

where $E[H|\mathbf{x},a]$ is the posterior mean under a $\mathrm{Dir}(a)$ prior, and $p(\mathbf{x}|a)$ denotes the evidence, which has a Pólya distribution (Minka, 2003)

$$\begin{aligned}p(\mathbf{x}|a) &= \int p(\mathbf{x}|\boldsymbol{\pi})p(\boldsymbol{\pi}|a)\,\mathrm{d}\boldsymbol{\pi} \\ &= \frac{(N!)\Gamma(\mathcal{A}a)}{\Gamma(a)^{\mathcal{A}}\Gamma(N+\mathcal{A}a)}\prod_{i=1}^{\mathcal{A}}\frac{\Gamma(n_i+a)}{n_i!}.\end{aligned}$$

The NSB estimate $\hat{H}_{nsb}$ and its posterior variance are easily computable via 1D numerical integration in a using closed-form expressions for the first two moments of the posterior distribution of H given a. The forms for these moments are discussed in previous work (Wolpert and Wolf, 1995; Nemenman et al., 2002), but the full formulae have to our knowledge never been explicitly shown. Here we state the results,

$$\mathbb{E}[H|\mathbf{x},a] = \psi_0(\tilde{N}+1) - \sum_i \frac{\tilde{n}_i}{\tilde{N}}\psi_0(\tilde{n}_i+1) \tag{6}$$

$$\mathbb{E}[H^2|\mathbf{x},a] = \sum_{i\neq k}\frac{\tilde{n}_i\tilde{n}_k}{(\tilde{N}+1)\tilde{N}}I_{i,k} + \sum_i \frac{(\tilde{n}_i+1)\tilde{n}_i}{(\tilde{N}+1)\tilde{N}}J_i \tag{7}$$

$$\begin{aligned}I_{i,k} &= \Big(\psi_0(\tilde{n}_k+1) - \psi_0(\tilde{N}+2)\Big)\Big(\psi_0(\tilde{n}_i+1) - \psi_0(\tilde{N}+2)\Big) - \psi_1(\tilde{N}+2) \\ J_i &= (\psi_0(\tilde{n}_i+2) - \psi_0(\tilde{N}+2))^2 + \psi_1(\tilde{n}_i+2) - \psi_1(\tilde{N}+2),\end{aligned}$$

where $\tilde{n}_i = n_i + a$ are counts plus prior "pseudocount" a, $\tilde{N} = \sum \tilde{n}_i$ is the total of counts plus pseudocounts, and ψ_n is the polygamma of n-th order (i.e., ψ_0 is the digamma function). Finally, $\mathrm{var}[H|\mathbf{n},a] = \mathbb{E}[H^2|\mathbf{n},a] - \mathbb{E}[H|\mathbf{n},a]^2$. We derive these formulae in the Appendix, and in addition provide an alternative derivation using a size-biased sampling formulae discussed in Section 3.

2.4 Asymptotic NSB Estimator

Nemenman and colleagues have proposed an extension of the NSB estimator to countably infinite distributions (or distributions with unknown cardinality), using a zeroth order approximation to $\hat{H}_{nsb}$ in the limit $\mathcal{A} \to \infty$ which we refer to as asymptotic-NSB (ANSB) (Nemenman et al., 2004; Nemenman, 2011),

$$\hat{H}_{ansb} = 2\log(N) + \psi_0(N-K) - \psi_0(1) - \log(2), \tag{8}$$

where K is the number of distinct symbols in the sample. Note that the ANSB estimator is designed specifically for an extremely under-sampled regime ($K \sim N$), which we refer to as the "ANSB approximation regime". The fact that ANSB diverges with N in the well-sampled regime (Vu et al.,

2007) is therefore consistent with its design. In our experiments with ANSB in subsequent sections, we follow the work of Nemenman (2011) to define the ANSB approximation regime to be that region such that $E[K_N]/N > 0.9$, where K_N is the number of unique symbols appearing in a sample of size N.

3. Dirichlet and Pitman-Yor Process Priors

To construct a prior over unknown or countably-infinite discrete distributions, we borrow tools from nonparametric Bayesian statistics. The Dirichlet Process (DP) and Pitman-Yor process (PYP) define stochastic processes whose samples are countably infinite discrete distributions (Ferguson, 1973; Pitman and Yor, 1997). A sample from a DP or PYP may be written as $\sum_{i=1}^{\infty} \pi_i \delta_{\phi_i}$, where now $\boldsymbol{\pi} = \{\pi_i\}$ denotes a countably infinite set of 'weights' on a set of atoms $\{\phi_i\}$ drawn from some base probability measure, where δ_{ϕ_i} is a delta function on the atom ϕ_i.[1] We use DP and PYP to define a prior distribution on the infinite-dimensional simplex. The prior distribution over $\boldsymbol{\pi}$ under the DP or PYP is technically called the GEM[2] distribution or the two-parameter Poisson-Dirichlet distribution, but we will abuse terminology by referring to both the process and its associated weight distribution by the same symbol, DP or PY (Ishwaran and Zarepour, 2002).

The DP distribution over $\boldsymbol{\pi}$ results from a limit of the (finite) Dirichlet distribution where alphabet size grows and concentration parameter shrinks: $\mathcal{A} \to \infty$ and $a \to 0$ s.t. $a\mathcal{A} \to \alpha$. The PYP distribution over $\boldsymbol{\pi}$ generalizes the DP to allow power-law tails, and includes DP as a special case (Kingman, 1975; Pitman and Yor, 1997). For $\mathrm{PY}(d, \alpha)$ with $d \neq 0$, the tails approximately follow a power-law: $\pi_i \propto (i)^{-\frac{1}{d}}$ (pp. 867, Pitman and Yor (1997)).[3] Many natural phenomena such as city size, language, spike responses, etc., also exhibit power-law tails (Zipf, 1949; Newman, 2005). Figure 2 shows two such examples, along with a sample drawn from the best-fitting DP and PYP distributions.

Let $\mathrm{PY}(d, \alpha)$ denote the PYP with *discount* parameter d and *concentration* parameter α (also called the "Dirichlet parameter"), for $d \in [0, 1), \alpha > -d$. When $d = 0$, this reduces to the Dirichlet process, $\mathrm{DP}(\alpha)$. To gain intuition for the DP and PYP, it is useful to consider typical samples $\boldsymbol{\pi}$ with weights $\{\pi_i\}$ sorted in decreasing order of probability, so that $\pi_{(1)} > \pi_{(2)} > \cdots$. The concentration parameter α controls how much of the probability mass is concentrated in the first few samples, that is, in the head instead of the tail of the sorted distribution. For small α the first few weights carry most of the probability mass whereas, for large α, the probability mass is more spread out so that $\boldsymbol{\pi}$ is more uniform. As noted above the discount parameter d controls the shape of the tail. Larger d gives heavier power-law tails, while $d = 0$ yields exponential tails.

We can draw samples $\boldsymbol{\pi} \sim \mathrm{PY}(d, \alpha)$ using an infinite sequence of independent Beta random variables in a process known as "stick-breaking" (Ishwaran and James, 2001)

$$\beta_i \sim \mathrm{Beta}(1 - d, \alpha + id), \qquad \tilde{\pi}_i = \prod_{j=1}^{i-1} (1 - \beta_j)\beta_i, \tag{9}$$

where $\tilde{\pi}_i$ is known as the i'th *size-biased permutation* from $\boldsymbol{\pi}$ (Pitman, 1996). The $\tilde{\pi}_i$ sampled in this manner are not strictly decreasing, but decrease on average such that $\sum_{i=1}^{\infty} \tilde{\pi}_i = 1$ with probability 1 (Pitman and Yor, 1997).

1. Here, we will assume the base measure is non-atomic, so that the atoms ϕ_i's are distinct with probability one. This allows us to ignore the base measure, making entropy of the distribution equal to the entropy of the weights $\boldsymbol{\pi}$.
2. GEM stands for "Griffiths, Engen and McCloskey", after three researchers who considered these ideas early on (Ewens, 1990).
3. The power-law exponent has been given incorrectly in previous work (Goldwater et al., 2006; Teh, 2006).

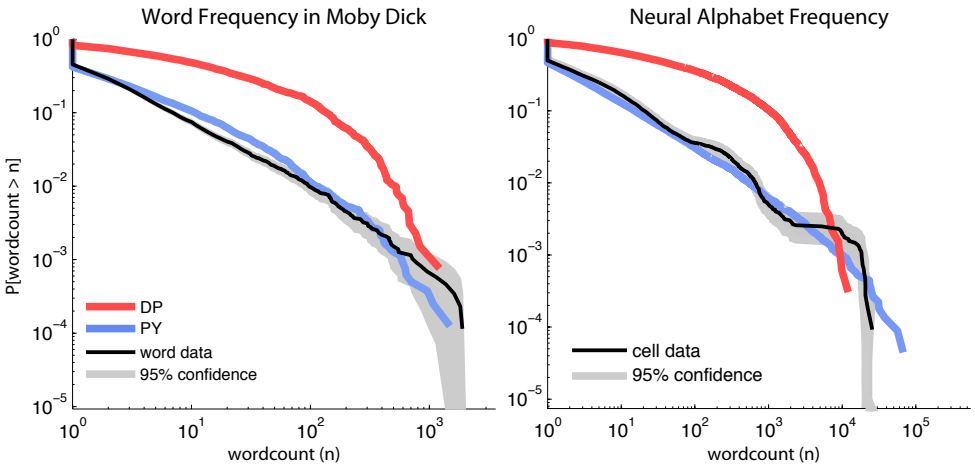


Figure 2: Empirical cumulative distribution functions of words in natural language (**left**) and neural spike patterns (**right**). We compare samples from the DP (red) and PYP (blue) priors for two data sets with heavy tails (black). In both cases, we compare the empirical CDF estimated from data to distributions drawn from DP and PYP using the ML values of α and (d, α) respectively. For both data sets, PYP better captures the heavy-tailed behavior of the data. **(left)** Frequency of $N = 217826$ words in the novel Moby Dick by Herman Melville. **(right)** Frequencies among $N = 1.2 \times 10^6$ neural spike words from 27 simultaneously-recorded retinal ganglion cells, binarized and binned at 10ms.

3.1 Expectations over DP and PYP Priors

For our purposes, a key virtue of PYP priors is a mathematical property called *invariance under size-biased sampling*. This property allows us to convert expectations over $\boldsymbol{\pi}$ on the infinite-dimensional simplex (which are required for computing the mean and variance of H given data) into one- or two-dimensional integrals with respect to the distribution of the first two size-biased samples (Perman et al., 1992; Pitman, 1996).

Proposition 1 (Expectations with first two size-biased samples) *For $\boldsymbol{\pi} \sim PY(d, \alpha)$,*

$$\mathbb{E}_{(\boldsymbol{\pi}|d,\alpha)}\left[\sum_{i=1}^{\infty} f(\pi_i)\right] = \mathbb{E}_{(\tilde{\pi}_1|d,\alpha)}\left[\frac{f(\tilde{\pi}_1)}{\tilde{\pi}_1}\right], \tag{10}$$

$$\mathbb{E}_{(\boldsymbol{\pi}|d,\alpha)}\left[\sum_{i,j\neq i} g(\pi_i, \pi_j)\right] = \mathbb{E}_{(\tilde{\pi}_1,\tilde{\pi}_2|d,\alpha)}\left[\frac{g(\tilde{\pi}_1, \tilde{\pi}_2)}{\tilde{\pi}_1\tilde{\pi}_2}(1 - \tilde{\pi}_1)\right], \tag{11}$$

where $\tilde{\pi}_1$ and $\tilde{\pi}_2$ are the first two size-biased samples from $\boldsymbol{\pi}$.

The first result (10) appears in (Pitman and Yor, 1997), and an analogous proof can be constructed for (11) (see Appendix).

The direct consequence of this proposition is that the first two moments of $H(\boldsymbol{\pi})$ under the PYP and DP priors have closed forms[4]

$$\mathbb{E}[H|d, \alpha] = \psi_0(\alpha + 1) - \psi_0(1 - d), \tag{12}$$

$$\mathrm{var}[H|d, \alpha] = \frac{\alpha + d}{(\alpha + 1)^2(1 - d)} + \frac{1 - d}{\alpha + 1}\psi_1(2 - d) - \psi_1(2 + \alpha). \tag{13}$$

4. Note that (12) and (13) follow from (6) and (7), respectively, under the PY limit.

The derivation can be found in the Appendix.

3.2 Expectations over DP and PYP Posteriors

A useful property of PYP priors (for multinomial observations) is that the posterior $p(\boldsymbol{\pi}|\mathbf{x}, d, \alpha)$ takes the form of a mixture of a Dirichlet distribution (over the observed symbols) and a Pitman-Yor process (over the unobserved symbols) (Ishwaran and James, 2003). This makes the integrals over the infinite-dimensional simplex tractable and, as a result, we obtain closed-form solutions for the posterior mean and variance of H. Let K be the number of unique symbols observed in N samples, i.e., $K = \sum_{i=1}^{\mathcal{A}} \mathbf{1}_{\{n_i>0\}}$.[5] Further, let $\alpha_i = n_i - d$, $N = \sum n_i$, and $A = \sum \alpha_i = \sum_i n_i - Kd = N - Kd$. Now, following Ishwaran and colleagues (Ishwaran and Zarepour, 2002), we write the posterior as an infinite random vector $\boldsymbol{\pi}|\mathbf{x}, d, \alpha = (p_1, p_2, p_3, \ldots, p_K, p_*\boldsymbol{\pi}')$, where

$$\begin{aligned}(p_1, p_2, \ldots, p_K, p_*) &\sim \mathrm{Dir}(n_1 - d, \ldots, n_K - d, \alpha + Kd) \\ \boldsymbol{\pi}' := (\pi_1, \pi_2, \pi_3, \ldots) &\sim \mathrm{PY}(d, \alpha + Kd).\end{aligned} \tag{14}$$

The posterior mean $E[H|\mathbf{x}, d, \alpha]$ is given by

$$\mathbb{E}[H|\alpha, d, \mathbf{x}] = \psi_0(\alpha + N + 1) - \frac{\alpha + Kd}{\alpha + N}\psi_0(1 - d) - \frac{1}{\alpha + N}\left[\sum_{i=1}^{K}(n_i - d)\psi_0(n_i - d + 1)\right]. \tag{15}$$

The variance, $\mathrm{var}[H|\mathbf{x}, d, \alpha]$, may also be expressed in an easily-computable closed-form. As we discuss in detail in Appendix A.4, $\mathrm{var}[H|\mathbf{x}, d, \alpha]$ may be expressed in terms of the first two moments of p_*, $\boldsymbol{\pi}$, and $\mathbf{p} = (p_1, \ldots, p_K)$ appearing in the posterior (14). Applying the law of total variance and using the independence properties of the posterior, we find

$$\begin{aligned}\mathrm{var}[H|d, \alpha] = \mathbb{E}_{p_*}[(1 - p_*)^2] \operatorname*{var}_{\mathbf{p}}[H(\mathbf{p})] + \mathbb{E}_{p_*}[p_*^2] \operatorname*{var}_{\boldsymbol{\pi}}[H(\boldsymbol{\pi})] \\ + \mathbb{E}_{p_*}[\Omega^2(p_*)] - \mathbb{E}_{p_*}[\Omega(p_*)]^2,\end{aligned}$$

where $\Omega(p_*) = (1-p_*)\mathbb{E}_{\mathbf{p}}[H(\mathbf{p})] + p_*\mathbb{E}_{\boldsymbol{\pi}}[H(\boldsymbol{\pi})] + H(p_*)$, and $H(p_*) = -p_*\log(p_*) - (1-p_*)\log(1-p_*)$. To specify $\Omega(p_*)$, we let $\mathbf{A} = \mathbb{E}_{\mathbf{p}}[H(\mathbf{p})]$, $\mathbf{B} = \mathbb{E}_{\boldsymbol{\pi}}[H(\boldsymbol{\pi})]$ so that

$$\begin{aligned}\mathbb{E}[\Omega] &= \mathbb{E}_{p_*}[1 - p_*]\mathbb{E}_{\mathbf{p}}[H(\mathbf{p})] + \mathbb{E}_{p_*}[p_*]\mathbb{E}_{\boldsymbol{\pi}}[H(\boldsymbol{\pi})] + H(p_*), \\ \mathbb{E}[\Omega^2] &= 2\mathbb{E}_{p_*}[p_* H(p_*)][\mathbf{B} - \mathbf{A}] + 2\mathbf{A}\mathbb{E}_{p_*}[H(p_*)] + \mathbb{E}_{p_*}[h^2(p_*)] \\ &\quad + \mathbb{E}_{p_*}[p_*^2]\left[\mathbf{B}^2 - 2\mathbf{A}\mathbf{B}\right] + 2\mathbb{E}_{p_*}[p_*]\mathbf{A}\mathbf{B} + \mathbb{E}_{p_*}[(1 - p_*)^2]\mathbf{A}^2.\end{aligned}$$

4. Entropy Inference under DP and PYP priors

The posterior expectations computed in Section 3.2 provide a class of entropy estimators for distributions with countably-infinite support. For each choice of (d, α), $\mathbb{E}[H|\alpha, d, \mathbf{x}]$ is the posterior mean under a $PY(d, \alpha)$ prior, analogous to the fixed-α Dirichlet priors discussed in Section 2.2. Unfortunately, fixed $PY(d, \alpha)$ priors carry the same difficulties as fixed Dirichlet priors. A fixed-parameter $\mathrm{PY}(d, \alpha)$ prior on $\boldsymbol{\pi}$ results in a highly concentrated prior distribution on entropy (Figure 3).

We address this problem by introducing a mixture prior $p(d, \alpha)$ on $\mathrm{PY}(d, \alpha)$ under which the implied prior on entropy is flat.[6] We then define the BLS entropy estimator under this mixture prior,

5. We note that the quantity K has been studied in Bayesian nonparametrics in its own right, for instance to study species diversity in ecological applications (Favaro et al., 2009).

6. Notice, however, that by constructing a flat prior on entropy, we do not obtain an objective prior. Here, we are not interested in estimating the underlying high-dimensional probabilities $\{\pi_i\}$, but rather in estimating a single statistic. An objective prior on the model parameters is not necessarily optimal for estimating entropy: entropy is not a parameter in our model. In fact, Jeffreys' prior for multinomial observations is exactly a Dirichlet distribution with a fixed $\alpha = 1/2$. As mentioned in the text, such Bayesian priors are highly informative about the entropy.

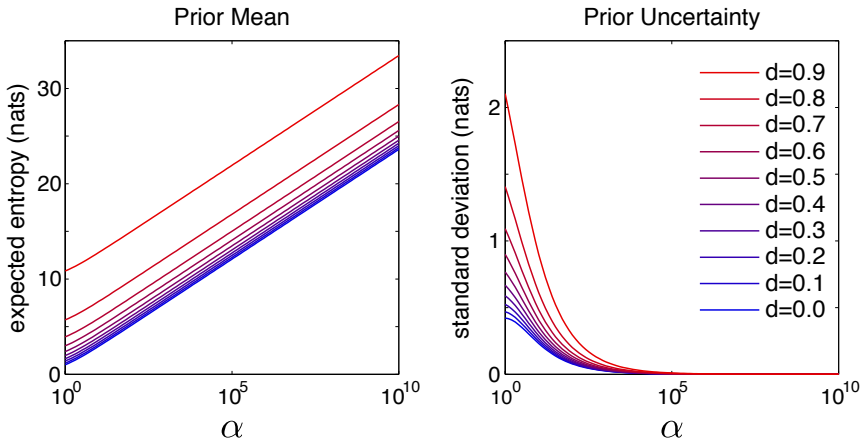


Figure 3: Prior entropy mean and variance (12) and 13 as a function of α and d. Note that entropy is approximately linear in $\log \alpha$. For large values of α, $p(H(\boldsymbol{\pi})|d, \alpha)$ is highly concentrated around the mean.

the Pitman-Yor Mixture (PYM) estimator, and discuss some of its theoretical properties. Finally, we turn to the computation of PYM, discussing methods for sampling, and numerical quadrature integration.

4.1 Pitman-Yor Process Mixture (PYM) Prior

One way of constructing a flat mixture prior is to follow the approach of Nemenman and colleagues (Nemenman et al., 2002), setting $p(d, \alpha)$ proportional to the derivative of the expected entropy (12). Unlike NSB, we have two parameters through which to control the prior expected entropy. For instance, large prior (expected) entropies can arise either from large values of α (as in the DP) or from values of d near 1 (see Figure 3A). We can explicitly control this trade-off by reparameterizing PYP as follows

$$h = \psi_0(\alpha + 1) - \psi_0(1 - d), \quad \gamma = \frac{\psi_0(1) - \psi_0(1 - d)}{\psi_0(\alpha + 1) - \psi_0(1 - d)},$$

where $h > 0$ is equal to the expected prior entropy (12), and $\gamma \in [0, \infty)$ captures prior beliefs about tail behavior (Figure 4A). For $\gamma = 0$, we have the DP (i.e., $d = 0$, giving $\boldsymbol{\pi}$ with exponential tails), while for $\gamma = 1$ we have a PY$(d, 0)$ process (i.e., $\alpha = 0$, yielding $\boldsymbol{\pi}$ with power-law tails). In the limit where $\alpha \to -1$ and $d \to 1$, $\gamma \to \infty$. Where required, the inverse transformation to standard PY parameters is given by: $\alpha = {\psi_0}^{-1}\left(h(1 - \gamma) + \psi_0(1)\right) - 1$, $d = 1 - {\psi_0}^{-1}\left(\psi_0(1) - h\gamma\right)$, where ${\psi_0}^{-1}(\cdot)$ denotes the inverse digamma function.

We can construct an approximately flat improper distribution over H on $[0, \infty]$ by setting $p(h, \gamma) = q(\gamma)$ for all h, where q is any density on $[0, \infty)$. We call this the Pitman-Yor process mixture (PYM) prior. The induced prior on entropy is thus

$$p(H) = \iint p(H|\boldsymbol{\pi}) p(\boldsymbol{\pi}|\gamma, h) p(\gamma, h) \, \mathrm{d}\gamma \, \mathrm{d}h,$$

where $p(\boldsymbol{\pi}|\gamma, h)$ denotes a PYP on $\boldsymbol{\pi}$ with parameters γ, h. We compare only three choices of $q(\gamma)$ here. However, the prior $q(\gamma)$ is not fixed but may be adapted to reflect prior beliefs about the data set at hand. A $q(\gamma)$ that places probability mass on larger γ (near 1) results in a prior that prefers heavy-tailed behavior and high entropy, whereas weight on small γ prefers exponential-tailed distributions. As a result, priors with more mass on large γ will also tend to yield wider credible

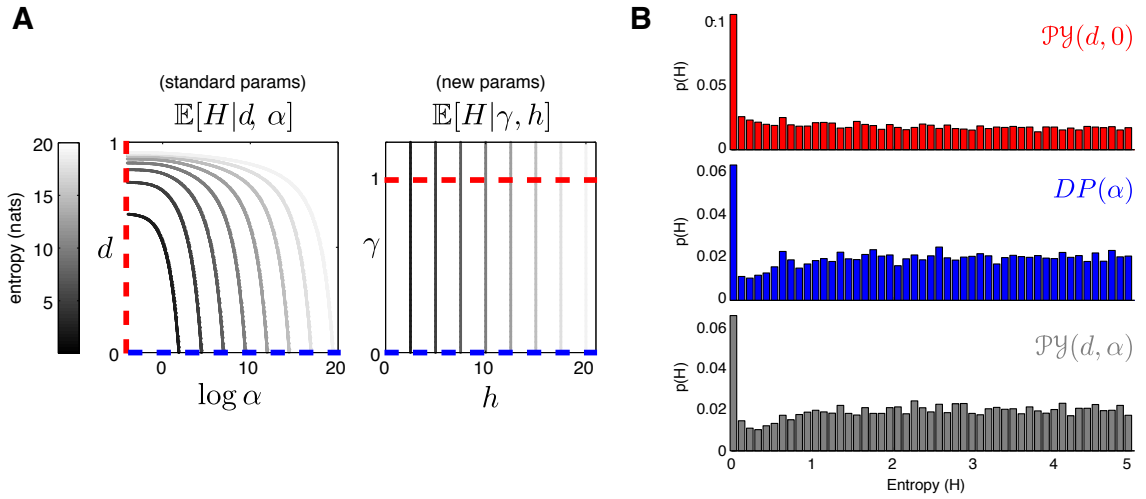


Figure 4: Prior over expected entropy under Pitman-Yor process prior. **(A)** Left: expected entropy as a function of the natural parameters (d, α). Right: expected entropy as a function of transformed parameters (h, γ). The dotted red and blue lines indicate the contours on which the PY$(d, 0)$ and DP(α) priors are defined, respectively. **(B)** Sampled prior distributions $(N = 5e3)$ over entropy implied by three different PYM priors, each with a different mixing density over α and d. We formulate each prior in the transformed parameters γ and h. We place a uniform prior on h and show three different choices of prior $q(\gamma)$. Each resulting PYM prior is a mixture of Pitman-Yor processes: PY$(d, 0)$ (red) uses a mixing density over d: $q(\gamma) = \delta_{\gamma-1}$; PY$(0, \alpha) =$ DP(α) (blue) uses a mixing density over α: $q(\gamma) = \delta_{\gamma}$; and PY$(d, \alpha)$ (grey) uses a mixture over both hyperparameters: $q(\gamma) = \exp(-\frac{10}{1-\gamma})\mathbf{1}_{\{\gamma<1\}}$. Note that for all of these examples, the "true" $p(H)$ is an improper prior supported on $[0, \infty)$. We visualize the sampled distributions only on the range from 0 to 5 nats, since sampling from PY becomes prohibitively expensive with increasing expected entropy (especially as $d \to 1$).

intervals and higher estimates of entropy. PYM mixture priors resulting from different choices of $q(\gamma)$ are all approximately flat on H, but each favors distributions with different tail behavior; the ability to select $q(\gamma)$ greatly enhances the flexibility of PYM, allowing the practitioner to adapt it to her own data.

Figure 4B shows samples from this prior under three different choices of $q(\gamma)$, for h uniform on $[0, 3]$. For the experiments, we use $q(\gamma) = \exp(-\frac{10}{1-\gamma})\mathbf{1}_{\{\gamma<1\}}$ which yields good results by weighting less on extremely heavy-tailed distributions.[7] Combined with the likelihood, the posterior $p(d, \alpha|\mathbf{x}) \propto p(\mathbf{x}|d, \alpha)p(d, \alpha)$ quickly concentrates as more data are given (see Figure 5).

4.2 The Pitman-Yor Mixture Entropy Estimator

Now that we have determined a prior on the infinite simplex, we turn to the problem of inference given observations $\mathbf{x}$. The Bayes least squares entropy estimator under the mixture prior $p(d, \alpha)$, the

7. In particular, the restriction $\gamma < 1$ omits the corner $d \to 1$ and $\alpha \to -d$. In this region, one can obtain arbitrarily large prior variance over H for a given mean. However, such priors have very heavy tails and seem poorly-suited to data with finite or exponential tails, and we therefore do not explore them further here.

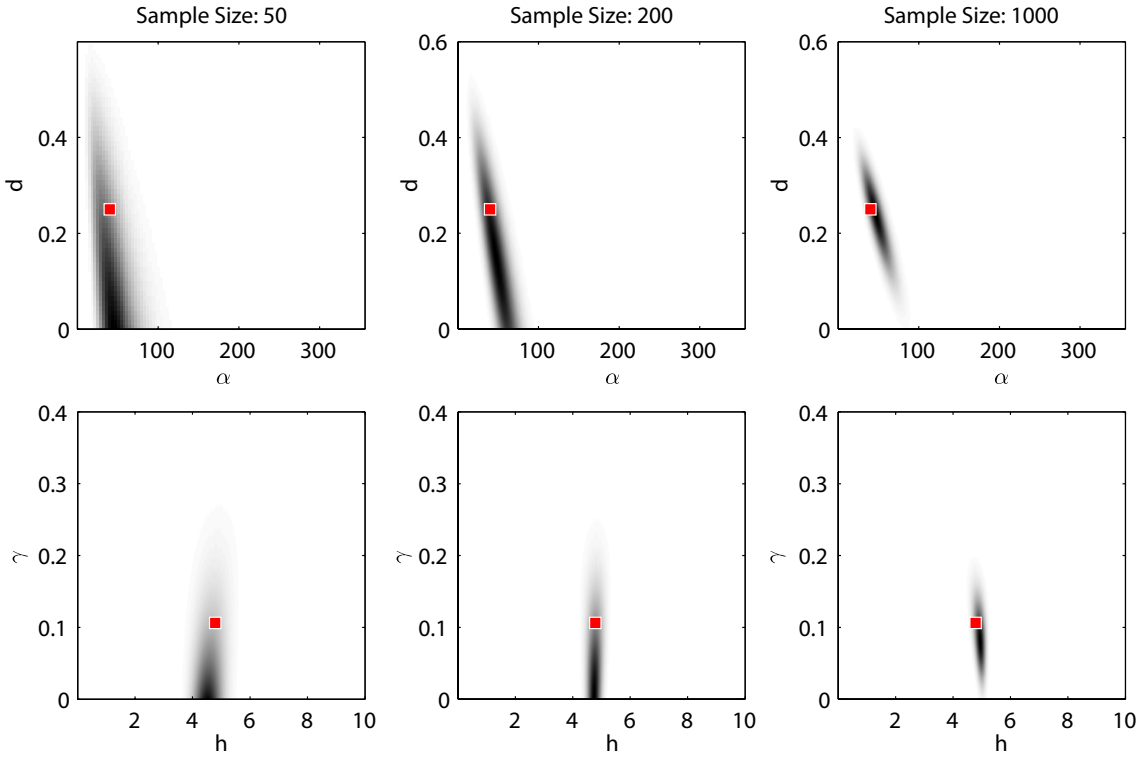


Figure 5: Convergence of $p(d, \alpha|\mathbf{x})$ for increasing sample size. Both parameterization (d, α) and (γ, h) are shown. Data are simulated from a PY$(0.25, 40)$ whose parameters are indicated by the red dot.

Pitman-Yor Mixture (PYM) estimator, takes the form

$$\hat{H}_{\mathrm{PYM}} = \mathbb{E}[H|\mathbf{x}] = \int \mathbb{E}[H|\mathbf{x}, d, \alpha] \frac{p(\mathbf{x}|d, \alpha)p(d, \alpha)}{p(\mathbf{x})} \, \mathrm{d}(d, \alpha), \tag{16}$$

where $\mathbb{E}[H|\mathbf{x}, d, \alpha]$ is the expected posterior entropy for a fixed (d, α) (see Section 3.2). The quantity $p(\mathbf{x}|d, \alpha)$ is the evidence, given by

$$p(\mathbf{x}|d, \alpha) = \frac{\left(\prod_{l=1}^{K-1}(\alpha + ld)\right)\left(\prod_{i=1}^{K}\Gamma(n_i - d)\right)\Gamma(1 + \alpha)}{\Gamma(1 - d)^K \Gamma(\alpha + N)}. \tag{17}$$

We can obtain posterior credible intervals for $\hat{H}_{\mathrm{PYM}}$ by estimating the posterior variance $\mathbb{E}[(H - \hat{H}_{\mathrm{PYM}})^2|\mathbf{x}]$. The estimate takes the same form as (16), except that we replace $\mathbb{E}[H|\mathbf{x}, d, \alpha]$ with $\mathrm{var}[H|\mathbf{x}, d, \alpha]$ in the integrand.

4.3 Computation

Because of the improper prior $p(d, \alpha)$, and because by (16) it must be integrated over all $\alpha > 0$, it is not obvious that the PYM estimate $\hat{H}_{\mathrm{PYM}}$ is computationally tractable. In this section we discuss techniques for efficient and accurate computation of $\hat{H}_{\mathrm{PYM}}$. First, we outline a compressed data representation we call the "multiplicities" representation, which substantially reduces computational

cost. Then, we outline a fast method for performing the numerical integration over a suitable range of α and d.

4.3.1 Multiplicities

We can compute the expected entropy $\mathbb{E}[H|\mathbf{x}, d, \alpha]$ more efficiently by using a representation in terms of *multiplicities*, a compressed statistic often used under other names (e.g., the *empirical histogram distribution function* as discussed by Paninski 2003). Multiplicities are the number of symbols that have occurred with a given frequency in the sample. Letting $m_k = |\{i : n_i = k\}|$ denote the total number of symbols with exactly k observations in the sample gives the compressed statistic $\mathbf{m} = [m_0, m_1, \ldots, m_M]^\top$, where M is the largest number of samples for any symbol. Note that the dot product $[0, 1, \ldots, M]^\top \mathbf{m} = N$, is the total number of samples.

The multiplicities representation significantly reduces the time and space complexity of our computations for most data sets as we need only compute sums and products involving the number of symbols with distinct frequencies (at most M), rather than the total number of symbols K. In practice we compute all expressions not explicitly involving $\boldsymbol{\pi}$ using the multiplicities representation. For instance, when expressed in terms of the multiplicities the evidence takes the compressed form

$$\begin{aligned} p(\mathbf{x}|d, \alpha) &= p(m_1, \ldots, m_M|d, \alpha) \\ &= \frac{\Gamma(1+\alpha)\prod_{l=1}^{K-1}(\alpha + ld)}{\Gamma(\alpha + N)} \prod_{i=1}^{M} \left(\frac{\Gamma(i-d)}{i!\Gamma(1-d)} \right)^{m_i} \frac{M!}{m_i!}. \end{aligned}$$

4.3.2 Heuristic for Integral Computation

In principle the PYM integral over α is supported on the range $[0, \infty)$. In practice, however, the posterior concentrates on a relatively small region of parameter space. It is generally unnecessary to consider the full integral over a semi-infinite domain. Instead, we select a subregion of $[0, 1] \times [0, \infty)$ which supports the posterior up to ϵ probability mass. The posterior is unimodal in each variable α and d separately (see Appendix D); however, we do not have a proof for the unimodality of the evidence. Nevertheless, if there are multiple modes, they must lie on a strictly decreasing line of d as a function of α and, in practice, we find the posterior to be unimodal. We illustrate the concentration of the evidence visually in Figure 5.

We compute the hessian at the MAP parameter value, $(d_{\text{MAP}}, \alpha_{\text{MAP}})$. Using the inverse hessian as the covariance of a Gaussian approximation to the posterior, we select a grid spanning ± 6 std. We use numerical integration (Gauss-Legendre quadrature) on this region to compute the integral. When the hessian is rank-deficient (which may occur, for instance, when the $\alpha_{\text{MAP}} = 0$ or $d_{\text{MAP}} = 0$), we use Gauss-Legendre quadrature to perform the integral in d over $[0, 1)$, but employ a Fourier-Chebyshev numerical quadrature routine to integrate α over $[0, \infty)$ (Boyd, 1987).

4.4 Sampling the Full Posterior Over H

The closed-form expressions for the conditional moments derived in the previous section allow us to compute PYM and its variance by 2-dimensional numerical integration. PYM's posterior mean and variance provide, essentially, a Gaussian approximation to the posterior, and corresponding credible regions. However, in some situations (see Figure 6), variance-based credible intervals are a poor approximation to the true posterior credible intervals. In these cases we may wish to examine the full posterior distribution over H.

Stick-breaking, as described by (9), provides a straightforward algorithm for sampling distributions $\boldsymbol{\pi} \sim \text{PY}(d, \alpha)$. With enough stick-breaking samples, it is always possible to approximate π to arbitrary

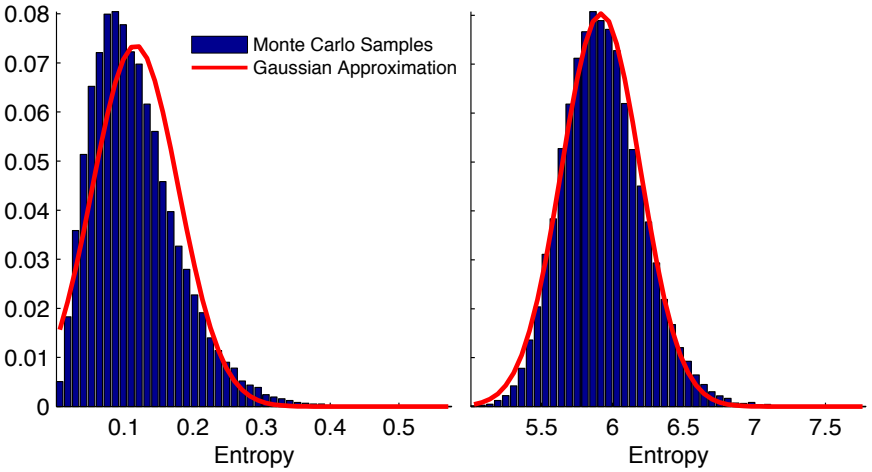


Figure 6: The posterior distributions of entropy for two data sets of 100 samples drawn from distinct distributions and the Gaussian approximation to each distribution based upon the posterior mean and variance. **(left)** Simulated example with low entropy. Notice that the true posterior is highly asymmetric, and that the Gaussian approximation does not respect the positivity of H. **(right)** Simulated example with higher entropy. The Gaussian approximation is a much closer approximation to the true distribution.

accuracy.[8] Even so, sampling $\pi \sim \mathrm{PY}(d, \alpha)$ for d near 1, where $\boldsymbol{\pi}$ is likely to be heavy-tailed, may require intractably large number of samples to obtain a good approximation.

We address this problem by directly estimating the entropy of the tail, $\mathrm{PY}(d, \alpha + N_s d)$, using (12). As shown in Figure 3, the prior variance of PY becomes arbitrarily small as for large α. We need only enough samples to assure that the variance of the tail entropy is small. The resulting final sample is the entropy of the (finite) samples plus the expected entropy of the tail, $H(\boldsymbol{\pi}^*) + \mathbb{E}[H|d, \alpha + Kd]$.[9]

Sampling entropy is most useful for very small amounts of data drawn from distributions with low expected entropy. In Figure 5 we illustrate the posterior distributions of entropy in two simulated experiments. In general, as the expected entropy and sample size increase, the posterior becomes more approximately Gaussian.

5. Theoretical Properties of PYM

Having defined PYM and discussed its practical computation, we now establish conditions under which (16) is defined (i.e., the right–hand of the equation is finite), and also prove some basic facts about its asymptotic properties. While $\hat{H}_{\mathrm{PYM}}$ is a Bayesian estimator, we wish to build connections to the literature by showing frequentist properties.

Note that the prior expectation $\mathbb{E}[H]$ does not exist for the improper prior defined above, since $p(h = \mathbb{E}[H]) \propto 1$ on $[0, \infty]$. It is therefore reasonable to ask what conditions on the data are sufficient to obtain finite posterior expectation $\hat{H}_{\mathrm{PYM}} = E[H|\mathbf{x}]$. We give an answer to this question in the following short proposition (proofs of all statements may be found in the appendix),

8. Bounds on the number of samples necessary to reach ϵ on average have been considered by Ishwaran and James (2001).
9. Due to the generality of the expectation formula (10), this method may be applied to sample the distributions of other additive functionals of PY.

Theorem 2 *Given a fixed data set* $\mathbf{x}$ *of* N *samples,* $\hat{H}_{PYM} < \infty$ *for any prior distribution* $p(d, \alpha)$ *if* $N - K \geq 2$.

In other words, we require 2 coincidences in the data for $\hat{H}_{\mathrm{PYM}}$ to be finite. When no coincidences have occurred in $\mathbf{x}$, we have no evidence regarding the support of the $\boldsymbol{\pi}$ and our resulting entropy estimate is unbounded. In fact, in the absence of coincidences, no entropy estimator can give a reasonable estimate without prior knowledge or assumptions about $\mathcal{A}$.

Concerns about inadequate numbers of coincidences are peculiar to the under-sampled regime; as $N \to \infty$, we will almost surely observe each letter infinitely often. We now turn to asymptotic considerations, establishing consistency of $\hat{H}_{\mathrm{PYM}}$ in the limit of large N for a broad class of distributions. It is known that the plugin is consistent for any distribution (finite or countably infinite), although the rate of convergence can be arbitrarily slow (Antos and Kontoyiannis, 2001). Therefore, we establish consistency by showing asymptotic convergence to the plugin estimator.

For clarity, we explicitly denote a quantity's dependence upon sample size N by introducing a subscript. Thus $\mathbf{x}$ and K become $\mathbf{x}_N$ and K_N respectively. As a first step we show that $\mathbb{E}[H|\mathbf{x}_N, d, \alpha]$ converges to the plugin estimator.

Theorem 3 *Assuming* $\mathbf{x}_N$ *drawn from a fixed, finite or countably infinite discrete distribution* $\boldsymbol{\pi}$ *such that* $\frac{K_N}{N} \xrightarrow{P} 0$

$$|\mathbb{E}[H|\mathbf{x}_N, d, \alpha] - \mathbb{E}[H_{\mathrm{plugin}}|\mathbf{x}_N]| \xrightarrow{P} 0$$

The assumption $K_N/N \to 0$ is more general than it may seem. For any infinite discrete distribution it holds that $K_N \to \mathbb{E}[K_N]$ a.s. and $\mathbb{E}[K_N]/N \to 0$ a.s. (Gnedin et al., 2007). Hence, $K_N/N \to 0$ in probability for an arbitrary distribution. As a result, the right–hand–side of (15) shares its asymptotic behavior, in particular consistency, with $\hat{H}_{\mathrm{plugin}}$. As (15) is consistent for each value of α and d, it is intuitively plausible that $\hat{H}_{\mathrm{PYM}}$, as a mixture of such values, should be consistent as well. However, while (15) alone is well-behaved, it is not clear that $\hat{H}_{\mathrm{PYM}}$ should be. Since $\mathbb{E}[H|\mathbf{x}, d, \alpha] \to \infty$ as $\alpha \to \infty$, care must be taken when integrating over $p(d, \alpha|\mathbf{x})$. Our main consistency result is,

Theorem 4 *For any proper prior or bounded improper prior* $p(d, \alpha)$*, if data* $\mathbf{x}_N$ *are drawn from a fixed, countably infinite discrete distribution* $\boldsymbol{\pi}$ *such that for some constant* $C > 0$ $K_N = o(N^{1-1/C})$ *in probability, then*

$$|\mathbb{E}[H|\mathbf{x}_N] - \mathbb{E}[H_{\mathrm{plugin}}|\mathbf{x}_N]| \xrightarrow{P} 0$$

Intuitively, the asymptotic behavior of K_N/N is tightly related to the tail behavior of the distribution (Gnedin et al., 2007). In particular, $K_N \sim cN^b$ with $0 < b < 1$ if and only if $\pi_i \sim c' i^{\frac{1}{b}}$ where c and c' are constants, and we assume π_i is non-increasing (Gnedin et al., 2007). The class of distributions such that $K_N = o(N^{1-1/C})$ a.s. includes the class of power-law or thinner tailed distributions, i.e., $\pi_i = O(i^b)$ for some $b > 1$ (again π_i is assumed non-increasing).

While consistency is an important property for any estimator, we emphasize that PYM is designed to address the under-sampled regime. Indeed, since $\hat{H}_{\mathrm{plugin}}$ is consistent and has an optimal rate of convergence for a large class of distributions (Vu et al., 2007; Antos and Kontoyiannis, 2001; Zhang, 2012), asymptotic properties provide little reason to use $\hat{H}_{\mathrm{PYM}}$. Nevertheless, notice that Theorem 4 makes very weak assumptions about $p(d, \alpha)$. In particular, the result is not dependent upon the form of the PYM prior introduced in the previous section; it holds for any probability distribution $p(d, \alpha)$, or even a bounded improper prior. Thus, we can view Theorem 4 as a statement about a class of PYM estimators. Almost any prior we choose on (d, α) results in a consistent estimator of entropy.

6. Simulation Results

We compare $\hat{H}_{\mathrm{PYM}}$ to other proposed entropy estimators using several example data sets. Each plot in Figures 7, 8, 9, and 10 shows convergence as well as small sample performance. We compare

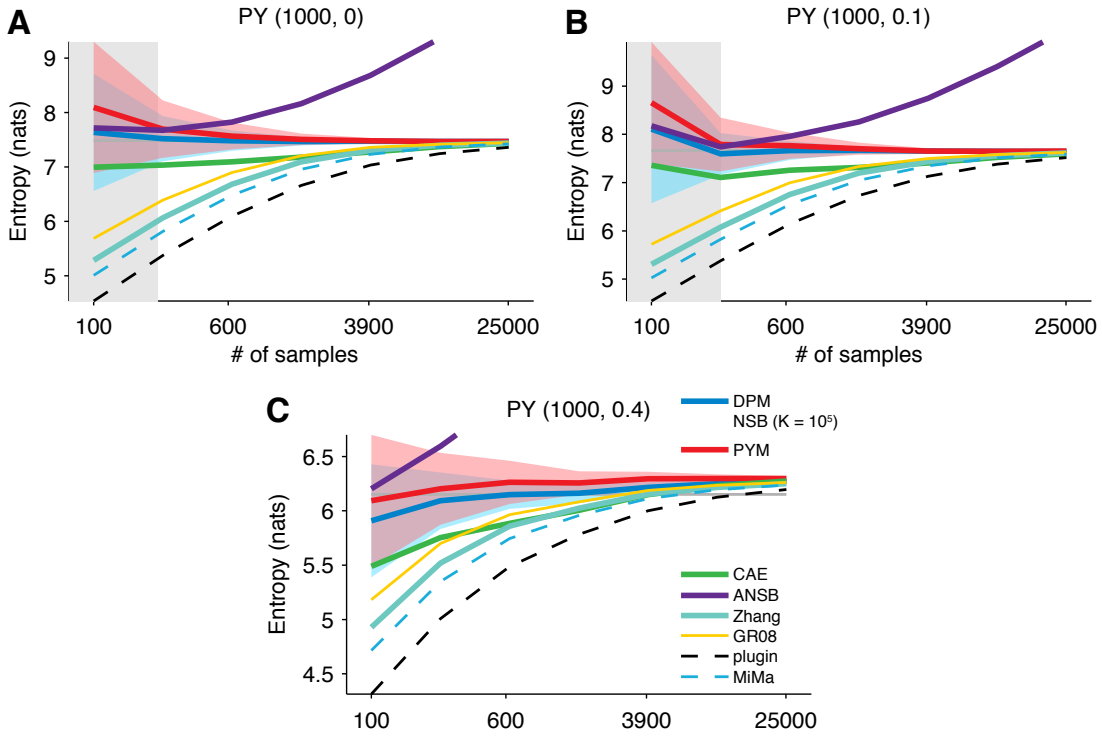


Figure 7: Comparison of estimators on stick-breaking distributions. Poisson-Dirichlet distribution with $(d = 0, \alpha = 1000)$ (**A**), $(d = 0.1, \alpha = 1000)$ (**B**), $(d = 0.4, \alpha = 100)$ (**C**). Recall that the Dirichlet Process is the Pitman-Yor Process with $d = 0$. We compare our estimators (DPM, PYM) with other enumerable support estimators (CAE, ANSB, Zhang, GR08), and finite support estimators (plugin, MiMa). Note that in these examples, the DPM estimator performs indistinguishably from NSB with alphabet size $\mathcal{A}$ fixed to a large value ($\mathcal{A} = 10^5$). For the experiments, we first sample a single $\boldsymbol{\pi} \sim \mathrm{PY}(d, \alpha)$ using the stick-breaking procedure of (9). For each N (x-axis), we apply all estimators to each of 10 sample data sets drawn randomly from $\boldsymbol{\pi}$. Solid lines are averages over all 10 realizations. Colored shaded area represent 95% credible intervals averaged over all 10 realizations. Gray shaded area represents the ANSB approximation regime defined as expected number of unique symbols being more than 90% the total number of samples.

our estimators, DPM ($d = 0$ only) and PYM ($\hat{H}_{\mathrm{PYM}}$), with other enumerable-support estimators: coverage-adjusted estimator (CAE) (Chao and Shen, 2003; Vu et al., 2007), asymptotic NSB (ANSB, Section 2.4) (Nemenman, 2011), Grassberger's asymptotic bias correction (GR08) (Grassberger, 2008), and Good-Turing estimator (Zhang, 2012). Note that similar to ANSB, DPM is an asymptotic (Poisson-Dirichlet) limit of NSB, and hence in practice behaves identically to NSB with large but finite K. We also compare with plugin (3) and a standard Miller-Maddow (MiMa) bias correction method with a conservative assumption that the number of uniquely observed symbols is K (Miller, 1955). To make comparisons more straightforward, we do not apply additional bias correction methods (e.g., jackknife) to any of the estimators.

In each simulation, we draw 10 sample distributions $\boldsymbol{\pi}$. From each $\boldsymbol{\pi}$ we draw a data set of N iid samples. In each figure we show the performance of all estimators averaged across the 10 sampled data sets.

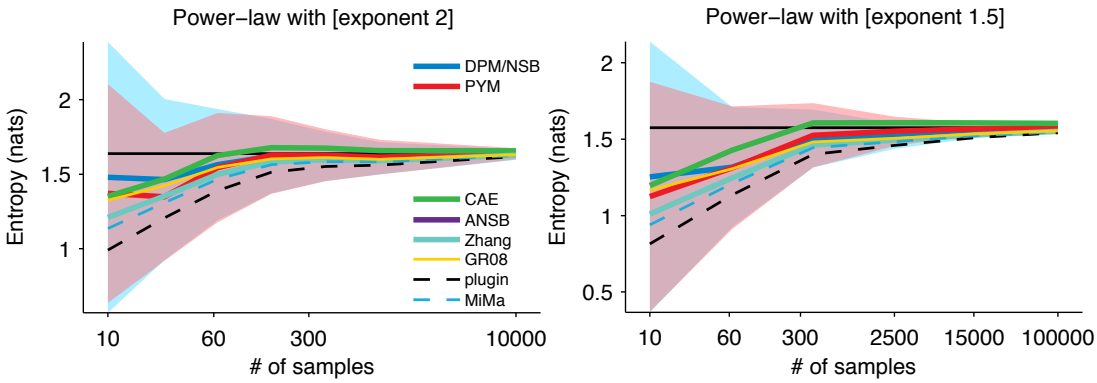


Figure 8: Comparison of estimators on power-law distributions. The highest probabilities in these power-law distributions were large enough that they were never effectively under-sampled.

The experiments of Figure 7 show performance on a single $\boldsymbol{\pi} \sim \mathrm{PY}(d, \alpha)$ drawn using the stick-breaking procedure of (9). We draw π_i according to (9) in blocks of size 10^3 until $1 - \sum_{N_s} \pi_i < 10^{-3}$, where N_s is the number of sticks. Unsurprisingly, PYM performs well when the data are truly generated by a Pitman-Yor process (Figure 7). Credible intervals for DPM tend to be smaller than PYM, although both shrink quickly (indicating high confidence). When the tail of the distribution is exponentially decaying, ($d = 0$ case; Figure 7 top), DPM shows slightly improved performance. When the tail has a strong power-law decay, (Figure 7 bottom), PYM performs better than DPM. Most of the other estimators are consistently biased down, with the exception of ANSB.

The shaded gray area indicates the ANSB approximation regime, where the approximation used to define the ANSB estimator is approximately valid. Although this region has no definitive boundary, it corresponds to a regime where where the average number of coincidences is small relative to the number of samples. Following Nemenman (2011), we define the under-sampled regime to be the region where $E[K_N]/N > 0.9$, where K_N is the number of unique symbols appearing in a sample of size N. Note that only 3 out of 10 results in Figures. 7, 8, 9, 10 have shaded area; the ANSB approximation regime is not large enough to appear in the plots. This regime appears to be designed for a relatively broad distribution (close to uniform distribution). In cases where a single symbol has high probability, the ANSB approximation regime is essentially never valid. In our example distributions, this is the case with for power-law distributions and $\mathcal{PY}$ distributions with large d. For example, Figure 8 is already outside of the ANSB approximation regime with only 4 samples.

Although Pitman-Yor process $\mathrm{PY}(d, \alpha)$ has a power-law tail controlled by d, the high probability portion is modulated by α and does not strictly follow a power-law distribution as a whole. In Figure 8, we evaluate the performance for $p_i \propto i^{-2}$ and $p_i \propto i^{-1.5}$. PYM and DPM have slight negative biases, but the credible interval covers the true entropy for all sample sizes. For small sample sizes, most estimators are negatively biased, again except for ANSB (which does not show up in the plot since it is severely biased upwards). Notably, CAE performs very well in moderate sample sizes.

In Figure 9 we compute the entropy per word of in the novel *Moby Dick* by Herman Melville and entropy per time bin of a population of retinal ganglion cells from monkey retina (Pillow et al., 2005). We tokenized the novel into individual words using the Python library NLTK.[10] Punctuation is disregarded. These real-world data sets have heavy, approximately power-law tails[11] as pointed out

10. Further information about the Natural Language Toolkit (NLTK) may be obtained at the project's website, `http://www.nltk.org/`.
11. We emphasize that we use the term "power-law" in a heuristic, descriptive sense only. We did not fit explicit power-law models to the data sets in questions, and neither do we rely upon the properties of power-law distributions in our analyses.

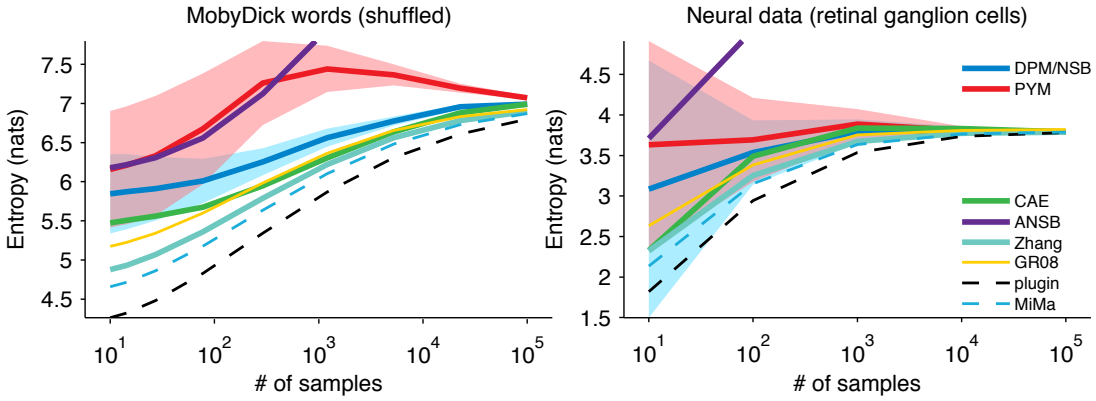


Figure 9: Comparison of estimators on real data sets.

earlier in Figure 2. For Moby Dick, PYM slightly overestimates, while DPM slightly underestimates, yet both methods are closer to the entropy estimated by the full data available than other estimators. DPM is overly confident (its credible interval is too narrow), while PYM becomes overly confident with more data. The neural data were preprocessed to be a binarized response (10 ms time bins) of 8 simultaneously recorded off-response retinal ganglion cells. PYM, DPM, and CAE all perform well on this data set with both PYM and DPM bracketing the asymptotic value with their credible intervals.

Finally, we applied the denumerable-support estimators to finite-support distributions (Figure 10). The power-law $p_n \propto n^{-1}$ has the heaviest tail among the simulations we consider but notice that it does not define a proper distribution (the probability mass does not integrate), and so we use a truncated $1/n$ distribution with the first 1000 symbols (Figure 10 top). Initially PYM shows the least bias, but DPM provides a better estimate for increasing sample size. However, notice that for both estimates the credible intervals consistently cover the true entropy. Interestingly, the finite support estimators perform poorly compared to DPM, CAE and PYM. For the uniform distribution over 1000 symbols, both DPM and PYM have slight upward bias, while CAE shows almost perfect performance (Figure 10 middle). For Poisson distribution, a theoretically enumerable-support distribution on the natural numbers, the tail decays so quickly that the effective support (due to machine precision) is very small (26 in this case). All the estimators, with the exception of ANSB, work quite well.

The novel Moby Dick provides the most challenging data: no estimator seems to have converged, even with the full data. Surprisingly, the Good-Turing estimator (Zhang, 2012) tends to perform similarly to the Grassberger and Miller-Maddow bias-correction methods. Among such the bias-correction methods, Grassberger's method tended to show the best performance, outperforming Zhang's method.

The computation time for our estimators is $O(LG)$, where L number symbols with distinct frequencies (bounded above by the quantity M defined in Section 4.3.1) and G is the number of grid points used to compute the numerical integral. Hence, computation time as a function of sample size depends upon how L scales with samples size N, always sublinearly, and $O(N^{1/2})$ in the worse case. In our examples, computation times for 10^5 samples are in the order of 0.1 seconds (Figure 11). Hence in practice, for the examples we have shown, more time is spent building the histogram from the data than computing the entropy estimate.

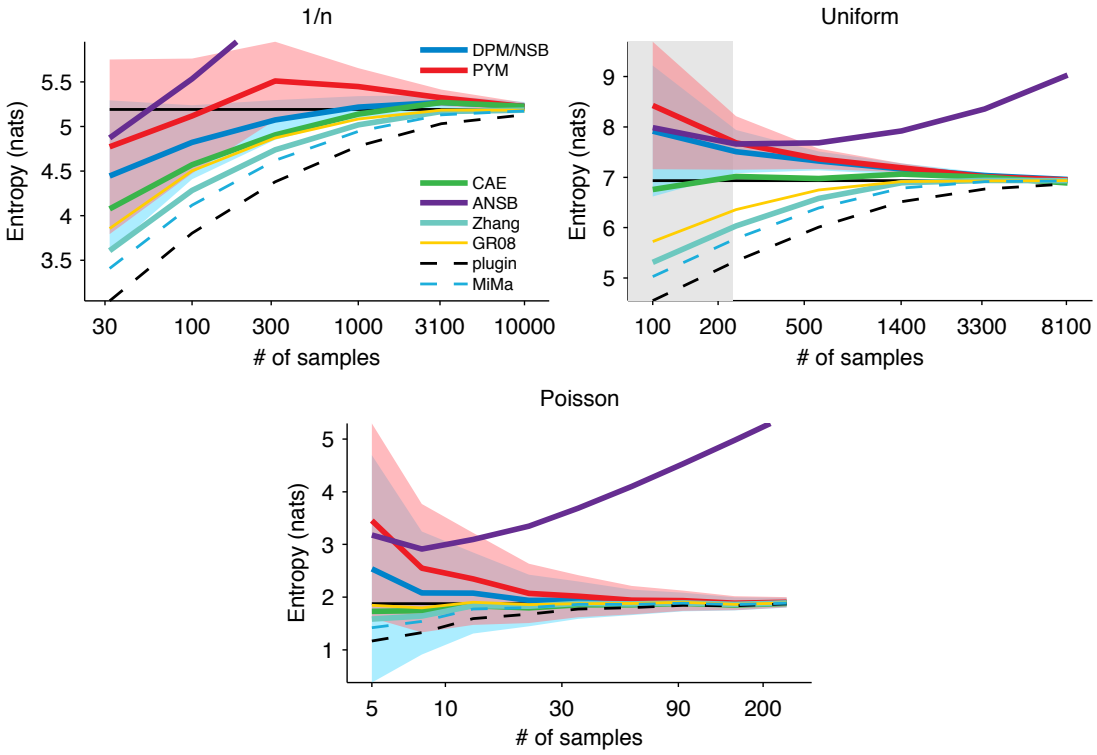


Figure 10: Comparison of estimators on finite support distributions. Black solid line indicates the true entropy. Poisson distribution ($\lambda = e$) has a countably infinite (but very thin) tail. All probability mass was concentrated on 26 symbols, within machine precision.

7. Conclusion

In this paper we introduced PYM, a new entropy estimator for distributions with unknown support. We derived analytic forms for the conditional mean and variance of entropy under a DP and PY prior for fixed parameters. Inspired by the work of Nemenman et al. (2002), we defined a novel PY mixture prior, PYM, which implies an approximately flat prior on entropy. PYM addresses two major issues with NSB: its dependence on knowledge of $\mathcal{A}$ and its inability (inherited from the Dirichlet distribution) to account for the heavy-tailed distributions which abound in biological and other natural data.

Further experiments on diverse data sets might reveal ways to improve PYM, such as new tactics or theory for selecting the prior on tail behavior, $q(\gamma)$. It may also prove fruitful to investigate ways to tailor PYM to a specific application, for instance by combining it with with more structured priors such as those employed by Archer et al. (2013). Further, while we have shown that PYM is consistent for any prior, an expanded theory might investigate the convergence rate, perhaps in relation to the choice of prior.

We have shown that PYM performs well in comparison to other entropy estimators, and indicated its practicality in example applications to data. A MATLAB implementation of the PYM estimator is available at `https://github.com/pillowlab/PYMentropy`.

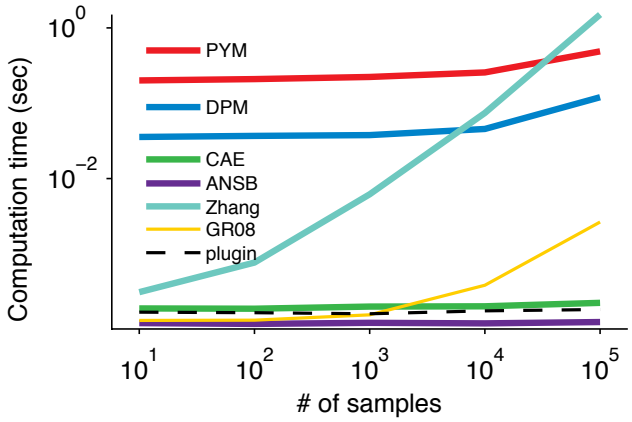


Figure 11: Median computation time to estimate entropy for the neural data. The computation time excludes the preprocessing required to build the histogram and convert to multiplicity representation. Note that for DPM and PYM this time also includes estimating the posterior variance.

Acknowledgments

We thank E. J. Chichilnisky, A. M. Litke, A. Sher and J. Shlens for retinal data, and Y. W. Teh and A. Cerquetti for helpful comments on the manuscript. This work was supported by a Sloan Research Fellowship, McKnight Scholar's Award, and NSF CAREER Award IIS-1150186 (JP). Parts of this manuscript were presented at the Advances in Neural Information Processing Systems (NIPS) 2012 conference.

Appendix A. Derivations of Dirichlet and PY Moments

In this Appendix we present as propositions a number of technical moment derivations used in the text.

A.1 Mean Entropy of Finite Dirichlet

Proposition 5 (Replica trick for entropy [Wolpert and Wolf, 1995])
For $\boldsymbol{\pi} \sim \mathrm{Dir}(\alpha_1, \alpha_2, \ldots, \alpha_{\mathcal{A}})$, *such that* $\sum_{i=1}^{\mathcal{A}} \alpha_i = A$, *and letting* $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_{\mathcal{A}})$, *we have*

$$\mathbb{E}[H(\boldsymbol{\pi})|\vec{\alpha}] \quad = \quad \psi_0(A+1) - \sum_{i=1}^{\mathcal{A}} \frac{\alpha_i}{A} \psi_0(\alpha_i + 1) \tag{18}$$

Proof First, let c be the normalizer of Dirichlet, $c = \frac{\prod_j \Gamma(\alpha_j)}{\Gamma(A)}$ and let $\mathcal{L}$ denote the Laplace transform (on π to s). Now we have that

$$\begin{aligned} c\mathbb{E}[H|\vec{\alpha}] &= \int \left(-\sum_i \pi_i \log_2 \pi_i \right) \delta(\textstyle\sum_i \pi_i - 1) \displaystyle\prod_j \pi_j^{\alpha_j - 1} d\boldsymbol{\pi} \\ &= -\sum_i \int (\pi_i^{\alpha_i} \log_2 \pi_i)\, \delta(\textstyle\sum_i \pi_i - 1) \displaystyle\prod_{j \neq i} \pi_j^{\alpha_j - 1} d\boldsymbol{\pi} \end{aligned}$$

$$
\begin{aligned}
&= -\sum_i \int \left(\frac{d}{d(\alpha_i)}\pi_i^{\alpha_i}\right)\delta(\textstyle\sum_i \pi_i - 1)\displaystyle\prod_{j\neq i}\pi_j^{\alpha_j - 1} d\boldsymbol{\pi} \\
&= -\sum_i \frac{d}{d(\alpha_i)}\int \pi_i^{\alpha_i}\delta(\textstyle\sum_i \pi_i - 1)\displaystyle\prod_{j\neq i}\pi_j^{\alpha_j - 1} d\boldsymbol{\pi} \\
&= -\sum_i \frac{d}{d(\alpha_i)}\mathcal{L}^{-1}\left[\mathcal{L}(\pi_i^{\alpha_i})\prod_{j\neq i}\mathcal{L}(\pi_j^{\alpha_j - 1})\right](1) \\
&= -\sum_i \frac{d}{d(\alpha_i)}\mathcal{L}^{-1}\left[\frac{\Gamma(\alpha_i+1)\prod_{j\neq i}\Gamma(\alpha_j)}{s^{\sum_k(\alpha_k)+1}}\right](1) \\
&= -\sum_i \frac{d}{d(\alpha_i)}\left[\frac{\Gamma(\alpha_i+1)}{\Gamma(\sum_k(\alpha_k)+1)}\right]\prod_{j\neq i}\Gamma(\alpha_j) \\
&= -\sum_i \frac{\Gamma(\alpha_i+1)}{\Gamma(\sum_k \alpha_k + 1)}\left[\psi_0(\alpha_i+1) - \psi_0(A+1)\right]\prod_{j\neq i}\Gamma(\alpha_j) \\
&= \left[\psi_0(A+1) - \sum_{i=1}^{\mathcal{A}}\frac{\alpha_i}{A}\psi_0(\alpha_i+1)\right]\frac{\prod_j \Gamma(\alpha_j)}{\Gamma(A)}.
\end{aligned}
$$

■

A.2 Variance of Entropy for Finite Dirichlet

We derive $\mathbb{E}[H^2(\boldsymbol{\pi})|\vec{\alpha}]$. In practice we compute $\text{var}[H(\boldsymbol{\pi})|\vec{\alpha}] = \mathbb{E}[H^2(\boldsymbol{\pi})|\vec{\alpha}] - \mathbb{E}[H(\boldsymbol{\pi})|\vec{\alpha}]^2$.

Proposition 6 *For $\boldsymbol{\pi} \sim \text{Dir}(\alpha_1, \alpha_2, \ldots, \alpha_{\mathcal{A}})$, such that $\sum_{i=1}^{\mathcal{A}} \alpha_i = A$, and letting $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_{\mathcal{A}})$, we have*

$$
\begin{aligned}
\mathbb{E}[H^2(\boldsymbol{\pi})|\vec{\alpha}] &= \sum_{i\neq k}\frac{\alpha_i\alpha_k}{(A+1)(A)}I_{ik} + \sum_i \frac{\alpha_i(\alpha_i+1)}{(A+1)(A)}J_i \qquad (19)\\
I_{ik} &= (\psi_0(\alpha_k+1) - \psi_0(A+2))\,(\psi_0(\alpha_i+1) \\
&\quad -\psi_0(A+2)) - \psi_1(A+2) \\
J_i &= (\psi_0(\alpha_i+2) - \psi_0(A+2))^2 + \psi_1(\alpha_i+2) \\
&\quad - \psi_1(A+2)
\end{aligned}
$$

Proof We can evaluate the second moment in a manner similar to the mean entropy above. First, we split the second moment into square and cross terms. To evaluate the integral over the cross terms, we apply the "replica trick" twice. Letting c be the normalizer of Dirichlet, $c = \frac{\prod_j \Gamma(\alpha_j)}{\Gamma(A)}$ we have

$$
\begin{aligned}
c\mathbb{E}[H^2|\vec{\alpha}] &= \int\left(-\sum_i \pi_i \log_2 \pi_i\right)^2 \delta(\textstyle\sum_i \pi_i - 1)\displaystyle\prod_j \pi_j^{\alpha_j - 1} d\boldsymbol{\pi} \\
&= \sum_i \int \left(\pi_i^2 {\log_2}^2 \pi_i\right)\delta(\textstyle\sum_i \pi_i - 1)\displaystyle\prod_j \pi_j^{\alpha_j - 1} d\boldsymbol{\pi} \\
&+ \sum_{i\neq k}\int (\pi_i \log_2 \pi_i)\,(\pi_k \log_2 \pi_k)\,\delta(\textstyle\sum_i \pi_i - 1)\displaystyle\prod_j \pi_j^{\alpha_j - 1} d\boldsymbol{\pi}
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_i \int \pi_i^{\alpha_i+1} {\log_2}^2 \pi_i \delta(\textstyle\sum_i \pi_i - 1) \prod_{j\neq i} \pi_j^{\alpha_j-1} d\boldsymbol{\pi} \\
&+ \sum_{i\neq k} \int (\pi_i^{\alpha_i} \log_2 \pi_i)\,(\pi_k^{\alpha_k} \log_2 \pi_k)\, \delta(\textstyle\sum_i \pi_i - 1) \prod_{j\notin\{i,k\}} \pi_j^{\alpha_j-1} d\boldsymbol{\pi} \\
&= \sum_i \frac{d^2}{d(\alpha_i+1)^2} \int \pi_i^{\alpha_i+1} \delta(\textstyle\sum_i \pi_i - 1) \prod_{j\neq i} \pi_j^{\alpha_j-1} d\boldsymbol{\pi} \\
&+ \sum_{i\neq k} \frac{d}{d\alpha_i}\frac{d}{d\alpha_k} \int (\pi_i^{\alpha_i})\,(\pi_k^{\alpha_k})\, \delta(\textstyle\sum_i \pi_i - 1) \prod_{j\notin\{i,k\}} \pi_j^{\alpha_j-1} d\boldsymbol{\pi}
\end{aligned}
$$

Assuming $i \neq k$, these will be the cross terms.

$$
\begin{aligned}
&\int (\pi_i \log_2 \pi_i)(\pi_k \log_2 \pi_k) \delta(\textstyle\sum_i \pi_i - 1) \prod_j \pi_j^{\alpha_j-1} d\boldsymbol{\pi} \\
&= \frac{d}{d\alpha_i}\frac{d}{d\alpha_k} \int (\pi_i^{\alpha_i})(\pi_k^{\alpha_k}) \delta(\textstyle\sum_i \pi_i - 1) \prod_{j\notin\{i,k\}} \pi_j^{\alpha_j-1} d\boldsymbol{\pi} \\
&= \frac{d}{d\alpha_i}\frac{d}{d\alpha_k} \left[\frac{\Gamma(\alpha_i+1)\Gamma(\alpha_k+1)}{\Gamma(A+2)}\right] \prod_{j\notin\{i,k\}} \Gamma(\alpha_j) \\
&= \frac{d}{d\alpha_k} \left[\frac{\alpha_i \Gamma(\alpha_k+1)}{\Gamma(A+2)} \psi_0(\alpha_i+1)\right. \\
&\qquad \left. - \frac{\alpha_i \Gamma(\alpha_k+1)}{\Gamma(A+2)} \psi_0(A+2)\right] \prod_{j\neq k} \Gamma(\alpha_j) \\
&= \frac{d}{d\alpha_k} \left[\frac{\alpha_i \psi_0(\alpha_k+1)}{\Gamma(A+2)} \psi_0(\alpha_i+1)\right. \\
&\qquad \left. - \frac{\alpha_i \Gamma(\alpha_k+1)}{\Gamma(A+2)} \psi_0(A+2)\right] \prod_{j\neq k} \Gamma(\alpha_j) \\
&= \frac{\alpha_i \alpha_k}{\Gamma(A+2)} \left[(\psi_0(\alpha_k+1) - \psi_0(A+2))\right. \\
&\qquad \left. (\psi_0(\alpha_i+1) - \psi_0(A+2)) - \psi_1(A+2)\right] \prod_j \Gamma(\alpha_j) \\
&= \frac{\alpha_i \alpha_k}{(A+1)(A)} \left[(\psi_0(\alpha_k+1) - \psi_0(A+2))\right. \\
&\qquad \left. (\psi_0(\alpha_i+1) - \psi_0(A+2)) - \psi_1(A+2)\right] \frac{\prod_j \Gamma(\alpha_j)}{\Gamma(A)}
\end{aligned}
$$

$$
\begin{aligned}
&\frac{d^2}{d(\alpha_i+1)^2} \int \pi_i^{\alpha_i+1} \delta(\textstyle\sum_i \pi_i - 1) \prod_{j\neq i} \pi_j^{\alpha_j-1} d\boldsymbol{\pi} \\
&= \frac{d^2}{d(\alpha_i+1)^2} \left[\frac{\Gamma(\alpha_i+2)}{\Gamma(A+2)}\right] \prod_{j\neq i} \Gamma(\alpha_j) \\
&= \frac{d^2}{dz^2} \left[\frac{\Gamma(z+1)}{\Gamma(z+c)}\right] \prod_{j\neq i} \Gamma(\alpha_j), \quad \{c = A+2-(\alpha_i+1)\}
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{\Gamma(1+z)}{\Gamma(c+z)}\left[(\psi_0(1+z)-\right. \\
&\qquad \left.\psi_0(c+z))^2+\psi_1(1+z)-\psi_1(c+z)\right]\prod_{j\neq i}\Gamma(\alpha_j) \\
&= \frac{z(z-1)}{(c+z-1)(c+z-2)}\left[(\psi_0(1+z)-\psi_0(c+z))^2\right. \\
&\qquad \left.+\psi_1(1+z)-\psi_1(c+z)\right]\frac{\prod_j\Gamma(\alpha_j)}{\Gamma(A)} \\
&= \frac{(\alpha_i+1)(\alpha_i)}{(A+1)(A)}\left[(\psi_0(\alpha_i+2)-\psi_0(A+2))^2+\psi_1(\alpha_i+2)\right. \\
&\qquad \left.-\psi_1(A+2)\right]\frac{\prod_j\Gamma(\alpha_j)}{\Gamma(A)}
\end{aligned}
$$

Summing over all terms and adding the cross and square terms, we recover the desired expression for $\mathbb{E}[H^2(\boldsymbol{\pi})|\vec{\alpha}]$. ■

A.3 Prior Entropy Mean and Variance Under PY

We derive the prior entropy mean and variance of a PY distribution with fixed parameters α and d, $\mathbb{E}_{\boldsymbol{\pi}}[H(\boldsymbol{\pi})|d,\alpha]$ and $\operatorname{var}_{\boldsymbol{\pi}}[H(\boldsymbol{\pi})|d,\alpha]$. We first prove our Proposition 1 (mentioned in (Pitman and Yor, 1997)). This proposition establishes the identity $\mathbb{E}\left[\sum_{i=1}^{\infty} f(\pi_i)\middle|\alpha\right] = \int_0^1 \frac{f(\tilde{\pi}_1)}{\tilde{\pi}_1} p(\tilde{\pi}_1|\alpha)d\tilde{\pi}_1$ which will allow us to compute expectations over PY using only the distribution of the first size biased sample, $\tilde{\pi}_1$.

Proof [Proof of Proposition 1]

First we validate (10). Writing out the general form of the size-biased sample

$$p(\tilde{\pi}_1 = x|\boldsymbol{\pi}) = \sum_{i=1}^{\infty}\pi_i\delta(x-\pi_i),$$

we see that

$$
\begin{aligned}
\mathbb{E}_{\tilde{\pi}_1}\left[\frac{f(\tilde{\pi}_1)}{\tilde{\pi}_1}\right] &= \int_0^1 \frac{f(x)}{x}p(\tilde{\pi}_1 = x)dx \\
&= \int_0^1 \mathbb{E}_{\boldsymbol{\pi}}\left[\frac{f(x)}{x}p(\tilde{\pi}_1 = x|\boldsymbol{\pi})\right]dx \\
&= \int_0^1 \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{i=1}^{\infty}\frac{f(x)}{x}\pi_i\delta(x-\pi_i)\right]dx \\
&= \mathbb{E}_{\boldsymbol{\pi}}\left[\int_0^1\sum_{i=1}^{\infty}\frac{f(x)}{x}\pi_i\delta(x-\pi_i)dx\right] \\
&= \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{i=1}^{\infty}\int_0^1\frac{f(x)}{x}\pi_i\delta(x-\pi_i)dx\right] \\
&= \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{i=1}^{\infty}f(\pi_i)\right],
\end{aligned}
$$

where the interchange of sums and integrals is justified by Fubini's theorem.

A similar method validates (11). We will need the second size-biased sample in addition to the first. We begin with the sum inside the expectation on the left–hand side of (11)

$$\sum_i \sum_{j \neq i} g(\pi_i, \pi_j) \tag{20}$$

$$= \frac{\sum_i \sum_{j \neq i} g(\pi_i, \pi_j)}{p(\tilde{\pi}_1 = \pi_i, \tilde{\pi}_2 = \pi_j)} p(\tilde{\pi}_1 = \pi_i, \tilde{\pi}_2 = \pi_j) \tag{21}$$

$$= \sum_i \sum_{j \neq i} \frac{g(\pi_i, \pi_j)}{\pi_i \pi_j} (1 - \pi_i) p(\tilde{\pi}_1 = \pi_i, \tilde{\pi}_2 = \pi_j) \tag{22}$$

$$= \mathbb{E}_{\tilde{\pi}_1, \tilde{\pi}_2} \left[\frac{g(\tilde{\pi}_1, \tilde{\pi}_2)}{\tilde{\pi}_1 \tilde{\pi}_2} (1 - \tilde{\pi}_1) \Big| \boldsymbol{\pi} \right] \tag{23}$$

where the joint distribution of size biased samples is given by

$$\begin{aligned} p(\tilde{\pi}_1 = \pi_i, \tilde{\pi}_2 = \pi_j) &= p(\tilde{\pi}_1 = \pi_i) p(\tilde{\pi}_2 = \pi_j | \tilde{\pi}_1 = \pi_i) \\ &= \pi_i \cdot \frac{\pi_j}{1 - \pi_i} \end{aligned}$$

■

As this identity is defined for any additive functional f of $\boldsymbol{\pi}$, we can employ it to compute the first two moments of entropy. For PYP (and DP when $d = 0$), the first size-biased sample is distributed according to

$$\tilde{\pi}_1 \sim \text{Beta}(1 - d, \alpha + d) \tag{24}$$

Proposition 1 gives the mean entropy directly. Taking $f(x) = -x \log(x)$ we have

$$\mathbb{E}[H(\boldsymbol{\pi}) | d, \alpha] = -\mathbb{E}_\alpha[\log(\pi_1)] = \psi_0(\alpha + 1) - \psi_0(1 - d),$$

The same method may be used to obtain the prior variance, although the computation is more involved. For the variance, we will need the second size-biased sample in addition to the first. The second size-biased sample is given by,

$$\tilde{\pi}_2 = (1 - \tilde{\pi}_1) v_2, \quad v_2 \sim \text{Beta}(1 - d, \alpha + 2d) \tag{25}$$

We will compute the second moment explicitly, splitting $H(\boldsymbol{\pi})^2$ into square and cross terms,

$$\mathbb{E}[(H(\boldsymbol{\pi}))^2 | d, \alpha] = \mathbb{E}\left[\left(-\sum_i \pi_i \log(\pi_i) \right)^2 \Big| d, \alpha \right]$$

$$= \mathbb{E}\left[\sum_i (\pi_i \log(\pi_i))^2 \Big| d, \alpha \right] \tag{26}$$

$$+ \mathbb{E}\left[\sum_i \sum_{j \neq i} \pi_i \pi_j \log(\pi_i) \log(\pi_j) \Big| d, \alpha \right] \tag{27}$$

The first term follows directly from (10)

$$\begin{aligned}
&\mathbb{E}\left[\sum_i (\pi_i \log(\pi_i))^2 \middle| d, \alpha\right] = \int_0^1 x(-\log(x))^2 p(x|d,\alpha)\,\mathrm{d}x \\
&= B^{-1}(1-d, \alpha+d)\int_0^1 x\log^2(x)x^{1-d}(1-x)^{\alpha+d-1}\,\mathrm{d}x \\
&= \frac{1-d}{\alpha+1}\left[(\psi_0(2-d)-\psi_0(2+\alpha))^2 + \psi_1(2-d) - \psi_1(2+\alpha)\right] \qquad (28)
\end{aligned}$$

The second term of (27), requires the first two size biased samples, and follows from (11) with $g(x,y) = \log(x)\log(y)$. For the PYP prior, it is easier to integrate on V_1 and V_2, rather than the size biased samples. Letting $\gamma = B^{-1}(1-d, \alpha+2d)$ and $\zeta = B^{-1}(1-d, \alpha+d)$, the second term is then,

$$\begin{aligned}
&\mathbb{E}\left[\mathbb{E}\left[\log(\tilde{\pi}_1)\log(\tilde{\pi}_2)(1-\pi_1)|\boldsymbol{\pi}\right]|\alpha\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[\log(V_1)\log((1-V_1)V_2)(1-V_1)|\boldsymbol{\pi}\right]|\alpha\right] \\
&= \zeta\gamma\int_0^1\int_0^1 \log(v_1)\log((1-v_1)v_2)(1-v_1)v_1^{1-d}(1-v_1)^{\alpha+d-1} \\
&\qquad\qquad \times v_2^{1-d}(1-v_2)^{\alpha+2d-1}\,\mathrm{d}v_1\,\mathrm{d}v_2 \\
&= \zeta\left[\int_0^1 \log(v_1)\log(1-v_1)(1-v_1)v_1^{1-d}(1-v_1)^{\alpha+d-1}\,\mathrm{d}v_1\right. \\
&\quad + \gamma\int_0^1 \log(v_1)(1-v_1)v_1^{1-d}(1-v_1)^{\alpha+d-1} \\
&\qquad\qquad \left.\times\int_0^1 \log(v_2)v_2^{1-d}(1-v_2)^{\alpha+2d-1}\,\mathrm{d}v_1\,\mathrm{d}v_2\right] \\
&= \frac{\alpha+d}{\alpha+1}\left[(\psi_0(1-d)-\psi_0(2+\alpha))^2 - \psi_1(2+\alpha)\right]
\end{aligned}$$

Finally combining the terms, the variance of the entropy under PYP prior is

$$\begin{aligned}
&\mathrm{var}[H(\boldsymbol{\pi})|d,\alpha] = \qquad (29) \\
&\frac{1-d}{\alpha+1}\left[(\psi_0(2-d)-\psi_0(2+\alpha))^2 + \psi_1(2-d) - \psi_1(2+\alpha)\right] \\
&\quad + \frac{\alpha+d}{\alpha+1}\left[(\psi_0(1-d)-\psi_0(2+\alpha))^2 - \psi_1(2+\alpha)\right] \\
&\quad - (\psi_0(1+\alpha)-\psi_0(1-d))^2 \\
&= \frac{\alpha+d}{(\alpha+1)^2(1-d)} + \frac{1-d}{\alpha+1}\psi_1(2-d) - \psi_1(2+\alpha) \qquad (30)
\end{aligned}$$

We note that the expectations over the finite Dirichlet may also be derived using this formula by letting the $\tilde{\boldsymbol{\pi}}$ be the first size-biased sample of a finite Dirichlet on $\Delta_{\mathcal{A}}$.

A.4 Posterior Moments of PYP

First, we discuss the form of the PYP posterior, and introduce independence properties that will be important in our derivation of the mean. We recall that the PYP posterior, $\boldsymbol{\pi}_{\mathrm{post}}$, of (14) has three stochastically independent components: Bernoulli p_*, PY $\boldsymbol{\pi}$, and Dirichlet $\mathbf{p}$.

Component expectations: From the above derivations for expectations under the PYP and Dirichlet distributions as well as the Beta integral identities (see, e.g., Archer et al., 2012), we find

expressions for $\mathbb{E}_{\mathbf{p}}\left[H(\mathbf{p})|d,\alpha\right]$, $E_{\boldsymbol{\pi}}\left[H(\boldsymbol{\pi})|d,\alpha\right]$, and $\mathbb{E}_{p_*}\left[H(p_*)\right]$.

$$\begin{aligned}
\mathbb{E}[H(\boldsymbol{\pi})|d,\alpha] &= \psi_0(\alpha+1) - \psi_0(1-d) \\
\mathbb{E}_{p_*}[H(p_*)] &= \psi_0(\alpha+N+1) - \frac{\alpha+Kd}{\alpha+N}\psi_0(\alpha+Kd+1) \\
&\quad - \frac{N-Kd}{\alpha+N}\psi_0(N-Kd+1) \\
\mathbb{E}_{\mathbf{p}}[H(\mathbf{p})|d,\alpha] &= \psi_0(N-Kd+1) - \sum_{i=1}^{K}\frac{n_i-d}{N-Kd}\psi_0(n_i-d+1)
\end{aligned}$$

where by a slight abuse of notation we define the entropy of p_* as $H(p_*) = -(1-p_*)\log(1-p_*) - p_*\log p_*$. We use these expectations below in our computation of the final posterior integral.

Derivation of posterior mean: We now derive the analytic form of the posterior mean, (15).

$$\begin{aligned}
\mathbb{E}[H(\pi_{\text{post}})|d,\alpha] &= \mathbb{E}\left[-\sum_{i=1}^{K} p_i\log p_i - p_*\sum_{i=1}^{\infty}\pi_i\log p_*\pi_i\Big|d,\alpha\right] \\
&= \mathbb{E}\left[-(1-p_*)\sum_{i=1}^{K}\frac{p_i}{1-p_*}\log\left(\frac{p_i}{1-p_*}\right)\right. \\
&\quad \left.-(1-p_*)\log(1-p_*) - p_*\sum_{i=1}^{\infty}\pi_i\log\pi_i - p_*\log p_*\Big|d,\alpha\right] \\
&= \mathbb{E}\left[-(1-p_*)\sum_{i=1}^{K}\frac{p_i}{1-p_*}\log\left(\frac{p_i}{1-p_*}\right)\right. \\
&\qquad \left.-p_*\sum_{i=1}^{\infty}\pi_i\log\pi_i + H(p_*)\Big|d,\alpha\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[-(1-p_*)\sum_{i=1}^{K}\frac{p_i}{1-p_*}\log\left(\frac{p_i}{1-p_*}\right)\right.\right. \\
&\qquad \left.\left.-p_*\sum_{i=1}^{\infty}\pi_i\log\pi_i + H(p_*)\;\Big|\;p_*\right]\Big|d,\alpha\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[(1-p_*)H(\mathbf{p}) + p_*H(\boldsymbol{\pi}) + H(p_*)\;\Big|\;p_*\right]\Big|d,\alpha\right] \\
&= \mathbb{E}_{p_*}\left[(1-p_*)\mathbb{E}_{\mathbf{p}}\left[H(\mathbf{p})|d,\alpha\right] + p_*\mathbb{E}_{\boldsymbol{\pi}}\left[H(\boldsymbol{\pi})|d,\alpha\right] + H(p_*)\right]
\end{aligned}$$

Using the formulae for $\mathbb{E}_{\mathbf{p}}\left[H(\mathbf{p})|d,\alpha\right]$, $\mathbb{E}_{\boldsymbol{\pi}}\left[H(\boldsymbol{\pi})|d,\alpha\right]$, and $\mathbb{E}_{p_*}[H(p_*)]$ and rearranging terms, we obtain (15)

$$\begin{aligned}
\mathbb{E}[H(\pi_{\text{post}})|d,\alpha] &= \frac{A}{\alpha+N}\mathbb{E}_{\mathbf{p}}[H(\mathbf{p})] \\
&\quad + \frac{\alpha+Kd}{\alpha+N}\mathbb{E}_{\boldsymbol{\pi}}[H(\boldsymbol{\pi})] + \mathbb{E}_{p_*}[H(p_*)] \\
&= \frac{A}{\alpha+N}\left[\psi_0(A+1) - \sum_{i=1}^{K}\frac{\alpha_i}{A}\psi_0(\alpha_i+1)\right] \\
&\quad + \frac{\alpha+Kd}{\alpha+N}\left[\psi_0(\alpha+Kd+1) - \psi_0(1-d)\right] + \\
&\quad \psi_0(\alpha+N+1) - \frac{\alpha+Kd}{\alpha+N}\psi_0(\alpha+Kd+1) - \frac{A}{\alpha+N}\psi_0(A+1)
\end{aligned}$$

$$= \psi_0(\alpha + N + 1) - \frac{\alpha + Kd}{\alpha + N}\psi_0(1-d) - \frac{A}{\alpha + N}\left[\sum_{i=1}^{K}\frac{\alpha_i}{A}\psi_0(\alpha_i + 1)\right]$$
$$= \psi_0(\alpha + N + 1) - \frac{\alpha + Kd}{\alpha + N}\psi_0(1-d) - \frac{1}{\alpha + N}\left[\sum_{i=1}^{K}(n_i - d)\psi_0(n_i - d + 1)\right].$$

Derivation of posterior variance: We continue the notation from the subsection above. In order to exploit the independence properties of π_{post} we first apply the law of total variance to obtain (31)

$$\operatorname{var}[H(\pi_{\text{post}})|d,\alpha] = \operatorname*{var}_{p_*}\left[\mathbb{E}_{\boldsymbol{\pi},\mathbf{p}}[H(\pi_{\text{post}})]\middle|d,\alpha\right] + \mathbb{E}_{p_*}\left[\operatorname*{var}_{\boldsymbol{\pi},\mathbf{p}}[H(\pi_{\text{post}})]\middle|d,\alpha\right] \tag{31}$$

We now seek expressions for each term in (31) in terms of the expectations already derived.

Step 1: For the right-hand term of (31), we use the independence properties of π_{post} to express the variance in terms of PYP, Dirichlet, and Beta variances

$$\mathbb{E}_{p_*}\left[\operatorname*{var}_{\boldsymbol{\pi},\mathbf{p}}[H(\pi_{\text{post}})|p_*]\middle|d,\alpha\right] \tag{32}$$
$$= \mathbb{E}_{p_*}\left[(1-p_*)^2\operatorname*{var}_{\mathbf{p}}[H(\mathbf{p})] + p_*^2\operatorname*{var}_{\boldsymbol{\pi}}[H(\boldsymbol{\pi})]\middle|d,\alpha\right]$$
$$= \frac{(N-Kd)(N-Kd+1)}{(\alpha+N)(\alpha+N+1)}\operatorname*{var}_{\mathbf{p}}[H(\mathbf{p})] + \frac{(\alpha+Kd)(\alpha+Kd+1)}{(\alpha+N)(\alpha+N+1)}\operatorname*{var}_{\boldsymbol{\pi}}[H(\boldsymbol{\pi})] \tag{33}$$

Step 2: In the left-hand term of (31) the variance is with respect to the Beta distribution, while the inner expectation is precisely the posterior mean we derived above. Expanding, we obtain

$$\operatorname*{var}_{p_*}\left[\mathbb{E}_{\boldsymbol{\pi},\mathbf{p}}[H(\pi_{\text{post}})|p_*]\middle|d,\alpha\right]$$
$$= \operatorname*{var}_{p_*}\left[(1-p_*)\mathbb{E}_{\mathbf{p}}\left[H(\mathbf{p})\right] + p_*\mathbb{E}_{\boldsymbol{\pi}}\left[H(\boldsymbol{\pi})|p_*\right] + H(p_*)\middle|d,\alpha\right] \tag{34}$$

To evaluate this integral, we introduce some new notation

$$\begin{aligned}\mathbf{A} &= \mathbb{E}_{\mathbf{p}}\left[H(\mathbf{p})\right]\\ \mathbf{B} &= \mathbb{E}_{\boldsymbol{\pi}}\left[H(\boldsymbol{\pi})\right]\\ \Omega(p_*) &= (1-p_*)\mathbb{E}_{\mathbf{p}}\left[H(\mathbf{p})\right] + p_*\mathbb{E}_{\boldsymbol{\pi}}\left[H(\boldsymbol{\pi})\right] + H(p_*)\\ &= (1-p_*)\mathbf{A} + p_*\mathbf{B} + H(p_*)\end{aligned}$$

so that

$$\Omega^2(p_*) = 2p_*H(p_*)[\mathbf{B}-\mathbf{A}] + 2\mathbf{A}H(p_*) + h^2(p_*) + p_*^2[\mathbf{B}^2 - 2\mathbf{A}\mathbf{B}] + 2p_*\mathbf{A}\mathbf{B} + (1-p_*)^2\mathbf{A}^2 \tag{35}$$

and we note that

$$\operatorname*{var}_{p_*}\left[\mathbb{E}_{\boldsymbol{\pi},\mathbf{p}}[H(\pi_{\text{post}})|p_*]\middle|d,\alpha\right] = \mathbb{E}_{p_*}[\Omega^2(p_*)] - \mathbb{E}_{p_*}[\Omega(p_*)]^2 \tag{36}$$

The components composing $\mathbb{E}_{p_*}[\Omega(p_*)]$, as well as each term of (35) are derived by Archer et al. (2012). Although less elegant than the posterior mean, the expressions derived above permit us to compute (31) numerically from its component expectations, without sampling.

Appendix B. Proof of Proposition 2

In this Appendix we give a proof for Proposition 2.

Proof PYM is given by

$$\hat{H}_{PYM} = \frac{1}{p(\mathbf{x})}\int_0^\infty \int_0^1 H_{(d,\alpha)}p(\mathbf{x}|d,\alpha)p(d,\alpha)\,\mathrm{d}\alpha\,\mathrm{d}d$$

where we have written $H_{(d,\alpha)} = \mathbb{E}[H|d,\alpha,\mathbf{x}]$. Note that $p(\mathbf{x}|d,\alpha)$ is the evidence, given by (17). We will assume $p(d,\alpha) = 1$ for all α and d to show conditions under which $H_{(d,\alpha)}$ is integrable for any prior. Using the identity $\frac{\Gamma(x+n)}{\Gamma(x)} = \prod_{i=1}^n (x+i-1)$ and the log convexity of the Gamma function we have

$$\begin{aligned} p(\mathbf{x}|d,\alpha) &\leq \prod_{i=1}^K \frac{\Gamma(n_i-d)}{\Gamma(1-d)}\frac{\Gamma(\alpha+K)}{\Gamma(\alpha+N)} \\ &\leq \frac{\Gamma(n_i-d)}{\Gamma(1-d)}\frac{1}{\alpha^{N-K}} \end{aligned}$$

Since $d \in [0,1)$, we have from the properties of the digamma function

$$\psi_0(1-d) = \psi_0(2-d) - \frac{1}{1-d} \geq \psi_0(1) - \frac{1}{1-d} = -\gamma - \frac{1}{1-d},$$

and thus the upper bound

$$H_{(d,\alpha)} \leq \psi_0(\alpha+N+1) + \frac{\alpha+Kd}{\alpha+N}\left(\gamma + \frac{1}{1-d}\right) \tag{37}$$

$$- \frac{1}{\alpha+N}\left[\sum_{i=1}^K (n_i-d)\psi_0(n_i-d+1)\right]. \tag{38}$$

Although second term is unbounded in d notice that $\frac{\Gamma(n_i-d)}{\Gamma(1-d)} = \prod_{i=1}^{n_i}(i-d)$; thus, so long as $N-K \geq 1$, $H_{(\alpha,d)}p(\mathbf{x}|d,\alpha)$ is integrable in d.

For the integral over α, it suffices to choose $\alpha_0 \gg N$. Consider the tail $\int_{\alpha_0}^\infty H_{(d,\alpha)}p(\mathbf{x}|d,\alpha)p(d,\alpha)\,\mathrm{d}\alpha$. From (37) and the asymptotic expansion $\psi(x) = \log(x) - \frac{1}{2x} - \frac{1}{12x^2} + O(\frac{1}{x^4})$ as $x \to \infty$ we see that in the limit of $\alpha \gg N$

$$H_{(d,\alpha)} \leq \log(\alpha+N+2) + \frac{c}{\alpha},$$

where c is a constant depending on K, N, and d. Thus, we have

$$\begin{aligned} &\int_{\alpha_0}^\infty H_{(d,\alpha)}p(\mathbf{x}|d,\alpha)p(d,\alpha)\,\mathrm{d}\alpha \\ &\qquad \leq \frac{\prod_{i=1}^K \Gamma(n_i-d)}{\Gamma(1-d)}\int_{\alpha_0}^\infty \left(\log(\alpha+N+2) + \frac{c}{\alpha}\right)\frac{1}{\alpha^{N-K}}\,\mathrm{d}\alpha \end{aligned}$$

and so $H_{(d,\alpha)}$ is integrable in α so long as $N-K \geq 2$. ■

Appendix C. Proofs of Consistency Results

Proof [proof of Theorem 3] We have

$$\begin{aligned}
&\lim_{N\to\infty} \mathbb{E}[H|\alpha, d, \mathbf{x}_N] \\
&\quad = \lim_{N\to\infty} \left[\psi_0(\alpha+N+1) - \frac{\alpha+K_N d}{\alpha+N}\psi_0(1-d) - \right. \\
&\qquad\qquad \left. \frac{1}{\alpha+N}\left[\sum_{i=1}^{K_N}(n_i-d)\psi_0(n_i-d+1)\right]\right] \\
&\quad = \lim_{N\to\infty} \left[\psi_0(\alpha+N+1) - \sum_{i=1}^{K_N}\frac{n_i}{N}\psi_0(n_i-d+1)\right] \\
&\quad = -\lim_{N\to\infty} \sum_{i=1}^{K_N}\frac{n_i}{N}\left[\psi_0(n_i-d+1) - \psi_0(\alpha+N+1)\right]
\end{aligned}$$

although we have made no assumptions about the tail behavior of $\boldsymbol{\pi}$, so long as $\pi_k > 0$, $\mathbb{E}[n_k] = \mathbb{E}[\sum_{i=1}^{\infty}\mathbf{1}_{\{x_i=k\}}] = \sum_{i=1}^{\infty}P\{x_i = k\} = \lim_{N\to\infty} N\pi_k \to \infty$, and we may apply the asymptotic expansion $\psi(x) = \log(x) - \frac{1}{2x} - \frac{1}{12x^2} + O(\frac{1}{x^4})$ as $x \to \infty$ to find

$$\lim_{N\to\infty} \mathbb{E}[H|\alpha, d, \mathbf{x}_N] = H_{\text{plugin}}$$

■

We now turn to the proof of consistency for PYM. Although consistency is an intuitively plausible property for PYM, due to the form of the estimator our proof involves a rather detailed technical argument. Because of this, we break the proof of Theorem 4 into two parts. First, we prove a supporting Lemma.

Lemma 7 *If the data $\mathbf{x}_N$ have at least two coincidences, and are sampled from a distribution such that, for some constant $c > 0$, $K_N = o(N^{1-1/c})$ in probability, the following sequence of integrals converge.*

$$\int_0^{K_N+c}\int_0^1 \mathbb{E}[H|\alpha, d, \mathbf{x}_N]\frac{p(\mathbf{x}_N|\alpha, d)p(\alpha, d)}{p(\mathbf{x}_N)}\mathrm{d}\alpha\mathrm{d}d \xrightarrow{P} \mathbb{E}[\hat{H}_{plugin}|\mathbf{x}_N]$$

where $c > 0$ is an arbitrary constant.

Proof

Notice first that $E[H|\alpha, d, \mathbf{x}_N]$ is monotonically increasing in α, and so

$$\begin{aligned}
&\int_{\alpha=0}^{K_N+c}\int_{d=0}^1 \mathbb{E}[H|\alpha, d, \mathbf{x}_N]\frac{p(\mathbf{x}_N|\alpha, d)}{p(\mathbf{x}_N)}\,\mathrm{d}\alpha\,\mathrm{d}d \\
&\qquad \le \int_{\alpha=0}^{K_N+c}\int_{d=0}^1 \mathbb{E}[H|K_N+c, d, \mathbf{x}_N]\frac{p(\mathbf{x}_N|\alpha, d)}{p(\mathbf{x}_N)}\,\mathrm{d}\alpha\,\mathrm{d}d.
\end{aligned}$$

As a result we have that

$$\begin{aligned}\mathbb{E}[H|K_N+c,d,\mathbf{x}_N] &= \psi_0(K_N+c+N+1) \\ &- \frac{(1+d)K_N+c}{K_N+N+c}\psi_0(1-d) \\ &- \frac{1}{K_N+c+N}\left(\sum_{i=1}^{K_N}(n_i-d)\psi_0(n_i-d+1)\right)\end{aligned} \tag{39}$$

As a consequence of Proposition 2, $\int_{d=0}^{1}(1+d)\psi(1-d)\frac{p(\mathbf{x}|\alpha,d)}{p(\mathbf{x}_N)}\,\mathrm{d}d<\infty$, and so the second term is bounded and controlled by K_N/N. We let

$$A(d,N) = -\frac{(1+d)K_N+c}{K_N+N+c}\psi_0(1-d)$$

and, since $\lim_{N\to\infty}\int_{d=0}^{1}A(d,N)\frac{p(\mathbf{x}|\alpha,d)}{p(\mathbf{x}_N)}\,\mathrm{d}d=0$, we focus on the remaining terms of (39). We also let $B(\mathbf{n})=\sum_{i=1}^{K_N}\left(\frac{n_i-1}{N}\log\left(\frac{n_i}{N}\right)\right)$, and note that $\lim_{N\to\infty}B=\hat{H}_{\text{plugin}}$. We find that

$$\begin{aligned}&\mathbb{E}[H|K_N+c,d,\mathbf{x}_N] \\ &\quad\le \log(N+K_N+c+1)+A(d,N) \\ &\quad- \sum_{i=1}^{K_N}\left(\frac{n_i-1}{K_N+N+c}\log(n_i)\right) \\ &\quad= \log(N+K_N+c+1)+A(d,N)- \\ &\quad\frac{N}{K_N+N+c}\left[\sum_{i=1}^{K_N}\left(\frac{n_i-1}{N}\log\left(\frac{n_i}{N}\right)\right)+\frac{N-K_N}{N}\log(N)\right] \\ &\quad= \log\left(1+\frac{K_N+c+1}{N}\right)+A(d,N) \\ &\qquad+\log(N)\left[\frac{2K_N+c}{N+K_N+c}\right]+\frac{N}{K_N+N+c}B \\ &\quad= \log\left(1+\frac{K_N+c+1}{N}\right)+A(d,N) \\ &\qquad+\frac{1}{1+(K_N+c)/N}\frac{2K_N+c}{N^{1-1/C}}\frac{\log(N)}{N^{1/C}}+\frac{N}{K_N+N+c}B \\ &\quad\to \hat{H}_{\text{plugin}}+o(1)\end{aligned}$$

As a result

$$\begin{aligned}\int_{\alpha=0}^{K_N+c}\int_{d=0}^{1}\mathbb{E}[H|\alpha,d,\mathbf{x}_N]\frac{p(\mathbf{x}_N|\alpha,d)}{p(\mathbf{x}_N)}\,\mathrm{d}d\,\mathrm{d}\alpha& \\ \le\left[\hat{H}_{\text{plugin}}\int_{\alpha=0}^{K_N+c}\int_{d=0}^{1}\frac{p(\mathbf{x}_N|\alpha,d)}{p(\mathbf{x}_N)}\,\mathrm{d}d\,\mathrm{d}\alpha+o(1)\right]& \\ \to\hat{H}_{\text{plugin}}&\end{aligned}$$

For the lower bound, we let $H_{(\alpha,d,N)}=\mathbb{E}[H|\alpha,d,\mathbf{x}_N]\mathbf{1}_{[0,K_N+c]}(\alpha)$. Notice that $\exp(-H_{(\alpha,d,N)})\le 1$, so by dominated convergence $\lim_{N\to\infty}\mathbb{E}[\exp(-H_{(\alpha,d,N)})]=\exp(-\hat{H}_{\text{plugin}})$ by Proposition 2. And so by Jensen's inequality

$$\exp(-\lim_{N\to\infty}\mathbb{E}[H_{(\alpha,d,N)}]) \leq \lim_{N\to\infty}\mathbb{E}[\exp(-H_{(\alpha,d,N)})] = \exp(-\hat{H}_{\text{plugin}})$$
$$\implies \lim_{N\to\infty}\mathbb{E}[H_{(\alpha,d,N)}] \geq \hat{H}_{\text{plugin}},$$

and the lemma follows. ■

We now turn to the proof of our primary consistency result.

Proof [proof of Theorem 4]

$$\begin{aligned}
&\iint \mathbb{E}[H|\alpha,d,\mathbf{x}_N]\frac{p(\mathbf{x}_N|\alpha,d)p(\alpha,d)}{p(\mathbf{x}_N)}\mathrm{d}\alpha\mathrm{d}d \\
&\quad= \int_0^{\alpha_0}\int_0^1 \mathbb{E}[H|\alpha,d,\mathbf{x}_N]\frac{p(\mathbf{x}_N|\alpha,d)p(\alpha,d)}{p(\mathbf{x}_N)}\mathrm{d}\alpha\mathrm{d}d \\
&\qquad+ \int_{\alpha_0}^{\infty}\int_0^1 \mathbb{E}[H|\alpha,d,\mathbf{x}_N]\frac{p(\mathbf{x}_N|\alpha,d)p(\alpha,d)}{p(\mathbf{x}_N)}\mathrm{d}\alpha\mathrm{d}d
\end{aligned}$$

If we let $\alpha_0 = K_N + 1$, by Lemma 7

$$\int_0^{\alpha_0}\int_0^1 \mathbb{E}[H|\alpha,d,\mathbf{x}_N]\frac{p(\mathbf{x}_N|\alpha,d)p(\alpha,d)}{p(\mathbf{x}_N)}\mathrm{d}\alpha\mathrm{d}d \to \mathbb{E}[H_{\text{plugin}}|\mathbf{x}_N].$$

Therefore, it remains to show that

$$\int_{\alpha_0}^{\infty}\int_0^1 \mathbb{E}[H|\alpha,d,\mathbf{x}_N]\frac{p(\mathbf{x}_N|\alpha,d)p(\alpha,d)}{p(\mathbf{x}_N)}\mathrm{d}\alpha\mathrm{d}d \to 0$$

For finite support distributions where $K_N \to K < \infty$, this is trivial. Hence, we only consider infinite support distributions where $K_N \to \infty$. In this case, there exists N_0 such that for all $N \geq N_0$, $p([0, K_N + 1], [0, 1)) \neq 0$.

Since $p(\alpha, d)$ has a decaying tail as $\alpha \to \infty$, $\exists N_0 \forall N \geq N_0$, $p(K_N + 1, d) \leq 1$, thus, it is sufficient demonstrate convergence under an improper prior $p(\alpha, d) = 1$.

Using

$$\mathbb{E}[H|\alpha,d,\mathbf{x}_N] \leq \psi_0(N+\alpha+1) \leq N+\alpha$$

we bound

$$\begin{aligned}
&\int_{\alpha_0}^{\infty}\int_0^1 \mathbb{E}[H|\alpha,d,\mathbf{x}_N]\frac{p(\mathbf{x}_N|\alpha,d)}{p(\mathbf{x}_N)}\mathrm{d}\alpha\mathrm{d}d \\
&\qquad\leq \frac{\int_{\alpha_0}^{\infty}\int_0^1 (N+\alpha-1)p(\mathbf{x}_N|\alpha,d)\mathrm{d}\alpha\mathrm{d}d}{p(\mathbf{x}_N)} \\
&\qquad\quad+ \frac{\int_{\alpha_0}^{\infty}\int_0^1 p(\mathbf{x}_N|\alpha,d)\mathrm{d}\alpha\mathrm{d}d}{p(\mathbf{x}_N)}.
\end{aligned}$$

We focus upon the first term on the RHS since its boundedness implies that of the smaller second term. Recall, that $p(\mathbf{x}) = \int_{\alpha=0}^{\infty}\int_{d=0}^{1} p(\mathbf{x}|\alpha,d)\,\mathrm{d}d\,\mathrm{d}\alpha$. We seek an upper bound for the numerator and a lower bound for $p(\mathbf{x}_N)$.

Upper Bound: First we integrate over d to find the upper bound of the numerator. (For the following display only we let $\gamma(d) = \prod_{i=1}^{K_N} \Gamma(n_i - d)$).

$$\begin{aligned}
&\int_{\alpha_0}^{\infty} \int_0^1 (N+\alpha-1) p(\mathbf{x}_N|\alpha, d) \mathrm{d}\alpha \mathrm{d}d \\
&= \int_{\alpha_0}^{\infty} \int_{d=0}^1 \frac{\left(\prod_{l=1}^{K_N-1}(\alpha+ld)\right)\gamma(d)\Gamma(1+\alpha)(N+\alpha-1)}{\Gamma(1-d)^{K_N}\Gamma(\alpha+N)} \,\mathrm{d}d\,\mathrm{d}\alpha \\
&\leq \int_{d=0}^1 \frac{\gamma(d)}{\Gamma(1-d)^{K_N}}\,\mathrm{d}d \int_{\alpha_0}^{\infty} \frac{\Gamma(\alpha+K_N)(N+\alpha-1)}{\Gamma(\alpha+N)}\,\mathrm{d}\alpha
\end{aligned}$$

Fortunately, the first integral on d will cancel with a term from the lower bound of $p(\mathbf{x}_N)$. Using[12] $\frac{(N+\alpha-1)\Gamma(\alpha+K_N)}{\Gamma(\alpha+N)} = \frac{\mathrm{Beta}(\alpha+K_N, N-K-1)}{\Gamma(N-K-1)}$,

$$\begin{aligned}
&\int_{\alpha_0}^{\infty} \frac{(N+\alpha-1)\Gamma(\alpha+K)}{\Gamma(\alpha+N)}\,\mathrm{d}\alpha \\
&= \frac{1}{\Gamma(N-K-1)} \int_{\alpha_0}^{\infty} \mathrm{Beta}(\alpha+K, N-K-1)\,\mathrm{d}\alpha \\
&= \frac{1}{\Gamma(N-K-1)} \int_{\alpha_0}^{\infty} \int_0^1 t^{\alpha+K-1}(1-t)^{N-K-2}\,\mathrm{d}t\,\mathrm{d}\alpha \\
&= \frac{1}{\Gamma(N-K-1)} \int_{t=0}^1 \frac{t^{\alpha_0+K-1}}{\log(\frac{1}{t})}(1-t)^{N-K-2}\,\mathrm{d}t \\
&\leq \frac{1}{\Gamma(N-K-1)} \int_{t=0}^1 \frac{t^{\alpha_0+K-1}}{(1-t)}(1-t)^{N-K-2}\,\mathrm{d}t \\
&= \frac{1}{\Gamma(N-K-1)} \mathrm{Beta}(\alpha_0+K, N-K-2) \\
&= \frac{1}{\Gamma(N-K-1)} \frac{\Gamma(\alpha_0+K)\Gamma(N-K-2)}{\Gamma(N+\alpha_0-2)} \\
&= \frac{\Gamma(\alpha_0+K)}{\Gamma(N+\alpha_0-2)(N-K-2)}
\end{aligned}$$

Lower Bound: Again, we first integrate d

$$\begin{aligned}
&\int_{\alpha=0}^{\infty} \int_{d=0}^1 p(\mathbf{x}|\alpha, d)\,\mathrm{d}d\,\mathrm{d}\alpha \\
&= \int_{\alpha=0}^{\infty} \int_{d=0}^1 \frac{\left(\prod_{l=1}^{K-1}(\alpha+ld)\right)\left(\prod_{i=1}^{K}\Gamma(n_i-d)\right)\Gamma(1+\alpha)}{\Gamma(1-d)^K\Gamma(\alpha+N)}\,\mathrm{d}d\,\mathrm{d}\alpha \\
&= \int_{d=0}^1 \frac{\left(\prod_{i=1}^{K}\Gamma(n_i-d)\right)}{\Gamma(1-d)^K}\,\mathrm{d}d \int_{\alpha=0}^{\infty} \frac{\alpha^{K-1}\Gamma(1+\alpha)}{\Gamma(\alpha+N)}\,\mathrm{d}\alpha
\end{aligned}$$

So, since $\frac{\Gamma(1+\alpha)}{\Gamma(\alpha+N)} = \frac{\mathrm{Beta}(1+\alpha, N-1)}{\Gamma(N-1)}$, then

12. Note that in the argument for the inequalities we use K rather than K_N for clarity of notation.

$$
\begin{aligned}
\Gamma(N-1)\int_{\alpha=0}^{\infty}\frac{\alpha^{K-1}\Gamma(1+\alpha)}{\Gamma(\alpha+N)}\,\mathrm{d}\alpha& \\
&\geq \int_{\alpha=0}^{\infty}\alpha^{K-1}\mathrm{Beta}(1+\alpha,N-1)\,\mathrm{d}\alpha \\
&= \int_{\alpha=0}^{\infty}\alpha^{K-1}\int_{t=0}^{1}t^{\alpha}(1-t)^{N-2}\,\mathrm{d}t\,\mathrm{d}\alpha \\
&= \int_{t=0}^{1}(1-t)^{N-2}\int_{\alpha=0}^{\infty}\alpha^{K-1}t^{\alpha}\,\mathrm{d}\alpha\,\mathrm{d}t \\
&= \Gamma(K)\int_{t=0}^{1}(1-t)^{N-2}\log\left(\frac{1}{t}\right)^{-K}\mathrm{d}t \\
&\geq \Gamma(K)\int_{t=0}^{1}(1-t)^{N-K-2}t^{K}\,\mathrm{d}t \\
&= \Gamma(K)\mathrm{Beta}(N-K-1,K+1)
\end{aligned}
$$

where we've used the fact that $\log(\frac{1}{t})^{-1} \geq \frac{t}{1-t}$. Finally, we obtain the bound

$$
\int_{\alpha=0}^{\infty}\frac{\alpha^{K_N-1}\Gamma(1+\alpha)}{\Gamma(\alpha+N)}\,\mathrm{d}\alpha \geq \frac{\Gamma(K)\Gamma(N-K-1)\Gamma(K+1)}{\Gamma(N-1)\Gamma(N)}.
$$

Now, we apply the upper and lower bounds to bound PYM. We have

$$
\begin{aligned}
&\frac{\int_{\alpha_0}^{\infty}\int_{0}^{1}(N+\alpha-1)p(\mathbf{x}_N|\alpha,d)\mathrm{d}\alpha\mathrm{d}d}{p(\mathbf{x}_N)} \\
&\leq \frac{\Gamma(\alpha_0+K_N)}{(N-K_N-2)\Gamma(N+\alpha_0-2)}\frac{\Gamma(N-1)\Gamma(N)}{\Gamma(K_N)\Gamma(N-K_N-1)\Gamma(K_N+1)} \\
&= \frac{1}{(N-K_N-2)}\frac{\Gamma(\alpha_0+K_N)}{\Gamma(K_N)}\frac{\Gamma(N-1)}{\Gamma(N+\alpha_0-2)} \\
&\qquad\qquad \times \frac{\Gamma(N)}{\Gamma(N-K_N-1)\Gamma(K_N+1)} \\
&\to \frac{N}{(N-K_N-2)}\left(\frac{K_N}{N}\right)^{\alpha_0}\frac{N^{N-1/2}}{(N-K_N-1)^{N-K_N-3/2}(K_N+1)^{K_N+1/2}} \\
&= \frac{N^2}{(K_N+1)^{1/2}(N-K_N-2)}\left(\frac{K_N}{N}\right)^{\alpha_0}\left(\frac{N}{N-K_N-1}\right)^{N-3/2} \\
&\qquad\qquad \times \left(\frac{N-K_N-1}{K_N+1}\right)^{K_N} \\
&\to \frac{N}{(K_N+1)^{1/2}}\left(\frac{K_N}{N}\right)^{\alpha_0}\left(\frac{N}{K_N}\right)^{K_N}
\end{aligned}
$$

Where we have applied the asymptotic expansion for the Beta function

$$
\mathrm{Beta}(x,y) \sim \sqrt{2\pi}\frac{x^{x-\frac{1}{2}}y^{y-\frac{1}{2}}}{(x+y)^{x+y-\frac{1}{2}}},
$$

a consequence of Stirling's formula. Finally, we take $\alpha_0 = K_N + (C+1)/2$ so that the limit becomes

$$\begin{aligned}&\to \frac{N}{K_N^{1/2}} \left(\frac{K_N}{N}\right)^{(C+1)/2} \\ &= \frac{K_N^{C/2}}{N^{C/2-1/2}}\end{aligned}$$

which tends to 0 with increasing N since, by assumption, $K_N = o(N^{1-1/C})$. ■

Appendix D. Results on Unimodality of Evidence

Theorem 8 (Unimodal evidence on d) *The evidence $p(\mathbf{x}|d,\alpha)$ given by (17) has only one local maximum (unimodal) for a fixed $\alpha > 0$.*

Proof Equivalently, we show that the log evidence is unimodal.

$$\begin{aligned}L &= \log p(\mathbf{x}|d,\alpha) \\ &= \sum_{l=1}^{K-1} \log(\alpha + ld) + \sum_{i=1}^{K} \log\Gamma(n_i - d) + \log\Gamma(1+\alpha) - K\log\Gamma(1-d) - \log\Gamma(\alpha+N)\end{aligned}$$

It is sufficient to show that the partial derivative w.r.t. d has at most one positive root.

$$\begin{aligned}\frac{\partial L}{\partial d} &= \sum_{l=1}^{K-1} \frac{l}{\alpha + ld} - \sum_{i=1}^{K} \left(\psi_0(n_i - d) - \psi_0(1-d)\right) \\ &= \sum_{l=1}^{K-1} \frac{l}{\alpha + ld} + \sum_{i=1}^{K} \sum_{j=1}^{n_i - 1} \frac{1}{d-j}\end{aligned}$$

Note that as $d \to 1$, the derivative tends to $-\infty$. Combined with the observation that it is a linear combination of convex functions, there is at most one root for $\frac{\partial L}{\partial d} = 0$. ■

Theorem 9 (Unimodal evidence on α) *The evidence $p(\mathbf{x}|d,\alpha)$ given by (17) has only one local maximum (unimodal), on the region $\alpha > 0$, for a fixed d.*

Proof Similar to Theorem 8, it is sufficient to show that the partial derivative w.r.t. α has at most one positive root.

$$\begin{aligned}\frac{\partial L}{\partial \alpha} &= \sum_{l=1}^{K-1} \frac{1}{\alpha + ld} + \psi_0(1+\alpha) - \psi_0(\alpha+N) \\ &= \sum_{l=1}^{K-1} \frac{1}{\alpha + ld} - \sum_{j=1}^{N-1} \frac{1}{j+\alpha}\end{aligned}$$

Let $\alpha = \frac{1}{x}$ be a root, then,

$$\sum_{i=1}^{K-1} \frac{1}{1+xid} = \sum_{j=1}^{N-1} \frac{1}{1+xj}. \tag{40}$$

Note that since $xid < xi$, $\frac{1}{1+xid} > \frac{1}{1+xi}$ for $1 \leq i \leq K-1$. Therefore, we can split the equality as follows:

$$f_i(x) = a_i \frac{1}{1+xid} = \frac{1}{1+xj} = g_i(x) \qquad \text{for } i \leq K-1 \tag{41}$$

$$f_{ij}(x) = b_{ij} \frac{1}{1+xid} = c_{ij} \frac{1}{1+xj} = g_{ij}(x) \qquad \text{for } i \leq K-1 \text{ and } K < j < N \tag{42}$$

where $0 \leq a_i, b_{ij}, c_{ij} \leq 1$, $\forall i < K$, $a_i + \sum_j b_{ij} = 1$, and $\forall j < N$, $\sum_i c_{ij} = 1$. Fix a_i, b_{ij}, c_{ij}'s, and now suppose $\frac{1}{y} > \frac{1}{x} > 0$ is another positive root. Then, we observe the following strict inequalities due to $0 \leq d < 1$,

$$\frac{f_i(x)}{f_i(y)} = \frac{1+yid}{1+xid} < \frac{1+yj}{1+xj} = \frac{g_i(x)}{g_i(y)} \qquad \text{for } i \leq K-1 \tag{43}$$

$$\frac{f_{ij}(x)}{f_{ij}(y)} = \frac{1+yid}{1+xid} < \frac{1+yj}{1+xj} = \frac{g_{ij}(x)}{g_{ij}(y)} \qquad \text{for } i \leq K-1 \text{ and } K < j < N \tag{44}$$

Using Lemma 10 to put the sum back together, we obtain,

$$\sum_{i=1}^{K-1} \frac{1}{1+yid} > \sum_{j=1}^{N-1} \frac{1}{1+yj}. \tag{45}$$

which is a contradiction to our assumption that $\frac{1}{y}$ is a positive root. ■

Lemma 10 *If $f_j, g_j > 0$, $f_j(x) = g_j(x)$ and $\frac{f_j(y)}{f_j(x)} > \frac{g_j(y)}{g_j(x)}$ for all j, then $\sum_j f_j(y) > \sum_j g_j(y)$.*

References

A. Antos and I. Kontoyiannis. Convergence properties of functional estimates for discrete distributions. *Random Structures & Algorithms*, 19(3-4):163–193, 2001.

E. Archer, I. M. Park, and J. Pillow. Bayesian estimation of discrete entropy with mixtures of stick-breaking priors. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, pages 2024–2032. MIT Press, Cambridge, MA, 2012.

E. Archer, I. M. Park, and J. W. Pillow. Bayesian entropy estimation for binary spike train data using parametric prior knowledge. In *Advances in Neural Information Processing Systems*, 2013.

R. Barbieri, L. Frank, D. Nguyen, M. Quirk, V. Solo, M. Wilson, and E. Brown. Dynamic analyses of information encoding in neural ensembles. *Neural Computation*, 16:277–307, 2004.

J. Boyd. Exponentially convergent Fourier-Chebshev quadrature schemes on bounded and infinite intervals. *Journal of Scientific Computing*, 2(2):99–109, 1987.

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(4):429–443, 2003.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

T. Dudok de Wit. When do finite sample effects significantly affect entropy estimates? *Eur. Phys. J. B - Cond. Matter and Complex Sys.*, 11(3):513–516, October 1999.

W. J. Ewens. Population genetics theory-the past and the future. In *Mathematical and Statistical Developments of Evolutionary Theory*, pages 177–227. Springer, 1990.

M. Farach, M. Noordewier, S. Savari, L. Shepp, A. Wyner, and J. Ziv. On the entropy of DNA: Algorithms and measurements based on memory and rapid convergence. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 48–57. Society for Industrial and Applied Mathematics, 1995.

S. Favaro, A. Lijoi, R. H. Mena, and I. Prünster. Bayesian non-parametric inference for species variety with a two-parameter Poisson-Dirichlet process prior. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(5):993–1008, November 2009.

T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2): 209–230, 1973.

A. Gnedin, B. Hansen, and J. Pitman. Notes on the occupancy problem with infinitely many boxes: general asymptotics and power laws. *Probability Surveys*, 4:146–171, 2007.

S. Goldwater, T. Griffiths, and M. Johnson. Interpolating between types and tokens by estimating power-law generators. In *Advances in Neural Information Processing Systems*, page 459. MIT Press, Cambridge, MA, 2006.

P. Grassberger. Entropy estimates from insufficient samplings. *arXiv preprint*, January 2008.

J. Hausser and K. Strimmer. Entropy inference and the James-Stein estimator, with application to nonlinear gene association networks. *The Journal of Machine Learning Research*, 10:1469–1484, 2009.

M. Hutter. Distribution of mutual information. In *Advances in Neural Information Processing Systems*, pages 399–406. MIT Press, Cambridge, MA, 2002.

H. Ishwaran and L. James. Generalized weighted Chinese restaurant processes for species sampling mixture models. *Statistica Sinica*, 13(4):1211–1236, 2003.

H. Ishwaran and M. Zarepour. Exact and approximate sum representations for the Dirichlet process. *Canadian Journal of Statistics*, 30(2):269–283, 2002.

H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161–173, March 2001.

J. Kingman. Random discrete distributions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 37(1):1–22, 1975.

K. C. Knudson and J. W. Pillow. Spike train entropy-rate estimation using hierarchical dirichlet process priors. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, pages 2076–2084. Curran Associates, Inc., 2013.

C. Letellier. Estimating the shannon entropy: recurrence plots versus symbolic dynamics. *Physical Review Letters*, 96(25):254102, 2006.

G. Miller. Note on the bias of information estimates. *Information Theory in Psychology: Problems and Methods*, 2:95–100, 1955.

T. Minka. Estimating a Dirichlet distribution. Technical report, MIT, 2003.

I. Nemenman. Coincidences and estimation of entropies of random variables with large cardinalities. *Entropy*, 13(12):2013–2023, 2011.

I. Nemenman, F. Shafee, and W. Bialek. Entropy and inference, revisited. In *Advances in Neural Information Processing Systems*, pages 471–478. MIT Press, Cambridge, MA, 2002.

I. Nemenman, W. Bialek, and R. Van Steveninck. Entropy and information in neural spike trains: Progress on the sampling problem. *Physical Review E*, 69(5):056111, 2004.

M. Newman. Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46(5):323–351, 2005.

L. Paninski. Estimation of entropy and mutual information. *Neural Computation*, 15:1191–1253, 2003.

S. Panzeri and A. Treves. Analytical estimates of limited sampling biases in different information measures. *Network: Computation in Neural Systems*, 7:87–107, 1996.

M. Perman, J. Pitman, and M. Yor. Size-biased sampling of poisson point processes and excursions. *Probability Theory and Related Fields*, 92(1):21–39, March 1992.

J. W. Pillow, L. Paninski, V. J. Uzzell, E. P. Simoncelli, and E. J. Chichilnisky. Prediction and decoding of retinal ganglion cell responses with a probabilistic spiking model. *The Journal of Neuroscience*, 25:11003–11013, 2005.

J. Pitman. Random discrete distributions invariant under size-biased permutation. *Advances in Applied Probability*, pages 525–539, 1996.

J. Pitman and M. Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, 25(2):855–900, 1997.

E. T. Rolls, M. J. Tovée, and S. Panzeri. The neurophysiology of backward visual masking: Information analysis. *Journal of Cognitive Neuroscience*, 11(3):300–311, May 1999.

K. H. Schindler, M. Palus, M. Vejmelka, and J. Bhattacharya. Causality detection based on information-theoretic approaches in time series analysis. *Physics Reports*, 441:1–46, 2007.

J. Shlens, M. B. Kennel, H. D. I. Abarbanel, and E. J. Chichilnisky. Estimating information rates with confidence intervals in neural spike trains. *Neural Computation*, 19(7):1683–1719, Jul 2007.

R. Strong, S. Koberle, de Ruyter van Steveninck R., and W. Bialek. Entropy and information in neural spike trains. *Physical Review Letters*, 80:197–202, 1998.

Y. Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 985–992, 2006.

A. Treves and S. Panzeri. The upward bias in measures of information derived from limited data samples. *Neural Computation*, 7:399–407, 1995.

V. Q. Vu, B. Yu, and R. E. Kass. Coverage-adjusted entropy estimation. *Statistics in Medicine*, 26 (21):4039–4060, 2007.

D. H. Wolpert and S. DeDeo. Estimating functions of distributions defined over spaces of unknown size. *Entropy*, 15(11):4668–4699, 2013.

D. Wolpert and D. Wolf. Estimating functions of probability distributions from a finite set of samples. *Physical Review E*, 52(6):6841–6854, 1995.

Z. Zhang. Entropy estimation in Turing's perspective. *Neural Computation*, pages 1–22, 2012.

G. K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley Press, 1949.

Journal of Machine Learning Research 15 (2014) 2869-2909 Submitted 3/14; Revised 4/14; Published 10/14

Confidence Intervals and Hypothesis Testing for High-Dimensional Regression

Adel Javanmard ADELJ@STANFORD.EDU
Department of Electrical Engineering
Stanford University
Stanford, CA 94305, USA

Andrea Montanari MONTANAR@STANFORD.EDU
Department of Electrical Engineering and Department of Statistics
Stanford University
Stanford, CA 94305, USA

Editor: Nicolai Meinshausen

Abstract

Fitting high-dimensional statistical models often requires the use of non-linear parameter estimation procedures. As a consequence, it is generally impossible to obtain an exact characterization of the probability distribution of the parameter estimates. This in turn implies that it is extremely challenging to quantify the *uncertainty* associated with a certain parameter estimate. Concretely, no commonly accepted procedure exists for computing classical measures of uncertainty and statistical significance as confidence intervals or p-values for these models.

We consider here high-dimensional linear regression problem, and propose an efficient algorithm for constructing confidence intervals and p-values. The resulting confidence intervals have nearly optimal size. When testing for the null hypothesis that a certain parameter is vanishing, our method has nearly optimal power.

Our approach is based on constructing a 'de-biased' version of regularized M-estimators. The new construction improves over recent work in the field in that it does not assume a special structure on the design matrix. We test our method on synthetic data and a high-throughput genomic data set about riboflavin production rate, made publicly available by Bühlmann et al. (2014).

Keywords: hypothesis testing, confidence intervals, LASSO, high-dimensional models, bias of an estimator

1. Introduction

It is widely recognized that modern statistical problems are increasingly high-dimensional, i.e., require estimation of more parameters than the number of observations/samples. Examples abound from signal processing (Lustig et al., 2008), to genomics (Peng et al., 2010), collaborative filtering (Koren et al., 2009) and so on. A number of successful estimation techniques have been developed over the last ten years to tackle these problems. A widely applicable approach consists in optimizing a suitably regularized likelihood function. Such estimators are, by necessity, non-linear and non-explicit (they are solution of certain optimization problems).

The use of non-linear parameter estimators comes at a price. In general, it is impossible to characterize the distribution of the estimator. This situation is very different from the one of classical statistics in which either exact characterizations are available, or asymptotically exact ones can be derived from large sample theory (Van der Vaart, 2000). This has an important and very concrete consequence. In classical statistics, generic and well accepted procedures are available for characterizing the uncertainty associated to a certain parameter estimate in terms of confidence intervals or p-values (Wasserman, 2004; Lehmann and Romano, 2005). However, no analogous procedures exist in high-dimensional statistics.

In this paper we develop a computationally efficient procedure for constructing confidence intervals and p-values for a broad class of high-dimensional regression problems. The salient features of our procedure are:

(*i*) Our approach guarantees nearly optimal confidence interval sizes and testing power.

(*ii*) It is the first one to achieve this goal under essentially no assumptions beyond the standard conditions for high-dimensional consistency.

(*iii*) It allows for a streamlined analysis with respect to earlier work in the same area.

For the sake of clarity, we will focus our presentation on the case of linear regression, under Gaussian noise. Section 4 provides a detailed study of the case of non-Gaussian noise. A preliminary report on our results was presented in NIPS 2013 (Javanmard and Montanari, 2013a), which also discusses generalizations of the same approach to generalized linear models, and regularized maximum likelihood estimation.

In a linear regression model, we are given n i.i.d. pairs $(Y_1, X_1), (Y_2, X_2), \dots, (Y_n, X_n)$, with vectors $X_i \in \mathbb{R}^p$ and response variables Y_i given by

$$Y_i \,=\, \langle \theta_0, X_i \rangle + W_i \,, \qquad W_i \sim \mathsf{N}(0, \sigma^2) \,. \tag{1}$$

Here $\theta_0 \in \mathbb{R}^p$ and $\langle \, \cdot \, , \, \cdot \, \rangle$ is the standard scalar product in $\mathbb{R}^p$. In matrix form, letting $Y = (Y_1, \dots, Y_n)^{\mathsf{T}}$ and denoting by $\mathbf{X}$ the design matrix with rows $X_1^{\mathsf{T}}, \dots, X_n^{\mathsf{T}}$, we have

$$Y \,=\, \mathbf{X}\, \theta_0 + W \,, \qquad W \sim \mathsf{N}(0, \sigma^2 \mathrm{I}_{n\times n}) \,. \tag{2}$$

The goal is to estimate the unknown (but fixed) vector of parameters $\theta_0 \in \mathbb{R}^p$.

In the classic setting, $n \gg p$ and the estimation method of choice is ordinary least squares yielding $\widehat{\theta}^{\mathrm{OLS}} = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}Y$. In particular $\widehat{\theta}^{\mathrm{OLS}}$ is Gaussian with mean θ_0 and covariance $\sigma^2(\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}$. This directly allows to construct confidence intervals.[1]

In the high-dimensional setting where $p > n$, the matrix $(\mathbf{X}^{\mathsf{T}}\mathbf{X})$ is rank deficient and one has to resort to biased estimators. A particularly successful approach is the LASSO (Tibshirani, 1996; Chen and Donoho, 1995) which promotes sparse reconstructions through an ℓ_1 penalty:

$$\widehat{\theta}^{n}(Y, \mathbf{X}; \lambda) \equiv \arg\min_{\theta \in \mathbb{R}^p} \Big\{ \frac{1}{2n} \|Y - \mathbf{X}\theta\|_2^2 + \lambda \|\theta\|_1 \Big\} \,. \tag{3}$$

1. For instance, letting $Q \equiv (\mathbf{X}^{\mathsf{T}}\mathbf{X}/n)^{-1}$, $\widehat{\theta}_i^{\mathrm{OLS}} - 1.96\sigma\sqrt{Q_{ii}/n}, \widehat{\theta}_i^{\mathrm{OLS}} + 1.96\sigma\sqrt{Q_{ii}/n}]$ is a 95% confidence interval (Wasserman, 2004).

Algorithm 1 Unbiased estimator for θ_0 in high-dimensional linear regression models

Input: Measurement vector y, design matrix $\mathbf{X}$, parameters λ, μ.
Output: Unbiased estimator $\widehat{\theta}^u$.

1: Let $\widehat{\theta}^n = \widehat{\theta}^n(Y, \mathbf{X}; \lambda)$ be the LASSO estimator as per Equation (3).
2: Set $\widehat{\Sigma} \equiv (\mathbf{X}^\mathsf{T}\mathbf{X})/n$.
3: **for** $i = 1, 2, \ldots, p$ **do**
4: Let m_i be a solution of the convex program:

$$\begin{aligned} &\text{minimize} && m^\mathsf{T}\widehat{\Sigma}m \\ &\text{subject to} && \|\widehat{\Sigma}m - e_i\|_\infty \le \mu\,, \end{aligned} \tag{4}$$

where $e_i \in \mathbb{R}^p$ is the vector with one at the i-th position and zero everywhere else.
5: Set $M = (m_1, \ldots, m_p)^\mathsf{T}$. If any of the above problems is not feasible, then set $M = \mathrm{I}_{p\times p}$.
6: Define the estimator $\widehat{\theta}^u$ as follows:

$$\widehat{\theta}^u = \widehat{\theta}^n(\lambda) + \frac{1}{n} M\mathbf{X}^\mathsf{T}(Y - \mathbf{X}\widehat{\theta}^n(\lambda)) \tag{5}$$

In case the right hand side has more than one minimizer, one of them can be selected arbitrarily for our purposes. We will often omit the arguments Y, $\mathbf{X}$, as they are clear from the context.

We denote by $S \equiv \mathrm{supp}(\theta_0)$ the support of $\theta_0 \in \mathbb{R}^p$, defined as

$$\mathrm{supp}(\theta_0) \equiv \{i \in [p] : \theta_{0,i} \neq 0\}\,,$$

where we use the notation $[p] = \{1, \ldots, p\}$. We further let $s_0 \equiv |S|$. A copious theoretical literature (Candès and Tao, 2005; Bickel et al., 2009; Bühlmann and van de Geer, 2011) shows that, under suitable assumptions on $\mathbf{X}$, the LASSO is nearly as accurate as if the support S was known *a priori*. Namely, for $n = \Omega(s_0 \log p)$, we have $\|\widehat{\theta}^n - \theta_0\|_2^2 = O(s_0\sigma^2(\log p)/n)$.

As mentioned above, these remarkable properties come at a price. Deriving an exact characterization for the distribution of $\widehat{\theta}^n$ is not tractable in general, and hence there is no simple procedure to construct confidence intervals and p-values. A closely related property is that $\widehat{\theta}^n$ is biased, an unavoidable property in high dimension, since a point estimate $\widehat{\theta}^n \in \mathbb{R}^p$ must be produced from data in lower dimension $Y \in \mathbb{R}^n$, $n < p$. We refer to Section 2.2 for further discussion of this point.

In order to overcome this challenge, we construct a de-biased estimator from the LASSO solution. The de-biased estimator is given by the simple formula $\widehat{\theta}^u = \widehat{\theta}^n + (1/n)\, M\mathbf{X}^\mathsf{T}(Y - \mathbf{X}\widehat{\theta}^n)$, as in Equation (5). The basic intuition is that $\mathbf{X}^\mathsf{T}(Y - \mathbf{X}\widehat{\theta}^n)/(n\lambda)$ is a subgradient of the ℓ_1 norm at the LASSO solution $\widehat{\theta}^n$. By adding a term proportional to this subgradient, our procedure compensates the bias introduced by the ℓ_1 penalty in the LASSO.

We will prove in Section 2.1 that $\widehat{\theta}^u$ is approximately Gaussian, with mean θ_0 and covariance $\sigma^2(M\widehat{\Sigma}M)/n$, where $\widehat{\Sigma} = (\mathbf{X}^\mathsf{T}\mathbf{X}/n)$ is the empirical covariance of the feature vectors. This result allows to construct confidence intervals and p-values in complete analogy with classical statistics procedures. For instance, letting $Q \equiv M\widehat{\Sigma}M$, $[\widehat{\theta}_i^u - 1.96\sigma\sqrt{Q_{ii}/n}, \widehat{\theta}_i^u + 1.96\sigma\sqrt{Q_{ii}/n}]$ is a 95% confidence interval. The size of this interval is of order $\sigma/\sqrt{n}$, which

is the optimal (minimum) one, i.e., the same that would have been obtained by knowing *a priori* the support of θ_0. In practice the noise standard deviation is not known, but σ can be replaced by any consistent estimator $\widehat{\sigma}$ (see Section 3 for more details on this).

A key role is played by the matrix $M \in \mathbb{R}^{p\times p}$ whose function is to 'decorrelate' the columns of $\mathbf{X}$. We propose here to construct M by solving a convex program that aims at optimizing two objectives. One one hand, we try to control $|M\widehat{\Sigma} - \mathrm{I}|_\infty$ (here and below $|\cdot|_\infty$ denotes the entrywise ℓ_∞ norm) which, as shown in Theorem 8, controls the non-Gaussianity and bias of $\widehat{\theta}^u$. On the other, we minimize $[M\widehat{\Sigma}M]_{i,i}$, for each $i \in [p]$, which controls the variance of $\widehat{\theta}^u_i$.

The idea of constructing a de-biased estimator of the form $\widehat{\theta}^u = \widehat{\theta}^n + (1/n)\, M\mathbf{X}^{\mathsf{T}}(Y - \mathbf{X}\widehat{\theta}^n)$ was used by the present authors in Javanmard and Montanari (2013b), that suggested the choice $M = c\Sigma^{-1}$, with $\Sigma = \mathbb{E}\{X_1 X_1^{\mathsf{T}}\}$ the population covariance matrix and c a positive constant. A simple estimator for Σ was proposed for sparse covariances, but asymptotic validity and optimality were proven only for uncorrelated Gaussian designs (i.e., Gaussian $\mathbf{X}$ with $\Sigma = \mathrm{I}$). Van de Geer, Bülhmann, Ritov and Dezeure (van de Geer et al., 2014) used the same construction with M an estimate of Σ^{-1} which is appropriate for sparse inverse covariances. These authors prove semi-parametric optimality in a non-asymptotic setting, provided the sample size is at least $n = \Omega((s_0 \log p)^2)$.

From a technical point of view, our proof starts from a simple decomposition of the de-biased estimator $\widehat{\theta}^u$ into a Gaussian part and an error term, already used in van de Geer et al. (2014). However, departing radically from earlier work, we realize that M need not be a good estimator of Σ^{-1} in order for the de-biasing procedure to work. We instead set M as to minimize the error term and the variance of the Gaussian term. As a consequence of this choice, our approach applies to general covariance structures Σ. By contrast, earlier approaches applied only to sparse Σ, as in Javanmard and Montanari (2013b), or sparse Σ^{-1} as in van de Geer et al. (2014). The only assumptions we make on Σ are the standard compatibility conditions required for high-dimensional consistency (Bühlmann and van de Geer, 2011). A detailed comparison of our results with the ones of van de Geer et al. (2014) can be found in Section 2.3.

1.1 Organization of the Paper

Our presentation is organized as follows.

Section 2 considers a general debiased estimator of the form $\widehat{\theta}^u = \widehat{\theta}^n + (1/n)\, M\mathbf{X}^{\mathsf{T}}(Y - \mathbf{X}\widehat{\theta}^n)$. We introduce a figure of merit of the pair $M, \mathbf{X}$, termed the generalized coherence parameter $\mu_*(\mathbf{X}; M)$. We show that, if the generalized coherence is small, then the debiasing procedure is effective (for a given deterministic design), see Theorem 6. We then turn to random designs, and show that the generalized coherence parameter can be made as small as $\sqrt{(\log p)/n}$, though a convex optimization procedure for computing M. This results in a bound on the bias of $\widehat{\theta}^u$, cf. Theorem 8: the largest entry of the bias is of order $(s_0 \log p)/n$. This must be compared with the standard deviation of $\widehat{\theta}^u_i$, which is of order $\sigma/\sqrt{n}$. The conclusion is that, for $s_0 = o(\sqrt{n}/\log p)$, the bias of $\widehat{\theta}^u$ is negligible.

Section 3 applies these distributional results to deriving confidence intervals and hypothesis testing procedures for low-dimensional marginals of θ_0. The basic intuition is that $\widehat{\theta}^u$ is approximately Gaussian with mean θ_0, and known covariance structure. Hence standard

optimal tests can be applied. We prove a general lower bound on the power of our testing procedure, in Theorem 16. In the special case of Gaussian random designs with i.i.d. rows, we can compare this with the upper bound proved in Javanmard and Montanari (2013b), cf. Theorem 17. As a consequence, the asymptotic efficiency of our approach is constant-optimal. Namely, it is lower bounded by a constant $1/\eta_{\Sigma,s_0}$ which is bounded away from 0, cf. Theorem 18. (For instance $\eta_{\mathrm{I},s_0} = 1$, and η_{Σ,s_0} is always upper bounded by the condition number of Σ.)

Section 4 uses the central limit theorem for triangular arrays to generalize the above results to non-Gaussian noise.

Section 5 illustrates the above results through numerical simulations both on synthetic and on real data. In the interest of reproducibility, an R implementation of our algorithm is available at `http://www.stanford.edu/~montanar/sslasso/`.

Note that our proofs require stricter sparsity s_0 (or larger sample size n) than required for consistent estimation. We assume $s_0 = o(\sqrt{n}/\log p)$ instead of $s_0 = o(n/\log p)$ (Candès and Tao, 2007; Bickel et al., 2009; Bühlmann and van de Geer, 2011). The same assumption is made in van de Geer et al. (2014), on top of additional assumptions on the sparsity of Σ^{-1}.

It is currently an open question whether successful hypothesis testing can be performed under the weaker assumption $s_0 = o(n/\log p)$. We refer to Javanmard and Montanari (2013c) for preliminary work in that direction. The barrier at $s_0 = o(\sqrt{n}/\log p)$ is possibly related to an analogous assumption that arises in Gaussian graphical models selection (Ren et al., 2013).

1.2 Further Related Work

The theoretical literature on high-dimensional statistical models is vast and rapidly growing. Estimating sparse linear regression models is the most studied problem in this area, and a source of many fruitful ideas. Limiting ourselves to linear regression, earlier work investigated prediction error (Greenshtein and Ritov, 2004), model selection properties (Meinshausen and Bühlmann, 2006; Zhao and Yu, 2006; Wainwright, 2009; Candès and Plan, 2009), ℓ_2 consistency (Candès and Tao, 2005; Bickel et al., 2009). . Of necessity, we do not provide a complete set of references, and instead refer the reader to Bühlmann and van de Geer (2011) for an in-depth introduction to this area.

The problem of quantifying statistical significance in high-dimensional parameter estimation is, by comparison, far less understood. Zhang and Zhang (2014) and Bühlmann (2013) proposed hypothesis testing procedures under restricted eigenvalue or compatibility conditions (Bühlmann and van de Geer, 2011). These papers provide deterministic guarantees but, in order to achieve a certain target significance level α and power $1-\beta$, they require $|\theta_{0,i}| \geq c\max\{\sigma s_0 \log p/\, n, \sigma/\sqrt{n}\}$. The best lower bound (Javanmard and Montanari, 2013b) shows that any such test requires instead $|\theta_{0,i}| \geq c(\alpha,\beta)\sigma/\sqrt{n}$. (The lower bound of Javanmard and Montanari 2013b is reproduced as Theorem 17 here, for the reader's convenience.)

In other words, the guarantees of Zhang and Zhang (2014); Bühlmann (2013) can be suboptimal by a factor as large as $\sqrt{s_0}$. Equivalently, in order for the coefficient $\theta_{0,i}$ to be detectable with appreciable probability, it needs to be larger than the overall ℓ_2 error. Here

we will propose a test that, for random designs, achieves significance level α and power $1-\beta$ for $|\theta_{0,i}| \geq c'(\alpha,\beta)\sigma/\sqrt{n}$.

Lockhart et al. (2014) develop a test for the hypothesis that a newly added coefficient along the LASSO regularization path is irrelevant. This however does not allow to test arbitrary coefficients at a given value of λ, which is instead the problem addressed in this paper. These authors further assume that the current LASSO support contains the actual support $\mathrm{supp}(\theta_0)$ and that the latter has bounded size.

Belloni et al. (2014, 2013) consider inference in a regression model with high-dimensional data. In this model the response variable relates to a scalar main regressor and a p-dimensional control vector. The main regressor is of primary interest and the control vector is treated as nuisance component. Assuming that the control vector is s_0-sparse, the authors propose a method to construct confidence regions for the parameter of interest under the sample size requirement $(s_0^2 \log p)/n \to 0$. The proposed method is shown to attain the semi-parametric efficiency bounds for this class of models. The key modeling assumption in this paper is that the scalar regressor of interest is random, and depends linearly on the p-dimensional control vector, with a sparse coefficient vector (with sparsity again of order $o(\sqrt{n/\log p})$. This assumption is closely related to the sparse inverse covariance assumption of van de Geer et al. (2014) (with the difference that only one regressor is tested).

Finally, resampling methods for hypothesis testing were studied in Meinshausen and Bühlmann (2010); Minnier et al. (2011). These methods are perturbation-based procedures to approximate the distribution of a general class of penalized parameter estimates for the case $n > p$. The idea is to consider the minimizer of a stochastically perturbed version of the regularized objective function, call it $\tilde{\theta}$, and characterize the limiting distribution of the regularized estimator $\widehat{\theta}$ in terms of the distribution of $\tilde{\theta}$. In order to estimate the latter, a large number of random samples of the perturbed objective function are generated, and for each sample the minimizer is computed. Finally the theoretical distribution of $\tilde{\theta}$ is approximated by the empirical distribution of these minimizers.

After the present paper was submitted for publication, we became aware that Dezeure and Bühlmann (2013) had independently worked on similar ideas.

1.3 Preliminaries and Notations

In this section we introduce some basic definitions used throughout the paper, starting with simple notations.

For a matrix A and set of indices I, J, we let $A_{I,J}$ denote the submatrix formed by the rows in I and columns in J. Also, $A_{I,\cdot}$ (resp. $A_{\cdot,I}$) denotes the submatrix containing just the rows (reps. columns) in I. Likewise, for a vector v, v_I is the restriction of v to indices in I. We use the shorthand $A^{-1}_{I,J} = (A^{-1})_{I,J}$. In particular, $A^{-1}_{i,i} = (A^{-1})_{i,i}$. The maximum and the minimum singular values of A are respectively denoted by $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$. We write $\|v\|_p$ for the standard ℓ_p norm of a vector v, i.e., $\|v\|_p = (\sum_i |v_i|^p)^{1/p}$. and $\|v\|_0$ for the number of nonzero entries of v. For a matrix A, $\|A\|_p$ is the ℓ_p operator norm, and $|A|_p$ is the elementwise ℓ_p norm. For a vector v, $\mathrm{supp}(v)$ represents the positions of nonzero entries of v. Throughout, $\Phi(x) \equiv \int_{-\infty}^{x} e^{-t^2/2}\mathrm{d}t/\sqrt{2\pi}$ denotes the CDF of the standard normal distribution. Finally, *with high probability* (w.h.p) means with probability converging to one as $n \to \infty$.

We let $\widehat{\Sigma} \equiv \mathbf{X}^{\mathsf{T}}\mathbf{X}/n$ be the sample covariance matrix. For $p > n$, $\widehat{\Sigma}$ is always singular. However, we may require $\widehat{\Sigma}$ to be nonsingular for a restricted set of directions.

Definition 1 *Given a symmetric matrix $\widehat{\Sigma} \in \mathbb{R}^{p\times p}$ and a set $S \subseteq [p]$, the corresponding* compatibility constant *is defined as*

$$\phi^2(\widehat{\Sigma}, S) \equiv \min_{\theta\in\mathbb{R}^p} \Big\{ \frac{|S|\,\langle\theta, \widehat{\Sigma}\,\theta\rangle}{\|\theta_S\|_1^2} : \quad \theta \in \mathbb{R}^p, \ \ \|\theta_{S^c}\|_1 \le 3\|\theta_S\|_1 \Big\}. \tag{6}$$

We say that $\widehat{\Sigma} \in \mathbb{R}^{p\times p}$ satisfies the compatibility condition *for the set $S \subseteq [p]$, with constant ϕ_0 if $\phi(\widehat{\Sigma}, S) \ge \phi_0$. We say that it holds for the design matrix $\mathbf{X}$, if it holds for $\widehat{\Sigma} = \mathbf{X}^{\mathsf{T}}\mathbf{X}/n$.*

In the following, we shall drop the argument $\widehat{\Sigma}$ if clear from the context. Note that a slightly more general definition is used normally (Bühlmann and van de Geer, 2011, Section 6.13), whereby the condition $\|\theta_{S^c}\|_1 \le 3\|\theta_S\|_1$, is replaced by $\|\theta_{S^c}\|_1 \le L\|\theta_S\|_1$. The resulting constant $\phi(\widehat{\Sigma}, S, L)$ depends on L. For the sake of simplicity, we restrict ourselves to the case $L = 3$.

Definition 2 *The* sub-Gaussian norm *of a random variable X, denoted by $\|X\|_{\psi_2}$, is defined as*

$$\|X\|_{\psi_2} = \sup_{q\ge 1} q^{-1/2}(\mathbb{E}|X|^q)^{1/q}\,.$$

For a random vector $X \in \mathbb{R}^n$, its sub-Gaussian norm is defined as

$$\|X\|_{\psi_2} = \sup_{x\in S^{n-1}} \|\langle X, x\rangle\|_{\psi_2}\,,$$

where S^{n-1} denotes the unit sphere in $\mathbb{R}^n$.

Definition 3 *The* sub-exponential norm *of a random variable X, denoted by $\|X\|_{\psi_1}$, is defined as*

$$\|X\|_{\psi_1} = \sup_{q\ge 1} q^{-1}(\mathbb{E}|X|^q)^{1/q}\,.$$

For a random vector $X \in \mathbb{R}^n$, its sub-exponential norm is defined as

$$\|X\|_{\psi_1} = \sup_{x\in S^{n-1}} \|\langle X, x\rangle\|_{\psi_1}\,,$$

where S^{n-1} denotes the unit sphere in $\mathbb{R}^n$.

2. Compensating the Bias of the LASSO

In this section we present our characterization of the de-biased estimator $\widehat{\theta}^u$ (Subsection 2.1). This characterization also clarifies in what sense the LASSO estimator is biased. We discuss this point in Subsection 2.2.

2.1 A De-biased Estimator for θ_0

As emphasized above, our approach is based on a de-biased estimator defined in Equation (5), and on its distributional properties. In order to clarify the latter, it is convenient to begin with a slightly broader setting and consider a general debiasing procedure that makes use of a an arbitrary $M \in \mathbb{R}^{p\times p}$. Namely, we define

$$\widehat{\theta}^*(Y, \mathbf{X}; M, \lambda) = \widehat{\theta}^n(\lambda) + \frac{1}{n} M\mathbf{X}^{\mathsf{T}}(Y - \mathbf{X}\widehat{\theta}^n(\lambda))\,. \tag{7}$$

For notational simplicity, we shall omit the arguments $Y, \mathbf{X}, M, \lambda$ unless they are required for clarity. The quality of this debiasing procedure depends of course on the choice of M, as well as on the design $\mathbf{X}$. We characterize the pair $(\mathbf{X}, M)$ by the following figure of merit.

Definition 4 *Given the pair $\mathbf{X} \in \mathbb{R}^{n\times p}$ and $M \in \mathbb{R}^{p\times p}$, let $\widehat{\Sigma} = \mathbf{X}^{\mathsf{T}}\mathbf{X}/n$ denote the associated sample covariance. Then, the* generalized coherence parameter of $\mathbf{X}, M$*, denoted by $\mu_*(\mathbf{X}; M)$, is*

$$\mu_*(\mathbf{X}; M) \equiv \left|M\widehat{\Sigma} - \mathrm{I}\right|_\infty\,. \tag{8}$$

The minimum (generalized) coherence *of $\mathbf{X}$ is $\mu_{\min}(\mathbf{X}) = \min_{M\in\mathbb{R}^{p\times p}} \mu_*(\mathbf{X}; M)$. We denote by $M_{\min}(\mathbf{X})$ any minimizer of $\mu_*(\mathbf{X}; M)$.*

Note that the minimum coherence can be computed efficiently since $M \mapsto \mu_*(\mathbf{X}; M)$ is a convex function (even more, the optimization problem is a linear program).

The motivation for our terminology can be grasped by considering the following special case.

Remark 5 *Assume that the columns of $\mathbf{X}$ are normalized to have ℓ_2 norm equal to $\sqrt{n}$ (i.e., $\|\mathbf{X}e_i\|_2 = \sqrt{n}$ for all $i \in [p]$), and $M = \mathrm{I}$. Then $(M\widehat{\Sigma} - \mathrm{I})_{i,i} = 0$, and the maximum $|M\widehat{\Sigma} - \mathrm{I}|_\infty = \max_{i\neq j} |(\widehat{\Sigma})_{ij}|$. In other words $\mu(\mathbf{X}; \mathrm{I})$ is the maximum normalized scalar product between distinct columns of $\mathbf{X}$:*

$$\mu_*(\mathbf{X}; \mathrm{I}) = \frac{1}{n}\max_{i\neq j} \left|\langle \mathbf{X}e_i, \mathbf{X}e_j\rangle\right|\,. \tag{9}$$

The quantity (9) is known as the *coherence parameter* of the matrix $\mathbf{X}/\sqrt{n}$ and was first defined in the context of approximation theory by Mallat and Zhang (1993), and by Donoho and Huo (2001).

Assuming, for the sake of simplicity, that the columns of $\mathbf{X}$ are normalized so that $\|\mathbf{X}e_i\|_2 = \sqrt{n}$, a small value of the coherence parameter $\mu_*(\mathbf{X}; \mathrm{I})$ means that the columns of $\mathbf{X}$ are roughly orthogonal. We emphasize however that $\mu_*(\mathbf{X}; M)$ can be much smaller than its classical coherence parameter $\mu_*(\mathbf{X}; \mathrm{I})$. For instance, $\mu_*(\mathbf{X}; \mathrm{I}) = 0$ if and only if $\mathbf{X}/\sqrt{n}$ is an orthogonal matrix. On the other hand, $\mu_{\min}(\mathbf{X}) = 0$ if and only if $\mathbf{X}$ has rank p.[2]

The following theorem is a slight generalization of a result of van de Geer et al. (2014). Let us emphasize that it applies to deterministic design matrices $\mathbf{X}$.

2. Of course this example requires $n \geq p$. It is the simplest example that illustrates the difference between coherence and generalized coherence, and it is not hard to find related examples with $n < p$.

Theorem 6 *Let $\mathbf{X} \in \mathbb{R}^{n\times p}$ be any (deterministic) design matrix, and $\widehat{\theta}^* = \widehat{\theta}^*(Y, \mathbf{X}; M, \lambda)$ be a general debiased estimator as per Equation (7). Then, setting $Z = M\mathbf{X}^{\mathsf{T}}W/\sqrt{n}$, we have*

$$\sqrt{n}(\widehat{\theta}^* - \theta_0) = Z + \Delta\,, \quad Z \sim \mathsf{N}(0, \sigma^2 M\widehat{\Sigma}M^{\mathsf{T}})\,, \quad \Delta = \sqrt{n}(M\widehat{\Sigma} - \mathrm{I})(\theta_0 - \widehat{\theta}^n)\,. \tag{10}$$

Further, assume that $\mathbf{X}$ satisfies the compatibility condition for the set $S = \mathrm{supp}(\theta_0)$, $|S| \le s_0$, with constant ϕ_0, and has generalized coherence parameter $\mu_* = \mu_*(\mathbf{X}; M)$, and let $K \equiv \max_{i\in[p]} \widehat{\Sigma}_{i,i}$. Then, letting $\lambda = \sigma\sqrt{(c^2\log p)/n}$, we have

$$\mathbb{P}\Big(\|\Delta\|_\infty \ge \frac{4c\mu_*\sigma s_0}{\phi_0^2}\sqrt{\log p}\Big) \le 2p^{-c_0}\,, \quad c_0 = \frac{c^2}{32K} - 1\,. \tag{11}$$

Further, if $M = M_{\min}(\mathbf{X})$ minimizes the convex cost function $|M\widehat{\Sigma} - \mathrm{I}|_\infty$, then μ_* can be replaced by $\mu_{\min}(\mathbf{X})$ in Equation (11).

The above theorem decomposes the estimation error $(\widehat{\theta}^* - \theta_0)$ into a zero mean Gaussian term $Z/\sqrt{n}$ and a bias term $\Delta/\sqrt{n}$ whose maximum entry is bounded as per Equation (11). This estimate on $\|\Delta\|_\infty$ depends on the design matrix through two constants: the compatibility constant ϕ_0 and the generalized coherence parameter $\mu_*(\mathbf{X}; M)$. The former is a well studied property of the design matrix (Bühlmann and van de Geer, 2011; van de Geer and Bühlmann, 2009), and assuming ϕ_0 of order one is nearly necessary for the LASSO to achieve optimal estimation rate in high dimension. On the contrary, the definition of $\mu_*(\mathbf{X}; M)$ is a new contribution of the present paper.

The next theorem establishes that, for a natural probabilistic model of the design matrix $\mathbf{X}$, both ϕ_0 and $\mu_*(\mathbf{X}; M)$ can be bounded with probability converging rapidly to one as $n, p \to \infty$. Further, the bound on $\mu_*(\mathbf{X}, M)$ hold for the special choice of M that is constructed by Algorithm 1.

Theorem 7 *Let $\Sigma \in \mathbb{R}^{p\times p}$ be such that $\sigma_{\min}(\Sigma) \ge C_{\min} > 0$, and $\sigma_{\max}(\Sigma) \le C_{\max} < \infty$, and $\max_{i\in[p]} \Sigma_{ii} \le 1$. Assume $\mathbf{X}\Sigma^{-1/2}$ to have independent sub-Gaussian rows, with zero mean and sub-Gaussian norm $\|\Sigma^{-1/2}X_1\|_{\psi_2} = \kappa$, for some constant $\kappa \in (0, \infty)$.*

(a) *For $\phi_0, s_0, K \in \mathbb{R}_{>0}$, let $\mathcal{E}_n = \mathcal{E}_n(\phi_0, s_0, K)$ be the event that the compatibility condition holds for $\widehat{\Sigma} = (\mathbf{X}^{\mathsf{T}}\mathbf{X}/n)$, for all sets $S \subseteq [p]$, $|S| \le s_0$ with constant $\phi_0 > 0$, and that $\max_{i\in[p]} \widehat{\Sigma}_{i,i} \le K$. Explicitly*

$$\mathcal{E}_n(\phi_0, s_0, K) \equiv \Big\{\mathbf{X} \in \mathbb{R}^{n\times p} : \min_{S:\,|S|\le s_0} \phi(\widehat{\Sigma}, S) \ge \phi_0,\ \max_{i\in[p]} \widehat{\Sigma}_{i,i} \le K,\ \ \widehat{\Sigma} = (\mathbf{X}^{\mathsf{T}}\mathbf{X}/n)\Big\}\,. \tag{12}$$

Then there exists $c_* \le 2000$ such that the following happens. If $n \ge \nu_0\, s_0 \log(p/s_0)$, $\nu_0 \equiv 5\times 10^4 c_*(C_{\max}/C_{\min})^2\kappa^4$, $\phi_0 = \sqrt{C_{\min}}/2$, and $K \ge 1 + 20\kappa^2\sqrt{(\log p)/n}$, then

$$\mathbb{P}\big(\mathbf{X} \in \mathcal{E}_n(\phi_0, s_0, K)\big) \ge 1 - 4\,e^{-c_1 n}\,, \qquad c_1 \equiv \frac{1}{4c_*\kappa^4}\,. \tag{13}$$

(*b*) *For $a > 0$, let $\mathcal{G}_n = \mathcal{G}_n(a)$ be the event that the problem (4) is feasible for $\mu = a\sqrt{(\log p)/n}$, or equivalently*

$$\mathcal{G}_n(a) \equiv \Big\{\mathbf{X} \in \mathbb{R}^{n\times p} : \ \mu_{\min}(\mathbf{X}) < a\sqrt{\frac{\log p}{n}}\Big\}. \tag{14}$$

Then, for $n \geq a^2 C_{\min} \log p/(4e^2 C_{\max}\kappa^4)$

$$\mathbb{P}\big(\mathbf{X} \in \mathcal{G}_n(a)\big) \geq 1 - 2\,p^{-c_2}\,, \qquad c_2 \equiv \frac{a^2 C_{\min}}{24e^2\kappa^4 C_{\max}} - 2\,. \tag{15}$$

The proof of this theorem is given in Section 6.2 (for part (*a*)) and Section 6.3 (part (*b*)).

The proof that event $\mathcal{E}_n$ holds with high probability relies crucially on a theorem by Rudelson and Shuheng (2013, Theorem 6). Simplifying somewhat, the latter states that, if the restricted eigenvalue condition of Bickel et al. (2009) holds for the population covariance Σ, then it holds with high probability for the sample covariance $\widehat{\Sigma}$. (Recall that the restricted eigenvalue condition is implied by a lower bound on the minimum singular value,[3] and that it implies the compatibility condition van de Geer and Bühlmann, 2009.)

Finally, by putting together Theorem 6 and Theorem 7, we obtain the following conclusion. We refer to Section 6.4 for the proof of Theorem 8.

Theorem 8 *Consider the linear model* (1) *and let $\widehat{\theta}^u$ be defined as per Equation* (5) *in Algorithm 1, with $\mu = a\sqrt{(\log p)/n}$. Then, setting $Z = M\mathbf{X}^{\mathsf{T}}W/\sqrt{n}$, we have*

$$\sqrt{n}(\widehat{\theta}^u - \theta_0) = Z + \Delta\,, \quad Z|\mathbf{X} \sim \mathsf{N}(0, \sigma^2 M\widehat{\Sigma}M^{\mathsf{T}})\,, \quad \Delta = \sqrt{n}(M\widehat{\Sigma} - \mathrm{I})(\theta_0 - \widehat{\theta}^n)\,. \tag{16}$$

Further, under the assumptions of Theorem 7, and for $n \geq \max(\nu_0 s_0 \log(p/s_0), \nu_1 \log p)$, $\nu_1 = \max(1600\kappa^4, a^2/(4e^2\kappa^4))$, and $\lambda = \sigma\sqrt{(c^2\log p)/n}$, we have

$$\mathbb{P}\left\{\|\Delta\|_\infty \geq \Big(\frac{16ac\,\sigma}{C_{\min}}\Big)\frac{s_0\log p}{\sqrt{n}}\right\} \leq 4\,e^{-c_1 n} + 4\,p^{-\tilde{c_0}\wedge c_2}\,. \tag{17}$$

where $\tilde{c_0} = (c^2/48) - 1$ and c_1, c_2 are given by Equations (13) and (15).

Finally, the tail bound (17) holds for any choice of M that is only function of the design matrix $\mathbf{X}$, and satisfies the feasibility condition in Equation (4), i.e., $|M\widehat{\Sigma} - \mathrm{I}|_\infty \leq \mu$.

Assuming $\sigma, C_{\min}$ of order one, the last theorem establishes that, for random designs, the maximum size of the 'bias term' Δ_i over $i \in [p]$ is:

$$\|\Delta\|_\infty = O\Big(\frac{s_0\log p}{\sqrt{n}}\Big) \tag{18}$$

On the other hand, the 'noise term' Z_i is roughly of order $\sqrt{[M\widehat{\Sigma}M^{\mathsf{T}}]_{ii}}$. Bounds on the variances $[M\widehat{\Sigma}M^{\mathsf{T}}]_{ii}$ will be given in Section 3.3 (cf. Equation 82 in the proof of Theorem 16) showing that, if M is computed through Algorithm 1, $[M\widehat{\Sigma}M^{\mathsf{T}}]_{ii}$ is of order one for a broad family of random designs. As a consequence $|\Delta_i|$ is much smaller than $|Z_i|$ whenever $s_0 = o(\sqrt{n}/\log p)$. We summarize these remarks below.

3. Note, in particular, at the cost of further complicating the last statement, the condition $\sigma_{\min}(\Sigma) = \Omega(1)$ can be further weakened.

Remark 9 *Theorem 8 only requires that the support size satisfies $s_0 = O(n/\log p)$. If we further assume $s_0 = o(\sqrt{n}/\log p)$, then we have $\|\Delta\|_\infty = o(1)$ with high probability. Hence, $\widehat{\theta}^u$ is an asymptotically unbiased estimator for θ_0.*

A more formal comparison of the bias of $\widehat{\theta}^u$, and of the one of the LASSO estimator $\widehat{\theta}^n$ can be found in Section 2.2 below. Section 2.3 compares our approach with the related one in van de Geer et al. (2014).

As it can be seen from the statement of Theorem 6 and Theorem 7, the claim of Theorem 8 does not rely on the specific choice of the objective function in optimization problem (4) and only uses the constraint on $\|\widehat{\Sigma}m - e_i\|_\infty$. In particular it holds for any matrix M that is feasible. On the other hand, the specific objective function problem (4) minimizes the variance of the noise term $\mathrm{Var}(Z_i)$.

2.2 Discussion: Bias of the LASSO

Theorems 6 and 7 provide a quantitative framework to discuss in what sense the LASSO estimator $\widehat{\theta}^n$ is asymptotically biased, while the de-biased estimator $\widehat{\theta}^u$ is asymptotically unbiased.

Given an estimator $\widehat{\theta}^n$ of the parameter vector θ_0, we define its *bias* to be the vector

$$\mathsf{Bias}(\widehat{\theta}^n) \equiv \mathbb{E}\{\widehat{\theta}^n - \theta_0 | \mathbf{X}\}\,. \tag{19}$$

Note that, if the design is random, $\mathsf{Bias}(\widehat{\theta}^n)$ is a measurable function of $\mathbf{X}$. If the design is deterministic, $\mathsf{Bias}(\widehat{\theta}^n)$ is a deterministic quantity as well, and the conditioning is redundant.

It follows from Equation (10) that

$$\mathsf{Bias}(\widehat{\theta}^u) = \frac{1}{\sqrt{n}}\mathbb{E}\{\Delta|\mathbf{X}\}\,. \tag{20}$$

Applying Theorem 8 with high probability, $\|\Delta\|_\infty = O(s_0 \log p/\sqrt{n})$. The next corollary establishes that this translates into a bound on $\mathsf{Bias}(\widehat{\theta}^u)$ for all $\mathbf{X}$ in a set that has probability rapidly converging to one as n, p get large.

Corollary 10 *Under the assumptions of Theorem 8, let c_1, c_2 be defined as per Equations (13), (15). Then we have*

$$\mathbf{X} \in \mathcal{E}_n(\sqrt{C_{\min}}/2, s_0, 3/2) \cap \mathcal{G}_n(a) \;\Rightarrow\; \|\mathsf{Bias}(\widehat{\theta}^u)\|_\infty \le \frac{160a}{C_{\min}} \cdot \frac{\sigma s_0 \log p}{n}\,, \tag{21}$$

$$\mathbb{P}\Big(\mathbf{X} \in \mathcal{E}_n(\sqrt{C_{\min}}/2, s_0, 3/2) \cap \mathcal{G}_n(a)\Big) \ge 1 - 4e^{-c_1 n} - 2\,p^{-c_2}\,. \tag{22}$$

The proof of this corollary can be found in Appendix B.1.

This result can be contrasted with a converse result for the LASSO estimator. Namely, as stated below, there are choices of the vector θ_0, and of the design covariance Σ, such that $\mathsf{Bias}(\widehat{\theta}^n)$ is the sum of two terms. One is of order order $\lambda = c\sigma\sqrt{(\log p)/n}$ and the second is of order $\|\mathsf{Bias}(\widehat{\theta}^u)\|_\infty$. If s_0 is significantly smaller than $\sqrt{n/\log p}$ (which is the main regime studied in the rest of the paper), the first term dominates and $\|\mathsf{Bias}(\widehat{\theta}^n)\|_\infty$ is much larger than $\|\mathsf{Bias}(\widehat{\theta}^u)\|_\infty$. On the other hand, if s_0 is significantly larger than $\sqrt{n/\log p}$

then $\|\mathsf{Bias}(\widehat{\theta}^n)\|_\infty$ is of the same order as $\|\mathsf{Bias}(\widehat{\theta}^u)\|_\infty$. This justifies referring to $\widehat{\theta}^u$ as an *unbiased estimator*.

Notice that, since we want to establish a negative result about the LASSO, it is sufficient to exhibit a specific covariance structure Σ satisfying the assumptions of the previous corollary. Remarkably it is sufficient to consider standard designs, i.e., $\Sigma = \mathrm{I}_{p\times p}$.

Corollary 11 *Under the assumptions of Theorem 8, further consider the case $\Sigma = \mathrm{I}$. Then, there exist a set of design matrices $\mathcal{B}_n \subseteq \mathbb{R}^{n\times p}$, and coefficient vectors $\theta_0 \in \mathbb{R}^p$, $\|\theta_0\|_0 \le s_0$, such that*

$$\mathbf{X} \in \mathcal{B}_n \;\Rightarrow\; \|\mathsf{Bias}(\widehat{\theta}^n)\|_\infty \ge \left|\frac{2}{3}\lambda - \|\mathsf{Bias}(\widehat{\theta}^*)\|_\infty\right|, \tag{23}$$

$$\mathbb{P}(\mathcal{B}_n) \ge 1 - 4\,e^{-c_1 n} - 2\,p^{-3}, \tag{24}$$

where $\widehat{\theta}^* = \widehat{\theta}^*(Y, \mathbf{X}; \mathrm{I}, \lambda)$, with $\lambda = c\sigma\sqrt{(\log p)/n}$. In particular, there exists $c_{**} \le 4800$ such that if $n \ge (3c_{**}s_0/c)^2 \log p$ and $p \ge 13^{48/(c^2-48)}$, then the following hold true:

$$\|\mathsf{Bias}(\widehat{\theta}^*)\|_\infty \le \lambda/3\,, \tag{25}$$

$$\|\mathsf{Bias}(\widehat{\theta}^n)\|_\infty \ge \frac{c\sigma}{3}\sqrt{\frac{\log p}{n}} \gg \|\mathsf{Bias}(\widehat{\theta}^u)\|_\infty\,, \tag{26}$$

where $\widehat{\theta}^u$ is given by Equation (5) in Algorithm 1, with $\mu = 30\sqrt{(\log p)/n}$.

A formal proof of this statement is deferred to Appendix B.2, but the underlying mathematical mechanism is quite simple. Recall that the KKT condition for the LASSO estimator (3) reads

$$\frac{1}{n}\mathbf{X}^{\mathsf{T}}(Y - \mathbf{X}\widehat{\theta}^n) = \lambda\, v(\widehat{\theta}^n)\,, \tag{27}$$

with $v(\widehat{\theta}^n) \in \mathbb{R}^p$ being a vector in the subgradient of the ℓ_1 norm at $\widehat{\theta}^n$. Adding $\widehat{\theta}^n - \theta_0$ to both sides, and taking expectation over the noise, we get

$$\mathsf{Bias}(\widehat{\theta}^*) = \mathsf{Bias}(\widehat{\theta}^n) + \lambda\mathbb{E}\{v(\widehat{\theta}^n)|\mathbf{X}\}\,, \tag{28}$$

where $\widehat{\theta}^*$ is a debiased estimator of the general form (7), for $M = \mathrm{I}$. As shown formally in Appendix B.2, $\|\mathbb{E}\{v(\widehat{\theta}^n)|\mathbf{X}\}\|_\infty \ge 2/3$, which directly implies Equation (23) using triangle inequality.

2.3 Comparison with Earlier Results

In this Section we briefly compare the above debiasing procedure and in particular Theorems 6, 7 and 8 to the results of van de Geer et al. (2014). In the case of linear statistical models considered here, the authors of van de Geer et al. (2014) construct a debiased estimator of the form (7). However, instead of solving the optimization problem (4), they follow Zhang and Zhang (2014) and use the regression coefficients of the i-th column of $\mathbf{X}$ on the other columns to construct the i-th row of M. These regression coefficients are computed, once again, using the LASSO (node-wise LASSO).

It useful to spell out the most important differences between our contribution and the ones of van de Geer et al. (2014):

1. The case of fixed non-random designs is covered by van de Geer et al. (2014, Theorem 2.1), which should be compared to our Theorem 6. While in our case the bias is controlled by the generalized coherence parameter, a similar role is played in van de Geer et al. (2014) by the regularization parameters of the nodewise LASSO.

2. The case of random designs is covered by van de Geer et al. (2014, Theorem 2.2, Theorem 2.4), which should be compared with our Theorem 8. In this case, the assumptions underlying our result are less restrictive. More precisely:

   (a) van de Geer et al. (2014, Theorem 2.2, Theorem 2.4) assume $\mathbf{X}$ has i.i.d. rows, while we only assume the rows are independent.

   (b) van de Geer et al. (2014, Theorem 2.2, Theorem 2.4) assumes the rows of the inverse covariance matrix Σ^{-1} are sparse. More precisely, letting s_j be the number of non-zero entries of the j-th row of Σ^{-1}, van de Geer et al. (2014) assumes $\max_{j\in[p]} s_j = o(n/\log p)$, that is much smaller than p. We do not make any sparsity assumption on Σ^{-1}, and s_j can be as large as p.
   van de Geer et al. (2014, Theorem 2.4) also considers a slightly different setting, where $\mathbf{X}$ has bounded entries, under analogous sparsity assumptions.

   It is currently unknown whether the sparsity assumption in van de Geer et al. (2014) is required for that approach to work, or it is rather an artifact of the specific analysis. Indeed van de Geer et al. (2014, Theorem 2.1) can in principle be used to weaken this condition.

In addition our Theorem 8 provides the specific dependence on the maximum and minimum singular value of $\widehat{\Sigma}$.

Note that solving the convex problem (4) is not more burdensome than solving the nodewise LASSO as in Zhang and Zhang (2014); van de Geer et al. (2014), This can be confirmed by checking that the dual of problem (4) is an ℓ_1-regularized quadratic optimization problem. It has therefore the same complexity as the nodewise LASSO (but it is different from the nodewise LASSO).

3. Statistical Inference

A direct application of Theorem 8 is to derive confidence intervals and statistical hypothesis tests for high-dimensional models. Throughout, we make the sparsity assumption $s_0 = o(\sqrt{n}/\log p)$ and omit explicit constants that can be readily derived from Theorem 8.

3.1 Preliminary Lemmas

As discussed above, the bias term Δ is negligible with respect to the random term Z in the decomposition (16), provided the latter has variance of order one. Our first lemma establishes that this is indeed the case.

Lemma 12 *Let $M = (m_1, \dots, m_p)^{\mathsf{T}}$ be the matrix with rows m_i^{T} obtained by solving convex program (4) in Algorithm 1. Then for all $i \in [p]$,*

$$[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i} \geq \frac{(1-\mu)^2}{\widehat{\Sigma}_{i,i}}\,.$$

Lemma 12 is proved in Appendix A.1.

Using this fact, we can then characterize the asymptotic distribution of the residuals $(\widehat{\theta}^u - \theta_{0,i})$. Theorem 8 naturally suggests to consider the scaled residual $\sqrt{n}(\widehat{\theta}^u_i - \theta_{0,i})/(\sigma[M\widehat{\Sigma}M^{\mathsf{T}}]^{1/2}_{i,i})$. In the next lemma we consider a slightly more general scaling, replacing σ by a consistent estimator $\widehat{\sigma}$.

Lemma 13 *Consider a sequence of design matrices $\mathbf{X} \in \mathbb{R}^{n\times p}$, with dimensions $n \to \infty$, $p = p(n) \to \infty$ satisfying the following assumptions, for constants $C_{\min}, C_{\max}, \kappa \in (0,\infty)$ independent of n. For each n, $\Sigma \in \mathbb{R}^{p\times p}$ is such that $\sigma_{\min}(\Sigma) \geq C_{\min} > 0$, and $\sigma_{\max}(\Sigma) \leq C_{\max} < \infty$, and $\max_{i\in[p]} \Sigma_{ii} \leq 1$. Assume $\mathbf{X}\Sigma^{-1/2}$ to have independent sub-Gaussian rows, with zero mean and sub-Gaussian norm $\|\Sigma^{-1/2}X_1\|_{\psi_2} \leq \kappa$,*

Consider the linear model (1) and let $\widehat{\theta}^u$ be defined as per Equation (5) in Algorithm 1, with $\mu = a\sqrt{(\log p)/n}$ and $\lambda = \sigma\sqrt{(c^2\log p)/n}$, with a, c large enough constants. Finally, let $\widehat{\sigma} = \widehat{\sigma}(y, \mathbf{X})$ be an estimator of the noise level satisfying, for any $\varepsilon > 0$,

$$\lim_{n\to\infty} \sup_{\theta_0\in\mathbb{R}^p;\, \|\theta_0\|_0 \leq s_0} \mathbb{P}\Big(\Big|\frac{\widehat{\sigma}}{\sigma} - 1\Big| \geq \varepsilon\Big) = 0\,. \tag{29}$$

If $s_0 = o(\sqrt{n}/\log p)$ ($s_0 \geq 1$), then, for all $x \in \mathbb{R}$, we have

$$\lim_{n\to\infty} \sup_{\theta_0\in\mathbb{R}^p;\, \|\theta_0\|_0 \leq s_0} \left|\mathbb{P}\left\{\frac{\sqrt{n}(\widehat{\theta}^u_i - \theta_{0,i})}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]^{1/2}_{i,i}} \leq x\right\} - \Phi(x)\right| = 0\,. \tag{30}$$

The proof of this lemma can be found in Section 6.5. We also note that the dependence of a, c on $C_{\min}, C_{\max}, \kappa$ can be easily reconstructed from Theorem 7.

The last lemma requires a consistent estimator of σ, in the sense of Equation (29). Several proposals have been made to estimate the noise level in high-dimensional linear regression. A short list of references includes Fan and Li (2001); Fan and Lv (2008); Städler et al. (2010); Zhang (2010); Sun and Zhang (2012); Belloni and Chernozhukov (2013); Fan et al. (2012); Reid et al. (2013); Dicker (2012); Fan et al. (2009); Bayati et al. (2013). Consistency results have been proved or can be proved for several of these estimators.

In order to demonstrate that the consistency criterion (29) can be achieved, we use the scaled LASSO (Sun and Zhang, 2012) given by

$$\{\widehat{\theta}^n(\widetilde{\lambda}), \widehat{\sigma}(\widetilde{\lambda})\} \equiv \operatorname*{arg\,min}_{\theta\in\mathbb{R}^p, \sigma>0} \Big\{\frac{1}{2\sigma n}\|Y - \mathbf{X}\theta\|_2^2 + \frac{\sigma}{2} + \widetilde{\lambda}\|\theta\|_1\Big\}\,. \tag{31}$$

This is a joint convex optimization problem which provides an estimate of the noise level in addition to an estimate of θ_0.

The following lemma uses the analysis of Sun and Zhang (2012) to show that $\widehat{\sigma}$ satisfies the consistency criterion (29).

Lemma 14 *Under the assumptions of Lemma 13, let $\widehat{\sigma} = \widehat{\sigma}(\widetilde{\lambda})$ be the scaled LASSO estimator of the noise level, see Equation (31), with $\widetilde{\lambda} = 10\sqrt{(2\log p)/n}$. Then $\widehat{\sigma}$ satisfies Equation (29).*

The proof of this lemma is fairly straightforward and can be found in Appendix C.

3.2 Confidence Intervals

In view of Lemma 13, it is quite straightforward to construct asymptotically valid confidence intervals. Namely, for $i \in [p]$ and significance level $\alpha \in (0,1)$, we let

$$\begin{aligned} J_i(\alpha) &\equiv [\widehat{\theta}_i^u - \delta(\alpha, n), \widehat{\theta}_i^u + \delta(\alpha, n)]\,, \\ \delta(\alpha, n) &\equiv \Phi^{-1}(1-\alpha/2)\,\frac{\widehat{\sigma}}{\sqrt{n}}[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}\,. \end{aligned} \tag{32}$$

Theorem 15 *Consider a sequence of design matrices $\mathbf{X} \in \mathbb{R}^{n\times p}$, with dimensions $n \to \infty$, $p = p(n) \to \infty$ satisfying the assumptions of Lemma 13.*

Consider the linear model (1) and let $\widehat{\theta}^u$ be defined as per Equation (5) in Algorithm 1, with $\mu = a\sqrt{(\log p)/n}$ and $\lambda = \sigma\sqrt{(c^2 \log p)/n}$, with a, c large enough constants. Finally, let $\widehat{\sigma} = \widehat{\sigma}(y, \mathbf{X})$ a consistent estimator of the noise level in the sense of Equation (29). Then the confidence interval $J_i(\alpha)$ is asymptotically valid, namely

$$\lim_{n\to\infty} \mathbb{P}\Big(\theta_{0,i} \in J_i(\alpha)\Big) = 1 - \alpha\,. \tag{33}$$

Proof The proof is an immediate consequence of Lemma 13 since

$$\begin{aligned} \lim_{n\to\infty} \mathbb{P}\Big(\theta_{0,i} \in J_i(\alpha)\Big) =& \lim_{n\to\infty} \mathbb{P}\left\{\frac{\sqrt{n}(\widehat{\theta}_i^u - \theta_{0,i})}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}} \le \Phi^{-1}(1-\alpha/2)\right\} \\ &- \lim_{n\to\infty} \mathbb{P}\left\{\frac{\sqrt{n}(\widehat{\theta}_i^u - \theta_{0,i})}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}} \le -\Phi^{-1}(1-\alpha/2)\right\} \\ =& 1 - \alpha\,. \end{aligned} \tag{34}$$

■

3.3 Hypothesis Testing

An important advantage of sparse linear regression models is that they provide parsimonious explanations of the data in terms of a small number of covariates. The easiest way to select the 'active' covariates is to choose the indexes i for which $\widehat{\theta}_i^n \neq 0$. This approach however does not provide a measure of statistical significance for the finding that the coefficient is non-zero.

More precisely, we are interested in testing an individual null hypothesis $H_{0,i} : \theta_{0,i} = 0$ versus the alternative $H_{A,i} : \theta_{0,i} \neq 0$, and assigning p-values for these tests. We construct a p-value P_i for the test $H_{0,i}$ as follows:

$$P_i = 2\left(1 - \Phi\left(\frac{\sqrt{n}\,|\widehat{\theta}_i^u|}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}}\right)\right). \tag{35}$$

The decision rule is then based on the p-value P_i:

$$\widehat{T}_{i,\mathbf{X}}(y) = \begin{cases} 1 & \text{if } P_i \le \alpha \qquad (\text{reject } H_{0,i})\,, \\ 0 & \text{otherwise} \qquad (\text{accept } H_{0,i})\,, \end{cases} \tag{36}$$

where α is the fixed target Type I error probability. We measure the quality of the test $\widehat{T}_{i,\mathbf{X}}(y)$ in terms of its significance level α_i and statistical power $1-\beta_i$. Here α_i is the probability of type I error (i.e., of a false positive at i) and β_i is the probability of type II error (i.e., of a false negative at i).

Note that it is important to consider the tradeoff between statistical significance and power. Indeed any significance level α can be achieved by randomly rejecting $H_{0,i}$ with probability α. This test achieves power $1-\beta=\alpha$. Further note that, without further assumption, no nontrivial power can be achieved. In fact, choosing $\theta_{0,i}\neq 0$ arbitrarily close to zero, $H_{0,i}$ becomes indistinguishable from its alternative. We will therefore assume that, whenever $\theta_{0,i}\neq 0$, we have $|\theta_{0,i}|>\gamma$ as well. We take a minimax perspective and require the test to behave uniformly well over s_0-sparse vectors. Formally, given a family of tests $T_{i,\mathbf{X}}:\mathbb{R}^n\to\{0,1\}$, indexed by $i\in[p]$, $\mathbf{X}\in\mathbb{R}^{n\times p}$, we define, for $\gamma>0$ a lower bound on the non-zero entries:

$$\alpha_{i,n}(T)\equiv \sup\Big\{\mathbb{P}_{\theta_0}(T_{i,\mathbf{X}}(y)=1):\ \theta_0\in\mathbb{R}^p,\ \|\theta_0\|_0\le s_0(n),\ \theta_{0,i}=0\Big\}. \tag{37}$$

$$\beta_{i,n}(T;\gamma)\equiv \sup\Big\{\mathbb{P}_{\theta_0}(T_{i,\mathbf{X}}(y)=0):\ \theta_0\in\mathbb{R}^p,\ \|\theta_0\|_0\le s_0(n),\ |\theta_{0,i}|\ge\gamma\Big\}. \tag{38}$$

Here, we made dependence on n explicit. Also, $\mathbb{P}_\theta(\,\cdot\,)$ denotes the induced probability for random design $\mathbf{X}$ and noise realization w, given the fixed parameter vector θ. Our next theorem establishes bounds on $\alpha_{i,n}(\widehat{T})$ and $\beta_{i,n}(\widehat{T};\gamma)$ for our decision rule (36).

Theorem 16 *Consider a sequence of design matrices $\mathbf{X}\in\mathbb{R}^{n\times p}$, with dimensions $n\to\infty$, $p=p(n)\to\infty$ satisfying the assumptions of Lemma 13.*

Consider the linear model (1) *and let $\widehat{\theta}^u$ be defined as per Equation* (5) *in Algorithm 1, with $\mu=a\sqrt{(\log p)/n}$ and $\lambda=\sigma\sqrt{(c^2\log p)/n}$, with a,c large enough constants. Finally, let $\widehat{\sigma}=\widehat{\sigma}(y,\mathbf{X})$ a consistent estimator of the noise level in the sense of Equation (29), and $\widehat{T}$ be the test defined in Equation (36).*

Then the following holds true for any fixed sequence of integers $i=i(n)$:

$$\lim_{n\to\infty}\alpha_{i,n}(\widehat{T})\le\alpha\,. \tag{39}$$

$$\liminf_{n\to\infty}\frac{1-\beta_{i,n}(\widehat{T};\gamma)}{1-\beta^*_{i,n}(\gamma)}\ge 1\,,\qquad 1-\beta^*_{i,n}(\gamma)\equiv G\Big(\alpha,\frac{\sqrt{n}\,\gamma}{\sigma[\Sigma^{-1}_{i,i}]^{1/2}}\Big)\,, \tag{40}$$

where, for $\alpha\in[0,1]$ and $u\in\mathbb{R}_+$, the function $G(\alpha,u)$ is defined as follows:

$$G(\alpha,u)=2-\Phi(\Phi^{-1}(1-\frac{\alpha}{2})+u)-\Phi(\Phi^{-1}(1-\frac{\alpha}{2})-u)\,.$$

Theorem 16 is proved in Section 6.6. It is easy to see that, for any $\alpha>0$, $u\mapsto G(\alpha,u)$ is continuous and monotone increasing. Moreover, $G(\alpha,0)=\alpha$ which is the trivial power obtained by randomly rejecting $H_{0,i}$ with probability α. As γ deviates from zero, we obtain nontrivial power. Notice that in order to achieve a specific power $\beta>\alpha$, our scheme requires $\gamma\ge c_\beta(\sigma/\sqrt{n})$, for some constant c_β that depends on β. This is because $\Sigma^{-1}_{i,i}\le\sigma_{\max}(\Sigma^{-1})\le(\sigma_{\min}(\Sigma))^{-1}=O(1)$.

3.3.1 Near optimality of the hypothesis testing procedure

The authors of Javanmard and Montanari (2013b) prove an upper bound for the minimax power of tests with a given significance level α, under random designs. For the reader's convenience, we recall this result here. (The following is a restatement of Javanmard and Montanari (2013b, Theorem 2.3), together with a standard estimate on the tail of chi-squared random variables.)

Theorem 17 ((Javanmard and Montanari, 2013b)) *Assume $\mathbf{X} \in \mathbb{R}^{n\times p}$ to be a random design matrix with i.i.d. Gaussian rows with zero mean and covariance Σ. For $i \in [p]$, let $T_{i,\mathbf{X}} : \mathbb{R}^n \to \mathbb{R}^n$ be a hypothesis testing procedure for testing $H_{0,i} : \theta_{0,i} = 0$, and denote by $\alpha_i(T)$ and $\beta_{i,n}(T;\gamma)$ its fraction of type I and type II errors, cf. Equations (37) and (38). Finally, for $S \subseteq [p] \setminus \{i\}$, define $\Sigma_{i|S} \equiv \Sigma_{ii} - \Sigma_{i,S}\Sigma_{S,S}^{-1}\Sigma_{S,i} \in \mathbb{R}$.*

For any $\ell \in \mathbb{R}$ and $|S| < s_0 < n$, if $\alpha_{i,n}(T) \le \alpha$, then

$$1 - \beta_{i,n}(T;\gamma) \le G\Big(\alpha, \frac{\gamma}{\sigma_{\text{eff}}(\xi)}\Big) + e^{-\xi^2/8}\,, \tag{41}$$

$$\sigma_{\text{eff}}(\xi) \equiv \frac{\sigma}{\Sigma_{i|S}^{1/2}(\sqrt{n - s_0 + 1} + \xi)}\,, \tag{42}$$

for any $\xi \in [0, (3/2)\sqrt{n - s_0 + 1}]$.

The intuition behind this bound is straightforward: the power of any test for $H_{0,i} : \theta_{0,i} = 0$ is upper bounded by the power of an oracle test that is given access to $\text{supp}(\theta_0) \setminus \{i\}$ and outputs a test for $H_{0,i}$. Computing the minimax power of such oracle reduces to a classical hypothesis testing problem.

Let us emphasize that the last theorem applies to *Gaussian* random designs. Since this theorem establishes a negative result (an upper bound on power), it makes sense to consider this somewhat more specialized setting.

Using this upper bound, we can restate Theorem 16 as follows.

Corollary 18 *Consider a Gaussian random design model that satisfies the conditions of Theorem 16, and let $\widehat{T}$ be the testing procedure defined in Equation (36), with $\widehat{\theta}^u$ as in Algorithm 1. Further, let*

$$\eta_{\Sigma,s_0} \equiv \min_{i\in[p];S} \Big\{\Sigma_{i|S}\,\Sigma_{ii}^{-1} :\ S \subseteq [p]\backslash\{i\},\, |S| < s_0\Big\}\,. \tag{43}$$

Under the sparsity assumption $s_0 = o(\sqrt{n}/\log p)$, the following holds true. If $\{T_{i,\mathbf{X}}\}$ is any sequence of tests with $\limsup_{n\to\infty} \alpha_{i,n}(T) \le \alpha$, then

$$\liminf_{n\to\infty} \frac{1 - \beta_{i,n}(\widehat{T};\gamma)}{1 - \beta_{i,n/\eta_{\Sigma,s_0}}(T;\gamma)} \ge 1\,. \tag{44}$$

In other words, the asymptotic efficiency of the test $\widehat{T}$ is at least $1/\eta_{\Sigma,s_0}$.

Hence, our test $\widehat{T}$ has nearly optimal power in the following sense. It has power at least as large as the power of any other test T, provided the latter is applied to a sample size decreased by a factor η_{Σ,s_0}.

Further, under the assumptions of Theorem 8, the factor η_{Σ,s_0} is a bounded constant. Indeed

$$\eta_{\Sigma,s_0} \le \Sigma^{-1}_{i,i}\Sigma_{i,i} \le \frac{\sigma_{\max}(\Sigma)}{\sigma_{\min}(\Sigma)} \le \frac{C_{\max}}{C_{\min}}\,, \tag{45}$$

since $\Sigma^{-1}_{ii} \le (\sigma_{\min}(\Sigma))^{-1}$, and $\Sigma_{i|S} \le \Sigma_{i,i} \le \sigma_{\max}(\Sigma)$ due to $\Sigma_{S,S} \succ 0$.

Note that n, γ and σ appears in our upper bound (41) in the combination $\gamma\sqrt{n}/\sigma$, which is the natural measure of the signal-to-noise ratio (where, for simplicity, we neglected $s_0 = o(\sqrt{n}/\log p)$ with respect to n). Hence, the above result can be restated as follows. The test $\widehat{T}$ has power at least as large as the power of any other test T, provided the latter is applied at a noise level augmented by a factor $\sqrt{\eta_{\Sigma,s_0}}$.

3.4 Generalization to Simultaneous Confidence Intervals

In many situations, it is necessary to perform statistical inference on more than one of the parameters simultaneously. For instance, we might be interested in performing inference about $\theta_{0,R} \equiv (\theta_{0,i})_{i\in R}$ for some set $R \subseteq [p]$.

The simplest generalization of our method is to the case in which $|R|$ stays finite as $n, p \to \infty$. In this case we have the following generalization of Lemma 13. (The proof is the same as for Lemma 13, and hence we omit it.)

Lemma 19 *Under the assumptions of Lemma 13, define*

$$Q^{(n)} \equiv \frac{\widehat{\sigma}^2}{n}\,[M\widehat{\Sigma}M^{\mathsf{T}}]\,. \tag{46}$$

Let $R = R(n)$ be a sequence of sets $R(n) \subseteq [p]$, with $|R(n)| = k$ fixed as $n, p \to \infty$, and further assume $s_0 = o(\sqrt{n}/\log p)$, with $s_0 \ge 1$. Then, for all $x = (x_1, \dots, x_k) \in \mathbb{R}^k$, we have

$$\lim_{n\to\infty} \sup_{\theta_0\in\mathbb{R}^p;\, \|\theta_0\|_0\le s_0} \left|\mathbb{P}\left\{(Q^{(n)}_{R,R})^{-1/2}(\widehat{\theta}^u_R - \theta_{0,R}) \le x\right\} - \Phi_k(x)\right| = 0\,, \tag{47}$$

where $(a_1, \dots, a_k) \le (b_1, \dots, b_k)$ indicates that $a_1 \le b_1, \dots a_k \le b_k$, and $\Phi_k(x) = \Phi(x_1) \times \cdots \times \Phi(x_k)$.

This lemma allows to construct confidence regions for low-dimensional projections of θ_0, much in the same way as we used Lemma 13 to compute confidence intervals for one-dimensional projections in Section 3.2.

Explicitly, let $\mathcal{C}_{k,\alpha} \subseteq \mathbb{R}^k$ be any Borel set such that $\int_{\mathcal{C}_{k,\alpha}} \phi_k(x)\,\mathrm{d}x \ge 1-\alpha$, where

$$\phi_k(x) = \frac{1}{(2\pi)^{k/2}}\,\exp\Big(-\frac{\|x\|^2}{2}\Big)\,,$$

is the k-dimensional Gaussian density. Then, for $R \subseteq [p]$, we define $J_R(\alpha) \subseteq \mathbb{R}^k$ as follows

$$J_R(\alpha) \equiv \widehat{\theta}^u_R + (Q^{(n)}_{R,R})^{1/2}\mathcal{C}_{k,\alpha}\,. \tag{48}$$

Then Lemma 19 implies (under the assumptions stated there) that $J_R(\alpha)$ is a valid confidence region

$$\lim_{n\to\infty} \mathbb{P}\big(\theta_{0,R} \in J_R(\alpha)\big) = 1-\alpha\,. \tag{49}$$

A more challenging regime is the one of large-scale inference, that corresponds to $|R(n)| \to \infty$ with n. Even in the seemingly simple case in which a correct p-value is given for each individual coordinate, the problem of aggregating them has attracted considerable amount of work, see e.g., Efron (2010) for an overview.

Here we limit ourselves to designing a testing procedure for the family of hypotheses $\{H_{0,i} : \theta_{0,i} = 0\}_{i\in[p]}$ that controls the familywise error rate (FWER). Namely we want to define $T_{i,\mathbf{X}} : \mathbb{R}^n \to \{0,1\}$, for each $i \in [p]$, $\mathbf{X} \in \mathbb{R}^{n\times p}$ such that

$$\mathrm{FWER}(T,n) \equiv \sup_{\theta_0\in\mathbb{R}^p, \|\theta_0\|_0\le s_0} \mathbb{P}\Big\{\exists i\in[p] :\ \ \theta_{0,i}=0,\, T_{i,\mathbf{X}}(y)=1\Big\}\,, \tag{50}$$

Here $T = \{T_{i,\mathbf{X}}\}_{i\in[p]}$ represents the family of tests.

In order to achieve familywise error control, we adopt a standard trick based on Bonferroni inequality. Given p-values defined as per Equation (35), we let

$$\widehat{T}^{\mathrm{F}}_{i,\mathbf{X}}(y) = \begin{cases} 1 & \text{if } P_i \le \alpha/p \qquad (\text{reject } H_{0,i})\,,\\ 0 & \text{otherwise} \qquad\quad (\text{accept } H_{0,i})\,. \end{cases} \tag{51}$$

Then we have the following error control guarantee.

Theorem 20 *Consider a sequence of design matrices $\mathbf{X} \in \mathbb{R}^{n\times p}$, with dimensions $n \to \infty$, $p = p(n) \to \infty$ satisfying the assumptions of Lemma 13.*

Consider the linear model (1) *and let $\widehat{\theta}^u$ be defined as per Equation* (5) *in Algorithm 1, with $\mu = a\sqrt{(\log p)/n}$ and $\lambda = \sigma\sqrt{(c^2\log p)/n}$, with a, c large enough constants. Finally, let $\widehat{\sigma} = \widehat{\sigma}(y,\mathbf{X})$ be a consistent estimator of the noise level in the sense of Equation (29), and $\widehat{T}$ be the test defined in Equation (51). Then:*

$$\limsup_{n\to\infty} \mathrm{FWER}(\widehat{T}^{\mathrm{F}}, n) \le \alpha\,. \tag{52}$$

The proof of this theorem is similar to the one of Lemma 13 and Theorem 16, and is deferred to Appendix D.

4. Non-Gaussian Noise

As can be seen from the proof of Theorem 8, $Z = M\mathbf{X}^{\mathsf{T}}W/\sqrt{n}$, and since the noise is Gaussian, i.e., $W \sim \mathsf{N}(0,\sigma^2\mathrm{I})$, we have $Z|\mathbf{X} \sim \mathsf{N}(0,\sigma^2 M\widehat{\Sigma}M^{\mathsf{T}})$. We claim that the distribution of the coordinates of Z is asymptotically Gaussian, even if W is non-Gaussian, provided the definition of M is modified slightly. As a consequence, the definition of confidence intervals and p-values in Corollary 15 and (35) remain valid in this broader setting.

In case of non-Gaussian noise, we write

$$\frac{\sqrt{n}(\widehat{\theta}_i^u - \theta_{0,i})}{\sigma[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}} = \frac{1}{\sqrt{n}} \frac{m_i^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}W}{\sigma[m_i^{\mathsf{T}}\widehat{\Sigma}m_i]^{1/2}} + o(1)$$
$$= \frac{1}{\sqrt{n}} \sum_{j=1}^{n} \frac{m_i^{\mathsf{T}}X_jW_j}{\sigma[m_i^{\mathsf{T}}\widehat{\Sigma}m_i]^{1/2}} + o(1)\,.$$

Conditional on $\mathbf{X}$, the summands $\xi_j = m_i^{\mathsf{T}}X_jW_j/(\sigma[m_i^{\mathsf{T}}\widehat{\Sigma}m_i]^{1/2})$ are independent and zero mean. Further, $\sum_{j=1}^n \mathbb{E}(\xi_j^2|\mathbf{X}) = 1$. Therefore, if Lindeberg condition holds, namely for every $\varepsilon > 0$, almost surely

$$\lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} \mathbb{E}(\xi_j^2 \mathbb{I}_{\{|\xi_j|>\varepsilon\sqrt{n}\}}|\mathbf{X}) = 0\,, \tag{53}$$

then $\sum_{j=1}^n \xi_j/\sqrt{n}|\mathbf{X} \xrightarrow{\mathrm{d}} \mathsf{N}(0,1)$, from which we can build the valid p-values as in (35).

In order to ensure that the Lindeberg condition holds, we modify the optimization problem (54) as follows:

$$\begin{aligned} &\text{minimize} && m^{\mathsf{T}}\widehat{\Sigma}m \\ &\text{subject to} && \|\widehat{\Sigma}m - e_i\|_\infty \le \mu \\ &&& \|\mathbf{X}m\|_\infty \le n^\beta \quad \text{for arbitrary fixed } 1/4 < \beta < 1/2 \end{aligned} \tag{54}$$

Next theorem shows the validity of the proposed p-values in the non-Gaussian noise setting.

Theorem 21 *Suppose that the noise variables W_i are independent with $\mathbb{E}(W_i) = 0$, $\mathbb{E}(W_i^2) = \sigma^2$, and $\mathbb{E}(|W_i|^{2+a}) \le C\,\sigma^{2+a}$ for some $a > 0$.*

Let $M = (m_1, \dots, m_p)^{\mathsf{T}}$ be the matrix with rows m_i^{T} obtained by solving optimization problem (54). Then under the assumptions of Theorem 8, and for sparsity level $s_0 = o(\sqrt{n}/\log p)$, an asymptotic two-sided confidence interval for $\theta_{0,i}$ with significance α is given by $I_i = [\widehat{\theta}_i^u - \delta(\alpha,n), \widehat{\theta}_i^u + \delta(\alpha,n)]$ where

$$\delta(\alpha,n) = \Phi^{-1}(1-\alpha/2)\widehat{\sigma}\,n^{-1/2}\sqrt{[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}}\,. \tag{55}$$

Further, an asymptotically valid p-value P_i for testing null hypothesis $H_{0,i}$ is constructed as:

$$P_i = 2\bigg(1 - \Phi\bigg(\frac{\sqrt{n}|\widehat{\theta}_i^u|}{[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}}\bigg)\bigg)\,.$$

Theorem 21 is proved in Section 6.7.

5. Numerical Experiments

We corroborate our theoretical results with numerical experiments on both synthetic and real data examples. We further compare performance of our approach with the previous proposals.

5.1 Synthetic Data

We consider linear model (2), where the rows of design matrix $\mathbf{X}$ are fixed i.i.d. realizations from $\mathsf{N}(0, \Sigma)$, where $\Sigma \in \mathbb{R}^{p\times p}$ is a circulant symmetric matrix with entries Σ_{jk} given as follows for $j \le k$:

$$\Sigma_{jk} = \begin{cases} 1 & \text{if } k = j\,, \\ 0.1 & \text{if } k \in \{j+1, \dots, j+5\} \\ & \text{or } k \in \{j+p-5, \dots, j+p-1\}\,, \\ 0 & \text{for all other } j \le k\,. \end{cases} \tag{56}$$

Regarding the regression coefficient, we consider a uniformly random support $S \subseteq [p]$, with $|S| = s_0$ and let $\theta_{0,i} = b$ for $i \in S$ and $\theta_{0,i} = 0$ otherwise. The measurement errors are $W_i \sim \mathsf{N}(0, 1)$, for $i \in [n]$. We consider several configurations of (n, p, s_0, b) and for each configuration report our results based on 20 independent realizations of the model with fixed design and fixed regression coefficients. In other words, we repeat experiments over 20 independent realization of the measurement errors.

We use the regularization parameter $\lambda = 4\widehat{\sigma}\sqrt{(2\log p)/n}$, where $\widehat{\sigma}$ is given by the scaled LASSO as per equation (31) with $\widetilde{\lambda} = 10\sqrt{(2\log p)/n}$. Furthermore, parameter μ (cf. Equation 4) is set to

$$\mu = 2\sqrt{\frac{\log p}{n}}\,.$$

This choice of μ is guided by Theorem 7 (b).

Throughout, we set the significance level $\alpha = 0.05$.

Confidence intervals. For each configuration, we consider 20 independent realizations of measurement noise and for each parameter $\theta_{0,i}$, we compute the average length of the corresponding confidence interval, denoted by $\mathrm{Avglength}(J_i(\alpha))$ where $J_i(\alpha)$ is given by equation (32) and the average is taken over the realizations. We then define

$$\ell \equiv p^{-1}\sum_{i\in[p]} \mathrm{Avglength}(J_i(\alpha))\,. \tag{57}$$

We also consider the average length of intervals for the active and inactive parameters, as follows:

$$\ell_S \equiv s_0^{-1}\sum_{i\in S} \mathrm{Avglength}(J_i(\alpha))\,, \quad \ell_{S^c} \equiv (p-s_0)^{-1}\sum_{i\in S^c} \mathrm{Avglength}(J_i(\alpha))\,. \tag{58}$$

Similarly, we consider average coverage for individual parameters. We define the following three metrics:

$$\widehat{\mathsf{Cov}} \equiv p^{-1}\sum_{i\in[p]} \widehat{\mathbb{P}}[\theta_{0,i} \in J_i(\alpha)]\,, \tag{59}$$

$$\widehat{\mathsf{Cov}}_S \equiv s_0^{-1}\sum_{i\in S} \widehat{\mathbb{P}}[\theta_{0,i} \in J_i(\alpha)]\,, \tag{60}$$

$$\widehat{\mathsf{Cov}}_{S^c} \equiv (p-s_0)^{-1}\sum_{i\in S^c} \widehat{\mathbb{P}}[0 \in J_i(\alpha)]\,, \tag{61}$$

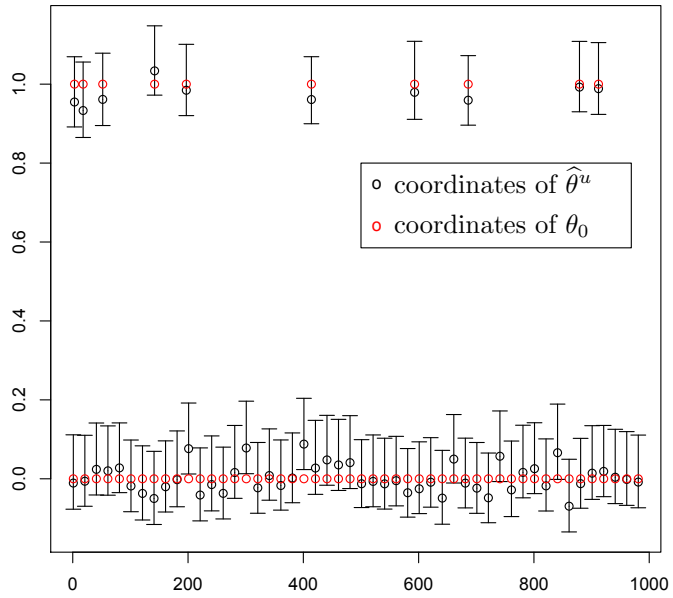


Figure 1: 95% confidence intervals for one realization of configuration $(n, p, s_0, b) = (1000, 600, 10, 1)$. For clarity, we plot the confidence intervals for only 100 of the 1000 parameters. The true parameters $\theta_{0,i}$ are in red and the coordinates of the debiased estimator $\widehat{\theta}^u$ are in black.

where $\widehat{\mathbb{P}}$ denotes the empirical probability computed based on the 20 realizations for each configuration. The results are reported in Table 1. In Figure 1, we plot the constructed 95%-confidence intervals for one realization of configuration $(n, p, s_0, b) = (1000, 600, 10, 1)$. For sake of clarity, we plot the confidence intervals for only 100 of the 1000 parameters.

False positive rates and statistical powers. Table 2 summarizes the false positive rates and the statistical powers achieved by our proposed method, the multisample-splitting method (Meinshausen et al., 2009), and the ridge-type projection estimator (Bühlmann, 2013) for several configurations. The results are obtained by taking average over 20 independent realizations of measurement errors for each configuration. As we see the multisample-splitting achieves false positive rate 0 on all of the configurations considered here, making no type I error. However, the true positive rate is always smaller than that of our proposed method. By contrast, our method achieves false positive rate close to the pre-assigned significance level $\alpha = 0.05$ and obtains much higher true positive rate. Similar to the multisample-splitting, the ridge-type projection estimator is conservative and achieves false positive rate smaller than α. This, however, comes at the cost of a smaller true positive rate than our method. It is worth noting that an ideal testing procedure should allow to control the level of statistical significance α, and obtain the maximum true positive rate at that level.

Here, we used the R-package hdi to test multisample-splitting and the ridge-type projection estimator.

Configuration \ Measure	ℓ	ℓ_S	ℓ_{S^c}	$\widehat{\mathsf{Cov}}$	$\widehat{\mathsf{Cov}}_S$	$\widehat{\mathsf{Cov}}_{S^c}$
$(1000, 600, 10, 0.5)$	0.1870	0.1834	0.1870	0.9766	0.9600	0.9767
$(1000, 600, 10, 0.25)$	0.1757	0.1780	0.1757	0.9810	0.9000	0.9818
$(1000, 600, 10, 0.1)$	0.1809	0.1823	0.1809	0.9760	1	0.9757
$(1000, 600, 30, 0.5)$	0.2107	0.2108	0.2107	0.9780	0.9866	0.9777
$(1000, 600, 30, 0.25)$	0.1956	0.1961	0.1956	0.9660	0.9660	0.9659
$(1000, 600, 30, 0.1)$	0.2023	0.2043	0.2023	0.9720	0.9333	0.9732
$(2000, 1500, 50, 0.5)$	0.1383	0.1391	0.1383	0.9754	0.9800	0.9752
$(2000, 1500, 50, 0.25)$	0.1356	0.1363	0.1355	0.9720	0.9600	0.9723
$(2000, 1500, 50, 0.1)$	0.1361	0.1361	0.1361	0.9805	1	0.9800
$(2000, 1500, 25, 0.5)$	0.1233	0.1233	0.1233	0.9731	0.9680	0.9731
$(2000, 1500, 25, 0.25)$	0.1208	0.1208	0.1208	0.9735	1	0.9731
$(2000, 1500, 25, 0.1)$	0.1242	0.1237	0.1242	0.9670	0.9200	0.9676

Table 1: Simulation results for the synthetic data described in Section 5.1. The results corresponds to 95% confidence intervals.

Let $Z = (z_i)_{i=1}^p$ denote the vector with $z_i \equiv \sqrt{n}(\widehat{\theta}_i^u - \theta_{0,i})/\widehat{\sigma}\sqrt{[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}}$. Figure 2 shows the sample quantiles of Z versus the quantiles of the standard normal distribution for one realization of the configuration $(n, p, s_0, b) = (1000, 600, 10, 1)$. The scattered points are close to the line with unit slope and zero intercept. This confirms the result of Theorem 13 regarding the Gaussianity of the entries z_i.

For the same problem, in Figure 3 we plot the empirical CDF of the computed p-values restricted to the variables outside the support. Clearly, the p-values for these entries are uniformly distributed as expected.

5.2 Real Data

As a real data example, we consider a high-throughput genomic data set concerning riboflavin (vitamin B_2) production rate. This data set is made publicly available by Bühlmann et al. (2014) and contains $n = 71$ samples and $p = 4,088$ covariates corresponding to $p = 4,088$ genes. For each sample, there is a real-valued response variable indicating the logarithm of the riboflavin production rate along with the logarithm of the expression level of the $p = 4,088$ genes as the covariates.

Following Bühlmann et al. (2014), we model the riboflavin production rate as a linear model with $p = 4,088$ covariates and $n = 71$ samples, as in Equation (1). We use the R package glmnet (Friedman et al., 2010) to fit the LASSO estimator. Similar to the previous section, we use the regularization parameter $\lambda = 4\widehat{\sigma}\sqrt{(2\log p)/n}$, where $\widehat{\sigma}$ is given by the scaled LASSO as per equation (31) with $\widetilde{\lambda} = 10\sqrt{(2\log p)/n}$. This leads to the choice $\lambda = 0.036$. The resulting model contains 30 genes (plus an intercept term) corresponding to the nonzero parameters of the lasso estimator.

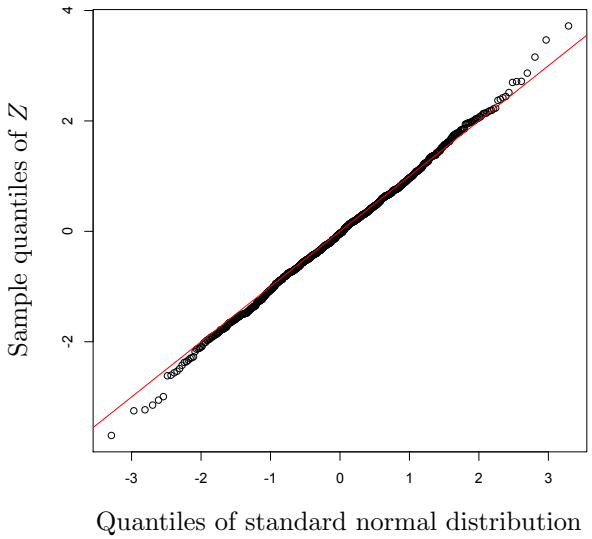


Figure 2: Q-Q plot of Z for one realization of configuration $(n, p, s_0, b) = (1000, 600, 10, 1)$.

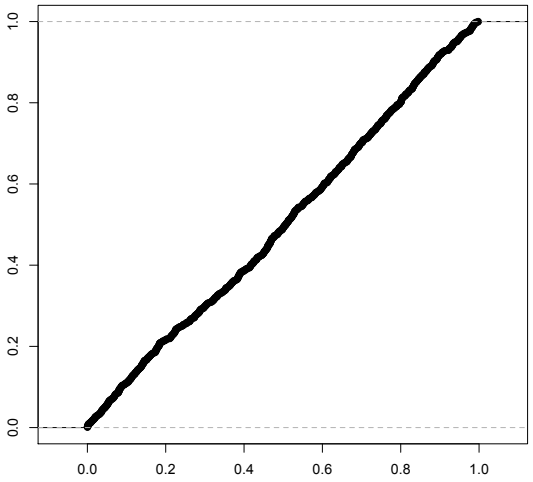


Figure 3: Empirical CDF of the computed p-values (restricted to entries outside the support) for one realization of configuration $(n, p, s_0, b) = (1000, 600, 10, 1)$. Clearly the plot confirms that the p-values are distributed according to uniform distribution.

	Our method		Multisample-splitting		Ridge projection estimator	
Configuration	FP	TP	FP	TP	FP	TP
$(1000, 600, 10, 0.5)$	0.0452	1	0	1	0.0284	0.8531
$(1000, 600, 10, 0.25)$	0.0393	1	0	0.4	0.02691	0.7506
$(1000, 600, 10, 0.1)$	0.0383	0.8	0	0	0.2638	0.6523
$(1000, 600, 30, 0.5)$	0.0433	1	0	1	0.0263	0.8700
$(1000, 600, 30, 0.25)$	0.0525	1	0	0.4	0.2844	0.8403
$(1000, 600, 30, 0.1)$	0.0402	0.7330	0	0	0.2238	0.6180
$(2000, 1500, 50, 0.5)$	0.0421	1	0	1	0.0301	0.9013
$(2000, 1500, 50, 0.25)$	0.0415	1	0	1	0.0292	0.8835
$(2000, 1500, 50, 0.1)$	0.0384	0.9400	0	0	0.02655	0.7603
$(2000, 1500, 25, 0.5)$	0.0509	1	0	1	0.0361	0.9101
$(2000, 1500, 25, 0.25)$	0.0481	1	0	1	0.3470	0.8904
$(2000, 1500, 25, 0.1)$	0.0551	1	0	0.16	0.0401	0.8203

Table 2: Simulation results for the synthetic data described in Section 5.1. The false positive rates (FP) and the true positive rates (TP) are computed at significance level $\alpha = 0.05$.

We use Equation (35) to construct p-values for different genes. Adjusting FWER to 5% significance level, we find two significant genes, namely genes YXLD-at and YXLE-at. By contrast, the multisample-splitting method proposed in Meinshausen et al. (2009) finds only the gene YXLD-at at the FWER-adjusted 5% significance level. Also the Ridge-type projection estimator, proposed in Bühlmann (2013), returns no significance gene. (See Bühlmann et al. 2014 for further discussion on these methods.) This indicates that these methods are more conservative and produce typically larger p-values.

In Figure 4 we plot the empirical CDF of the computed p-values for riboflavin example. Clearly the plot confirms that the p-values are distributed according to uniform distribution.

6. Proofs

This section is devoted to the proofs of theorems and main lemmas.

6.1 Proof of Theorem 6

Substituting $Y = \mathbf{X}\theta_0 + W$ in the definition (7), we get

$$\begin{aligned} \widehat{\theta}^* &= \widehat{\theta}^n + \frac{1}{n} M \mathbf{X}^{\mathsf{T}} \mathbf{X}(\theta_0 - \widehat{\theta}^n) + \frac{1}{n} M \mathbf{X}^{\mathsf{T}} W \\ &= \theta_0 + \frac{1}{\sqrt{n}} Z + \frac{1}{\sqrt{n}} \Delta \,, \end{aligned} \tag{62}$$

with Z, Δ defined as per the theorem statement. Further Z is Gaussian with the stated covariance because it is a linear function of the Gaussian vector $W \sim \mathsf{N}(0, \sigma^2 \, \mathrm{I}_{p \times p})$.

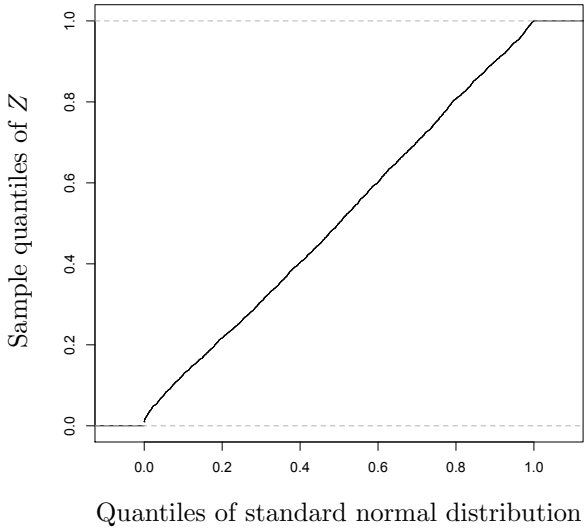


Figure 4: Empirical CDF of the computed p-values for riboflavin example. Clearly the plot confirms that the p-values are distributed according to uniform distribution.

We are left with the task of proving the bound (11) on Δ. Note that by definition (4), we have

$$\|\Delta\|_\infty \le \sqrt{n}\,|M\widehat{\Sigma} - \mathrm{I}|_\infty\,\|\widehat{\theta}^n - \theta_0\|_1 = \sqrt{n}\,\mu_*\|\widehat{\theta}^n - \theta_0\|_1\,. \tag{63}$$

By Bühlmann and van de Geer (2011, Theorem 6.1, Lemma 6.2), we have, for any $\lambda \ge 4\sigma\sqrt{2K\log(pe^{t^2/2})/n}$

$$\mathbb{P}\Big(\|\widehat{\theta}^n - \theta_0\|_1 \ge \frac{4\lambda s_0}{\phi_0^2}\Big) \le 2\,e^{-t^2/2}\,. \tag{64}$$

(More precisely, we consider the trivial generalization of Bühlmann and van de Geer 2011, Lemma 6.2 to the case $(\mathbf{X}^T\mathbf{X}/n)_{ii} \le K$, instead of $(\mathbf{X}^T\mathbf{X}/n)_{ii} = 1$ for all $i \in [p]$.)

Substituting Equation (63) in the last bound, we get

$$\mathbb{P}\Big(\|\Delta\|_\infty \ge \frac{4\lambda\mu_* s_0\sqrt{n}}{\phi_0^2}\Big) \le 2\,e^{-t^2/2}\,. \tag{65}$$

Finally, the claim follows by selecting t so that $e^{t^2/2} = p^{c_0}$.

6.2 Proof of Theorem 7.(a)

Note that the event $\mathcal{E}_n$ requires two conditions. Hence, its complement is given by

$$\mathcal{E}_n(\phi_0, s_0, K)^c = \mathcal{B}_{1,n}(\phi_0, s_0) \cup \mathcal{B}_{2,n}(K)\,, \tag{66}$$

$$\mathcal{B}_{1,n}(\phi_0, s_0) \equiv \Big\{\mathbf{X} \in \mathbb{R}^{n\times p} : \min_{S:\,|S|\le s_0} \phi(\widehat{\Sigma}, S) < \phi_0,\ \ \widehat{\Sigma} = (\mathbf{X}^\mathsf{T}\mathbf{X}/n)\Big\}\,, \tag{67}$$

$$\mathcal{B}_{2,n}(K) \equiv \Big\{\mathbf{X} \in \mathbb{R}^{n\times p} : \max_{i\in[p]} \widehat{\Sigma}_{i,i} > K,\ \ \widehat{\Sigma} = (\mathbf{X}^\mathsf{T}\mathbf{X}/n)\Big\}\,. \tag{68}$$

We will bound separately the probability of $\mathcal{B}_{1,n}$ and the probability of $\mathcal{B}_{2,n}$. The claim of Theorem 7.(a) follows by union bound.

6.2.1 Controlling $\mathcal{B}_{1,n}(\phi_0, s_0)$

It is also useful to recall the notion of restricted eigenvalue, introduced by Bickel, Ritov and Tsybakov (Bickel et al., 2009).

Definition 22 *Given a symmetric matrix $Q \in \mathbb{R}^{p\times p}$ an integer $s_0 \geq 1$, and $L > 0$, the restricted eigenvalue of Q is defined as*

$$\phi^2_{\mathrm{RE}}(Q, s_0, L) \equiv \min_{S\subseteq[p],|S|\leq s_0} \min_{\theta\in\mathbb{R}^p} \Big\{ \frac{\langle\theta, Q\,\theta\rangle}{\|\theta_S\|_2^2} : \ \ \theta \in \mathbb{R}^p, \ \ \|\theta_{S^c}\|_1 \leq L\|\theta_S\|_1 \Big\}. \tag{69}$$

Rudelson and Shuheng (2013) prove that, if the population covariance satisfies the restricted eigenvalue condition, then the sample covariance satisfies it as well, with high probability. More precisely, by Rudelson and Shuheng (2013, Theorem 6) we have

$$\mathbb{P}\Big(\phi_{\mathrm{RE}}(\widehat{\Sigma}, s_0, 3) \geq \frac{1}{2}\phi_{\mathrm{RE}}(\Sigma, s_0, 9)\Big) \geq 1 - 2e^{-n/(4c_*\kappa^4)}\,, \tag{70}$$

for some $c_* \leq 2000$, $m \equiv 6\times 10^4 s_0 C^2_{\max}/\phi^2_{\mathrm{RE}}(\Sigma, s_0, 9)$, and every $n \geq 4c_* m\kappa^4 \log(120ep/m)$.

Note that $\phi_{\mathrm{RE}}(\Sigma, s_0, 9) \geq \sigma_{\min}(\Sigma)^{1/2} \geq \sqrt{C_{\min}}$, and by Cauchy-Schwartz,

$$\min_{S:|S|\leq s_0} \phi(\widehat{\Sigma}, S) \geq \phi_{\mathrm{RE}}(\widehat{\Sigma}, s_0, 3)\,.$$

With the definitions in the statement (cf. Equation 13), we therefore have

$$\mathbb{P}\Big(\min_{S:|S|\leq s_0} \phi(\widehat{\Sigma}, S) \geq \frac{1}{2}\sqrt{C_{\min}}\Big) \geq 1 - 2e^{-c_1 n}\,. \tag{71}$$

Equivalently, $\mathbb{P}(\mathcal{B}_{1,n}(\phi_0, s_0)) \leq 2\,e^{-c_1 n}$.

6.2.2 Controlling $\mathcal{B}_{2,n}(K)$

By definition

$$\widehat{\Sigma}_{ii} - 1 = \frac{1}{n}\sum_{\ell=1}^n (\langle X_\ell, e_i\rangle^2 - 1) = \frac{1}{n}\sum_{\ell=1}^n u_\ell, \ . \tag{72}$$

Note that u_ℓ are independent centered random variables. Further, (recalling that, for any random variables U, V, $\|U + V\|_{\psi_1} \leq \|U\|_{\psi_1} + \|V\|_{\psi_1}$, and $\|U^2\|_{\psi_1} \leq 2\|U\|^2_{\psi_2}$) they are subexponential with subexponential norm

$$\begin{aligned}
\|u_\ell\|_{\psi_1} &\leq 2\|\langle X_\ell, e_i\rangle^2\|_{\psi_1} \leq 4\|\langle X_\ell, e_i\rangle\|^2_{\psi_1} \\
&\leq 4\|\langle \Sigma^{-1/2} X_\ell, \Sigma^{1/2} e_i\rangle\|^2_{\psi_1} \\
&\leq 4\kappa^2\|\Sigma^{1/2} e_i\|_2^2 = 4\kappa^2\Sigma_{ii} = 4\kappa^2\,.
\end{aligned}$$

By Bernstein-type inequality for centered subexponential random variables, cf. Vershynin (2012), we get

$$\mathbb{P}\Big\{\frac{1}{n}\Big|\sum_{\ell=1}^{n} u_\ell\Big| \ge \varepsilon\Big\} \le 2\exp\Big[-\frac{n}{6}\min\Big((\frac{\varepsilon}{4e\kappa^2})^2, \frac{\varepsilon}{4e\kappa^2}\Big)\Big]. \tag{73}$$

Hence, for all ε such that $\varepsilon/(e\kappa^2) \in [\sqrt{(48\log p)/n}, 4]$,

$$\mathbb{P}\Big(\max_{i\in[p]} \widehat{\Sigma}_{ii} \ge 1+\varepsilon\Big) \le 2p\,\exp\Big(-\frac{n\varepsilon^2}{24e^2\kappa^4}\Big) \le 2e^{-c_1 n}\,, \tag{74}$$

which implies $\mathbb{P}(\mathbf{X}\in\mathcal{B}_{2,n}(K)) \le 2\,e^{-c_1 n}$ for all $K-1 \ge 20\kappa^2\sqrt{(\log p)/n} \ge \sqrt{(48e^2\kappa^4\log p)/n}$.

6.3 Proof of Theorem 7.(*b*)

Obviously, we have

$$\mu_{\min}(\mathbf{X}) \le \big|\Sigma^{-1}\widehat{\Sigma} - \mathrm{I}\big|\,, \tag{75}$$

and hence the statement follows immediately from the following estimate.

Lemma 23 *Consider a random design matrix $\mathbf{X}\in\mathbb{R}^{p\times p}$, with i.i.d. rows having mean zero and population covariance Σ. Assume that*

(*i*) *We have $\sigma_{\min}(\Sigma) \ge C_{\min} > 0$, and $\sigma_{\max}(\Sigma) \le C_{\max} < \infty$.*

(*ii*) *The rows of $X\Sigma^{-1/2}$ are sub-Gaussian with $\kappa = \|\Sigma^{-1/2}X_1\|_{\psi_2}$.*

Let $\widehat{\Sigma} = (\mathbf{X}^{\mathsf{T}}\mathbf{X})/n$ be the empirical covariance. Then, for any constant $C>0$, the following holds true.

$$\mathbb{P}\left\{\Big|\Sigma^{-1}\widehat{\Sigma} - \mathrm{I}\Big|_\infty \ge a\sqrt{\frac{\log p}{n}}\right\} \le 2p^{-c_2}\,, \tag{76}$$

with $c_2 = (a^2 C_{\min})/(24e^2\kappa^4 C_{\max}) - 2$.

Proof [Proof of Lemma 23] The proof is based on Bernstein-type inequality for subexponential random variables (Vershynin, 2012). Let $\tilde{X}_\ell = \Sigma^{-1/2}X_\ell$, for $\ell\in[n]$, and write

$$Z \equiv \Sigma^{-1}\widehat{\Sigma} - \mathrm{I} = \frac{1}{n}\sum_{\ell=1}^{n}\Big\{\Sigma^{-1}X_\ell X_\ell^{\mathsf{T}} - \mathrm{I}\Big\} = \frac{1}{n}\sum_{\ell=1}^{n}\Big\{\Sigma^{-1/2}\tilde{X}_\ell\tilde{X}_\ell^{\mathsf{T}}\Sigma^{1/2} - \mathrm{I}\Big\}\,.$$

Fix $i,j\in[p]$, and for $\ell\in[n]$, let $v_\ell^{(ij)} = \langle\Sigma^{-1/2}_{i,\cdot}, \tilde{X}_\ell\rangle\langle\Sigma^{1/2}_{j,\cdot}, \tilde{X}_\ell\rangle - \delta_{i,j}$, where $\delta_{i,j} = \mathbf{1}_{\{i=j\}}$. Notice that $\mathbb{E}(v_\ell^{(ij)}) = 0$, and the $v_\ell^{(ij)}$ are independent for $\ell\in[n]$. Also, $Z_{i,j} = (1/n)\sum_{\ell=1}^n v_\ell^{(ij)}$. By Vershynin (2012, Remark 5.18), we have

$$\|v_\ell^{(ij)}\|_{\psi_1} \le 2\|\langle\Sigma^{-1/2}_{i,\cdot}, \tilde{X}_\ell\rangle\langle\Sigma^{1/2}_{j,\cdot}, \tilde{X}_\ell\rangle\|_{\psi_1}.$$

Moreover, for any two random variables X and Y, we have

$$\begin{aligned}\|XY\|_{\psi_1} &= \sup_{p\geq 1} p^{-1}\mathbb{E}(|XY|^p)^{1/p} \\ &\leq \sup_{p\geq 1} p^{-1}\mathbb{E}(|X|^{2p})^{1/2p}\,\mathbb{E}(|Y|^{2p})^{1/2p} \\ &\leq 2\left(\sup_{q\geq 2} q^{-1/2}\mathbb{E}(|X|^q)^{1/q}\right)\left(\sup_{q\geq 2} q^{-1/2}\mathbb{E}(|Y|^q)^{1/q}\right) \\ &\leq 2\|X\|_{\psi_2}\,\|Y\|_{\psi_2}\,.\end{aligned}$$

Hence, by assumption (ii), we obtain

$$\begin{aligned}\|v_\ell^{(ij)}\|_{\psi_1} &\leq 2\|\langle\Sigma_{i,\cdot}^{-1/2},\tilde{X}_\ell\rangle\|_{\psi_2}\|\langle\Sigma_{j,\cdot}^{1/2},\tilde{X}_\ell\rangle\|_{\psi_2} \\ &\leq 2\|\Sigma_{i,\cdot}^{-1/2}\|_2\|\Sigma_{j,\cdot}^{1/2}\|_2\kappa^2 \leq 2\sqrt{C_{\max}/C_{\min}}\,\kappa^2\,.\end{aligned}$$

Let $\kappa' = 2\sqrt{C_{\max}/C_{\min}}\kappa^2$. Applying Bernstein-type inequality for centered sub-exponential random variables (Vershynin, 2012), we get

$$\mathbb{P}\Big\{\frac{1}{n}\Big|\sum_{\ell=1}^n v_\ell^{(ij)}\Big| \geq \varepsilon\Big\} \leq 2\exp\Big[-\frac{n}{6}\min\Big((\frac{\varepsilon}{e\kappa'})^2,\frac{\varepsilon}{e\kappa'}\Big)\Big]\,.$$

Choosing $\varepsilon = a\sqrt{(\log p)/n}$, and assuming $n \geq [a/(e\kappa')]^2\log p$, we arrive at

$$\mathbb{P}\left\{\frac{1}{n}\Big|\sum_{\ell=1}^n v_\ell^{(ij)}\Big| \geq a\sqrt{\frac{\log p}{n}}\right\} \leq 2p^{-a^2/(6e^2\kappa'^2)}\,.$$

The result follows by union bounding over all possible pairs $i,j\in[p]$. ■

6.4 Proof of Theorem 8

Let

$$\Delta_0 \equiv \Big(\frac{16ac\,\sigma}{C_{\min}}\Big)\frac{s_0\log p}{\sqrt{n}} \tag{77}$$

be a shorthand for the bound on $\|\Delta\|_\infty$ appearing in Equation (17). Then we have

$$\begin{aligned}\mathbb{P}\Big(\|\Delta\|_\infty \geq \Delta_0\Big) \leq& \mathbb{P}\Big(\big\{\|\Delta\|_\infty \geq \Delta_0\big\}\cap\mathcal{E}_n(\sqrt{C_{\min}}/2,s_0,3/2)\cap\mathcal{G}_n(a)\Big) \\ &+\mathbb{P}\big(\mathcal{E}_n(\sqrt{C_{\min}}/2,s_0,3/2)\big)+\mathbb{P}\big(\mathcal{G}_n^c(a)\big) \\ \leq& \mathbb{P}\Big(\big\{\|\Delta\|_\infty \geq \Delta_0\big\}\cap\mathcal{E}_n(\sqrt{C_{\min}}/2,s_0,3/2)\cap\mathcal{G}_n(a)\Big)+4\,e^{-c_1 n}+2\,p^{-c_2}\,,\end{aligned}$$

where, in the first equation $\mathcal{A}^c$ denotes the complement of event $\mathcal{A}$ and the second inequality follows from Theorem 7. Notice, in particular, that the bound (13) can be applied for

$K = 3/2$ since, under the present assumptions $20\kappa^2\sqrt{(\log p)/n} \le 1/2$. Finally

$$\mathbb{P}\Big(\big\{\|\Delta\|_\infty \ge \Delta_0\big\} \cap \mathcal{E}_n(\sqrt{C_{\min}}/2, s_0, 3/2) \cap \mathcal{G}_n(a)\Big) \le \sup_{\mathbf{X}\in\mathcal{E}_n(\sqrt{C_{\min}}/2,s_0,3/2)\cap\mathcal{G}_n(a)} \mathbb{P}\Big(\|\Delta\|_\infty \ge \Delta_0 \Big| \mathbf{X}\Big) \le 2\, p^{-\tilde{c}_0}\,. \quad (78)$$

Here the last inequality follows from Theorem 6 applied per given $\mathbf{X} \in \mathcal{E}_n(\sqrt{C_{\min}}/2, s_0, 3/2)\cap \mathcal{G}_n(a)$ and hence using the bound (11) with $\phi_0 = \sqrt{C_{\min}}/2$, $K = 3/2$, $\mu_* = a\sqrt{(\log p)/n}$.

6.5 Proof of Lemma 13

We will prove that, under the stated assumptions

$$\limsup_{n\to\infty} \sup_{\|\theta_0\|_0\le s_0} \mathbb{P}\left\{\frac{\sqrt{n}(\widehat{\theta}_i^u - \theta_{0,i})}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}} \le x\right\} \le \Phi(x)\,. \quad (79)$$

A matching lower bound follows by a completely analogous argument.

Notice that by Equation (16), we have

$$\frac{\sqrt{n}(\widehat{\theta}_i^u - \theta_{0,i})}{\sigma[M\widehat{\Sigma}M^{\mathsf{T}}]_{ii}^{1/2}} = \frac{e_i^{\mathsf{T}} M\mathbf{X}^{\mathsf{T}}W}{\sigma[M\widehat{\Sigma}M^{\mathsf{T}}]_{ii}^{1/2}} + \frac{\Delta_i}{\sigma[M\widehat{\Sigma}M^{\mathsf{T}}]_{ii}^{1/2}}\,. \quad (80)$$

Let $V = \mathbf{X}M^{\mathsf{T}}e_i/(\sigma[M\widehat{\Sigma}M^{\mathsf{T}}]_{ii}^{1/2})$ and $\widetilde{Z} \equiv V^{\mathsf{T}}W$. We claim that $\tilde{Z} \sim \mathsf{N}(0,1)$. To see this, note that $\|V\|_2 = 1$, and V and W are independent. Hence,

$$\mathbb{P}(\widetilde{Z} \le x) = \mathbb{E}\{\mathbb{P}(V^{\mathsf{T}}W \le x|V)\} = \mathbb{E}\{\Phi(x)|V\} = \Phi(x)\,, \quad (81)$$

which proves our claim. In order to prove Equation (79), fix $\varepsilon > 0$ and write

$$\begin{aligned}
\mathbb{P}\left(\frac{\sqrt{n}(\widehat{\theta}_i^u - \theta_{0,i})}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}} \le x\right) &= \mathbb{P}\left(\frac{\sigma}{\widehat{\sigma}}\widetilde{Z} + \frac{\Delta_i}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}} \le x\right)\\
&\le \mathbb{P}\Big(\frac{\sigma}{\widehat{\sigma}}\widetilde{Z} \le x + \varepsilon\Big) + \mathbb{P}\left(\frac{|\Delta_i|}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}} \ge \varepsilon\right)\\
&\le \mathbb{P}\Big(\widetilde{Z} \le x + 2\varepsilon + \varepsilon|x|\Big) + \mathbb{P}\left(\frac{|\Delta_i|}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}} \ge \varepsilon\right)\\
&\quad + \mathbb{P}\Big(\Big|\frac{\widehat{\sigma}}{\sigma} - 1\Big| \ge \varepsilon\Big)\,.
\end{aligned}$$

By taking the limit and using assumption (29), we obtain

$$\limsup_{n\to\infty} \sup_{\|\theta_0\|_0\le s_0} \mathbb{P}\left(\frac{\sqrt{n}(\widehat{\theta}_i^u - \theta_{0,i})}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}} \le x\right) \le \Phi(x + 2\varepsilon + \varepsilon|x|) + \limsup_{n\to\infty} \sup_{\|\theta_0\|_0\le s_0} \mathbb{P}\left(\frac{|\Delta_i|}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}} \ge \varepsilon\right)\,.$$

Since $\varepsilon > 0$ is arbitrary, it is therefore sufficient to show that the limit on the right hand side vanishes for any $\varepsilon > 0$.

Note that $[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i} \geq 1/(4\widehat{\Sigma}_{ii})$ for all n large enough, by Lemma 12, and since $\mu = a\sqrt{(\log p)/n} \to 0$ as $n, p \to \infty$. We have therefore

$$\begin{aligned}\mathbb{P}\left(\frac{|\Delta_i|}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}} \geq \varepsilon\right) &\leq \mathbb{P}\Big(\frac{2}{\widehat{\sigma}}\widehat{\Sigma}_{ii}^{1/2}\,|\Delta_i| \geq \varepsilon\Big)\\ &\leq \mathbb{P}\Big(\frac{5}{\sigma}\,|\Delta_i| \geq \varepsilon\Big) + \mathbb{P}\Big(\frac{\widehat{\sigma}}{\sigma} \leq \frac{1}{2}\Big) + \mathbb{P}(\widehat{\Sigma}_{ii} \geq \sqrt{2})\,.\end{aligned}$$

Note that $\mathbb{P}\big((\widehat{\sigma}/\sigma) \leq 1/2\big) \to 0$ by assumption (29), and $\mathbb{P}(\widehat{\Sigma}_{ii} \geq \sqrt{2}) \to 0$ by Theorem 7.(b). Hence

$$\begin{aligned}\limsup_{n\to\infty}\sup_{\|\theta_0\|_0\leq s_0}\mathbb{P}\left(\frac{|\Delta_i|}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}} \geq \varepsilon\right) &\leq \limsup_{n\to\infty}\sup_{\|\theta_0\|_0\leq s_0}\mathbb{P}\Big(\|\Delta\|_\infty \geq \frac{\varepsilon\sigma}{5}\Big)\\ &\leq \limsup_{n\to\infty}\big(4\,e^{-c_1 n} + 4\,p^{-(\tilde{c}_0\wedge c_2)}\big) = 0\,,\end{aligned}$$

where the last inequality follows from Equation (17), recalling that $s_0 = o(\sqrt{n}/\log p)$ and hence $(16acs_0\log p)/(C_{\min}\sqrt{n}) \leq \varepsilon/5$ for all n large enough.

This completes the proof of Equation (79). The matching lower bound follows by the same argument.

6.6 Proof of Theorem 16

We begin with proving Equation (39). Defining $Z_i \equiv \sqrt{n}(\widehat{\theta}_i^u - \theta_{0,i})/(\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2})$, we have

$$\begin{aligned}\lim_{n\to\infty}\alpha_{i,n}(\widehat{T}) &= \lim_{n\to\infty}\sup_{\theta_0}\Big\{\mathbb{P}(P_i \leq \alpha) : i \in [p],\, \|\theta_0\|_0 \leq s_0,\, \theta_{0,i} = 0\Big\}\\ &= \lim_{n\to\infty}\sup_{\theta_0}\Big\{\mathbb{P}\Big(\Phi^{-1}(1-\frac{\alpha}{2}) \leq \frac{\sqrt{n}|\widehat{\theta}_i^u|}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}}\Big) :\, i \in [p],\, \|\theta_0\|_0 \leq s_0,\, \theta_{0,i} = 0\Big\}\\ &= \lim_{n\to\infty}\sup_{\theta_0}\Big\{\mathbb{P}\Big(\Phi^{-1}(1-\frac{\alpha}{2}) \leq |Z_i|\Big) :\, i \in [p],\, \|\theta_0\|_0 \leq s_0\Big\} \leq \alpha\,,\end{aligned}$$

where the last inequality follows from Lemma 13.

We next prove Equation (40). Recall that $\Sigma_{\cdot,i}^{-1}$ is a feasible solution of (4), for $1 \leq i \leq p$ with probability at least $1 - 2p^{-c_2}$, as per Lemma 23). On this event, letting m_i be the solution of the optimization problem (4), we have

$$\begin{aligned}m_i^{\mathsf{T}}\widehat{\Sigma}m_i &\leq \Sigma_{i,\cdot}^{-1}\widehat{\Sigma}\Sigma_{\cdot,i}^{-1}\\ &= (\Sigma_{i,\cdot}^{-1}\widehat{\Sigma}\Sigma_{\cdot,i}^{-1} - \Sigma_{ii}^{-1}) + \Sigma_{i,i}^{-1}\\ &= \frac{1}{n}\sum_{j=1}^{N}(V_j^2 - \Sigma_{ii}^{-1}) + \Sigma_{i,i}^{-1}\,,\end{aligned}$$

where $V_j = \Sigma^{-1}_{i,\cdot} X_j$ are i.i.d. random variables with $\mathbb{E}(V_j^2) = \Sigma^{-1}_{ii}$ and sub-Gaussian norm

$$\|V_j\|_{\psi_2} \le \|\Sigma^{-1/2}_{i,\cdot}\|_2 \|\Sigma^{-1/2} X_j\|_{\psi_2} \le \kappa \sqrt{\Sigma^{-1}_{i,i}}.$$

Letting $U_j = V_j^2 - \Sigma^{-1}_{ii}$, we have that U_j is zero mean and sub-exponential with $\|U_j\|_{\psi_1} \le 2\|V_j^2\|_{\psi_1} \le 4\|V_j\|^2_{\psi_2} \le 4\kappa^2 \Sigma^{-1}_{ii} \le 4\kappa^2 \sigma_{\min}(\Sigma)^{-1} \le 4\kappa^2 C^{-1}_{\min} \equiv \kappa'$. Hence, by applying Bernstein inequality (as, for instance, in the proof of Lemma 23), we have, for $\varepsilon \le e\kappa'$,

$$\mathbb{P}\Big(m_i^{\mathsf{T}} \widehat{\Sigma} m_i \ge \Sigma^{-1}_{i,i} + \varepsilon \Big) \le 2\, e^{-(n/6)(\varepsilon/e\kappa')^2} + 2\, p^{-c_2}\,.$$

We can make $c_2 \ge 2$ by a suitable choice of a and therefore, by Borel-Cantelli we have the following almost surely

$$\limsup_{n\to\infty} [m_i^{\mathsf{T}} \widehat{\Sigma} m_i - \Sigma^{-1}_{i,i}] \le 0\,. \tag{82}$$

Now we are ready to prove the lower bound for the power. Let $z_* \equiv \Phi^{-1}(1-\alpha/2)$. Then,

$$\begin{aligned}
&\liminf_{n\to\infty} \frac{1-\beta_{i,n}(\widehat{T};\gamma)}{1-\beta^*_{i,n}(\gamma)} \\
&= \liminf_{n\to\infty} \frac{1}{1-\beta^*_i(\gamma;n)} \inf_{\theta_0} \Big\{ \mathbb{P}(P_i \le \alpha) :\ \|\theta_0\|_0 \le s_0,\ |\theta_{0,i}| \ge \gamma \Big\} \\
&= \liminf_{n\to\infty} \frac{1}{1-\beta^*_{i,n}(\gamma)} \inf_{\theta_0} \Big\{ \mathbb{P}\Big(z_* \le \frac{\sqrt{n}|\widehat{\theta}^u_i|}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]^{1/2}_{i,i}} \Big) : \|\theta_0\|_0 \le s_0,\ |\theta_{0,i}| \ge \gamma \Big\} \\
&= \liminf_{n\to\infty} \frac{1}{1-\beta^*_{i,n}(\gamma)} \inf_{\theta_0} \Big\{ \mathbb{P}\Big(z_* \le \Big| Z_i + \frac{\sqrt{n}\theta_{0,i}}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]^{1/2}_{i,i}} \Big| \Big) : \|\theta_0\|_0 \le s_0,\ |\theta_{0,i}| \ge \gamma \Big\} \\
&\overset{(a)}{\ge} \liminf_{n\to\infty} \frac{1}{1-\beta^*_{i,n}(\gamma)} \inf_{\theta_0} \Big\{ \mathbb{P}\Big(z_* \le \Big| Z_i + \frac{\sqrt{n}\gamma}{\sigma[\Sigma^{-1}_{i,i}]^{1/2}} \Big| \Big) : \|\theta_0\|_0 \le s_0 \Big\} \\
&= \liminf_{n\to\infty} \frac{1}{1-\beta^*_{i,n}(\gamma)} \Big\{ 1 - \Phi\Big(z_* - \frac{\sqrt{n}\gamma}{\sigma[\Sigma^{-1}_{i,i}]^{1/2}} \Big) + \Phi\Big(-z_* - \frac{\sqrt{n}\gamma}{\sigma[\Sigma^{-1}_{i,i}]^{1/2}} \Big) \Big\} \\
&= \liminf_{n\to\infty} \frac{1}{1-\beta^*_{i,n}(\gamma)} G\Big(\alpha, \frac{\sqrt{n}\gamma}{\sigma[\Sigma^{-1}_{i,i}]^{1/2}} \Big) = 1\,.
\end{aligned}$$

Here (a) follows from Equation (82) and the fact $|\theta_{0,i}| \ge \gamma$.

6.7 Proof of Theorem 21

Under the assumptions of Theorem 8 and assuming $s_0 = o(\sqrt{n}/\log p)$, we have

$$\sqrt{n}(\widehat{\theta}^u - \theta_0) = \frac{1}{\sqrt{n}} M \mathbf{X}^{\mathsf{T}} W + \Delta\,,$$

with $\|\Delta\|_\infty = o(1)$. Using Lemma 12, we have

$$\frac{\sqrt{n}(\widehat{\theta}_i^u - \theta_{0,i})}{\sigma[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}} = Z_i + o(1)\,, \quad \text{with } Z_i \equiv \frac{1}{\sqrt{n}}\frac{m_i^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}W}{\sigma[m_i^{\mathsf{T}}\widehat{\Sigma}m_i]^{1/2}}\,.$$

The following lemma characterizes the limiting distribution of $Z_i|\mathbf{X}$ which implies the validity of the proposed p-value P_i and confidence intervals.

Lemma 24 *Suppose that the noise variables W_i are independent with $\mathbb{E}(W_i) = 0$, and $\mathbb{E}(W_i^2) = \sigma^2$, and $E(|W_i|^{2+a}) \le C\,\sigma^{2+a}$ for some $a > 0$. Let $M = (m_1, \dots, m_p)^{\mathsf{T}}$ be the matrix with rows m_i^{T} obtained by solving optimization problem (54). For $i \in [p]$, define*

$$Z_i = \frac{1}{\sqrt{n}}\frac{m_i^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}W}{\sigma[m_i^{\mathsf{T}}\widehat{\Sigma}m_i]^{1/2}}\,.$$

Under the assumptions of Theorem 8, for any sequence $i = i(n) \in [p]$, and any $x \in \mathbb{R}$, we have

$$\lim_{n\to\infty} \mathbb{P}(Z_i \le x|\mathbf{X}) = \Phi(x)\,.$$

Lemma 24 is proved in Appendix A.2.

Acknowledgments

A.J. is supported by a Caroline and Fabian Pease Stanford Graduate Fellowship. This work was partially supported by the NSF CAREER award CCF-0743978, the NSF grant DMS-0806211, and the grants AFOSR/DARPA FA9550-12-1-0411 and FA9550-13-1-0036.

Appendix A. Proof of Technical Lemmas

This appendix contains the proofs of several technical steps needed in establishing our theoretical results.

A.1 Proof of Lemma 12

Let $C_i(\mu)$ be the optimal value of the optimization problem (4). We claim that

$$C_i(\mu) \ge \frac{(1-\mu)^2}{\widehat{\Sigma}_{ii}}\,. \tag{83}$$

To prove this claim notice that the constraint implies (by considering its i-th component):

$$1 - \langle e_i, \widehat{\Sigma}m\rangle \le \mu\,.$$

Therefore if $\tilde{m}$ is feasible and $c \ge 0$, then

$$\langle \tilde{m}, \widehat{\Sigma}\tilde{m}\rangle \ge \langle \tilde{m}, \widehat{\Sigma}\tilde{m}\rangle + c(1-\mu) - c\langle e_i, \widehat{\Sigma}\tilde{m}\rangle \ge \min_m \Big\{\langle m, \widehat{\Sigma}m\rangle + c(1-\mu) - c\langle e_i, \widehat{\Sigma}m\rangle\Big\}\,.$$

Minimizing over all feasible $\tilde{m}$ gives

$$C_i(\mu) \geq \min_m \left\{ \langle m, \widehat{\Sigma} m\rangle + c(1-\mu) - c\langle e_i, \widehat{\Sigma} m\rangle \right\}. \tag{84}$$

The minimum over m is achieved at $m = ce_i/2$. Plugging in for m, we get

$$C_i(\mu) \geq c(1-\mu) - \frac{c^2}{4}\widehat{\Sigma}_{ii} \tag{85}$$

Optimizing this bound over c, we obtain the claim (83), with the optimal choice being $c = 2(1-\mu)/\widehat{\Sigma}_{ii}$.

A.2 Proof of Lemma 24

Write

$$Z_i = \frac{1}{\sqrt{n}} \sum_{j=1}^n \xi_j \qquad \text{with} \quad \xi_j \equiv \frac{m_i^{\mathsf{T}} X_j W_j}{\sigma [m_i^{\mathsf{T}} \widehat{\Sigma} m_i]^{1/2}}\,.$$

Conditional on $\mathbf{X}$, the summands ξ_j are zero mean and independent. Further, $\sum_{j=1}^n \mathbb{E}(\xi_j^2|\mathbf{X}) = n$. We next prove the Lindeberg condition as per Equation (53). Let $c_n \equiv (m_i^{\mathsf{T}} \widehat{\Sigma} m_i)^{1/2}$. By Lemma 12, we have $\liminf_{n\to\infty} c_n \geq c_\infty > 0$, almost surely. If all the optimization problems in (54) are feasible, then $|\xi_j| \leq c_n^{-1} \|\mathbf{X} m_i\|_\infty \|W\|_\infty/\sigma \leq c_n^{-1} n^\beta (\|W\|_\infty/\sigma)$. Hence,

$$\begin{aligned}
\lim_{n\to\infty} \frac{1}{n}\sum_{j=1}^n \mathbb{E}\Big(\xi_j^2 \mathbb{I}_{\{|\xi_j|>\varepsilon\sqrt{n}\}}|\mathbf{X}\Big) &\leq \lim_{n\to\infty} \frac{1}{n}\sum_{j=1}^n \mathbb{E}\Big(\xi_j^2 \mathbb{I}_{\{\|W\|_\infty/\sigma > \varepsilon c_n n^{1/2-\beta}\}}|\mathbf{X}\Big) \\
&= \lim_{n\to\infty} \frac{1}{n}\sum_{j=1}^n \frac{m_i^{\mathsf{T}} X_j X_j^{\mathsf{T}} m_i}{m_i^{\mathsf{T}} \widehat{\Sigma} m_i} \mathbb{E}\Big(\widetilde{W}_j^2 \mathbb{I}_{\{\|\widetilde{W}\|_\infty > \varepsilon c_\infty n^{1/2-\beta}\}}\Big) \\
&\leq \lim_{n\to\infty} \mathbb{E}\Big(\widetilde{W}_1^2 \mathbb{I}_{\{|\widetilde{W}_1| > \varepsilon c_\infty n^{1/2-\beta}\}}\Big) \\
&\leq c'(\varepsilon) \lim_{n\to\infty} n^{-a(1/2-\beta)} \mathbb{E}\{|\widetilde{W}_1|^{2+a}\} = 0\,.
\end{aligned}$$

where $\widetilde{W}_j = W_j/\sigma$ and the last limit follows since $\beta < 1/2$ and $a > 0$.

Using Lindeberg central limit theorem, we obtain $Z_i|\mathbf{X}$ converges weakly to standard normal distribution, and hence, $\mathbf{X}$-almost surely

$$\lim_{n\to\infty} \mathbb{P}(Z_i \leq x|\mathbf{X}) = \Phi(x)\,.$$

What remains is to show that with high probability all the p optimization problems in (54) are feasible. In particular, we show that $\Sigma^{-1}_{i,\cdot}$ is a feasible solution to the i-th optimization problem, for $i \in [p]$. By Lemma 23, $|\Sigma^{-1}\widehat{\Sigma} - \mathrm{I}|_\infty \leq \mu$, with high probability. Moreover,

$$\begin{aligned}
\sup_{j\in[p]} \|\Sigma^{-1}_{i,\cdot} X_j\|_{\psi_2} &= \sup_{j\in[p]} \|\Sigma^{-1/2}_{i,\cdot} \Sigma^{-1/2} X_j\|_{\psi_2} \\
&= \|\Sigma^{-1/2}_{i,\cdot}\|_2 \sup_{j\in[p]} \|\Sigma^{-1/2} X_j\|_{\psi_2} \\
&= [\Sigma^{-1}_{i,i}]^{1/2} \sup_{j\in[p]} \|\Sigma^{-1/2} X_j\|_{\psi_2} = O(1)\,.
\end{aligned}$$

Using tail bound for sub-Gaussian variables $\Sigma_{i,\cdot}^{-1}X_j$ and union bounding over $j \in [n]$, we get

$$\mathbb{P}(\|\mathbf{X}\Sigma_{\cdot,i}^{-1}\|_\infty > n^\beta) \le ne^{-cn^{2\beta}},$$

for some constant $c > 0$. Note that $s_0 = o(\sqrt{n}/\log p)$ and $\beta > 1/4$ imply $p = e^{o(n^{2\beta})}$. Hence, almost surely, $\Sigma_{i,\cdot}^{-1}$ is a feasible solution to optimization problem (54), for all $i \in [p]$.

Appendix B. Corollaries of Theorem 8

In this appendix, we prove Corollary 10 and Corollary 11.

B.1 Proof of Corollary 10

By Theorem 6, for any $\mathbf{X} \in \mathcal{E}_n(\sqrt{C_{\min}}/2, s_0, 3/2) \cap \mathcal{G}_n(a)$, we have

$$\mathbb{P}\left\{\|\Delta\|_\infty \ge L\,c\Big|\mathbf{X}\right\} \le 2\,p^{1-(c^2/48)}, \quad L \equiv \frac{16a\sigma}{C_{\min}}\,\frac{s_0\log p}{\sqrt{n}}. \tag{86}$$

This is obtained by setting $\phi_0 = \sqrt{C_{\min}}/2$, $K = 3/2$, $\mu_* = a\sqrt{(\log p)/n}$ in Equation (11). Hence

$$\begin{aligned}\|\mathsf{Bias}(\widehat{\theta}^u)\|_\infty &\le \frac{1}{\sqrt{n}}\mathbb{E}\{\|\Delta\|_\infty\big|\mathbf{X}\}\\ &= \frac{L}{\sqrt{n}}\int_0^\infty \mathbb{P}\left\{\|\Delta\|_\infty \ge L\,c\Big|\mathbf{X}\right\}\,\mathrm{d}c\\ &\le \frac{2L}{\sqrt{n}}\int_0^\infty \min(1, p^{1-(c^2/48)})\,\mathrm{d}c \le \frac{10L}{\sqrt{n}},\end{aligned} \tag{87}$$

which coincides with Equation (21). The probability estimate (22) simply follows from Theorem 7 using union bound.

B.2 Proof of Corollary 11

By Theorem 7.(a), we have

$$\mathbb{P}\Big(\mathbf{X} \in \mathcal{E}_n(1/2, s_0, 3/2)\Big) \ge 1 - 4\,e^{-c_1 n}. \tag{88}$$

Further, by Lemma 23, with $\widehat{\Sigma} \equiv \mathbf{X}^\mathsf{T}\mathbf{X}/n$, we have

$$\mathbb{P}\Big(\mu_*(\mathbf{X};\mathrm{I}) \le 30\sqrt{\frac{\log p}{n}}\Big) \ge 1 - 2\,p^{-3}. \tag{89}$$

Hence, defining

$$\mathcal{B}_n \equiv \mathcal{E}_n(1/2, s_0, 3/2) \cap \left\{\mathbf{X} \in \mathbb{R}^{n\times p} :\ \mu_*(\mathbf{X};\mathrm{I}) \le 30\sqrt{\frac{\log p}{n}}\right\} \tag{90}$$

we have the desired probability bound (24). Let $\widehat{\theta}^n = \widehat{\theta}^n(Y, \mathbf{X}; \mathrm{I}, \lambda)$. By Theorem 6, we have, for any $\mathbf{X} \in \mathcal{B}_n$

$$\widehat{\theta}^* = \theta_0 + \frac{1}{\sqrt{n}} Z + \frac{1}{\sqrt{n}} \Delta\,, \quad Z|\mathbf{X} \sim \mathsf{N}(0, \sigma^2 \widehat{\Sigma})\,, \tag{91}$$

and further

$$\mathbb{P}\Big\{ \|\Delta\|_\infty \geq \frac{480 c\sigma s_0 \log p}{\sqrt{n}} \Big| \mathbf{X} \Big\} \leq 2p^{1-(c^2/48)}\,, \tag{92}$$

whence, proceeding as in the proof in the last section, we get, for some universal numerical constant $c_{**} \leq 4800$,

$$\|\mathsf{Bias}(\widehat{\theta}^*)\|_\infty \leq \frac{1}{\sqrt{n}} \mathbb{E}\Big\{ \|\Delta\|_\infty \Big| \mathbf{X} \Big\} \leq c_{**} \sigma \frac{s_0 \log p}{n}\,. \tag{93}$$

Next by Equation (28) we have

$$\big\|\mathsf{Bias}(\widehat{\theta}^n)\big\|_\infty \geq \Big| \lambda \big\| \mathbb{E}\{v(\widehat{\theta}^n)|\mathbf{X}\} \big\|_\infty - \big\|\mathsf{Bias}(\widehat{\theta}^*)\big\|_\infty \Big|\,. \tag{94}$$

Hence, in order to prove Equation (23), it is sufficient to prove that $\|\mathbb{E}\{v(\widehat{\theta}^n)|\mathbf{X}\}\|_\infty \geq 2/3$.

Note that $v(\widehat{\theta}^n)_i = 1$ whenever $\widehat{\theta}^n_i > 0$, and $|v(\widehat{\theta}^n)_i| \leq 1$ for all coordinates i. Therefore, letting $b_0 \equiv 480 c\sigma(s_0 \log p)/n$ we have

$$\begin{aligned}
1 - \mathbb{E}\{v(\widehat{\theta}^n)_i|\mathbf{X}\} &\leq 2\mathbb{P}\Big(\widehat{\theta}^n_i \leq 0 \Big| \mathbf{X}\Big) \leq 2\mathbb{P}\Big(\widehat{\theta}^u_i \leq \lambda \Big| \mathbf{X}\Big) \qquad (95)\\
&\leq 2\mathbb{P}\Big(\theta_{0,i} + \frac{1}{\sqrt{n}} Z_i + \frac{1}{\sqrt{n}} \Delta_i \leq \lambda \Big| \mathbf{X}\Big)\\
&\leq 2\mathbb{P}\Big(\frac{1}{\sqrt{n}} Z_i \leq \lambda + b_0 - \theta_{0,i} \Big| \mathbf{X}\Big) + 2\mathbb{P}\Big(\|\Delta\|_\infty > \sqrt{n} b_0\Big)\\
&= 2\Phi\Big((\lambda + b_0 - \theta_{0,i})\sqrt{n/(\sigma^2 \widehat{\Sigma}_{ii})}\Big) + 4p^{1-(c^2/48)}\\
&\leq 2\Phi\Big((\lambda + b_0 - \theta_{0,i})\sqrt{2n/(3\sigma^2)}\Big) + 4p^{1-(c^2/48)} \qquad (96)
\end{aligned}$$

with $\Phi(x)$ the standard normal distribution function. Here, we used the relation $\widehat{\theta}^u = \widehat{\theta} + \lambda v(\widehat{\theta})$ in Equation (95) and Equation (96) holds because $\max_{i\in[p]} \widehat{\Sigma}_{ii} \leq 3/2$ on $\mathcal{B}_n$. We then choose θ_0 so that $\theta_{0,i} \geq b_0 + \lambda + \sqrt{30\sigma^2/n}$, for $i \in [p]$ in the support of θ_0. We therefore obtain

$$\mathbb{E}\{v(\widehat{\theta}^n)_i|\mathbf{X}\} \geq 1 - 2\Phi(-\sqrt{20}) - 4p^{1-(c^2/48)} \geq \frac{2}{3}\,, \tag{97}$$

where in the last step we used the assumption $p \geq 13^{48/(c^2-48)}$. This finishes the proof of Equation (23).

Equation (25) follows readily from Equation (93), substituting $\lambda = c\sigma\sqrt{(\log p)/n}$ and recalling the assumption $n \geq (3c_{**}s_0/c)^2 \log p$.

Finally, combining Equation (25) and Equation (23), we get

$$\|\mathsf{Bias}(\widehat{\theta}^n)\|_\infty \geq \frac{\lambda}{3}\,. \tag{98}$$

Therefore, Equation (26) is derived by substituting $\lambda = c\sigma\sqrt{(\log p)/n}$ and using Corollary 10, Equation (21) with $a = 30$.

Appendix C. Proof of Lemma 14

Let $\mathcal{E}_n = \mathcal{E}_n(\phi_0, s_0, K)$ be the event defined as per Theorem 7.(a). In particular, we take $\phi_0 = \sqrt{C_{\min}}/2$, and $K \geq 1 + 20\kappa^2\sqrt{(\log p)/n}$.[4] Further note that we can assume without loss of generality $n \geq \nu_0\, s_0 \log(p/s_0)$, since $s_0 = o(\sqrt{n}/\log p)$. Fixing $\varepsilon > 0$, we have

$$\begin{aligned}\mathbb{P}\Big(\Big|\frac{\widehat{\sigma}}{\sigma} - 1\Big| \geq \varepsilon\Big) &\leq \sup_{\mathbf{X}\in\mathcal{E}_n} \mathbb{P}\Big(\Big|\frac{\widehat{\sigma}}{\sigma} - 1\Big| \geq \varepsilon \,\Big|\, \mathbf{X}\Big) + \mathbb{P}\big(\mathbf{X} \notin \mathcal{E}_n\big)\\ &\leq \sup_{\mathbf{X}\in\mathcal{E}_n} \mathbb{P}\Big(\Big|\frac{\widehat{\sigma}}{\sigma} - 1\Big| \geq \varepsilon \,\Big|\, \mathbf{X}\Big) + 4\, e^{-c_1 n}\,,\end{aligned}$$

where $c_1 > 0$ is a constant defined as per Theorem 7.(a).

We are therefore left with the task of bounding the first term in the last expression above, uniformly over $\theta_0 \in \mathbb{R}^p$, $\|\theta_0\|_0 \leq s_0$. For $\mathbf{X} \in \mathcal{E}_n$, we apply the result of Sun and Zhang (2012, Theorem 1). More precisely, using the notations of Sun and Zhang (2012), with $\lambda_0 = \widetilde{\lambda}$, $\xi = 3$, $T = \mathrm{supp}(\theta_0)$, $\kappa(\xi, T) \geq \phi_0$, we have $\eta_*(\widetilde{\lambda}, \xi) \leq 4s_0\widetilde{\lambda}^2/\phi_0^2$. Further, let σ^* be the oracle estimator of σ introduced there. If $\|\mathbf{X}^{\mathsf{T}}W/(n\sigma^*)\|_\infty \leq \widetilde{\lambda}/4$, using Equation (13) in Sun and Zhang (2012), we obtain

$$\Big|\frac{\widehat{\sigma}}{\sigma^*} - 1\Big| \leq \frac{2\sqrt{s_0}\widetilde{\lambda}}{\sigma^*\phi_0} \leq \frac{\varepsilon}{2}\,, \tag{99}$$

where the last inequality follows for all n large enough since $s_0 = o(\sqrt{n}/\log p)$.

Hence

$$\sup_{\mathbf{X}\in\mathcal{E}_n} \mathbb{P}\Big(\Big|\frac{\widehat{\sigma}}{\sigma} - 1\Big| \geq \varepsilon \,\Big|\, \mathbf{X}\Big) \leq \sup_{\mathbf{X}\in\mathcal{E}_n} \mathbb{P}\Big(\, \|\mathbf{X}^{\mathsf{T}}W/n\|_\infty > \widetilde{\lambda}/4 \Big|\, \mathbf{X}\Big) + \sup_{\mathbf{X}\in\mathcal{E}_n} \mathbb{P}\Big(\Big|\frac{\sigma^*}{\sigma} - 1\Big| \geq \frac{\varepsilon}{10} \,\Big|\, \mathbf{X}\Big)\,, \tag{100}$$

where we note that the right hand side is independent of θ_0. The first term vanishes as $n \to \infty$ by a standard tail bound on the supremum of p Gaussian random variables. The second term also vanishes because it is controlled by the tail of a chi-squared random variable (see Sun and Zhang, 2012).

Appendix D. Proof of Theorem 20

Let $\mathcal{F}_{p,s_0} \equiv \{x \in \mathbb{R}^p : \|x\|_0 \leq s_0\}$, and fix $\varepsilon \in (0, 1/10)$. By definition,

$$\begin{aligned}\mathrm{FWER}(\widehat{T}^{\mathrm{F}}, n) &= \sup_{\theta_0\in\mathcal{F}_{p,s_0}} \mathbb{P}\left\{\exists i \in [p]\setminus \mathrm{supp}(\theta_0),\ \text{s.t.}\ \frac{\sqrt{n}\,|\widehat{\theta}_i^u - \theta_{0,i}|}{\widehat{\sigma}[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}} \geq \Phi^{-1}\Big(1 - \frac{\alpha}{2p}\Big)\right\}\\ &\leq \sup_{\theta_0\in\mathcal{F}_{p,s_0}} \mathbb{P}\left\{\exists i \in [p]\setminus \mathrm{supp}(\theta_0),\ \text{s.t.}\ \frac{\sqrt{n}\,|\widehat{\theta}_i^u - \theta_{0,i}|}{\sigma[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i}^{1/2}} \geq (1-\varepsilon)\Phi^{-1}\Big(1 - \frac{\alpha}{2p}\Big)\right\}\\ &\quad + \sup_{\theta_0\in\mathcal{F}_{p,s_0}} \mathbb{P}\Big(\Big|\frac{\widehat{\sigma}}{\sigma} - 1\Big| \geq \varepsilon\Big)\,.\end{aligned}$$

4. For instance $K = 1.1$ will work for all n large enough since $(s_0 \log p)^2/n \to 0$, with $s_0 \geq 1$, by assumption.

Since the second term vanishes as $n \to \infty$ by Equation (29). Using Bonferroni inequality, letting $z_\alpha(\varepsilon) \equiv (1-\varepsilon)\Phi^{-1}\big(1-\frac{\alpha}{2p}\big)$, we have

$$\begin{aligned}\limsup_{n\to\infty} \mathrm{FWER}(\widehat{T}^{\mathrm{F}}, n) &\le \limsup_{n\to\infty} \sum_{i=1}^{p} \sup_{\theta_0\in\mathcal{F}_{p,s_0},\theta_{0,i}=0} \mathbb{P}\left\{\frac{\sqrt{n}\,|\widehat{\theta}^u_i - \theta_{0,i}|}{\sigma[M\widehat{\Sigma}M^{\mathsf{T}}]^{1/2}_{i,i}} \ge z_\alpha(\varepsilon)\right\}\\ &= \limsup_{n\to\infty} \sum_{i=1}^{p} \sup_{\theta_0\in\mathcal{F}_{p,s_0},\theta_{0,i}=0} \mathbb{P}\left\{\left|\widetilde{Z}_i + \frac{\Delta_i}{\sigma[M\widehat{\Sigma}M^{\mathsf{T}}]^{1/2}_{ii}}\right| \ge z_\alpha(\varepsilon)\right\},\end{aligned}$$

where, by Theorem 8, $\widetilde{Z}_i \sim \mathsf{N}(0,1)$ and Δ_i is given by Equation (16). We then have

$$\begin{aligned}\limsup_{n\to\infty} \mathrm{FWER}(\widehat{T}^{\mathrm{F}}, n) &\le \limsup_{n\to\infty} \sum_{i=1}^{p} \mathbb{P}\big\{|\widetilde{Z}_i| \ge z_\alpha(\varepsilon) - \varepsilon\big\}\\ &\quad + \limsup_{n\to\infty} \sum_{i=1}^{p} \sup_{\theta_0\in\mathcal{F}_{p,s_0},\theta_{0,i}=0} \mathbb{P}\left\{\|\Delta\|_\infty \ge \frac{\varepsilon\sigma}{2\widehat{\Sigma}^{1/2}_{ii}}\right\}\\ &\le 2p\big(1-\Phi(z_\alpha(\varepsilon)-\varepsilon)\big) + \limsup_{n\to\infty} p \max_{i\in[p]} \mathbb{P}(\widehat{\Sigma}_{ii} \ge 2)\\ &\quad + \limsup_{n\to\infty} \sup_{\theta_0\in\mathcal{F}_{p,s_0},\theta_{0,i}=0} p\,\mathbb{P}\left\{\|\Delta\|_\infty \ge \frac{\varepsilon\sigma}{4}\right\}, \qquad (101)\end{aligned}$$

where in the first inequality, we used $[M\widehat{\Sigma}M^{\mathsf{T}}]_{i,i} \ge 1/(4\widehat{\Sigma}_{ii})$ for all n large enough, by Lemma 12, and since $\mu = a\sqrt{(\log p)/n} \to 0$ as $n, p \to \infty$. Now, the second term in the right hand side of Equation (101) vanishes by Theorem 7.(a), and the last term is zero by Theorem 8, since $s_0 = o(\sqrt{n}/\log p)$. Therefore

$$\limsup_{n\to\infty} \mathrm{FWER}(\widehat{T}^{\mathrm{F}}, n) \le 2p\big(1-\Phi(z_\alpha(\varepsilon)-\varepsilon)\big). \qquad (102)$$

The claim follows by letting $\varepsilon \to 0$.

References

M. Bayati, M. A. Erdogdu, and A. Montanari. Estimating Lasso risk and noise level. In *Advances in Neural Information Processing Systems*, pages 944–952, 2013.

A. Belloni and V. Chernozhukov. Least squares after model selection in high-dimensional sparse models. *Bernoulli*, 19(2):521–547, 2013.

A. Belloni, V. Chernozhukov, and Y. Wei. Honest confidence regions for a regression parameter in logistic regression with a large number of controls. *arXiv Preprint* arXiv:1304.3969, 2013.

A. Belloni, V. Chernozhukov, and C. Hansen. Inference on treatment effects after selection amongst high-dimensional controls. *arXiv Preprint* arXiv:1201.0224, The Review of Economic Studies, 2014.

P. J. Bickel, Y. Ritov, and A. B. Tsybakov. Simultaneous analysis of Lasso and Dantzig selector. *The Annals of Statistics*, 37:1705–1732, 2009.

P. Bühlmann. Statistical significance in high-dimensional linear models. *Bernoulli*, 19(4): 1212–1242, 2013.

P. Bühlmann and S. van de Geer. *Statistics for High-Dimensional Data*. Springer-Verlag, 2011.

P. Bühlmann, M. Kalisch, and L. Meier. High-dimensional statistics with a view toward applications in biology. *Annual Review of Statistics and Its Application*, 1(1):255–278, 2014.

E. Candès and T. Tao. The Dantzig selector: statistical estimation when p is much larger than n. *The Annals of Statistics*, 35:2313–2351, 2007.

E. J. Candès and Y. Plan. Near-ideal model selection by ℓ_1 minimization. *The Annals of Statistics*, 37(5A):2145–2177, 2009.

E. J. Candès and T. Tao. Decoding by linear programming. *IEEE Trans. on Inform. Theory*, 51:4203–4215, 2005.

S. S. Chen and D. L. Donoho. Examples of basis pursuit. In *Proceedings of Wavelet Applications in Signal and Image Processing III*, San Diego, CA, 1995.

R. Dezeure and P. Bühlmann. *Private Communication*, 2013.

L. H. Dicker. Residual variance and the signal-to-noise ratio in high-dimensional linear models. *arXiv Preprint* arXiv:1209.0012, 2012.

D. L. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Transactions on Information Theory*, 47(7):2845–2862, 2001.

B. Efron. *Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing, and Prediction*, volume 1. Cambridge University Press, 2010.

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, 2001.

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.

J. Fan, R. Samworth, and Y. Wu. Ultrahigh dimensional feature selection: beyond the linear model. *The Journal of Machine Learning Research*, 10:2013–2038, 2009.

J. Fan, S. Guo, and N. Hao. Variance estimation using refitted cross-validation in ultrahigh dimensional regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(1):37–65, 2012.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. URL `http://www.jstatsoft.org/v33/i01/`.

E. Greenshtein and Y. Ritov. Persistence in high-dimensional predictor selection and the virtue of over-parametrization. *Bernoulli*, 10:971–988, 2004.

A. Javanmard and A. Montanari. Confidence intervals and hypothesis testing for high-dimensional statistical models. In *Advances in Neural Information Processing Systems*, pages 1187–1195, 2013a.

A. Javanmard and A. Montanari. Hypothesis testing in high-dimensional regression under the gaussian random design model: Asymptotic theory. *arXiv Preprint* arXiv:1301.4240, (To appear in *IEEE Transactions on Information Theory*), 2013b.

A. Javanmard and A. Montanari. Nearly optimal sample size in hypothesis testing for high-dimensional regression. In *51st Annual Allerton Conference*, pages 1427–1434, Monticello, IL, June 2013c.

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

E. L. Lehmann and J. P. Romano. *Testing Statistical Hypotheses*. Springer, 2005.

R. Lockhart, J. Taylor, R. J. Tibshirani, and R. Tibshirani. A significance test for the Lasso. *Annals of Statistics*, 42(2):413–468, 2014.

M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly. Compressed sensing MRI. *IEEE Signal Processing Magazine*, 25:72–82, 2008.

S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the Lasso. *The Annals of Statistics*, 34:1436–1462, 2006.

N. Meinshausen and P. Bühlmann. Stability selection. *J. R. Statist. Soc. B*, 72:417–473, 2010.

N. Meinshausen, L. Meier, and P. Bühlmann. p-values for high-dimensional regression. *Journal of the American Statistical Association*, 104(488):1671–1681, 2009.

J. Minnier, L. Tian, and T. Cai. A perturbation method for inference on regularized regression estimates. *Journal of the American Statistical Association*, 106(496), 2011.

J. Peng, J. Zhu, A. Bergamaschi, W. Han, D.-Y. Noh, J. R. Pollack, and P. Wang. Regularized multivariate regression for identifying master predictors with application to integrative genomics study of breast cancer. *The Annals of Applied Statistics*, 4(1):53–77, 2010.

S. Reid, R. Tibshirani, and J. Friedman. A study of error variance estimation in Lasso regression. *arXiv Preprint* arXiv:1311.5274, 2013.

Z. Ren, T. Sun, C.-H. Zhang, and H. H. Zhou. Asymptotic normality and optimalities in estimation of large gaussian graphical model. *arXiv Preprint* arXiv:1309.6024, 2013.

M. Rudelson and Z. Shuheng. Reconstruction from anisotropic random measurements. *IEEE Transactions on Information Theory*, 59(6):3434–3447, 2013.

N. Städler, P. Bühlmann, and S. van de Geer. ℓ_1-penalization for mixture regression models (with discussion). *Test*, 19(2):209–256, 2010.

T. Sun and C.-H. Zhang. Scaled sparse linear regression. *Biometrika*, 99(4):879–898, 2012.

R. Tibshirani. Regression shrinkage and selection with the Lasso. *J. Royal. Statist. Soc B*, 58:267–288, 1996.

S. van de Geer and P. Bühlmann. On the conditions used to prove oracle results for the Lasso. *Electron. J. Statist.*, 3:1360–1392, 2009.

S. van de Geer, P. Bühlmann, Y. Ritov, and R. Dezeure. On asymptotically optimal confidence regions and tests for high-dimensional models. *arXiv Preprint* arXiv:1303.0518, (To appear in *Annals of Statistics*), 2014.

A. W. Van der Vaart. *Asymptotic Statistics*, volume 3. Cambridge university press, 2000.

R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. In Y.C. Eldar and G. Kutyniok, editors, *Compressed Sensing: Theory and Applications*, pages 210–268. Cambridge University Press, 2012.

M. J. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ_1-constrained quadratic programming. *IEEE Trans. on Inform. Theory*, 55:2183–2202, 2009.

L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Verlag, 2004.

C-H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894–942, 2010.

C.-H. Zhang and S. S. Zhang. Confidence intervals for low-dimensional parameters in high-dimensional linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76:217–242, 2014.

P. Zhao and B. Yu. On model selection consistency of Lasso. *The Journal of Machine Learning Research*, 7:2541–2563, 2006.

Journal of Machine Learning Research 15 (2014) 2911-2947 Submitted 4/13; Revised 4/14; Published 10/14

QUIC: Quadratic Approximation for Sparse Inverse Covariance Estimation

Cho-Jui Hsieh CJHSIEH@CS.UTEXAS.EDU
Mátyás A. Sustik MSUSTIK@GMAIL.COM
Inderjit S. Dhillon INDERJIT@CS.UTEXAS.EDU
Pradeep Ravikumar PRADEEPR@CS.UTEXAS.EDU
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, USA

Editor: Xiaotong Shen

Abstract

The ℓ_1-regularized Gaussian maximum likelihood estimator (MLE) has been shown to have strong statistical guarantees in recovering a sparse inverse covariance matrix, or alternatively the underlying graph structure of a Gaussian Markov Random Field, from very limited samples. We propose a novel algorithm for solving the resulting optimization problem which is a regularized log-determinant program. In contrast to recent state-of-the-art methods that largely use first order gradient information, our algorithm is based on Newton's method and employs a quadratic approximation, but with some modifications that leverage the structure of the sparse Gaussian MLE problem. We show that our method is superlinearly convergent, and present experimental results using synthetic and real-world application data that demonstrate the considerable improvements in performance of our method when compared to previous methods.

Keywords: covariance, graphical model, regularization, optimization, Gaussian Markov random field

1. Introduction

Statistical problems under modern data settings are increasingly high-dimensional, so that the number of parameters is very large, potentially outnumbering even the number of observations. An important class of such problems involves estimating the graph structure of a Gaussian Markov random field (GMRF), with applications ranging from biological inference in gene networks, analysis of fMRI brain connectivity data and analysis of interactions in social networks. Specifically, given n independently drawn samples $\{\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_n}\}$ from a p-variate Gaussian distribution, so that $\mathbf{y_i} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, the task is to estimate its inverse covariance matrix Σ^{-1}, also referred to as the *precision* or *concentration* matrix. The non-zero pattern of this inverse covariance matrix Σ^{-1} can be shown to correspond to the underlying graph structure of the GMRF. An active line of work in high-dimensional settings, where $p \gg n$, is based on imposing constraints on the model space; in the GMRF case a common structured constraint is that of sparsity of the inverse covariance matrix. Accordingly, recent papers by Banerjee et al. (2008); Friedman et al. (2008); Yuan and Lin (2007) have

proposed an estimator that minimizes the Gaussian negative log-likelihood regularized by the ℓ_1 norm of the entries (typically restricted to those on the off-diagonal) of the inverse covariance matrix, which encourages sparsity in its entries. This estimator has been shown to have very strong statistical guarantees even under very high-dimensional settings, including convergence in Frobenius and spectral norms (Rothman et al., 2008; Lam and Fan, 2009; Ravikumar et al., 2011), as well as in recovering the non-zero pattern of the inverse covariance matrix, or alternatively the graph structure of the underlying GMRF (Ravikumar et al., 2011). Moreover, the resulting optimization problem is a log-determinant program, which is convex, and can be solved in polynomial time.

For such large-scale optimization problems arising from high-dimensional statistical estimation however, standard optimization methods typically suffer sub-linear rates of convergence (Agarwal et al., 2010). This would be too expensive for the Gaussian MLE problem, since the number of matrix entries scales quadratically with the number of nodes. Luckily, the log-determinant problem has special structure; the log-determinant function is strongly convex and one can thus obtain linear (i.e., geometric) rates of convergence via the state-of-the-art methods. However, even linear rates in turn become infeasible when the problem size is very large, with the number of nodes in the thousands and the number of matrix entries to be estimated in the millions. Here we ask the question: *can we obtain superlinear rates of convergence for the optimization problem underlying the ℓ_1-regularized Gaussian MLE?*

For superlinear rates, one has to consider second-order methods which at least in part use the Hessian of the objective function. There are however some caveats to the use of such second-order methods in high-dimensional settings. First, a straightforward implementation of each second-order step would be very expensive for high-dimensional problems. Secondly, the log-determinant function in the Gaussian MLE objective acts as a barrier function for the positive definite cone. This barrier property would be lost under quadratic approximations so there is a danger that Newton-like updates will not yield positive-definite matrices, unless one explicitly enforces such a constraint in some manner.

In this paper, we present QUIC (QUadratic approximation of Inverse Covariance matrices), a second-order algorithm, that solves the ℓ_1-regularized Gaussian MLE. We perform Newton steps that use iterative quadratic approximations of the Gaussian negative log-likelihood. The computation of the Newton direction is a *Lasso* problem (Meier et al., 2008; Friedman et al., 2010), which we then solve using coordinate descent. A key facet of our method is that we are able to reduce the computational cost of a coordinate descent update from the naive $O(p^2)$ to $O(p)$ complexity by exploiting the structure present in the problem, and by a careful arrangement and caching of the computations. Furthermore, an Armijo-rule based step size selection rule ensures sufficient descent and positive definiteness of the intermediate iterates. Finally, we use the form of the stationary condition characterizing the optimal solution to *focus* the Newton direction computation on a small subset of *free* variables, but in a manner that preserves the strong convergence guarantees of second-order descent. We note that when the solution has a block-diagonal structure as described in Mazumder and Hastie (2012); Witten et al. (2011), the *fixed/free* set selection in QUIC can automatically identify this block diagonal structure and avoid updates to the off-diagonal block elements. A preliminary version of this paper appeared in Hsieh et al. (2011). In this paper, we provide a more detailed analysis along with proofs of our algorithm, and cover a

more general weighted regularization case of the regularized inverse covariance estimation problem. We show that QUIC can automatically identify the sparsity structure under the block-diagonal case. We also conduct more experiments on both synthetic and real data sets to compare QUIC with other solvers. Our software package QUIC with MATLAB and R interface[1] is public available at `http://www.cs.utexas.edu/~sustik/QUIC/`.

The outline of the paper is as follows. We start with a review of related work and the problem setup in Section 2. In Section 3, we present our algorithm that combines quadratic approximation, Newton's method and coordinate descent. In Section 4, we show superlinear convergence of our method. We summarize the experimental results in Section 5, where we compare the algorithm using both real data and synthetic examples from Li and Toh (2010). We observe that our algorithm performs overwhelmingly better (quadratic instead of linear convergence) than existing solutions described in the literature.

Notation. In this paper, boldfaced lowercase letters denote vectors and uppercase letters denote $p \times p$ real matrices. $\mathcal{S}^p_{++}$ denotes the space of $p \times p$ symmetric positive definite matrices while $X \succ 0$ and $X \succeq 0$ means that X is positive definite and positive semidefinite, respectively. The vectorized listing of the elements of a $p \times p$ matrix X is denoted by $\mathrm{vec}(X) \in \mathbb{R}^{p^2}$ and the Kronecker product of the matrices X and Y is denoted by $X \otimes Y$. For a real-valued function $f(X)$, $\nabla f(X)$ is a $p \times p$ matrix with (i,j) element equal to $\frac{\partial}{\partial X_{ij}} f(X)$ and denoted by $\nabla_{ij} f(X)$, while $\nabla^2 f(X)$ is the $p^2 \times p^2$ Hessian matrix. We will use the ℓ_1 and ℓ_∞ norms defined on the vectorized form of matrix X: $\|X\|_1 := \sum_{i,j} |X_{ij}|$ and $\|X\|_\infty := \max_{i,j} |X_{ij}|$. We also employ elementwise ℓ_1-regularization, $\|X\|_{1,\Lambda} := \sum_{i,j} \lambda_{ij} |X_{ij}|$, where $\Lambda = [\lambda_{ij}]$ with $\lambda_{ij} > 0$ for off-diagonal elements, and $\lambda_{ii} \geq 0$ for diagonal elements.

2. Background and Related Work

Let $\mathbf{y}$ be a p-variate Gaussian random vector, with distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$. Given n independently drawn samples $\{\mathbf{y_1}, \ldots, \mathbf{y_n}\}$ of this random vector, the sample covariance matrix can be written as

$$S = \frac{1}{n-1} \sum_{k=1}^{n} (\mathbf{y_k} - \hat{\boldsymbol{\mu}})(\mathbf{y_k} - \hat{\boldsymbol{\mu}})^T, \text{ where } \hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_{k=1}^{n} \mathbf{y_k}. \tag{1}$$

Given a regularization penalty $\lambda > 0$, the ℓ_1-regularized Gaussian MLE for the inverse covariance matrix can be written as the solution of the following regularized *log-determinant* program:

$$\arg\min_{X \succ 0} \left\{ -\log\det X + \mathrm{tr}(SX) + \lambda \sum_{i,j=1}^{p} |X_{ij}| \right\}. \tag{2}$$

The ℓ_1 regularization promotes sparsity in the inverse covariance matrix, and thus encourages a sparse graphical model structure. We consider a generalized weighted ℓ_1 regularization, where given a symmetric nonnegative weight matrix $\Lambda = [\lambda_{ij}]$, we can assign different nonnegative weights to different entries, obtaining the regularization term $\|X\|_{1,\Lambda} = \sum_{i,j=1}^{p} \lambda_{ij} |X_{ij}|$. In this paper we will focus on solving the following generalized

1. The QUIC R package is also available from CRAN.

sparse inverse covariance estimation problem:

$$X^* = \arg\min_{X \succ 0} \left\{ -\log\det X + \mathrm{tr}(SX) + \|X\|_{1,\Lambda} \right\} = \arg\min_{X \succ 0} f(X), \tag{3}$$

where $X^* = (\Sigma^*)^{-1}$. In order to ensure that problem (3) has a unique minimizer, as we show later, it is sufficient to require that $\lambda_{ij} > 0$ for off-diagonal entries, and $\lambda_{ii} \geq 0$ for diagonal entries. The standard off-diagonal ℓ_1 regularization variant $\lambda \sum_{i \neq j} |X_{ij}|$ is a special case of this weighted regularization function. For further details on the background and utility of ℓ_1 regularization in the context of GMRFs, we refer the reader to Yuan and Lin (2007); Banerjee et al. (2008); Friedman et al. (2008); Duchi et al. (2008); Ravikumar et al. (2011).

Due in part to its importance, there has been an active line of work on efficient optimization methods for solving (2) and (3). Since the regularization term is non-smooth and hard to solve, many methods aim to solve the dual problem of (3):

$$\Sigma^* = \operatorname*{argmax}_{|W_{ij} - S_{ij}| \leq \lambda_{ij}} \log\det W, \tag{4}$$

which has a smooth objective function with bound constraints. Banerjee et al. (2008) propose a block-coordinate descent method to solve the dual problem (4), by updating one row and column of W at a time. They show that the dual of the corresponding row subproblem can be written as a standard Lasso problem, which they then solve by Nesterov's first order method. Friedman et al. (2008) follow the same strategy, but propose to use a coordinate descent method to solve the row subproblems instead; their method is implemented in the widely used R package called GLASSO. In other approaches, the dual problem (4) is treated as a constrained optimization problem, for which Duchi et al. (2008) apply a projected subgradient method called PSM, while Lu (2009) proposes an accelerated gradient descent method called VSM.

Other first-order methods have been pursued to solve the primal optimization problem (2). d'Aspremont et al. (2008) apply Nesterov's first order method to (2) after smoothing the objective function; Scheinberg et al. (2010) apply an augmented Lagrangian method to handle the smooth and nonsmooth parts separately; the resulting algorithm is implemented in the ALM software package. In Scheinberg and Rish (2010), the authors propose to directly solve the primal problem by a greedy coordinate descent method called SINCO. However, each coordinate update of SINCO has a time complexity of $O(p^2)$, which becomes computationally prohibitive when handling large problems. We will show in this paper that after forming the quadratic approximation, each coordinate descent update can be performed in $O(p)$ operations. This trick is one of the key advantages of our proposed method, QUIC.

One common characteristic of the above methods is that they are first-order iterative methods that mainly use gradient information at each step. Such first-order methods have become increasingly popular in recent years for high-dimensional problems in part due to their ease of implementation, and because they require very little computation and memory at each step. The caveat is that they have at most linear rates of convergence (Bertsekas, 1995). To achieve superlinear convergence rates, one has to consider second-order methods, which have only recently attracted some attention for the sparse inverse covariance estimation problem. Li and Toh (2010) handle the non-smoothness of the ℓ_1 regularization in the

objective function by doubling the number of variables, and solving the resulting constrained optimization problem by an inexact interior point method. Schmidt et al. (2009) propose a second order Projected Quasi-Newton method (PQN) that solves the dual problem (4), since the dual objective function is smooth. The key difference of our method when compared to these recent second order solvers is that we directly solve the ℓ_1-regularized primal objective using a second-order method. As we show, this allows us to leverage structure in the problem, and efficiently approximate the generalized Newton direction using coordinate descent. Subsequent to the preliminary version of this paper (Hsieh et al., 2011), Olsen et al. (2012) have proposed generalizations to our framework to allow various inner solvers such as FISTA, conjugate gradient (CG), and LBFGS to be used, in addition to our proposed coordinate descent scheme. Also, Lee et al. (2012) have extended the quadratic approximation algorithm to solve general composite functions and analyze the convergence properties.

3. Quadratic Approximation Method

We first note that the objective $f(X)$ in the non-differentiable optimization problem (3), can be written as the sum of two parts, $f(X) = g(X) + h(X)$, where

$$g(X) = -\log\det X + \mathrm{tr}(SX) \text{ and } h(X) = \|X\|_{1,\Lambda}. \tag{5}$$

The first component $g(X)$ is twice differentiable, and strictly convex. The second part, $h(X)$, is convex but non-differentiable. Following the approach of Tseng and Yun (2007) and Yun and Toh (2011), we build a quadratic approximation around any iterate X_t for this composite function by first considering the second-order Taylor expansion of the smooth component $g(X)$:

$$\bar{g}_{X_t}(\Delta) \equiv g(X_t) + \mathrm{vec}(\nabla g(X_t))^T \mathrm{vec}(\Delta) + \frac{1}{2}\mathrm{vec}(\Delta)^T \nabla^2 g(X_t)\,\mathrm{vec}(\Delta). \tag{6}$$

The Newton direction D_t^* for the entire objective $f(X)$ can then be written as the solution of the regularized quadratic program:

$$D_t^* = \arg\min_{\Delta} \big\{\bar{g}_{X_t}(\Delta) + h(X_t + \Delta)\big\}. \tag{7}$$

We use this Newton direction to compute our iterative estimates $\{X_t\}$ for the solution of the optimization problem (3). This variant of Newton method for such composite objectives is also referred to as a "proximal Newton-type method," and was empirically studied in Schmidt (2010). Tseng and Yun (2007) considered the more general case where the Hessian $\nabla^2 g(X_t)$ is replaced by any positive definite matrix. See also the recent paper by Lee et al. (2012), where convergence properties of such general proximal Newton-type methods are discussed. We note that a key caveat to applying such second-order methods in high-dimensional settings is that the computation of the Newton direction appears to have a large time complexity, which is one reason why first-order methods have been so popular for solving the high-dimensional ℓ_1-regularized Gaussian MLE.

Let us delve into the Newton direction computation in (7). Note that it can be rewritten as a standard Lasso regression problem (Tibshirani, 1996):

$$\arg\min_{\Delta} \frac{1}{2}\|H^{\frac{1}{2}}\,\mathrm{vec}(\Delta) + H^{-\frac{1}{2}}\boldsymbol{b}\|_2^2 + \|X_t + \Delta\|_{1,\Lambda}, \tag{8}$$

where $H = \nabla^2 g(X_t)$ and $\boldsymbol{b} = \text{vec}(\nabla g(X_t))$. Many efficient optimization methods exist that solve Lasso regression problems, such as the coordinate descent method (Friedman et al., 2007), the gradient projection method (Polyak, 1969), and iterative shrinking methods (Daubechies et al., 2004; Beck and Teboulle, 2009). When applied to the Lasso problem of (7), most of these optimization methods would require the computation of the gradient of $\bar{g}_{X_t}(\Delta)$:

$$\nabla \bar{g}_{X_t}(\Delta) = H \,\text{vec}(\Delta) + \boldsymbol{b}. \tag{9}$$

The straightforward approach for computing (9) for a general $p^2 \times p^2$ Hessian matrix H would take $O(p^4)$ time, making it impractical for large problems. Fortunately, for the sparse inverse covariance problem (3), the Hessian matrix H has the following special form (see for instance Boyd and Vandenberghe, 2009, Chapter A.4.3):

$$H = \nabla^2 g(X_t) = X_t^{-1} \otimes X_t^{-1},$$

where $\otimes$ denotes the Kronecker product. In Section 3.1, we show how to exploit this special form of the Hessian matrix to perform one coordinate descent step that updates one element of Δ in $O(p)$ time. Hence a full sweep of coordinate descent steps over all the variables requires $O(p^3)$ time. This key observation is one of the reasons that makes our Newton-like method viable for solving the inverse covariance estimation problem.

There exist other functions which allow efficient Hessian times vector multiplication. As an example, we consider the case of ℓ_1-regularized logistic regression. Suppose we are given n samples with feature vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^p$ and labels $y_1, \ldots, y_n$, and we solve the following ℓ_1-regularized logistic regression problem to compute the model parameter $\boldsymbol{w}$:

$$\arg\min_{\boldsymbol{w} \in \mathbb{R}^p} \sum_{i=1}^{n} \log(1 + e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i}) + \lambda \|\boldsymbol{w}\|_1.$$

Following our earlier approach, we can decompose this objective function into smooth and non-smooth parts, $g(\boldsymbol{w}) + h(\boldsymbol{w})$, where

$$g(\boldsymbol{w}) = \sum_{i=1}^{n} \log(1 + e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i}) \text{ and } h(\boldsymbol{w}) = \lambda \|\boldsymbol{w}\|_1.$$

In order to apply coordinate descent to solve the quadratic approximation, we have to compute the gradient as in (9). The Hessian matrix $\nabla^2 g(\boldsymbol{w})$ is a $p \times p$ matrix, so direct computation of this gradient costs $O(p^2)$ flops. However, the Hessian matrix for logistic regression has the following simple form

$$H = \nabla^2 g(\boldsymbol{w}) = XDX^T,$$

where D is a diagonal matrix with $D_{ii} = \frac{e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i}}{(1+e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i})^2}$ and $X = [\boldsymbol{x}_1,\ \boldsymbol{x}_2,\ \ldots,\ \boldsymbol{x}_n]$. Therefore we can write

$$\nabla g(\boldsymbol{w} + \Delta) = (\nabla^2 g(\boldsymbol{w}))\,\text{vec}(\Delta) + \boldsymbol{b} = XD(X^T \,\text{vec}(\Delta)) + \boldsymbol{b}. \tag{10}$$

The time complexity to compute (10) is only proportional to the number of nonzero elements in the data matrix X, which can be much smaller than $O(p^2)$ for high-dimensional sparse

data sets. Therefore similar quadratic approximation approaches are also efficient for solving the ℓ_1-regularized logistic regression problem as shown by Friedman et al. (2010); Yuan et al. (2012).

In the following three subsections, we detail three innovations which make our quadratic approximation algorithm feasible for solving (3). In Section 3.1, we show how to compute the Newton direction using an efficient coordinate descent method that exploits the structure of Hessian matrix, so that we reduce the time complexity of each coordinate descent update step from $O(p^2)$ to $O(p)$. In Section 3.2, we employ an Armijo-rule based step size selection to ensure sufficient descent *and* positive-definiteness of the next iterate. Finally, in Section 3.3 we use the form of the stationary condition characterizing the optimal solution, to *focus* the Newton direction computation to a small subset of *free* variables, in a manner that preserves the strong convergence guarantees of second-order descent. A high level overview of our method is presented in Algorithm 1. Note that the initial point X_0 has to be a feasible solution, thus $X_0 \succ 0$, and the positive definiteness of all the following iterates X_t will be guaranteed by the step size selection procedure (step 6 in Algorithm 1).

Algorithm 1: QUadratic approximation for sparse Inverse Covariance estimation (QUIC overview)

Input : Empirical covariance matrix S (positive semi-definite, $p \times p$), regularization parameter matrix Λ, initial iterate $X_0 \succ 0$.

Output: Sequence $\{X_t\}$ that converges to $\arg\min_{X \succ 0} f(X)$, where $f(X) = g(X) + h(X)$, where $g(X) = -\log\det X + \mathrm{tr}(SX), h(X) = \|X\|_{1,\Lambda}$.

1 **for** $t = 0, 1, \ldots$ **do**
2 Compute $W_t = X_t^{-1}$.
3 Form the second order approximation $\bar{f}_{X_t}(\Delta) := \bar{g}_{X_t}(\Delta) + h(X_t + \Delta)$ to $f(X_t + \Delta)$.
4 Partition the variables into free and fixed sets based on the gradient, see Section 3.3.
5 Use coordinate descent to find the Newton direction $D_t^* = \arg\min_\Delta \bar{f}_{X_t}(X_t + \Delta)$ over the set of free variables, see (13) and (16) in Section 3.1. (A *Lasso* problem.)
6 Use an *Armijo*-rule based step-size selection to get α such that $X_{t+1} = X_t + \alpha D_t^*$ is positive definite and there is sufficient decrease in the objective function, see (21) in Section 3.2.
7 **end**

3.1 Computing the Newton Direction

In order to compute the Newton direction, we have to solve the Lasso problem (7). The gradient and Hessian for $g(X) = -\log\det X + \mathrm{tr}(SX)$ are (see, for instance, Boyd and Vandenberghe, 2009, Chapter A.4.3)

$$\nabla g(X) = S - X^{-1} \text{ and } \nabla^2 g(X) = X^{-1} \otimes X^{-1}. \tag{11}$$

In order to formulate our problem accordingly, we can verify that for a symmetric matrix Δ we have $\mathrm{tr}(X_t^{-1}\Delta X_t^{-1}\Delta) = \mathrm{vec}(\Delta)^T(X_t^{-1} \otimes X_t^{-1})\,\mathrm{vec}(\Delta)$, so that $\bar{g}_{X_t}(\Delta)$ in (7) can be

rewritten as

$$\bar{g}_{X_t}(\Delta) = -\log\det X_t + \mathrm{tr}(SX_t) + \mathrm{tr}((S - W_t)^T\Delta) + \frac{1}{2}\mathrm{tr}(W_t\Delta W_t\Delta), \tag{12}$$

where $W_t = X_t^{-1}$.

In Friedman et al. (2007), Wu and Lange (2008), the authors show that coordinate descent methods are very efficient for solving Lasso type problems. An obvious way to update each element of Δ in (7) requires $O(p^2)$ floating point operations since $W_t \otimes W_t$ is a $p^2 \times p^2$ matrix, thus yielding an $O(p^4)$ procedure for computing the Newton direction. As we show below, our implementation reduces the cost of updating one variable to $O(p)$ by exploiting the structure of the second order term $\mathrm{tr}(W_t\Delta W_t\Delta)$.

For notational simplicity, we will omit the iteration index t in the derivations below where we only discuss a single Newton iteration; this applies to the rest of the this section and Section 3.2 as well. (Hence, the notation for $\bar{g}_{X_t}$ is also simplified to $\bar{g}$.) Furthermore, we omit the use of a separate index for the coordinate descent updates. Thus, we simply use D to denote the current iterate approximating the Newton direction and use D' for the updated direction. Consider the coordinate descent update for the variable X_{ij}, with $i < j$ that preserves symmetry: $D' = D + \mu(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T)$. The solution of the one-variable problem corresponding to (7) is:

$$\arg\min_{\mu} \quad \bar{g}(D + \mu(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T)) + 2\lambda_{ij}|X_{ij} + D_{ij} + \mu|. \tag{13}$$

We expand the terms appearing in the definition of $\bar{g}$ after substituting $D' = D + \mu(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T)$ for Δ in (12) and omit the terms not dependent on μ. The contribution of $\mathrm{tr}(SD') - \mathrm{tr}(WD')$ yields $2\mu(S_{ij} - W_{ij})$, while the regularization term contributes $2\lambda_{ij}|X_{ij} + D_{ij} + \mu|$, as seen from (13). The quadratic term can be rewritten (using the fact that $\mathrm{tr}(AB) = \mathrm{tr}(BA)$ and the symmetry of D and W) to yield:

$$\mathrm{tr}(WD'WD') = \mathrm{tr}(WDWD) \quad + \quad 4\mu\mathbf{w}_i^T D\mathbf{w}_j + 2\mu^2(W_{ij}^2 + W_{ii}W_{jj}), \tag{14}$$

where $\mathbf{w}_i$ refers to the i-th column of W. In order to compute the single variable update we seek the minimum of the following quadratic function of μ:

$$\frac{1}{2}(W_{ij}^2 + W_{ii}W_{jj})\mu^2 + (S_{ij} - W_{ij} + \mathbf{w}_i^T D\mathbf{w}_j)\mu + \lambda_{ij}|X_{ij} + D_{ij} + \mu|. \tag{15}$$

Letting $a = W_{ij}^2 + W_{ii}W_{jj}$, $b = S_{ij} - W_{ij} + \mathbf{w}_i^T D\mathbf{w}_j$, and $c = X_{ij} + D_{ij}$ the minimum is achieved for:

$$\mu = -c + \mathcal{S}(c - b/a, \lambda_{ij}/a), \tag{16}$$

where

$$\mathcal{S}(z, r) = \mathrm{sign}(z)\max\{|z| - r, 0\} \tag{17}$$

is the soft-thresholding function. Similarly, when $i = j$, for $D' = D + \mu\mathbf{e}_i\mathbf{e}_i^T$, we get

$$\mathrm{tr}(WD'WD') = \mathrm{tr}(WDWD) \quad + \quad 2\mu\mathbf{w}_i^T D\mathbf{w}_i + \mu^2(W_{ii}^2). \tag{18}$$

Therefore the update rule for D_{ii} can be computed by (16) with $a = W_{ii}^2, b = S_{ii} - W_{ii} + \mathbf{w}_i^T D \mathbf{w}_i$, and $c = X_{ii} + D_{ii}$.

Since a and c are easy to compute, the main computational cost arises while evaluating $\mathbf{w}_i^T D \mathbf{w}_j$, the third term contributing to coefficient b above. Direct computation requires $O(p^2)$ time. Instead, we maintain a $p \times p$ matrix $U = DW$, and then compute $\mathbf{w}_i^T D \mathbf{w}_j$ by $\mathbf{w}_i^T \mathbf{u}_j$ using $O(p)$ flops, where $\mathbf{u}_j$ is the j-th column of matrix U. In order to maintain the matrix U, we also need to update $2p$ elements, namely two coordinates of each $\mathbf{u}_k$ when D_{ij} is modified. We can compactly write the row updates of U as follows: $\mathbf{u}_{i\cdot} \leftarrow \mathbf{u}_{i\cdot} + \mu \mathbf{w}_{j\cdot}$ and $\mathbf{u}_{j\cdot} \leftarrow \mathbf{u}_{j\cdot} + \mu \mathbf{w}_{i\cdot}$, where $\mathbf{u}_{i\cdot}$ refers to the i-th *row* vector of U.

3.1.1 Update Rule when X is Diagonal

The calculation of the Newton direction can be simplified if X is also a diagonal matrix. For example, this occurs in the first Newton iteration when we initialize QUIC using the identity (or diagonal) matrix. When X is diagonal, the Hessian $\nabla^2 g(X) = X^{-1} \otimes X^{-1}$ is also a diagonal matrix, which indicates that all one variable sub-problems are independent of each other. Therefore, we only need to update each variable once to reach the optimum of (7). In particular, by examining (16), the optimal solution D_{ij}^* is

$$D_{ij}^* = \begin{cases} \mathcal{S}\left(-\frac{S_{ij}}{W_{ii}W_{jj}}, \frac{\lambda_{ij}}{W_{ii}W_{jj}}\right) & \text{if } i \neq j, \\ -X_{ii} + \mathcal{S}\left(X_{ii} - \frac{S_{ii}-W_{ii}}{W_{ii}^2}, \frac{\lambda_{ii}}{W_{ii}^2}\right) & \text{if } i = j, \end{cases} \tag{19}$$

where, as a reminder, $W_{ii} = 1/X_{ii}$. Thus, in this case, the closed form solution for each variable can be computed in $O(1)$ time, so the time complexity for the first Newton direction is further reduced from $O(p^3)$ to $O(p^2)$.

3.1.2 Updating Only a Subset of Variables

In our QUIC algorithm we compute the Newton direction using only a subset of the variables we call the *free* set. We identify these variables in each Newton iteration based on the value of the gradient (we will discuss the details of the selection in Section 3.3). In the following, we define the Newton direction restricted to a subset J of the variables.

Definition 1 *Let J denote a (symmetric) subset of variables. The Newton direction restricted to J is defined as:*

$$D_J^*(X) \equiv \arg \min_{\substack{D: D_{ij}=0 \\ \forall (i,j) \notin J}} \operatorname{tr}(\nabla g(X)^T D) + \frac{1}{2} \operatorname{vec}(D)^T \nabla^2 g(X) \operatorname{vec}(D) + \|X + D\|_{1,\Lambda}. \tag{20}$$

The cost to compute the Newton direction is thus substantially reduced when the free set J is small, which as we will show in Section 3.3, occurs when the optimal solution of the ℓ_1-regularized Gaussian MLE is sparse.

3.2 Computing the Step Size

Following the computation of the Newton direction $D^* = D_J^*(X)$ (restricted to the subset of variables J), we need to find a step size $\alpha \in (0, 1]$ that ensures positive definiteness of the next iterate $X + \alpha D^*$ and leads to a sufficient decrease of the objective function.

We adopt Armijo's rule (Bertsekas, 1995; Tseng and Yun, 2007) and try step-sizes $\alpha \in \{\beta^0, \beta^1, \beta^2, \dots\}$ with a constant decrease rate $0 < \beta < 1$ (typically $\beta = 0.5$), until we find the smallest $k \in \mathbb{N}$ with $\alpha = \beta^k$ such that $X + \alpha D^*$ is (a) positive-definite, and (b) satisfies the following sufficient decrease condition:

$$f(X + \alpha D^*) \leq f(X) + \alpha\sigma\delta, \quad \delta = \mathrm{tr}(\nabla g(X)^T D^*) + \|X + D^*\|_{1,\Lambda} - \|X\|_{1,\Lambda}, \tag{21}$$

where $0 < \sigma < 0.5$. Notice that Condition (21) is a generalized version of Armijo line search rule for ℓ_1-regularized problems (see (Tseng and Yun, 2007; Yun and Toh, 2011) for the detail). We can verify positive definiteness while we compute the Cholesky factorization (costs $O(p^3)$ flops) needed for the objective function evaluation that requires the computation of $\log\det(X + \alpha D^*)$. The Cholesky factorization dominates the computational cost in the step-size computations. We use the standard convention in convex analysis that $f(X) = +\infty$ when X is not in the effective domain of f, i.e., X is not positive definite. Using this convention, (21) enforces positive definiteness of $X + \alpha D^*$. Condition (21) has been proposed in Tseng and Yun (2007); Yun and Toh (2011) to ensure that the objective function value not only decreases but decreases by a certain amount $\alpha\sigma\delta$, where δ measures the closeness of the current solution to the global optimal. Our convergence proofs presented in Section 4 rely on this sufficient decrease condition.

In the rest of this section we present several lemmas about the step size computation. The reader mostly interested in the algorithm description may skip forward to Section 3.3 and revisit the details afterwards.

We start out by proving three important properties that we call (P1–P3) regarding the line search procedure governed by (21):

P1. The condition (21) is satisfied for some (sufficiently small) α, establishing that the algorithm does not enter into an infinite line search step. We note that in Proposition 3 below we show that the line search condition (21) can be satisfied for any symmetric matrix D (even one which is not the Newton direction).

P2. For the Newton direction D^*, the quantity δ in (21) is negative, which ensures that the objective function decreases. Moreover, to guarantee that X_t converges to the global optimum, $|\delta|$ should be large enough when the current iterate X_t is far from the optimal solution. In Proposition 4 we will prove the stronger condition that $\delta \leq -(1/M^2)\|D^*\|_F^2$ for some constant M. $\|D^*\|_F^2$ can be viewed as a measure of the distance from optimality of the current iterate X_t, and this bound ensures that the objective function decrease is proportional to $\|D^*\|_F^2$.

P3. When X is close enough to the global optimum, the step size $\alpha = 1$ will satisfy the line search condition (21). We will show this property in Proposition 5. Moreover, combined with the global convergence of QUIC proved in Theorem 12, this property suggests that after a finite number of iterations α will always be 1; this also implies that eventually only one Cholesky factorization is needed per iteration (to evaluate $\log\det(X + \alpha D^*)$ for computing $f(X + \alpha D)$).

3.2.1 Detailed Proofs for P1-3

We first show the following useful property. For any matrices X, D, real number $0 \leq \alpha \leq 1$ and $\Lambda \geq 0$ that generates the norm $\|\cdot\|_{1,\Lambda}$, we have

$$\|X + \alpha D\|_{1,\Lambda} = \|\alpha(X + D) + (1 - \alpha)X\|_{1,\Lambda} \leq \alpha\|X + D\|_{1,\Lambda} + (1 - \alpha)\|X\|_{1,\Lambda}. \tag{22}$$

The above inequality can be proved by the convexity of $\|\cdot\|_{1,\Lambda}$, and will be used repeatedly in this paper. Next we show an important property that all the iterates X_t will have eigenvalues bounded away from zero. Since the updates in our algorithm satisfy the line search condition (21), and δ is always a negative number (see Proposition 4), the function value is always decreasing. It also follows that all the iterates $\{X_t\}_{t=0,1,\ldots}$ belong to the level set U defined by:

$$U = \{X \mid f(X) \leq f(X_0) \text{ and } X \in S^p_{++}\}. \tag{23}$$

Lemma 2 *The level set U defined in (23) is contained in the set $\{X \mid mI \preceq X \preceq MI\}$ for some constants $m, M > 0$, if we assume that the off-diagonal elements of Λ and the diagonal elements of S are positive.*

Proof We begin the proof by showing that the largest eigenvalue of any $X \in U$ is bounded by M, a constant that depends only on Λ, $f(X_0)$ and the matrix S. We note that $S \succeq 0$ and $X \succ 0$ implies $\operatorname{tr}(SX) \geq 0$ and therefore:

$$f(X_0) \geq f(X) \geq -\log\det X + \|X\|_{1,\Lambda}. \tag{24}$$

Since $\|X\|_2$ is the largest eigenvalue of the $p \times p$ matrix X, we have $\log\det X \leq p\log(\|X\|_2)$. Combine with (24) and the fact that the off-diagonal elements of Λ are no smaller than some $\lambda > 0$:

$$\lambda \sum_{i \neq j} |X_{ij}| < \|X\|_{1,\Lambda} \leq f(X_0) + p\log(\|X\|_2). \tag{25}$$

Similarly, $\|X\|_{1,\Lambda} \geq 0$ implies that:

$$\operatorname{tr}(SX) < f(X_0) + p\log(\|X\|_2). \tag{26}$$

Next, we introduce $\gamma = \min_i S_{ii}$ and $\beta = \max_{i \neq j} |S_{ij}|$ and split $\operatorname{tr}(SX)$ into diagonal and off-diagonal terms in order to bound it:

$$\operatorname{tr}(SX) = \sum_i S_{ii}X_{ii} + \sum_{i \neq j} S_{ij}X_{ij} \geq \gamma \operatorname{tr}(X) - \beta \sum_{i \neq j} |X_{ij}|.$$

Since $\|X\|_2 \leq \operatorname{tr}(X)$,

$$\gamma\|X\|_2 \leq \gamma \operatorname{tr}(X) \leq \operatorname{tr}(SX) + \beta \sum_{i \neq j} |X_{ij}|.$$

Combine with (25) and (26) to get:

$$\gamma\|X\|_2 \leq (1 + \beta/\lambda)(f(X_0) + p\log(\|X\|_2)). \tag{27}$$

The left hand side of inequality (27), as a function of $\|X\|_2$, grows much faster than the right hand side (note $\gamma > 0$), and therefore $\|X\|_2$ can be upper bounded by M, where M depends on the values of $f(X_0)$, S and Λ.

In order to prove the lower bound, we consider the smallest eigenvalue of X denoted by a and use the upper bound on the other eigenvalues to get:

$$f(X_0) > f(X) > -\log\det X \geq -\log a - (p-1)\log M, \tag{28}$$

which shows that $m = e^{-f(X_0)}M^{-(p-1)}$ is a lower bound for a. ■

We note that the conclusion of the lemma also holds if the conditions on Λ and S are replaced by only the requirement that the diagonal elements of Λ are positive, see Banerjee et al. (2008). We emphasize that Lemma 2 allows the extension of the convergence results to the practically important case when the regularization does not penalize the diagonal, i.e., $\Lambda_{ii} = 0\ \forall i$. In subsequent arguments we will continue to refer to the minimum and maximum eigenvalues m and M established in Lemma 2.

Proposition 3 (corresponds to Property P1) *For any $X \succ 0$ and symmetric D, there exists an $\bar{\alpha} > 0$ such that for all $\alpha < \bar{\alpha}$, the matrix $X + \alpha D$ satisfies the line search condition* (21)*.*

Proof When $\alpha < \sigma_n(X)/\|D\|_2$ (where $\sigma_n(X)$ stands for the smallest eigenvalue of X and $\|D\|_2$ is the induced 2-norm of D, i.e., the largest eigenvalue in magnitude of D), we have $\|\alpha D\|_2 < \sigma_n(X)$, which implies that $X + \alpha D \succ 0$. So we can write:

$$\begin{aligned} f(X+\alpha D) - f(X) &= g(X+\alpha D) - g(X) + \|X+\alpha D\|_{1,\Lambda} - \|X\|_{1,\Lambda} \\ &\leq g(X+\alpha D) - g(X) + \alpha(\|X+D\|_{1,\Lambda} - \|X\|_{1,\Lambda}), \quad \text{by (22)} \\ &= \alpha\,\mathrm{tr}((\nabla g(X))^T D) + O(\alpha^2) + \alpha(\|X+D\|_{1,\Lambda} - \|X\|_{1,\Lambda}) \\ &= \alpha\delta + O(\alpha^2). \end{aligned}$$

Therefore for any fixed $0 < \sigma < 1$ and sufficiently small α, the line search condition (21) must hold. ■

Proposition 4 (corresponds to Property P2) *$\delta = \delta_J(X)$ as defined in the line search condition* (21) *satisfies*

$$\delta \leq -(1/\|X\|_2^2)\|D^*\|_F^2 \leq -(1/M^2)\|D^*\|_F^2, \tag{29}$$

where M is as in Lemma 2.

Proof We first show that $\delta = \delta_J(X)$ in the line search condition (21) satisfies

$$\delta = \mathrm{tr}((\nabla g(X))^T D^*) + \|X + D^*\|_{1,\Lambda} - \|X\|_{1,\Lambda} \leq -\,\mathrm{vec}(D^*)^T \nabla^2 g(X)\,\mathrm{vec}(D^*), \tag{30}$$

where $D^* = D_J^*(X)$ is the minimizer of the ℓ_1-regularized quadratic approximation defined in (20).

According to the definition of $D^* \equiv D_J^*(X)$ in (20), for all $0 < \alpha < 1$ we have:

$$\begin{aligned}\operatorname{tr}(\nabla g(X)^T D^*) + \frac{1}{2}\operatorname{vec}(D^*)^T \nabla^2 g(X) \operatorname{vec}(D^*) + \|X + D^*\|_{1,\Lambda} \leq \\ \operatorname{tr}(\nabla g(X)^T \alpha D^*) + \frac{1}{2}\operatorname{vec}(\alpha D^*)^T \nabla^2 g(X) \operatorname{vec}(\alpha D^*) + \|X + \alpha D^*\|_{1,\Lambda}. \end{aligned} \tag{31}$$

We combine (31) and 22 to yield:

$$\begin{aligned}\operatorname{tr}(\nabla g(X)^T D^*) + \frac{1}{2}\operatorname{vec}(D^*)^T \nabla^2 g(X) \operatorname{vec}(D^*) + \|X + D^*\|_{1,\Lambda} \leq \\ \alpha \operatorname{tr}(\nabla g(X)^T D^*) + \frac{1}{2}\alpha^2 \operatorname{vec}(D^*)^T \nabla^2 g(X) \operatorname{vec}(D^*) + \alpha\|X + D^*\|_{1,\Lambda} + (1-\alpha)\|X\|_{1,\Lambda}.\end{aligned}$$

Therefore

$$(1-\alpha)[\operatorname{tr}(\nabla g(X)^T D^*) + \|X + D^*\|_{1,\Lambda} - \|X\|_{1,\Lambda}] + \frac{1}{2}(1-\alpha^2)\operatorname{vec}(D^*)^T \nabla^2 g(X) \operatorname{vec}(D^*) \leq 0.$$

Divide both sides by $1 - \alpha > 0$ to get:

$$\operatorname{tr}(\nabla g(X)^T D^*) + \|X + D^*\|_{1,\Lambda} - \|X\|_{1,\Lambda} + \frac{1}{2}(1+\alpha)\operatorname{vec}(D^*)^T \nabla^2 g(X) \operatorname{vec}(D^*) \leq 0.$$

By taking the limit as $\alpha \uparrow 1$, we get:

$$\operatorname{tr}(\nabla g(X)^T D^*) + \|X + D^*\|_{1,\Lambda} - \|X\|_{1,\Lambda} \leq -\operatorname{vec}(D^*)^T \nabla^2 g(X) \operatorname{vec}(D^*),$$

which proves (30).

Since $\nabla^2 g(X) = X^{-1} \otimes X^{-1}$ is positive definite, (30) ensures that $\delta < 0$ for all $X \succ 0$. Since the updates in our algorithm satisfy the line search condition (21), we have established that the function value is decreasing. It also follows that all the iterates $\{X_t\}_{t=0,1,\ldots}$ belong to the level set U defined by (23). Since $\nabla^2 g(X) = X^{-1} \otimes X^{-1}$, the smallest eigenvalue of $\nabla^2 g(X)$ is $1/\|X\|_2^2$, and we combine with Lemma 2 to get (29). ■

The eigenvalues of any iterate X are bounded by Lemma 2, and therefore $\nabla^2 g(X) = X^{-1} \otimes X^{-1}$ is Lipschitz continuous. Next, we prove that $\alpha = 1$ satisfies the line search condition in a neighborhood of the global optimum X^*.

Proposition 5 (corresponds to Property P3) *Assume that $\nabla^2 g$ is Lipschitz continuous, i.e., $\exists L > 0$ such that $\forall t > 0$ and any symmetric matrix D,*

$$\|\nabla^2 g(X + tD) - \nabla^2 g(X)\|_F \leq L\|tD\|_F = tL\|D\|_F. \tag{32}$$

Then, if X is close enough to X^*, the line search condition (21) *will be satisfied with step size $\alpha = 1$.*

Proof We need to derive a bound for the decrease in the objective function value. We define $\tilde{g}(t) = g(X + tD)$, which yields $\tilde{g}''(t) = \operatorname{vec}(D)^T \nabla^2 g(X + tD) \operatorname{vec}(D)$. First, we bound

$|\tilde{g}''(t) - \tilde{g}''(0)|$:

$$\begin{aligned}
|\tilde{g}''(t) - \tilde{g}''(0)| &= |\operatorname{vec}(D)^T(\nabla^2 g(X+tD) - \nabla^2 g(X))\operatorname{vec}(D)| \\
&\leq \|\operatorname{vec}(D)^T(\nabla^2 g(X+tD) - \nabla^2 g(X))\|_2 \|\operatorname{vec}(D)\|_2 \text{ (by Cauchy-Schwartz)} \\
&\leq \|\operatorname{vec}(D)\|_2^2 \|\nabla^2 g(X+tD) - \nabla^2 g(X)\|_2 \qquad \text{(by definition of } \|\cdot\|_2 \text{ norm)} \\
&\leq \|D\|_F^2 \|\nabla^2 g(X+tD) - \nabla^2 g(X)\|_F \quad \text{(since } \|\cdot\|_2 \leq \|\cdot\|_F \text{ for any matrix)} \\
&\leq \|D\|_F^2 tL\|D\|_F \quad \text{by (32)} \\
&= tL\|D\|_F^3.
\end{aligned}$$

Therefore, an upper bound for $\tilde{g}''(t)$:

$$\tilde{g}''(t) \leq \tilde{g}''(0) + tL\|D\|_F^3 = \operatorname{vec}(D)^T \nabla^2 g(X) \operatorname{vec}(D) + tL\|D\|_F^3.$$

Integrate both sides to get

$$\begin{aligned}
\tilde{g}'(t) &\leq \tilde{g}'(0) + t\operatorname{vec}(D)^T \nabla^2 g(X) \operatorname{vec}(D) + \frac{1}{2}t^2 L\|D\|_F^3 \\
&= \operatorname{tr}((\nabla g(X))^T D) + t\operatorname{vec}(D)^T \nabla^2 g(X) \operatorname{vec}(D) + \frac{1}{2}t^2 L\|D\|_F^3.
\end{aligned}$$

Integrate both sides again:

$$\tilde{g}(t) \leq \tilde{g}(0) + t\operatorname{tr}((\nabla g(X))^T D) + \frac{1}{2}t^2 \operatorname{vec}(D)^T \nabla^2 g(X) \operatorname{vec}(D) + \frac{1}{6}t^3 L\|D\|_F^3.$$

Taking $t = 1$ we have

$$\begin{aligned}
g(X+D) &\leq g(X) + \operatorname{tr}(\nabla g(X)^T D) + \frac{1}{2}\operatorname{vec}(D)^T \nabla^2 g(X) \operatorname{vec}(D) + \frac{1}{6}L\|D\|_F^3 \\
f(X+D) &\leq g(X) + \|X\|_{1,\Lambda} + (\operatorname{tr}(\nabla g(X)^T D) + \|X+D\|_{1,\Lambda} - \|X\|_{1,\Lambda}) \\
&\quad + \frac{1}{2}\operatorname{vec}(D)^T \nabla^2 g(X) \operatorname{vec}(D) + \frac{1}{6}L\|D\|_F^3 \\
&\leq f(X) + \delta + \frac{1}{2}\operatorname{vec}(D)^T \nabla^2 g(X) \operatorname{vec}(D) + \frac{1}{6}L\|D\|_F^3 \\
&\leq f(X) + \frac{\delta}{2} + \frac{1}{6}L\|D\|_F^3 \text{ by (30)} \\
&\leq f(X) + (\frac{1}{2} - \frac{1}{6}LM^2\|D\|_F)\delta \text{ (by Proposition 4)} \\
&\leq f(X) + \sigma\delta \text{ (assuming } D \text{ is close to 0).}
\end{aligned}$$

The last inequality holds if $1/2 - LM^2\|D\|_F/6 > \sigma$ which is guaranteed if X is close enough to X^* and consequently D is close to 0 and $\sigma < 0.5$. (Note $\delta < 0$ as well.) In this case the line search condition (21) holds with $\alpha = 1$. ■

3.3 Identifying Which Variables to Update

In this section, we use the stationary condition of the Gaussian MLE problem to select a subset of variables to update in any Newton direction computation. Specifically, we partition the variables into *free* and *fixed* sets based on the value of the gradient at the start of the outer loop that computes the Newton direction. We define the *free* set S_{free} and *fixed* set S_{fixed} as:

$$\begin{aligned} &X_{ij} \in S_{fixed} \text{ if } |\nabla_{ij} g(X)| \leq \lambda_{ij}, \text{ and } X_{ij} = 0, \\ &X_{ij} \in S_{free} \text{ otherwise.} \end{aligned} \tag{33}$$

We will now show that a Newton update restricted to the *fixed set* of variables would not change any of the coordinates in that set. In brief, the gradient condition $|\nabla_{ij} g(X)| \leq \lambda_{ij}$ entails that the inner coordinate descent steps, according to the update in (16), would set these coordinates to zero, so they would not change since they were zero to begin with.

To derive the optimality condition, we begin by introducing the minimum-norm subgradient of f and relate it to the optimal solution X^* of (3).

Definition 6 *The minimum-norm subgradient* $\text{grad}^S_{ij} f(X)$ *is defined as follows:*

$$\text{grad}^S_{ij} f(X) = \begin{cases} \nabla_{ij} g(X) + \lambda_{ij} & \text{if } X_{ij} > 0, \\ \nabla_{ij} g(X) - \lambda_{ij} & \text{if } X_{ij} < 0, \\ \text{sign}(\nabla_{ij} g(X)) \max(|\nabla_{ij} g(X)| - \lambda_{ij}, 0) & \text{if } X_{ij} = 0. \end{cases}$$

Lemma 7 *For any index set* J, $grad^S_{ij} f(X) = 0 \ \forall (i,j) \in J$ *if and only if* $\Delta^* = 0$ *is a solution of the following optimization problem:*

$$\arg\min_{\Delta} f(X + \Delta) \text{ such that } \Delta_{ij} = 0 \ \ \forall (i,j) \notin J. \tag{34}$$

Proof Any optimal solution Δ^* for (34) must satisfy the following, for all $(i,j) \in J$,

$$\nabla_{ij} g(X + \Delta^*) \begin{cases} = -\lambda_{ij} & \text{if } X_{ij} + \Delta^*_{ij} > 0, \\ = \lambda_{ij} & \text{if } X_{ij} + \Delta^*_{ij} < 0, \\ \in [-\lambda_{ij} \ \ \lambda_{ij}] & \text{if } X_{ij} + \Delta^*_{ij} = 0. \end{cases} \tag{35}$$

It can be seen immediately that $\Delta^* = 0$ satisfies (35) if and only if $\text{grad}^S_{ij} f(X) = 0$ for all $(i,j) \in J$. ■

In our case, $\nabla g(X) = S - X^{-1}$ and therefore

$$\text{grad}^S_{ij} f(X) = \begin{cases} (S - X^{-1})_{ij} + \lambda_{ij} & \text{if } X_{ij} > 0, \\ (S - X^{-1})_{ij} - \lambda_{ij} & \text{if } X_{ij} < 0, \\ \text{sign}((S - X^{-1})_{ij}) \max(|(S - X^{-1})_{ij}| - \lambda_{ij}, 0) & \text{if } X_{ij} = 0. \end{cases}$$

Our definition of the *fixed* and *free* sets is clearly motivated by the minimum norm subgradient. A variable X_{ij} belongs to the *fixed* set if and only if $X_{ij} = 0$ and $\text{grad}^S_{ij} f(X) = 0$.

Therefore, taking $J = S_{fixed}$ in Lemma 7, we can show that for any X_t and corresponding fixed and free sets S_{fixed} and S_{free} as defined by (33), $\Delta^* = 0$ is the solution of the following optimization problem:

$$\arg\min_{\Delta} f(X_t + \Delta) \text{ such that } \Delta_{ij} = 0 \ \ \forall (i,j) \in S_{free}.$$

Based on the above property, if we perform block coordinate descent restricted to the fixed set, then no updates would occur. We then perform the coordinate descent updates restricted to only the free set to find the Newton direction. With this modification, the number of variables over which we perform the coordinate descent update (16) can be potentially reduced from p^2 to the number of non-zeros in X_t. When the solution is sparse (depending on the value of Λ) the number of free variables can be much smaller than p^2 and we can obtain huge computational gains as a result. In essence, we very efficiently select a subset of the coordinates that need to be updated.

The attractive facet of this modification is that it leverages sparsity of the solution and intermediate iterates in a manner that falls within the block coordinate descent framework of Tseng and Yun (2007). The index sets $J_0, J_1, \ldots$ corresponding to the block coordinate descent steps in the general setting of Tseng and Yun (2007)[p. 392] need to satisfy a Gauss-Seidel type of condition:

$$\bigcup_{j=0,\ldots,T-1} J_{t+j} \supseteq \mathcal{N} \ \ \forall t = 1, 2, \ldots \tag{36}$$

for some fixed T, where $\mathcal{N}$ denotes the full index set. In our framework $J_0, J_2, \ldots$ denote the fixed sets at various iterations, and $J_1, J_3, \ldots$ denote the free sets. Since J_{2i} and J_{2i+1} is a partitioning of $\mathcal{N}$ the choice $T = 3$ will suffice. But will the size of the free set be small? We initialize X_0 to a diagonal matrix, which is sparse. The following lemma shows that after a *finite* number of iterations, the iterates X_t will have a similar sparsity pattern as the limit X^*. Lemma 8 is actually an immediate consequence of Lemma 14 in Section 4.

Lemma 8 *Assume that $\{X_t\}$ converges to X^*, the optimal solution of (3). If for some index pair (i,j), $|\nabla_{ij} g(X^*)| < \lambda_{ij}$ (so that $X^*_{ij} = 0$), then there exists a constant $\bar{t} > 0$ such that for all $t > \bar{t}$, the iterates X_t satisfy*

$$|\nabla_{ij} g(X_t)| < \lambda_{ij} \quad \textit{and} \ (X_t)_{ij} = 0. \tag{37}$$

Note that $|\nabla_{ij} g(X^*)| < \lambda_{ij}$ implies $X^*_{ij} = 0$ from the optimality condition of (3). This theorem shows that after $\bar{t}$-th iteration we can ignore all the indexes that satisfies (37), and in practice we can use (37) as a criterion for identifying the *fixed* set. A similar variable selection strategy is used in SVM (so called shrinking) and ℓ_1-regularized logistic regression problems as mentioned in Yuan et al. (2010). In our experiments, we demonstrate that this strategy reduces the size of the free set very quickly.

Lemma 8 suggests that QUIC can identify the zero pattern in finite steps. As we will prove later, QUIC has an asymptotic quadratic convergence rate and therefore once the zero pattern is correctly recognized, the algorithm often converges in a few additional iterations. Hence, the time needed to converge to the global optimum is not much more than the time needed to arrive at the zero pattern of the inverse covariance matrix.

3.4 The Block-Diagonal Structure of X^*

It has been shown recently (Mazumder and Hastie, 2012; Witten et al., 2011) that when the thresholded covariance matrix E defined by $E_{ij} = \mathcal{S}(S_{ij}, \lambda) = \text{sign}(S_{ij}) \max(|S_{ij}| - \lambda, 0)$ has the following block-diagonal structure:

$$E = \begin{bmatrix} E_1 & 0 & \dots & 0 \\ 0 & E_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & E_k \end{bmatrix}, \tag{38}$$

then the solution X^* of the inverse covariance estimation problem (2) also has the same block-diagonal structure:

$$X^* = \begin{bmatrix} X_1^* & 0 & \dots & 0 \\ 0 & X_2^* & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & X_k^* \end{bmatrix}.$$

This result can be extended to the case when the elements are penalized differently, i.e., λ_{ij}'s are different. Then, if $E_{ij} = \mathcal{S}(S_{ij}, \lambda_{ij})$ is block diagonal, so is the solution X^* of (3), see Hsieh et al. (2012). Thus each X_i^* can be computed independently. Based on this observation one can decompose the problem into sub-problems of smaller sizes, which can be solved much faster. In the following, we show that our updating rule and *fixed/free* set selection technique can automatically detect this block-diagonal structure for free.

Recall that we have a closed form solution in the first iteration when the input is a diagonal matrix. Based on (19), since $X_{ij} = 0$ for all $i \neq j$ in this step, we have

$$D_{ij} = X_{ii}X_{jj}\mathcal{S}(-S_{ij}, \lambda_{ij}) = -X_{ii}X_{jj}\mathcal{S}(S_{ij}, \lambda_{ij}) \quad \text{for all } i \neq j.$$

We see that after the first iteration the nonzero pattern of X will be exactly the same as the nonzero pattern of the thresholded covariance matrix E as depicted in (38). In order to establish that the same is true at each subsequent step, we complete our argument using induction, by showing that the non-zero structure is preserved.

More precisely, we show that the off-diagonal blocks always belong to the *fixed* set if $|S_{ij}| \leq \lambda_{ij}$. Recall the definition of the *fixed* set in (33). We need to check whether $|\nabla_{ij} g(X)| \leq \lambda_{ij}$ for all (i, j) in the off-diagonal blocks of E, whenever X has the same block-diagonal structure as E. Taking the inverse preserves the diagonal structure, and therefore $\nabla_{ij} g(X) = S_{ij} - X_{ij}^{-1} = S_{ij}$ for all such (i, j). We conclude noting that $E_{ij} = 0$ implies that $|\nabla_{ij} g(X)| \leq \lambda_{ij}$, meaning that (i, j) will belong to the *fixed* set.

We decompose the matrix into smaller blocks prior to running Cholesky factorization to avoid the $O(p^3)$ time complexity on the whole problem. The connected components of X can be detected in $O(\|X\|_0)$ time, which is very efficient when X is sparse. A detailed description of QUIC is presented in Algorithm 2.

4. Convergence Analysis

In Section 3, we introduced the main ideas behind our QUIC algorithm. In this section, we first prove that QUIC converges to the global optimum, and then show that the convergence

Algorithm 2: QUadratic approximation for sparse Inverse Covariance estimation (*QUIC*)

Input : Empirical covariance matrix S (positive semi-definite $p \times p$), regularization parameter matrix Λ, initial $X_0 \succ 0$, parameters $0 < \sigma < 0.5,\ 0 < \beta < 1$

Output: Sequence of X_t converging to $\arg\min_{X \succ 0} f(X)$, where $f(X) = g(X) + h(X)$, where $g(X) = -\log\det X + \mathrm{tr}(SX), h(X) = \|X\|_{1,\Lambda}$.

Compute $W_0 = X_0^{-1}$.
for $t = 0, 1, \ldots$ **do**
  $D = 0$, $U = 0$
  while *not converged* **do**
    Partition the variables into fixed and free sets:
    $S_{fixed} := \{(i,j) \mid |\nabla_{ij} g(X_t)| \leq \lambda_{ij} \text{ and } (X_t)_{ij} = 0\}$,
    $S_{free} := \{(i,j) \mid |\nabla_{ij} g(X_t)| > \lambda_{ij} \text{ or } (X_t)_{ij} \neq 0\}$.
    for $(i,j) \in S_{free}$ **do**
      $a = w_{ij}^2 + w_{ii} w_{jj},\ b = s_{ij} - w_{ij} + \mathbf{w}_{\cdot i}^T \mathbf{u}_{\cdot j},\ c = x_{ij} + d_{ij}$
      $\mu = -c + \mathcal{S}(c - b/a, \lambda_{ij}/a)$
      $d_{ij} \leftarrow d_{ij} + \mu,\ \mathbf{u}_{i\cdot} \leftarrow \mathbf{u}_{i\cdot} + \mu \mathbf{w}_{j\cdot},\ \mathbf{u}_{j\cdot} \leftarrow \mathbf{u}_{j\cdot} + \mu \mathbf{w}_{i\cdot}$
    end
  end
  for $\alpha = 1, \beta, \beta^2, \ldots$ **do**
    Compute the Cholesky factorization $LL^T = X_t + \alpha D$.
    if $X_t + \alpha D \succ 0$ **then**
      Compute $f(X_t + \alpha D)$ from L and $X_t + \alpha D$
      if $f(X_t + \alpha D) \leq f(X_t) + \alpha\sigma\left[\mathrm{tr}(\nabla g(X_t) D) + \|X_t + D\|_{1,\Lambda} - \|X\|_{1,\Lambda}\right]$ **then**
        break
      end
    end
  end
  $X_{t+1} = X_t + \alpha D$
  Compute $W_{t+1} = X_{t+1}^{-1}$ reusing the Cholesky factor.
end

rate is quadratic. Banerjee et al. (2008) showed that for the special case where $\Lambda_{ij} = \lambda$ the optimization problem (2) has a unique global optimum and that the eigenvalues of the primal optimal solution X^* are bounded. In the following, we show this result for more general Λ where only the off-diagonal elements need to be positive.

Theorem 9 *There exists a unique minimizer X^* for the optimization problem* (3)*, where $\lambda_{ij} > 0$ for $i \neq j$, and $\lambda_{ij} \geq 0$.*

Proof According to Lemma 2, the level set U defined in (23) contains all the iterates, and it is in turn contained in the compact set $S \equiv \{X \mid mI \preceq X \preceq MI\}$. According to the Weierstrass extreme value theorem (Apostol, 1974), any continuous function in a compact set attains its minimum. Furthermore, $\nabla^2 g(X) = X^{-1} \otimes X^{-1}$ implies $\nabla^2 g(X) \succeq M^{-2} I$.

Since $\|X\|_{1,\Lambda}$ is convex and $-\log\det(X)$ is strongly convex, we have that $f(X)$ is strongly convex on the compact set S, and therefore the minimizer X^* is unique (Apostol, 1974). ■

4.1 Convergence Guarantee

In order to show that QUIC converges to the optimal solution, we consider a more general setting of the quadratic approximation algorithm: at each iteration, the iterate Y_t is updated by $Y_{t+1} = Y_t + \alpha_t D^*_{J_t}(Y_t)$ where J_t is a subset of variables chosen to update at iteration t, $D^*_{J_t}(Y_t)$ is the Newton direction restricted to J_t defined by (20), and α_t is the step size selected by the Armijo rule given in Section 3.2. The algorithm is summarized in Algorithm 3. Similar to the block coordinate descent framework of Tseng and Yun (2007), we assume the index set J_t satisfies a Gauss-Seidel type of condition:

$$\bigcup_{j=0,\ldots,T-1} J_{t+j} \supseteq \mathcal{N} \quad \forall t = 1, 2, \ldots. \tag{39}$$

Algorithm 3: General Block Quadratic Approximation method for Sparse Inverse Covariance Estimation

Input : Empirical covariance matrix S (positive semi-definite $p \times p$), regularization parameter matrix Λ, initial Y_0, inner stopping tolerance ϵ

Output: Sequence of Y_t.

for $t = 0, 1, \ldots$ **do**
  Generate a variable subset J_t.
  Compute the Newton direction $D^*_t \equiv D^*_{J_t}(Y_t)$ by (20).
  Compute the step-size α_t using the *Armijo*-rule based step-size selection in (21).
  Update $Y_{t+1} = Y_t + \alpha_t D^*_t$.
end

In QUIC, $J_0, J_2, \ldots$ denote the fixed sets and $J_1, J_3, \ldots$ denote the free sets. If $\{X_t\}_{t=0,1,2,\ldots}$ denotes the sequence generated by QUIC, then

$$Y_0 = Y_1 = X_0, Y_2 = Y_3 = X_1, \ldots, Y_{2i} = Y_{2i+1} = X_i.$$

Moreover, since each J_{2i} and J_{2i+1} is a partitioning of $\mathcal{N}$, the choice $T = 3$ will satisfy (39). In the rest of this section, we show that $\{Y_t\}_{t=0,1,2,\ldots}$ converges to the global optimum, thus $\{X_t\}_{t=0,1,2,\ldots}$ generated by QUIC also converges to the global optimum.

Our first step towards the convergence proof is a lemma on convergent subsequences.

Lemma 10 *For any convergent subsequence $Y_{s_t} \to \bar{Y}$ where $\bar{Y}$ is a limit point, we have $D^*_{s_t} \equiv D^*_{J_{s_t}}(Y_{s_t}) \to 0$.*

Proof The objective value decreases according to the line search condition (21) and Proposition 4. According to Lemma 2, $f(Y_{s_t})$ cannot converge to negative infinity, so $f(Y_{s_t}) - f(Y_{s_{t+1}}) \to 0$. The line search condition (21) implies that $\alpha_{s_t}\delta_{s_t} \to 0$.

We proceed to prove the statement by contradiction. If $D^*_{s_t}$ does not converge to 0, then there exists an infinite index set $\mathcal{T} \subseteq \{s_1, s_2, \ldots\}$ and $\eta > 0$ such that $\|D^*_t\|_F > \eta$ for all $t \in \mathcal{T}$. According to Proposition 4, δ_{s_t} is bounded away from 0, therefore $\delta_{s_t} \not\to 0$, while $\alpha_{s_t} \to 0$. We can assume without loss of generality that $\alpha_{s_t} < 1\ \forall t$, that is the line search condition is not satisfied in the first attempt. We will work in this index set $\mathcal{T}$ in the derivations that follow.

The line search step size $\alpha_t < 1$ ($t \in \mathcal{T}$) satisfies (21), but $\overline{\alpha}_t = \alpha_t/\beta$ does not satisfy (21) by the minimality of our line search procedure. So we have:

$$f(Y_t + \overline{\alpha}_t D^*_t) - f(Y_t) > \sigma \overline{\alpha}_t \delta_t. \tag{40}$$

If $Y_t + \overline{\alpha}_t D^*_t$ is not positive definite, then as is standard, $f(Y_t + \overline{\alpha}_t D^*_t) = \infty$, so (40) still holds. We expand (40) and apply 22 to get

$$\begin{aligned}\sigma\overline{\alpha}_t\delta_t &\le g(Y_t + \overline{\alpha}_t D^*_t) - g(Y_t) + \|Y_t + \overline{\alpha}_t D^*_t\|_{1,\Lambda} - \|Y_t\|_{1,\Lambda} \\ &\le g(Y_t + \overline{\alpha}_t D^*_t) - g(Y_t) + \overline{\alpha}_t(\|Y_t + D^*_t\|_{1,\Lambda} - \|Y_t\|_{1,\Lambda}), \forall t \in \mathcal{T}.\end{aligned}$$

By the definition of δ_t, we have:

$$\begin{aligned}\sigma\delta_t &\le \frac{g(Y_t + \overline{\alpha}_t D^*_t) - g(Y_t)}{\overline{\alpha}_t} + \delta_t - \operatorname{tr}(\nabla g(Y_t)^T D^*_t), \\ (1-\sigma)(-\delta_t) &\le \frac{g(Y_t + \overline{\alpha}_t D^*_t) - g(Y_t)}{\overline{\alpha}_t} - \operatorname{tr}(\nabla g(Y_t)^T D^*_t).\end{aligned}$$

By Proposition 4 we have $\delta_t \le -(1/M^2)\|D^*_t\|_F^2$, so using $\|D^*_t\|$ to denote $\|D^*_t\|_F$ for the rest of the proof, we get

$$\begin{aligned}(1-\sigma)M^{-2}\|D^*_t\|^2 &\le \frac{g(Y_t + \overline{\alpha}_t D^*_t) - g(Y_t)}{\overline{\alpha}_t} - \operatorname{tr}(\nabla g(Y_t)^T D^*_t) \\ (1-\sigma)M^{-2}\|D^*_t\| &\le \frac{g\left(Y_t + \overline{\alpha}_t\|D^*_t\|\frac{D^*_t}{\|D^*_t\|}\right) - g(Y_t)}{\overline{\alpha}_t\|D^*_t\|} - \operatorname{tr}\left(\nabla g(Y_t)^T \frac{D^*_t}{\|D^*_t\|}\right).\end{aligned}$$

We set $\hat{\alpha}_t = \overline{\alpha}_t\|D^*_t\|$. Since $\|D^*_t\| > \eta$ for all $t \in \mathcal{T}$ we have:

$$\begin{aligned}(1-\sigma)M^{-2}\eta &< \frac{g\left(Y_t + \hat{\alpha}_t\frac{D^*_t}{\|D^*_t\|}\right) - g(Y_t)}{\hat{\alpha}_t} - \operatorname{tr}\left(\nabla g(Y_t)^T \frac{D^*_t}{\|D^*_t\|}\right) \\ &= \frac{\hat{\alpha}_t \operatorname{tr}\left(\nabla g(Y_t)\frac{D^*_t}{\|D^*_t\|}\right) + O(\hat{\alpha}_t^2)}{\hat{\alpha}_t} - \operatorname{tr}\left(\nabla g(Y_t)^T \frac{D^*_t}{\|D^*_t\|}\right) \\ &= O(\hat{\alpha}_t).\end{aligned} \tag{41}$$

Again, by Proposition 4,

$$-\alpha_t\delta_t \ge \alpha_t M^{-2}\|D^*_t\|^2 > M^{-2}\alpha_t\|D^*_t\|\eta.$$

Since $\{\alpha_t\delta_t\}_t \to 0$, it follows that $\{\alpha_t\|D^*_t\|\}_t \to 0$ and $\{\hat{\alpha}_t\}_t \to 0$. Taking limit of (41) as $t \in \mathcal{T}$ and $t \to \infty$, we have

$$(1-\sigma)M^{-2}\eta \le 0,$$

a contradiction, finishing the proof. ■

Now that we have proved that $D^*_{J_t}$ converges to zero for the converging subsequence, we next show that D^*_J is closely related to the minimum-norm subgradient $\text{grad}^S f(Y)$ (see Definition 6), which in turn is an indicator of optimality as proved in Lemma 7.

Lemma 11 *For any index set J and positive definite Y, $D^*_J(Y) = 0$ if and only if $\text{grad}^S_{ij} f(Y) = 0$ for all $(i,j) \in J$.*

Proof The optimality condition of (20) can be written as

$$\nabla_{ij} g(X) + (\nabla^2 g(X)\,\text{vec}(D))_{ij} \begin{cases} = -\lambda & \text{if } X_{ij} + D_{ij} > 0 \\ = \lambda & \text{if } X_{ij} + D_{ij} < 0 \\ \in [-\lambda,\ \lambda] & \text{if } X_{ij} + D_{ij} = 0, \end{cases} \qquad \forall (i,j) \in J. \tag{42}$$

$D^*_J(Y) = 0$ if and only if $D^* = 0$ satisfies (42), and this condition is equivalent to (35) restricted to $(i,j) \in J$, which in turn is equivalent to the optimality condition of f. Therefore $D^*_J(Y) = 0$ iff $\text{grad}^S_{ij} f(Y) = 0$ for all $(i,j) \in J$. ■

Based on these lemmas, we are now able to prove our main convergence theorem.

Theorem 12 *Algorithm 3 converges to the unique global optimum Y^*.*

Proof Since all the iterates Y_t are in a compact set (as shown in Lemma 2), there exists a subsequence $\{Y_t\}_{\mathcal{T}}$ that converges to a limit point $\bar{Y}$. Since the cardinality of each index set J_t selected is finite, we can further assume that $J_t = \bar{J}_0$ for all $t \in \bar{\mathcal{T}}$, where $\bar{\mathcal{T}}$ is a subsequence of $\mathcal{T}$. From Lemma 10, $D^*_{\bar{J}_0}(Y_t) \to 0$. By continuity of $\nabla g(Y)$ and $\nabla^2 g(Y)$, it is easy to show that $D^*_{\bar{J}_0}(Y_t) \to D^*_{\bar{J}_0}(\bar{Y})$. Therefore $D^*_{\bar{J}_0}(\bar{Y}) = 0$. Based on Lemma 11, we have

$$\text{grad}^S_{ij} f(Y) = 0 \quad \text{for all } (i,j) \in \bar{J}_0.$$

Furthermore, $\{D^*_{\bar{J}_0}(Y_t)\}_{\mathcal{T}} \to 0$ and $\|Y_t - Y_{t+1}\|_F \le \|D^*_{\bar{J}_0}(Y_t)\|_F$, so $\{Y_{t+1}\}_{t \in \mathcal{T}}$ also converges to $\bar{Y}$. By considering a subsequence of $\mathcal{T}$ if necessary, we can further assume that $J_{t+1} = \bar{J}_1$ for all $t \in \mathcal{T}$. By the same argument, we can show that $\{D^*_{J_{t+1}}(Y_t)\}_{\mathcal{T}} \to 0$, so $D^*_{\bar{J}_1}(\bar{Y}) = 0$. Similarly, we can show that $D^*_{\bar{J}_t}(\bar{Y}) = 0\ \forall t = 0, \ldots, T-1$ can be assumed for an appropriate subset of $\mathcal{T}$. With assumption (39) and Lemma 11 we have

$$\text{grad}^S_{ij} f(\bar{Y}) = 0\ \forall i, j. \tag{43}$$

Using Lemma 7 with J as the set of all variables, we can show that (43) implies that $\bar{Y}$ is the global optimum. ■

It is straightforward to generalize Theorem 12 to prove the convergence of block coordinate descent when the Hessian $\nabla^2 g(X)$ is replaced by another positive definite matrix. The proof strategies are similar to Tseng and Yun (2007) and we omit the detailed derivation in this paper.

4.2 Asymptotic Convergence Rate

Newton methods on constrained minimization problems:
The convergence rate of the Newton method on bounded constrained minimization has been studied in Levitin and Polyak (1966) and Dunn (1980). Here, we briefly mention their results.

Assume we want to solve a constrained minimization problem

$$\min_{\mathbf{x}\in\Omega} F(\mathbf{x}),$$

where Ω is a nonempty subset of R^n denoting the constraint set and $F : R^n \to R$ has a second derivative $\nabla^2 F(\mathbf{x})$. Then beginning from $\mathbf{x}_0$, the natural Newton updates entail computing the $(k+1)$-st iterate $\mathbf{x}_{k+1}$ as

$$\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}\in\Omega} \nabla F(\mathbf{x}_k)^T(\mathbf{x}-\mathbf{x}_k) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_k)^T\nabla^2 F(\mathbf{x}_k)(\mathbf{x}-\mathbf{x}_k). \tag{44}$$

For simplicity, we assume that F is strictly convex, and has a unique minimizer $\mathbf{x}^*$ in Ω. Then the following theorem holds.

Theorem 13 (Theorem 3.1 in Dunn, 1980) *Assume F is strictly convex, has a unique minimizer $\mathbf{x}^*$ in Ω, and that $\nabla^2 F(\mathbf{x})$ is Lipschitz continuous. Then for all $\mathbf{x}_0$ sufficiently close to $\mathbf{x}^*$, the sequence $\{\mathbf{x}_k\}$ generated by (44) converges quadratically to $\mathbf{x}^*$.*

This theorem is proved in Dunn (1980). In our case, the objective function $f(X)$ is non-smooth so Theorem 13 does not directly apply. Instead, we will first show that after a finite number of iterations the sign of the iterates $\{X_t\}$ generated by QUIC will not change, so that we can then use Theorem 13 to establish asymptotic quadratic convergence.

Quadratic convergence rate for QUIC**:**
Unlike as in (44), our Algorithm 3 does not perform an unrestricted Newton update: it iteratively selects subsets of variables $\{J_t\}_{t=1,\ldots}$ (*fixed* and *free* sets), and computes the Newton direction restricted to the free sets. In the following, we show that the sequence $\{X_t\}_{t=1,2,\ldots}$ generated by QUIC does ultimately converge quadratically to the global optimum.

Assume X^* is the optimal solution, then we can divide the index set with $\lambda_{ij} \neq 0$ into three subsets:

$$P = \{(i,j) \mid X^*_{ij} > 0\}, \quad N = \{(i,j) \mid X^*_{ij} < 0\}, \quad Z = \{(i,j) \mid X^*_{ij} = 0\}. \tag{45}$$

From the optimality condition for X^*,

$$\nabla_{ij} g(X^*) \begin{cases} = -\lambda_{ij} & \text{if } (i,j) \in P, \\ = \lambda_{ij} & \text{if } (i,j) \in N, \\ \in [-\lambda_{ij}, \lambda_{ij}] & \text{if } (i,j) \in Z. \end{cases} \tag{46}$$

Lemma 14 *Assume that the sequence $\{X_t\}$ converges to the global optimum X^*. Then there exists a $\bar{t}$ such that for all $t > \bar{t}$,*

$$(X_t)_{ij} \begin{cases} \geq 0 & \text{if } (i,j) \in P, \\ \leq 0 & \text{if } (i,j) \in N, \\ = 0 & \text{if } (i,j) \in Z. \end{cases} \tag{47}$$

Proof We prove the case for $(i,j) \in P$ by contradiction, the other two cases can be handled similarly. If we cannot find a $\bar{t}$ satisfying the first condition in (47), then there exists an infinite subsequence $\{X_{a_t}\}$ such that for each a_t there exists a $(i,j) \in P$ such that $(X_{a_t})_{ij} < 0$. Since the cardinality of P is finite, we can further find a specific pair $(i,j) \in P$ such that $(X_{s_t})_{ij} < 0$ for all s_t, where s_t is a subsequence of a_t. We consider the update from X_{s_t-1} to X_{s_t}. From Lemma 5, we can assume that s_t is large enough so that the step size equals 1, therefore $X_{s_t} = X_{s_t-1} + D^*(X_{s_t-1})$ where $D^*(X_{s_t-1})$ is defined in (20). Since $(X_{s_t})_{ij} = (X_{s_t-1})_{ij} + (D^*(X_{s_t-1}))_{ij} < 0$, from the optimality condition of (20) we have

$$\left(\nabla g(X_{s_t-1}) + \nabla^2 g(X_{s_t-1}) \operatorname{vec}(D^*(X_{s_t-1}))\right)_{ij} = \lambda_{ij}. \tag{48}$$

Since $D^*(X_{s_t-1})$ converges to 0, (48) implies that $\{\nabla_{ij} g(X_{s_t-1})\}$ will converge to λ_{ij}. However, (46) implies $\nabla_{ij} g(X^*) = -\lambda_{ij}$, and by the continuity of ∇g we get that $\{\nabla_{ij} g(X_t)\}$ converges to $\nabla_{ij} g(X^*) = -\lambda_{ij}$, a contradiction, finishing the proof. ■

The following lemma shows that the coordinates from the fixed set remain zero after a finite number of iterations.

Lemma 15 *Assume $X_t \to X^*$. There exists a $\bar{t} > 0$ such that variables in P or N will not be selected to be in the fixed set S_{fixed}, when $t > \bar{t}$. That is,*

$$S_{fixed} \subseteq Z.$$

Proof Since X_t converges to X^*, $(X_t)_{ij}$ converges to $X^*_{ij} > 0$ if $(i,j) \in P$ and to $X^*_{ij} < 0$ if $(i,j) \in N$. Recall that (i,j) belongs to the fixed set only if $(X_t)_{ij} = 0$. When t is large enough, $(X_t)_{ij} \neq 0$ when $X_t \in P \cup N$, therefore P and N will be disjoint from the fixed set. Moreover, by the definition of the fixed set (33), indexes with $\lambda_{ij} = 0$ will never be selected. We proved that the fixed set will be a subset of Z when t is large enough. ■

Theorem 16 *The sequence $\{X_t\}$ generated by the* QUIC *algorithm converges quadratically to X^*, that is for some constant $\kappa > 0$,*

$$\lim_{t \to \infty} \frac{\|X_{t+1} - X^*\|_F}{\|X_t - X^*\|_F^2} = \kappa.$$

Proof First, if the index sets P, N and Z (related to the optimal solution) are given, the optimum of (2) is the same as the optimum of the following constrained minimization problem:

$$\begin{aligned} \min_X \quad & -\log\det(X) + \operatorname{tr}(SX) + \sum_{(i,j)\in P} \lambda_{ij} X_{ij} - \sum_{(i,j)\in N} \lambda_{ij} X_{ij} \\ \text{s.t.} \quad & X_{ij} \geq 0 \ \ \forall (i,j) \in P, \quad X_{ij} \leq 0 \ \ \forall (i,j) \in N, \quad X_{ij} = 0 \ \ \forall (i,j) \in Z. \end{aligned} \tag{49}$$

In the following, we show that when t is large enough, QUIC solves the minimization problem described by (49).

1. The constraints in (49) are satisfied by QUIC iterates after a finite number of steps, as shown in Lemma 14. Thus, the ℓ_1-regularized Gaussian MLE (3) is equivalent to the smooth constrained objective (49), since the constraints in (49) are satisfied when solving (3).

2. Since the optimization problem in (49) is smooth, it can be solved using constrained Newton updates as in (44). The QUIC update direction $D_J^*(X_t)$ is restricted to a set of free variables in J. This is exactly equal to the unrestricted Newton update as in (44), after a finite number of steps, as established by Lemma 15. In particular, at each iteration the fixed set is contained in Z, which is the set which always satisfies $(D_t^*)_Z = 0$ for large enough t.

3. Moreover, by Lemma 5 the step size is $\alpha = 1$ when t is large enough.

Therefore our algorithm is equivalent to the constrained Newton method in (44), which in turn converges quadratically to the optimal solution of (49). Since the revised problem (49) and our original problem (3) has the same minimum, we have shown that QUIC converges quadratically to the optimum of (3). ■

Note that the constant κ is an increasing function of the Lipschitz constant of $\nabla^2 g(X)$ (as shown in Dunn, 1980), which is related to the quality of quadratic approximation. We have shown in Lemma 2 that $mI \preceq X \preceq MI$, therefore the Lipschitz constant of $\nabla^2 g(X) = X^{-1} \otimes X^{-1}$ is also upper bounded.

In the next section, we show that this asymptotic convergence behavior of QUIC is corroborated empirically as well.

5. Experimental Results

We begin this section by comparing QUIC to other methods on synthetic and real data sets. Then, we present some empirical analysis of QUIC regarding the use of approximate Newton directions and effects of parameterization.

5.1 Comparisons with Other Methods

We now compare the performance of QUIC on both synthetic and real data sets to other state-of-the-art methods. We have implemented QUIC in C++ with MATLAB interface, and all experiments were executed on 2.83GHz Xeon X5440 machines with 32G RAM and Linux OS.

We include the following algorithms in our comparisons:

- ALM: the Alternating Linearization Method proposed by Scheinberg et al. (2010). We use their MATLAB source code for the experiments.

- ADMM: another implementation of the alternating linearization method implemented by Boyd et al. (2012). The MATLAB code can be downloaded from `http://www.stanford.edu/~boyd/papers/admm/`. We found that the default parameters (which we note are independent of the regularization penalty) yielded slow convergence; we

set the augmented Lagrangian parameter to $\rho = 50$ and the over-relaxation parameter to $\alpha = 1.5$. These parameters achieved the best speed on the ER data set.

- GLASSO: the block coordinate descent method proposed by Friedman et al. (2008). We use the latest version GLASSO 1.7 downloaded from `http://www-stat.stanford.edu/~tibs/glasso/`. We directly call their Fortran procedure using a MATLAB interface.

- PSM: the Projected Subgradient Method proposed by Duchi et al. (2008). We use the MATLAB source code provided in the PQN package (available at `http://www.cs.ubc.ca/~schmidtm/Software/PQN.html`).

- SINCO: the greedy coordinate descent method proposed by Scheinberg and Rish (2010). The code can be downloaded from `https://projects.coin-or.org/OptiML/browser/trunk/sinco`.

- IPM: An inexact interior point method proposed by Li and Toh (2010). The source code can be downloaded from `http://www.math.nus.edu.sg/~mattohkc/Covsel-0.zip`.

- PQN: the projected quasi-Newton method proposed by Schmidt et al. (2009). The source code can be downloaded from `http://www.di.ens.fr/~mschmidt/Software/PQN.html`.

In the following, we compare QUIC and the above state-of-the-art methods on synthetic and real data sets with various settings of λ. Note that we use the identity matrix as the initial point for QUIC, ADMM, SINCO, and IPM. Since the identity matrix is not a dual feasible point for dual methods (including GLASSO, PSM and PQN), we use $S + \lambda I$ as the dual initial point, which is the default setting in their original package.

5.1.1 Experiments on Synthetic Data Sets

We first compare the run times of the different methods on synthetic data. We generate the two following types of graph structures for the underlying Gaussian Markov Random Fields:

- Chain Graphs: The ground truth inverse covariance matrix Σ^{-1} is set to be $\Sigma^{-1}_{i,i-1} = -0.5$ and $\Sigma^{-1}_{i,i} = 1.25$.

- Graphs with Random Sparsity Structures: We use the procedure given in Example 1 in Li and Toh (2010) to generate inverse covariance matrices with random non-zero patterns. Specifically, we first generate a sparse matrix U with nonzero elements equal to ± 1, set Σ^{-1} to be $U^T U$ and then add a diagonal term to ensure Σ^{-1} is positive definite. We control the number of nonzeros in U so that the resulting Σ^{-1} has approximately $10p$ nonzero elements.

Given the inverse covariance matrix Σ^{-1}, we draw a limited number, $n = p/2$ i.i.d. samples from the corresponding GMRF distribution, in order to simulate the high-dimensional setting.

Data set			Parameter	Properties of the solution		
pattern	p	$\|\Sigma^{-1}\|_0$	λ	$\|X^*\|_0$	TPR	FPR
chain	1000	2998	0.4	3028	1	3×10^{-5}
chain	4000	11998	0.4	11998	1	0
chain	10000	29998	0.4	29998	1	0
random	1000	10758	0.12	10414	0.69	4×10^{-3}
			0.075	55830	0.86	0.05
random	4000	41112	0.08	41936	0.83	6×10^{-3}
			0.05	234888	0.97	0.05
random	10000	91410	0.08	89652	0.90	4×10^{-6}
			0.04	392786	1	3×10^{-3}

Table 1: The parameters and properties of the solution for the synthetic data sets. p stands for dimension, $\|\Sigma^{-1}\|_0$ indicates the number of nonzeros in ground truth inverse covariance matrix, $\|X^*\|_0$ is the number of nonzeros in the solution. TPR and FPR denote the true and false recovery rates, respectively, defined in (50).

Table 1 shows the attributes of the synthetic data sets that we used in the timing comparisons. The dimensionality varies from $\{1000, 4000, 10000\}$. For chain graphs, we select λ so that the solution has (approximately) the correct number of nonzero elements. In order to test the performance of the algorithms under different values of λ, for the case of random-structured graphs we considered two λ values; one of which resulted in the discovery of the correct number of non-zeros and one which resulted in five-times thereof. We measured the accuracy of the graph structure recovered by the true positive rate (TPR) and false positive rate (FPR) defined as

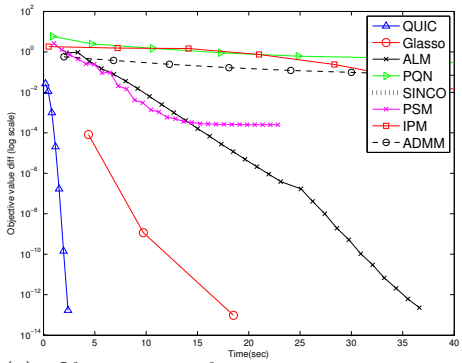
$$\text{TPR} = \frac{|\{(i,j) \mid (X^*)_{ij} > 0 \text{ and } Q_{ij} > 0\}|}{|\{(i,j) \mid Q_{ij} > 0\}|}, \text{FPR} = \frac{|\{(i,j) \mid (X^*)_{ij} > 0 \text{ and } Q_{ij} = 0\}|}{|\{(i,j) \mid Q_{ij} = 0\}|}, \tag{50}$$

where Q is the ground truth sparse inverse covariance.

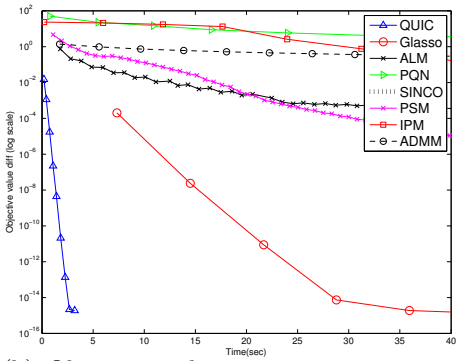
Since QUIC does not natively compute a dual solution, the duality gap cannot be used as a stopping condition.[2] In practice, we can use the minimum-norm sub-gradient (see Definition 6) as the stopping condition. There is no additional computational cost to this approach because X^{-1} is computed as part of the QUIC algorithm. In the experiments, we report the time for each algorithm to achieve ϵ-accurate solution defined by $f(X^k) - f(X^*) < \epsilon f(X^*)$. The global optimum X^* is computed by running QUIC until it converges to a solution with $\|\operatorname{grad}^S f(X_t)\| < 10^{-13}$.

Table 2 shows the results for $\epsilon = 10^{-2}$ and 10^{-6}, where $\epsilon = 10^{-2}$ tests the ability of the algorithm to get a good initial guess (the nonzero structure), and $\epsilon = 10^{-6}$ tests whether the algorithm can achieve an accurate solution. Table 2 shows that QUIC is consistently and

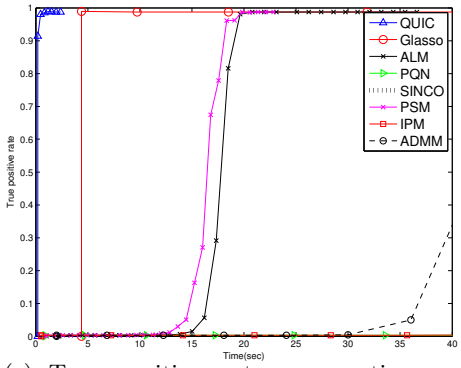
2. Note that $W = X^{-1}$ cannot be expected to satisfy the dual constraints $|W_{ij} - S_{ij}| \leq \lambda_{ij}$. One could project X^{-1} in order to enforce the constraints and use the resulting matrix to compute the duality gap. Our implementation provides this computation only if the user requests it.



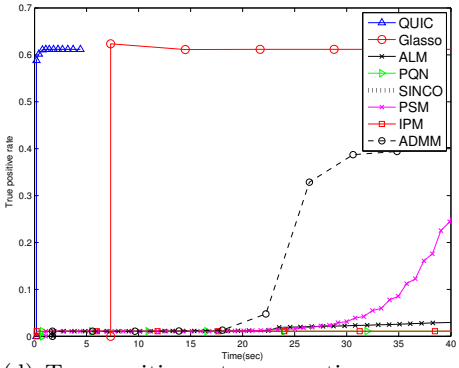
(a) Objective value versus time on chain1000



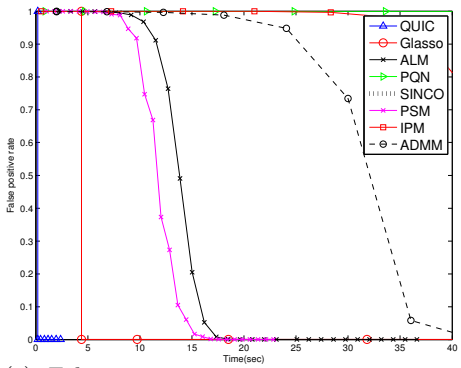
(b) Objective value versus time on random1000



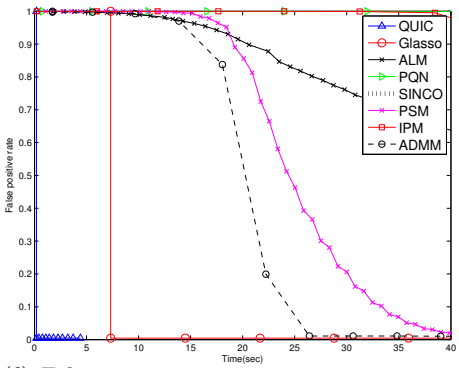
(c) True positive rate versus time on chain1000



(d) True positive rate versus time on random1000



(e) False positive rate versus time on chain1000



(f) False positive rate versus time on random1000

Figure 1: Comparison of algorithms on two synthetic data sets: chain1000 and random1000. The regularization parameter λ is chosen to recover (approximately) correct number of nonzero elements (see Table 1). We can see that QUIC achieves a solution with better objective function value as well as better true positive and false positive rates in both data sets. Notice that each marker in the figures indicates one iteration. Note that all results are averaged over 5 replicated runs.

overwhelmingly faster than other methods, both initially with $\epsilon = 10^{-2}$, and at $\epsilon = 10^{-6}$. Moreover, for the $p = 10000$ random pattern, there are $p^2 = 100$ million variables and the

Parameters				Time (in seconds)							
pattern	p	λ	ϵ	QUIC	ALM	Glasso	PSM	IPM	SINCO	PQN	ADMM
chain	1000	0.4	10^{-2}	**< 1**	19	9	16	86	120	110	62
			10^{-6}	**2**	42	20	35	151	521	210	281
chain	4000	0.4	10^{-2}	**11**	922	460	568	3458	5246	672	1028
			10^{-6}	**54**	1734	1371	1258	5754	*	10525	2584
chain	10000	0.4	10^{-2}	**217**	13820	10250	8450	*	*	*	*
			10^{-6}	**987**	28190	*	19251	*	*	*	*
random	1000	0.12	10^{-2}	**< 1**	42	7	20	72	61	33	35
			10^{-6}	**1**	28250	15	60	117	683	158	252
		0.075	10^{-2}	**1**	66	14	24	78	576	15	56
			10^{-6}	**7**	*	43	92	146	4449	83	*
random	4000	0.08	10^{-2}	**23**	1429	864	1479	4928	7375	2052	1025
			10^{-6}	**160**	*	1743	4232	8097	*	4387	*
		0.05	10^{-2}	**66**	*	2514	2963	5621	*	2746	*
			10^{-6}	**479**	*	5712	9541	13650	*	8718	*
random	10000	0.08	10^{-2}	**338**	26270	14296	*	*	*	*	*
			10^{-6}	**1125**	*	*	*	*	*	*	*
		0.04	10^{-2}	**804**	*	*	*	*	*	*	*
			10^{-6}	**2951**	*	*	*	*	*	*	*

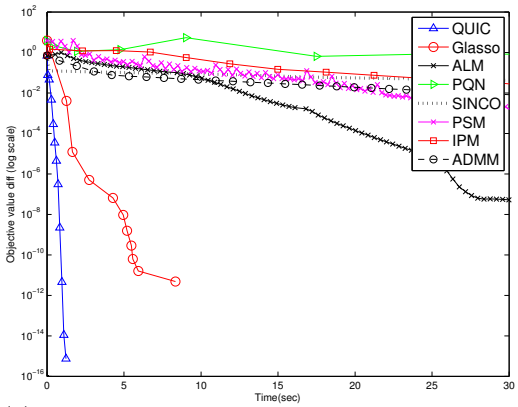
Table 2: Running time comparisons on synthetic data sets. See also Table 1 regarding the data set properties. We use $*$ to indicate that the run time exceeds 30,000 seconds (8.3 hours). The results show that QUIC is overwhelmingly faster than other methods, and is the only one which is able to scale up to solve problems with $p = 10000$.

selection of fixed/free sets helps QUIC to focus on a small subset of the variables. We converge to the solution in about 15 minutes, while other methods fail to obtain even an initial guess within 8 hours.

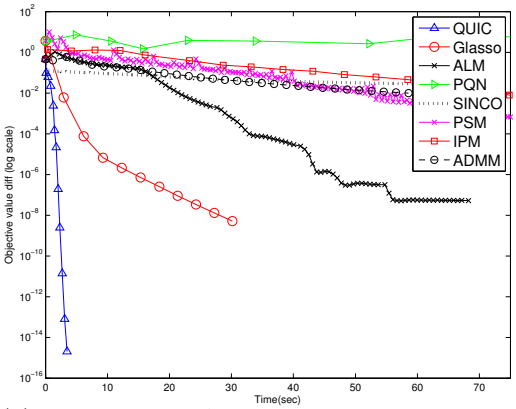
In some applications, researchers are primarily interested in just the graph structure represented by the solution. Therefore, in addition to the objective function value, we further compare the true positive and false positive rates of the nonzero pattern of the iterates X_t obtained by each algorithm. In Figure 1, we use two synthetic data sets, chain1000 and random1000, as examples. For each algorithm, we plot the objective function value, true positive rate, and false positive rate of the iterates X_t versus run time. For both ground truth pattern we generate 5 data sets and report the average results in Figure 1. For the methods that solve the dual problem, the sparse inverse covariance matrix $X_t = W_t^{-1}$ is usually dense, so we consider elements with absolute value larger than 10^{-6} as nonzero elements. We can see that QUIC not only obtains lower objective function value efficiently, but also recovers the ground truth structure of GMRF faster than other methods.

5.1.2 Experiments on Real Data sets

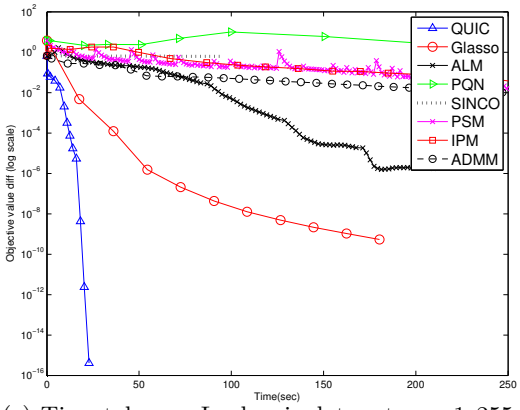
We use the real world biology data sets preprocessed by Li and Toh (2010) to compare the performance of our method with other state-of-the-art methods. In the first set of experiments, we set the regularization parameter λ to be 0.5, which achieves reasonable sparsity for the following data sets: Estrogen ($p = 692$), Arabidopsis ($p = 834$), Leukemia ($p = 1,225$), Hereditary ($p = 1,869$). In Figure 2 we plot the relative error $(f(X_t) - f(X^*))/f(X^*)$ (on a log scale) against time in seconds. We can observe from Figure 2 that under the setting of large λ and sparse solution, QUIC can be seen to achieve super-linear convergence while other methods exhibit at most a linear convergence. Overall, we see that QUIC can be five times faster than other methods, and can be expected to be even faster if a higher accuracy is desired.



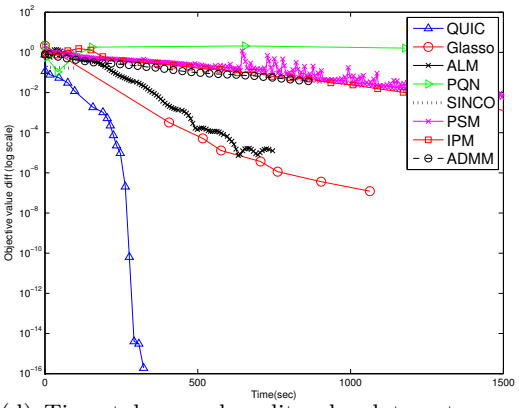
(a) Time taken on ER data set, $p = 692$, $\frac{\|X^*\|_0}{p^2} = 0.0222$



(b) Time taken on Arabidopsis data set, $p = 834$, $\frac{\|X^*\|_0}{p^2} = 0.0296$



(c) Time taken on Leukemia data set, $p = 1,255$, $\frac{\|X^*\|_0}{p^2} = 0.0221$



(d) Time taken on hereditarybc data set, $p = 1,869$, $\frac{\|X^*\|_0}{p^2} = 0.0198$

Figure 2: Comparison of algorithms on real data sets with $\lambda = 0.5$. The results show that QUIC converges faster than the other methods. Notice that each marker in the figures indicates one iteration. All the results are averaged over five runs.

In the second set of experiments, we compare the algorithms under different values of the regularization parameter λ on the ER data set. In Figure 2(a) we show the results for $\lambda = 0.5$. We then decrease λ to 0.1, 0.05, 0.01 using the same data sets and show the results in Figure 6. A smaller λ yields a denser solution, and we list the density of the converged solution X^* in Figure 6. From Figure 6 we can see that QUIC is the most efficient method when λ is large (solution is sparse), but IPM and PSM outperform QUIC when λ is small (solution is dense). However, such cases are usually not so useful in practice because when solving the ℓ_1-regularized MLE problem one usually wants a sparse graph structure for the GMRF. The main reason that QUIC is so efficient for large λ is that with *fixed/free* set selection, the coordinate descent method can focus on a small portion of variables, while in PSM and IPM the whole matrix is updated at each iteration.

5.2 Empirical Analysis of QUIC

Next we present some empirical analysis of QUIC regarding the effects of several parameters. We also demonstrate that the fixed/free set selection in QUIC significantly reduce the computational complexity.

5.2.1 Effect of Approximate Newton Directions

In the convergence analysis of Section 4, we assumed that each Newton direction D_t^* is computed exactly by solving the Lasso subproblem (20). In our implementation we use an iterative (coordinate descent) solver to compute D_t^*, which after a finite set of iterations only solves the problem approximately. In the first experiment we explore how varying the accuracy to which we compute the Newton direction affects overall performance. In Figure 3 we plot the total run times for the ER biology data set from Li and Toh (2010) corresponding to different numbers of inner iterations used in the coordinate descent solver.

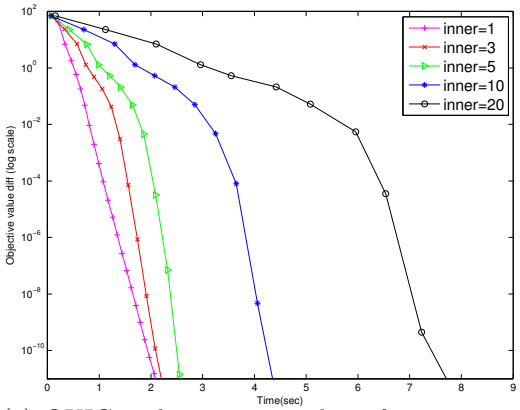
We can observe that QUIC with one inner iteration converges faster in the beginning, but eventually achieves just a linear convergence rate, while QUIC with 20 inner iterations converges more slowly in the beginning, but eventually achieves quadratic convergence. Based on this observation, we propose an adaptive stopping condition: we set the number of coordinate descent steps to be $\lceil \alpha t \rceil$ for the t-th outer iteration, where α is a constant; we use $\alpha = 1/3$ in our experiments. Figure 3(b) shows that by using this adaptive stopping condition, QUIC is not only efficient in the beginning, but also achieves quadratic convergence.

5.2.2 Line Search Parameters

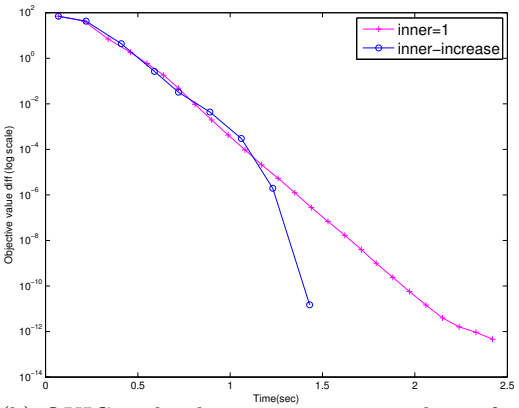
We demonstrate the robustness of QUIC to line search parameters σ and β. The results are shown in Figure 4.

5.2.3 Fixed/free Set Selection

To further demonstrate the power of *fixed/free* set selection, we use Hereditarybc data set as an example. In Figure 5, we plot the size of the free set versus the number of Newton iterations. Starting from a total of $1869^2 = 3,493,161$ variables, the size of the free set

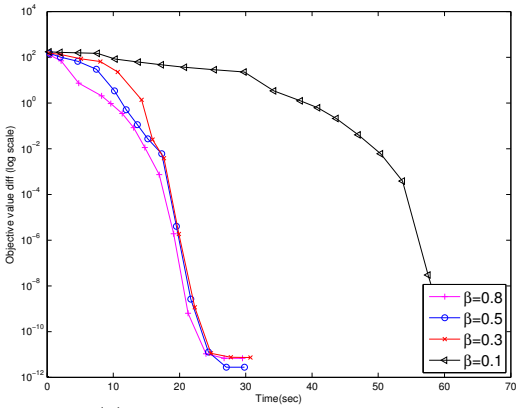


(a) QUIC with varying number of inner iterations (on ER data set).

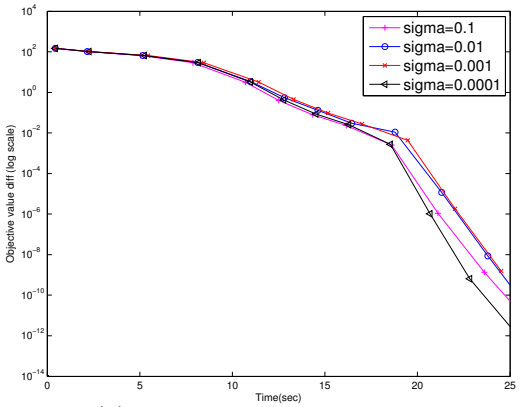


(b) QUIC with adaptive stopping condition for inner iterations.

Figure 3: The behavior of QUIC when varying the number of inner iterations. Figure 3(a) show that QUIC with one inner iteration converges faster in the beginning but eventually achieves just linear convergence, while QUIC with 20 inner iterations converges slower in the beginning, but has quadratic convergence. Figure 3(b) shows that by adaptively setting the number of iterations in QUIC, we get the advantages of both cases. Notice that each marker in the figures indicates one iteration.

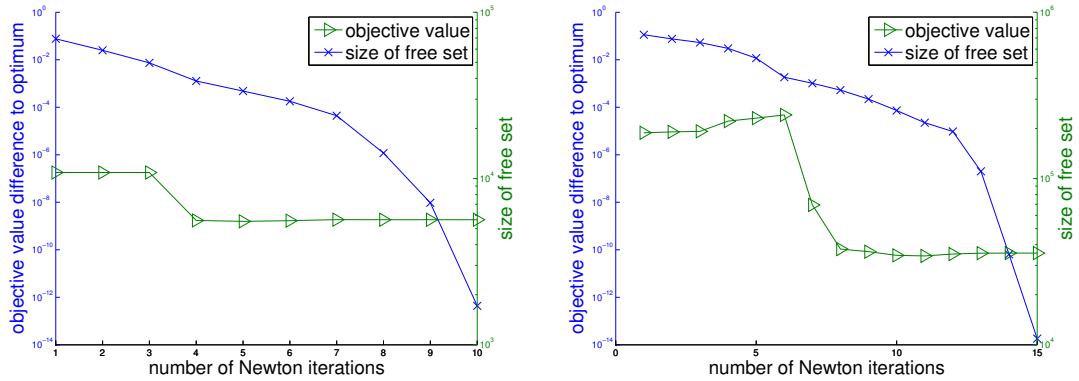


(a) Comparison on different β



(b) Comparison on different σ

Figure 4: Comparison of different line search parameters on the Leukemia data set. Figure 4(a) shows that QUIC is robust to a wide range of β values, but becomes slower when β is too small. Figure 4(b) demonstrates that QUIC is robust with respect to σ.



(a) ER data set (b) Hereditarybc data set

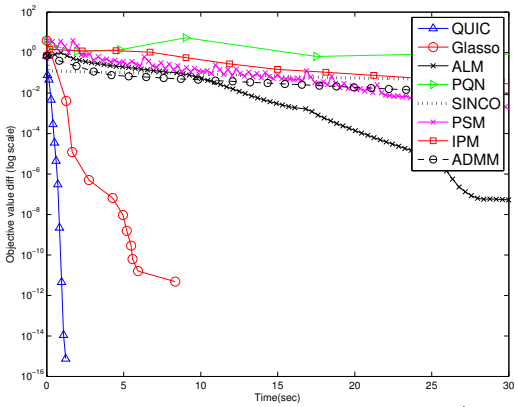
Figure 5: Size of free sets and objective value versus iterations. For both data sets, the sizes of free sets are always less than $6\|X^*\|_0$ when running QUIC algorithm.

progressively drops, in fact to less than $120,000$ in the very first iteration. We can see the super-linear convergence of QUIC even more clearly when we plot it against the number of iterations.

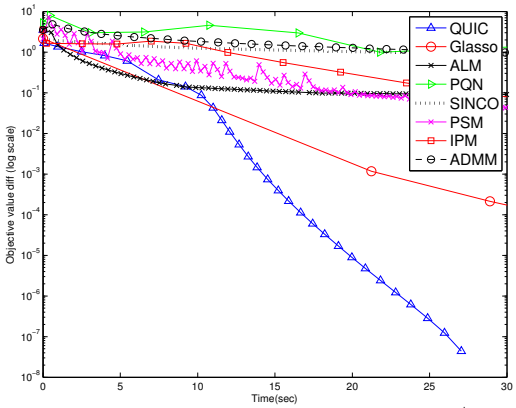
We further analyze the proportion of time taken by the two main steps of QUIC: coordinate descent updates and line search procedure (in line search, the most time intensive computation is the Cholesky factorization). We have looked at the ratio of time consumed by those two steps on different data sets. We found that when the size of the *free* set is large, QUIC will spend most of its time on coordinate updates. For example, in the Hereditarybc data set with $\lambda = 0.5$, where the size of free set is $0.11p^2$ in the beginning, coordinate descent takes 85.1% of the total run time, while line search only takes 14.9% of the total run time. In contrast, for data sets with small size of *free* set, coordinate descent updates will take noticeable less time. For example, when running on the ER data set with $\lambda = 0.5$, where the size of free set is $0.033p^2$ in the beginning, 59.6% of the total time is spent on coordinate descent.

5.2.4 Block-diagonal Structure

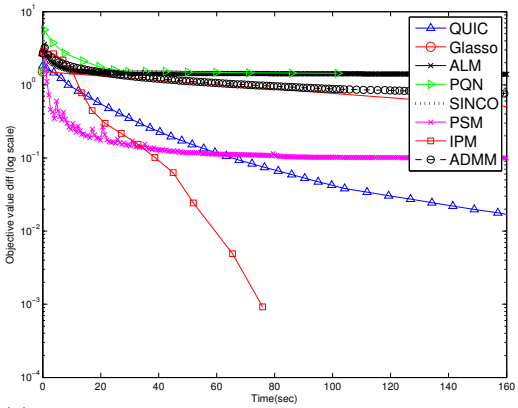
As discussed earlier, Mazumder and Hastie (2012); Witten et al. (2011) showed that when the thresholded covariance matrix $E = \max(|S| - \lambda, 0)$ is block-diagonal, then the problem can be naturally decomposed into sub-problems. This observation has been implemented in the latest version of glasso. In the end of Section 3, we showed that the *fixed/free* set selection can automatically identify the block-diagonal structure of the thresholded matrix, and thus QUIC can benefit from block-diagonal structure even when we do not explicitly decompose the matrix in the beginning. In the following experiment we show that with input sample covariance S with block-diagonal structure represented by E (see Section 3.4), QUIC still outperforms glasso. Moreover, we show that when some off-diagonal elements are added into the problem, while QUIC is still efficient because of its *fixed/free* set selection, glasso on the other hand suddenly becomes much slower.



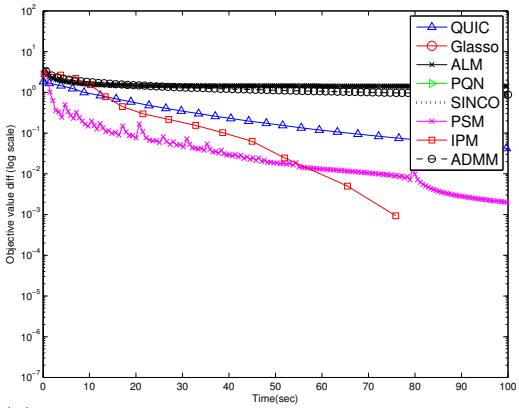
(a) Time taken on ER data set, $\lambda = 0.5$, $\frac{\|X^*\|_0}{p^2} = 0.0222$



(b) Time taken on ER data set, $\lambda = 0.1$, $\frac{\|X^*\|_0}{p^2} = 0.0724$



(c) Time taken on ER data set, $\lambda = 0.05$, $\frac{\|X^*\|_0}{p^2} = 0.125$



(d) Time taken on ER data set, $\lambda = 0.01$, $\frac{\|X^*\|_0}{p^2} = 0.322$

Figure 6: Comparison of algorithms on the ER data set ($p = 692$) under different λ. The results show that QUIC converges faster for larger λ where solutions are sparse, while IPM and PSM are faster for smaller λ which produces denser solutions. Note that each marker in the figures indicates one iteration, and that all the results are averaged over 5 replicated runs.

We generate synthetic data with block-diagonal structure as follows. We generate a sparse 150×150 inverse covariance matrix $\bar{\Theta}$ as discussed in Section 5.1.1, and then replicate $\bar{\Theta}$ eight times on the diagonal blocks to form a 1200×1200 block-diagonal matrix. Using this inverse covariance matrix to generate samples, we compare the following methods:

- QUIC: our proposed algorithm.
- GLASSO: In the latest version of GLASSO, the matrix is first decomposed into connected components based on the thresholded covariance matrix $\max(|S| - \lambda)$, and then each sub-problem is solved individually.

We then test the two algorithms for regularization parameter λ taking values from the set $\{0.017, \ldots, 0.011\}$. When $\lambda = 0.017$, the thresholded covariance matrix E has eight blocks, while when $\lambda = 0.011$ the block structure reduces to a single block. For each single λ trial, we compare the time taken by QUIC and GLASSO to achieve $(f(X_t) - f(X^*))/f(X^*) < 10^{-5}$. Figure 7 shows the experimental results. We can see that both methods are very fast for the case where the problem can be decomposed into 8 sub-problems (large λ); however, when we slightly increase λ so that there is only 1 connected component, QUIC is much faster than GLASSO. This is because even for the non-decomposable case, QUIC can still keep most of the elements of the very sparse off-diagonal blocks in the fixed set to speed up the process, while GLASSO cannot benefit from this sparse structure.

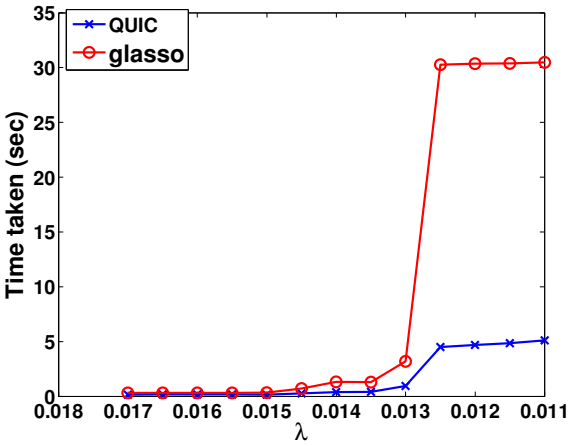


Figure 7: In this figure, we show the performance of QUIC and GLASSO for a sparse synthetic data with clustered structure. Using the same input covariance matrix S, we test the time for each algorithm to achieve $(f(X_t) - f(X^*))/f(X^*) < 10^{-5}$ under various values of λ. When $\lambda = 0.017$, the problem can be decomposed into 8 sub-problems, while when $\lambda = 0.011$ there is only one connected component. We can see that for the smaller values of λ, QUIC's approach of *free*/*fixed* set selection is able to exploit the sparsity structure of the solution, while GLASSO's training time increases dramatically.

Acknowledgements

This research was supported by NSF grant IIS-1018426. ISD also acknowledges support from the Moncrief Grand Challenge Award. PR also acknowledges support from NSF IIS-1149803. We would like to thank Kim-Chuan Toh for providing data sets and the IPM code as well as Katya Scheinberg and Shiqian Ma for providing the ALM implementation.

References

A. Agarwal, S. Negahban, and M. Wainwright. Convergence rates of gradient methods for high-dimensional statistical recovery. In *Advances in Neural Information Processing Systems*, 2010.

T. M. Apostol. *Mathematical Analysis*. Addison-Wesley, second edition, 1974.

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, 2008.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 7th printing edition, 2009.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Matlab scripts for alternating direction method of multipliers. `http://www.stanford.edu/~boyd/papers/admm/`, 2012.

A. d'Aspremont, O. Banerjee, and L. El Ghaoui. First-order methods for sparse covariance selection. *SIAM Journal on Matrix Analysis and its Applications*, 30(1):56–66, 2008.

I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

J. Duchi, S. Gould, and D. Koller. Projected subgradient methods for learning sparse Gaussians. In *Conference on Uncertainty in Artificial Intelligence*, 2008.

J.C. Dunn. Newton's method and the Goldstein step-length rule for constrained minimization problems. *SIAM J. Control and Optimization*, 18(6):659–674, 1980.

J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1(2):302–332, 2007.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, July 2008.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

C.-J. 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*. 2011.

C.-J. Hsieh, I. S. Dhillon, P. Ravikumar, and A. Banerjee. A divide-and-conquer method for sparse inverse covariance estimation. In *Advances in Neural Information Processing Systems*, 2012.

C. Lam and J. Fan. Sparsistency and rates of convergence in large covariance matrix estimation. *Annals of Statistics*, 37:4254–4278, 2009.

J. D. Lee, Y. Sun, and M. A. Saunders. Proximal Newton-type methods for convex optimization. In *Advances in Neural Information Processing Systems*, 2012.

E. S. Levitin and B. T. Polyak. Constrained minimization methods. *U.S.S.R. Computational Math. and Math. Phys.*, 6:1–50, 1966.

L. Li and K.-C. Toh. An inexact interior point method for L1-regularized sparse covariance selection. *Mathematical Programming Computation*, 2:291–315, 2010.

Z. Lu. Smooth optimization approach for sparse covariance selection. *SIAM Journal of Optimization*, 19:1807–1827, 2009.

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.

L. Meier, S. Van de Geer, and P. Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society, Series B*, 70:53–71, 2008.

P. Olsen, F. Oztoprak, J. Nocedal, and S. Rennie. Newton-like methods for sparse inverse covariance estimation. Technical report, Optimization Center, Northwestern University, 2012. URL `http://www.optimization-online.org/DB_HTML/2012/06/3506.html`.

B. T. Polyak. The conjugate gradient method in extremal problems. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 9:94–112, 1969.

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.

A. J. Rothman, P.J. Bickel, E. Levina, and J. Zhu. Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2:494–515, 2008.

K. Scheinberg and I. Rish. Learning sparse Gaussian Markov networks using a greedy coordinate ascent approach. In *Machine Learning and Knowledge Discovery in Databases*, volume 6323 of *Lecture Notes in Computer Science*, pages 196–212. Springer Berlin / Heidelberg, 2010.

K. Scheinberg, S. Ma, and D. Goldfarb. Sparse inverse covariance selection via alternating linearization methods. In *Advances in Neural Information Processing Systems*, 2010.

M. Schmidt. *Graphical Model Structure Learning with l1-regularization*. PhD thesis, University of British Columbia, 2010.

M. Schmidt, E. Van Den Berg, M. P. Friedl, and K. 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.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58:267–288, 1996.

P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117:387–423, 2007.

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.

T. T. Wu and K. Lange. Coordinate descent algorithms for lasso penalized regression. *Annals of Applied Statistics*, 2(1):224–244, 2008.

G.-X. Yuan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. A comparison of optimization methods and software for large-scale L1-regularized linear classification. *Journal of Machine Learning Research*, 11:3183–3234, 2010.

G.-X. Yuan, C.-H. Ho, and C.-J. Lin. An improved GLMNET for L1-regularized logistic regression. *Journal of Machine Learning Research*, 13:1999–2030, 2012.

M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94:19–35, 2007.

S. Yun and K.-C. Toh. A coordinate gradient descent method for L1-regularized convex minimization. *Computational Optimizations and Applications*, 48(2):273–307, 2011.

Journal of Machine Learning Research 15 (2014) 2949-2980 Submitted 6/13; Revised 3/14; Published 9/14

Multimodal Learning with Deep Boltzmann Machines

Nitish Srivastava NITISH@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
10 Kings College Road, Rm 3302
Toronto, Ontario, M5S 3G4, Canada.

Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Department of Statistics and Computer Science
University of Toronto
10 Kings College Road, Rm 3302
Toronto, Ontario, M5S 3G4, Canada.

Editor: Aapo Hyvarinen

Abstract

Data often consists of multiple diverse modalities. For example, images are tagged with textual information and videos are accompanied by audio. Each modality is characterized by having distinct statistical properties. We propose a Deep Boltzmann Machine for learning a generative model of such multimodal data. We show that the model can be used to create fused representations by combining features across modalities. These learned representations are useful for classification and information retrieval. By sampling from the conditional distributions over each data modality, it is possible to create these representations even when some data modalities are missing. We conduct experiments on bi-modal image-text and audio-video data. The fused representation achieves good classification results on the MIR-Flickr data set matching or outperforming other deep models as well as SVM based models that use Multiple Kernel Learning. We further demonstrate that this multimodal model helps classification and retrieval even when only unimodal data is available at test time.

Keywords: Boltzmann machines, unsupervised learning, multimodal learning, neural networks, deep learning

1. Introduction

Information in the real world comes through multiple input channels. Images are associated with captions and tags, videos contain visual and audio signals, sensory perception includes simultaneous inputs from visual, auditory, motor and haptic pathways. Each modality is characterized by very distinct statistical properties which makes it difficult to disregard the fact that they come from different input channels. Useful representations can potentially be learned for such data by combining the modalities into a joint representation that captures the real-world concept that the data corresponds to. For example, we would like a probabilistic model to correlate the occurrence of the words 'oak tree' and the visual properties of an image of an oak tree and represent them jointly, so that the model assigns high probability to one conditioned on the other.

Before we describe our model in detail, it is useful to understand why building such models is important. Different modalities can act like soft labels for each other. For example, consider bi-modal image-text data. If the same uncommon word was used in the context of several images, then there is some chance that all those images represent the same object. Conversely, if different words are used to describe similar looking images, then those words might mean the same thing. In other words, one modality might be somewhat invariant to large changes in another modality. This provides a rich learning signal. Moreover, different modalities typically carry different kinds of information. For example, people often caption an image to say things that may not be obvious from the image itself, such as the name of the person, place, or a particular object in the picture. Unless we do multimodal learning, it would not be possible to discover a lot of useful information about the world. We cannot afford to have discriminative models for every single concept. It would be useful, or at least elegant, to extract this information from unlabelled data.

In a multimodal setting, data consists of multiple modes, each modality having a different kind of representation and correlational structure. For example, text is usually represented as discrete sparse word count vectors, whereas an image is represented using pixel intensities or outputs of feature extractors which are real-valued and dense. Having very different statistical properties makes it much harder to discover relationships across modalities than relationships among features in the same modality. There is a lot of structure in the data but it is difficult to discover the highly non-linear relationships that exist between low-level features across different modalities. Moreover, the data is typically very noisy and there may be missing values.

A good multimodal learning model must satisfy certain properties. The joint representation must be such that similarity in the representation space implies similarity of the corresponding concepts so that the representation is useful for classification and retrieval. It is also desirable that the joint representation be easy to obtain even in the absence of some modalities. It should also be possible to fill-in missing modalities given the observed ones.

Our proposed multimodal Deep Boltzmann Machine (DBM) model satisfies the above desiderata. DBMs (Salakhutdinov and Hinton, 2009b) are undirected graphical models with bipartite connections between adjacent layers of hidden units. The key idea is to learn a joint density model over the space of multimodal inputs. Missing modalities can then be filled-in by sampling from the conditional distributions over them given the observed ones. For example, we use a large collection of user-tagged images to learn a joint distribution over images and text $P(\mathbf{v}_{img}, \mathbf{v}_{txt}; \theta)$. By drawing samples from $P(\mathbf{v}_{txt}|\mathbf{v}_{img}; \theta)$ and from $P(\mathbf{v}_{img}|\mathbf{v}_{txt}; \theta)$ we can fill-in missing data, thereby doing image annotation and image retrieval respectively, some of examples of which are shown in Figure 1.

There have been several approaches to learning from multimodal data. In particular, Huiskes et al. (2010) showed that using captions or tags, in addition to standard low-level image features significantly improves classification accuracy of Support Vector Machines (SVM) and Linear Discriminant Analysis (LDA) models. A similar approach of Guillaumin et al. (2010) based on the multiple kernel learning framework further demonstrated that an additional text modality can improve the accuracy of SVMs on various object recognition tasks. However, all of these approaches are discriminative by nature and cannot make use of large amounts of unlabelled data or deal easily with missing input modalities.

Image	Given Tags	Generated Tags	Input Tags	Nearest neighbors to generated image features
	pentax, k10d, kangarooisland, southaustralia, sa, 300mm, australia, australiansealion	beach, sea, surf, strand, shore, wave, seascape, sand, ocean, waves	nature, hill, scenery, green, clouds	 
	< no text >	night, lights, christmas, nightshot, nacht, nuit, notte, longexposure, noche, nocturna	flower, nature, green, flowers, petal, petals, bud	 
	aheram, 0505, sarahc, moo	portrait, bw, balckandwhite, people, faces, girl, blackwhite, person, man	blue, red, art, artwork, painted, paint, artistic, surreal, gallery, bleu	 
	unseulpixel, naturey crap	fall, autumn, trees, leaves, foliage, forest, woods, branches, path	bw, blackandwhite, noiretblanc, bianconero, blancoynegro	 

Figure 1: **Left:** Examples of text generated from a Deep Boltzmann Machine by sampling from $P(\mathbf{v}_{txt}|\mathbf{v}_{img};\theta)$. **Right:** Examples of images retrieved using features generated from a Deep Boltzmann Machine by sampling from $P(\mathbf{v}_{img}|\mathbf{v}_{txt};\theta)$.

On the generative side, Xing et al. (2005) used dual-wing harmoniums to build a joint model of images and text, which can be viewed as a linear Restricted Boltzmann Machine (RBM) model with Gaussian hidden units together with Gaussian and Poisson visible units. However, various data modalities will typically have very different statistical properties which makes it difficult to model them using shallow models. Most similar to our work is the recent approach of Ngiam et al. (2011) that used a deep autoencoder for speech and vision fusion. There are, however, several crucial differences. First, in this work we focus on jointly modelling very different data modalities: sparse word count vectors and real-valued dense image features. Second, we use a Deep Boltzmann Machine which is a probabilistic generative model as opposed to a feed-forward autoencoder. While both approaches have led to interesting results in several domains, using a generative model is important for applications we consider in this paper, as it allows our model to naturally handle missing and noisy data modalities.

2. Background: RBMs and Their Generalizations

Restricted Boltzmann Machines (RBMs) have been used effectively in modeling distributions over binary-valued data. Boltzmann machine models and their generalizations to exponential family distributions (Welling et al., 2005) have been successfully used in many application domains. For example, the Replicated Softmax model (Salakhutdinov and Hinton, 2009a) has been shown to be effective in modeling sparse word count vectors, whereas

Gaussian RBMs have been used for modeling real-valued inputs for image classification, video action recognition, and speech recognition (Lee et al., 2009; Taylor et al., 2010; Mohamed et al., 2011). In this section we briefly review these models, as they will serve as building blocks for our multimodal model.

2.1 Restricted Boltzmann Machines

A Restricted Boltzmann Machine (Smolensky, 1986) is an undirected graphical model with stochastic visible variables $\mathbf{v} \in \{0,1\}^D$ and stochastic hidden variables $\mathbf{h} \in \{0,1\}^F$, with each visible variable connected to each hidden variable. The model defines the following energy function $E : \{0,1\}^D \times \{0,1\}^F \to \mathbb{R}$

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij} v_i h_j - \sum_{i=1}^{D} b_i v_i - \sum_{j=1}^{F} a_j h_j, \tag{1}$$

where $\theta = \{\mathbf{a}, \mathbf{b}, \mathbf{W}\}$ are the model parameters: W_{ij} represents the symmetric interaction term between visible unit i and hidden unit j; b_i and a_j are bias terms. The joint distribution over the visible and hidden units is defined by

$$P(\mathbf{v}, \mathbf{h}; \theta) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(-E(\mathbf{v}, \mathbf{h}; \theta)\right), \quad \mathcal{Z}(\theta) = \sum_{\mathbf{v}}\sum_{\mathbf{h}} \exp\left(-E(\mathbf{v}, \mathbf{h}; \theta)\right), \tag{2}$$

where $\mathcal{Z}(\theta)$ is the normalizing constant. The conditional distributions over hidden $\mathbf{h}$ and visible $\mathbf{v}$ vectors factorize and can be easily derived from Equations 1, 2 as

$$\begin{aligned} P(\mathbf{h}|\mathbf{v}; \theta) &= \prod_{j=1}^{F} p(h_j|\mathbf{v}), \quad \text{with} \quad p(h_j = 1|\mathbf{v}) = g\left(\sum_{i=1}^{D} W_{ij} v_i + a_j\right), \\ P(\mathbf{v}|\mathbf{h}; \theta) &= \prod_{i=1}^{D} p(v_i|\mathbf{h}), \quad \text{with} \quad p(v_i = 1|\mathbf{h}) = g\left(\sum_{j=1}^{F} W_{ij} h_j + b_i\right), \end{aligned}$$

where $g(x) = 1/(1+\exp(-x))$ is the logistic function. Given a set of observations $\{\mathbf{v}_n\}_{n=1}^{N}$, the derivative of the log-likelihood with respect to the model parameters can be obtained from Equation 2 as

$$\frac{1}{N}\sum_{n=1}^{N} \frac{\partial \log P(\mathbf{v_n}; \theta)}{\partial W_{ij}} = \mathbb{E}_{P_{\text{data}}}\left[v_i h_j\right] - \mathbb{E}_{P_{\text{model}}}\left[v_i h_j\right],$$

where $\mathbb{E}_{P_{\text{data}}}[\cdot]$ denotes an expectation with respect to the data distribution $P_{\text{data}}(\mathbf{h}, \mathbf{v}; \theta) = P(\mathbf{h}|\mathbf{v}; \theta) P_{\text{data}}(\mathbf{v})$, with $P_{\text{data}}(\mathbf{v}) = \frac{1}{N}\sum_n \delta(\mathbf{v} - \mathbf{v}_n)$ representing the empirical distribution, and $\mathbb{E}_{P_{\text{model}}}[\cdot]$ is an expectation with respect to the distribution defined by the model, as in Equation 2. We will sometimes refer to $\mathbb{E}_{P_{\text{data}}}[\cdot]$ as the *data-dependent expectation*, and $\mathbb{E}_{P_{\text{model}}}[\cdot]$ as the *model's expectation*.

2.2 Gaussian-Bernoulli RBM

RBMs were originally developed for modeling binary vectors. Gaussian-Bernoulli RBMs (Freund and Haussler, 1994; Hinton and Salakhutdinov, 2006) are a variant that can be used for modeling real-valued vectors such as pixel intensities and filter responses. Consider modeling visible real-valued units $\mathbf{v} \in \mathbb{R}^D$, and let $\mathbf{h} \in \{0,1\}^F$ be binary stochastic hidden units. The energy of the state $\{\mathbf{v}, \mathbf{h}\}$ of the Gaussian-Bernoulli RBM is defined as

$$E(\mathbf{v}, \mathbf{h}; \theta) = \sum_{i=1}^{D} \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{i=1}^{D}\sum_{j=1}^{F} \frac{v_i}{\sigma_i} W_{ij} h_j - \sum_{j=1}^{F} a_j h_j,$$

where $\theta = \{\mathbf{a}, \mathbf{b}, \mathbf{W}, \boldsymbol{\sigma}\}$ are the model parameters. The density that the model assigns to a visible vector $\mathbf{v}$ is given by

$$P(\mathbf{v}; \theta) = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}} \exp\left(-E(\mathbf{v}, \mathbf{h}; \theta)\right), \quad \mathcal{Z}(\theta) = \int_{\mathbf{v}} \sum_{\mathbf{h}} \exp\left(-E(\mathbf{v}, \mathbf{h}; \theta)\right) d\mathbf{v}.$$

Similar to the standard RBMs, the conditional distributions factorize as

$$\begin{aligned} P(\mathbf{h}|\mathbf{v}; \theta) &= \prod_{j=1}^{F} p(h_j|\mathbf{v}), \quad \text{with} \quad p(h_j = 1|\mathbf{v}) = g\left(\sum_{i=1}^{D} W_{ij} \frac{v_i}{\sigma_i} + a_j\right), \\ P(\mathbf{v}|\mathbf{h}; \theta) &= \prod_{i=1}^{D} p(v_i|\mathbf{h}), \quad \text{with} \quad v_i|\mathbf{h} \sim \mathcal{N}\left(b_i + \sigma_i \sum_{j=1}^{F} W_{ij} h_j, \sigma_i^2\right), \end{aligned} \tag{3}$$

where $\mathcal{N}(\mu, \sigma^2)$ denotes a Gaussian distribution with mean μ and variance σ^2. Observe that conditioned on the states of the hidden units (Equation 3), each visible unit is modeled by a Gaussian distribution, whose mean is shifted by the weighted combination of the hidden unit activations.

Given a set of observations $\{\mathbf{v}_n\}_{n=1}^N$, the derivative of the log-likelihood with respect to the model parameters takes a very similar form when compared to binary RBMs.

$$\frac{1}{N} \sum_{n=1}^{N} \frac{\partial \log P(\mathbf{v_n}; \theta)}{\partial W_{ij}} = \mathbb{E}_{P_{\text{Data}}}\left[\frac{v_i}{\sigma_i} h_j\right] - \mathbb{E}_{P_{\text{model}}}\left[\frac{v_i}{\sigma_i} h_j\right].$$

2.3 Replicated Softmax Model

The Replicated Softmax Model is useful for modeling sparse count data, such as word count vectors in a document (Salakhutdinov and Hinton, 2009a). This model is a type of Restricted Boltzmann Machine in which the visible variables, that are usually binary, have been replaced by multinomial one of a number of different states. Specifically, let K be the dictionary size, M be the number of words appearing in a document, and $\mathbf{h} \in \{0,1\}^F$ be binary stochastic hidden topic features. Let $\mathbf{V}$ be a $M \times K$ observed binary matrix with $v_{ik} = 1$ iff the multinomial visible unit i takes on k^{th} value (meaning the i^{th} word in the document is the k^{th} dictionary word). The energy of the state $\{\mathbf{V}, \mathbf{h}\}$ can be defined as

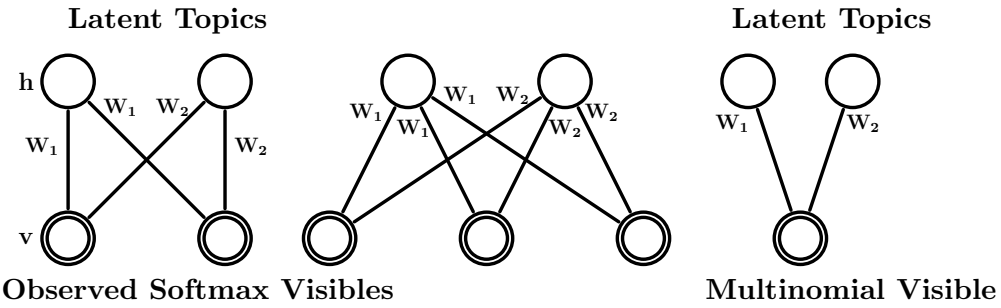


Figure 2: Replicated Softmax model. The top layer represents a vector $\mathbf{h}$ of stochastic, binary topic features and and the bottom layer represents softmax visible units $\mathbf{v}$. All visible units share the same set of weights, connecting them to binary hidden units. **Left:** The model for a document containing two and three words. **Right:** A different interpretation of the Replicated Softmax model, in which M softmax units with identical weights are replaced by a single multinomial unit which is sampled M times.

$$E(\mathbf{V},\mathbf{h}) = -\sum_{i=1}^{M}\sum_{j=1}^{F}\sum_{k=1}^{K} W_{ijk} v_{ik} h_j - \sum_{i=1}^{M}\sum_{k=1}^{K} b_{ik} v_{ik} - \sum_{j=1}^{F} a_j h_j,$$

where $\{\mathbf{W},\mathbf{a},\mathbf{b}\}$ are the model parameters: W_{ijk} is a symmetric interaction term between visible unit i that takes on value k, and hidden feature j, b_{ik} is the bias of unit i that takes on value k, and a_j is the bias of hidden feature j. The probability that the model assigns to a visible binary matrix $\mathbf{V}$ is

$$P(\mathbf{V},\mathbf{h};\theta) = \frac{1}{\mathcal{Z}(\theta)}\exp\left(-E(\mathbf{V},\mathbf{h};\theta)\right), \quad \mathcal{Z}(\theta) = \sum_{\mathbf{V}}\sum_{\mathbf{h}}\exp\left(-E(\mathbf{V},\mathbf{h};\theta)\right).$$

The key assumption of the Replicated Softmax model is that for each document we create a separate RBM with as many softmax units as there are words in the document, as shown in Figure 2. Assuming that the order of the words can be ignored, all of these softmax units can share the same set of weights, connecting them to binary hidden units. In this case, the energy of the state $\{\mathbf{V},\mathbf{h}\}$ for a document that contains M words is defined as

$$E(\mathbf{V},\mathbf{h}) = -\sum_{j=1}^{F}\sum_{k=1}^{K} W_{jk}\hat{v}_k h_j - \sum_{k=1}^{K} b_k \hat{v}_k - M\sum_{j=1}^{F} a_j h_j,$$

where $\hat{v}_k = \sum_{i=1}^{M} v_{ik}$ denotes the count for the k^{th} word. Observe that the bias terms of the hidden variables are scaled up by the length of the document. This scaling is important as it allows hidden units to behave sensibly when dealing with documents of different lengths. In the absence of bias scaling, the scale of the weights would get adjusted to work optimally for a typical document length. Documents longer than this would tend to saturate the units

Input	Reconstruction
chocolate, cake	cake, chocolate, sweets, dessert, cupcake, food, sugar, cream, birthday
nyc	nyc, newyork, brooklyn, queens, gothamist, manhattan, subway, streetart
dog	dog, puppy, perro, dogs, pet, filmshots, tongue, pets, nose, animal
flower, high, 花	flower, 花, high, japan, sakura, 日本, blossom, tokyo, lily, cherry
girl, rain, station, norway	norway, station, rain, girl, oslo, train, umbrella, wet, railway, weather
fun, life, children	children, fun, life, kids, child, playing, boys, kid, play, love
forest, blur	forest, blur, woods, motion, trees, movement, path, trail, green, focus
españa, agua, granada	españa, agua, spain, granada, water, andalucía, naturaleza, galicia, nieve

Table 1: Some examples of one-step reconstruction from the Replicated Softmax Model.

and shorter than this would lead to vague activations of the hidden units. The conditional distributions are given by

$$p(h_j = 1|\mathbf{V}) \;=\; g\left(Ma_j + \sum_{k=1}^{K} \hat{v}_k W_{jk}\right), \tag{4}$$

$$p(v_{ik} = 1|\mathbf{h}) \;=\; \frac{\exp\left(b_k + \sum_{j=1}^{F} h_j W_{jk}\right)}{\sum_{q=1}^{K} \exp\left(b_q + \sum_{j=1}^{F} h_j W_{jq}\right)}. \tag{5}$$

A pleasing property of using softmax units is that the mathematics underlying the learning algorithm for binary RBMs remains the same. Given a collection of N documents $\{\mathbf{V}_n\}_{n=1}^{N}$, the derivative of the log-likelihood with respect to parameters W is

$$\frac{1}{N}\sum_{n=1}^{N} \frac{\partial \log P(\mathbf{V_n})}{\partial W_{jk}} \;=\; \mathbb{E}_{P_{\text{data}}}\left[\hat{v}_k h_j\right] - \mathbb{E}_{P_{\text{model}}}\left[\hat{v}_k h_j\right].$$

The Replicated Softmax model can also be interpreted as an RBM model that uses a single visible multinomial unit with support $\{1, ..., K\}$ which is sampled M times (see Figure 2, right panel).

For all of the above models, exact maximum likelihood learning is intractable because exact computation of the expectation $\mathbb{E}_{P_{\text{model}}}[\cdot]$ takes time that is exponential in $\min\{D, F\}$, i.e the number of visible or hidden units. In practice, efficient learning is performed by following an approximation to the gradient of a different objective function, called the "Contrastive Divergence" (CD) (Hinton, 2002).

One way to illustrate the working of the model is to look at one-step reconstructions of some bags of words. Table 1 shows some examples. The words in the left column were given as input to the model. Then Equation 4 was used to compute a distribution over hidden units. Using these probabilities as states of the units, Equation 5 was used to obtain a distribution over words. The second column shows the words with the highest probability in that distribution. This model was trained using text from the MIR-Flickr data set which we use later in our experiments. We can see that the model has learned a basic understanding of which words are probable given the input words. For example, *chocolate, cake* generalizes to *sweets, desserts, food*, etc. The model often makes interesting inferences. For example, *girl, rain, station, norway* extends to *oslo, train, wet, umbrella, railway*, which are very plausible in that context. The model also captures regularities about language,

discovers synonyms across multiple languages and learns about geographical relationships. This shows that the hidden units can capture these regularities and represent them as binary features.

3. Deep Boltzmann Machines (DBMs)

A Deep Boltzmann Machine (Salakhutdinov and Hinton, 2009b) is a network of symmetrically coupled stochastic binary units. It contains a set of visible units $\mathbf{v} \in \{0,1\}^D$, and a sequence of layers of hidden units $\mathbf{h}^{(1)} \in \{0,1\}^{F_1}$, $\mathbf{h}^{(2)} \in \{0,1\}^{F_2}$,..., $\mathbf{h}^{(L)} \in \{0,1\}^{F_L}$. There are connections only between hidden units in adjacent layers, as well as between visible and hidden units in the first hidden layer. Consider a DBM with three hidden layers[1] (i.e., $L = 3$). The energy of the joint configuration $\{\mathbf{v}, \mathbf{h}\}$ is defined as

$$\begin{aligned} E(\mathbf{v}, \mathbf{h}; \theta) \ = \ & -\sum_{i=1}^{D}\sum_{j=1}^{F_1} W_{ij}^{(1)} v_i h_j^{(1)} - \sum_{j=1}^{F_1}\sum_{l=1}^{F_2} W_{jl}^{(2)} h_j^{(1)} h_l^{(2)} - \sum_{l=1}^{F_2}\sum_{p=1}^{F_3} W_{lp}^{(3)} h_l^{(2)} h_p^{(3)} \\ & - \sum_{i=1}^{D} b_i v_i - \sum_{j=1}^{F_1} b_j^{(1)} h_j^{(1)} - \sum_{l=1}^{F_2} b_l^{(2)} h_l^{(2)} - \sum_{p=1}^{F_3} b_p^{(3)} h_p^{(3)}, \end{aligned}$$

where $\mathbf{h} = \{\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}\}$ is the set of hidden units and $\theta = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \mathbf{W}^{(3)}, \mathbf{b}, \mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \mathbf{b}^{(3)}\}$ is the set of model parameters, representing visible-to-hidden and hidden-to-hidden symmetric interaction terms, as well as bias terms. Biases are equivalent to weights on a connection to a unit whose state is fixed at 1. The probability that the model assigns to a visible vector $\mathbf{v}$ is given by the Boltzmann distribution

$$P(\mathbf{v}; \theta) = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}} \exp\left(-E(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}; \theta)\right).$$

Observe that setting both $\mathbf{W}^{(2)}$=0 and $\mathbf{W}^{(3)}$=0 recovers the simpler Restricted Boltzmann Machine (RBM) model. The derivative of the log-likelihood with respect to the model parameters takes the form

$$\frac{\partial \log P(\mathbf{v}; \theta)}{\partial \mathbf{W}^{(1)}} \ = \ \mathbb{E}_{P_{\text{data}}}[\mathbf{v}\mathbf{h}^{(1)\top}] - \mathbb{E}_{P_{\text{model}}}[\mathbf{v}\mathbf{h}^{(1)\top}]. \tag{6}$$

The derivatives with respect to parameters $\mathbf{W}^{(2)}$ and $\mathbf{W}^{(3)}$ take similar forms but instead involve the cross-products $\mathbf{h}^{(1)}\mathbf{h}^{(2)\top}$ and $\mathbf{h}^{(2)}\mathbf{h}^{(3)\top}$ respectively. Unlike RBMs, the conditional distribution over the states of the hidden variables conditioned on the data is no longer factorial. The exact computation of the data-dependent expectation takes time that is exponential in the number of hidden units, whereas the exact computation of the model's expectation takes time that is exponential in the number of hidden and visible units.

1. For clarity, we use three hidden layers. Extensions to models with more than three layers is straightforward.

Deep Boltzmann Machines (DBMs) are interesting for several reasons. Firstly, like Deep Belief Networks (Hinton et al., 2006), DBMs can discover several layers of increasingly complex representations of the input, use an efficient layer-by-layer pretraining procedure (Salakhutdinov and Hinton, 2009b), can be trained on unlabelled data and can be fine-tuned for a specific task using (possibly limited) labelled data. Secondly, unlike other models with deep architectures, the approximate inference procedure for DBMs incorporates a top-down feedback in addition to the usual bottom-up pass, allowing Deep Boltzmann Machines to better incorporate uncertainty about missing or noisy inputs. Thirdly, and perhaps most importantly, parameters of all layers can be optimized jointly by following the approximate gradient of a variational lower-bound on the likelihood objective. As we show in our experimental results, this greatly facilitates learning better generative models, particularly when modeling the multimodal data.

4. Multimodal Deep Boltzmann Machines

Let us first consider constructing a multimodal DBM using an image-text bi-modal DBM as our running example. Let $\mathbf{v}^m \in \mathbb{R}^D$ denote a real-valued image input and $\mathbf{v}^t \in \{1, ..., K\}$ denote an associated text input containing M words, with v^t_k denoting the count for the k^{th} word.

We start by modeling each data modality using a separate two-layer DBM. (see Figure 3). Let $\mathbf{h}^{(1m)} \in \{0,1\}^{F^m_1}$ and $\mathbf{h}^{(2m)} \in \{0,1\}^{F^m_2}$ denote the two layers of hidden units. The probability that the image-specific two-layer DBM assigns to a visible vector $\mathbf{v}^m$ is given by

$$
\begin{aligned}
P(\mathbf{v}^m;\theta^m) &= \sum_{\mathbf{h}^{(1m)},\mathbf{h}^{(2m)}} P(\mathbf{v}^m,\mathbf{h}^{(2m)},\mathbf{h}^{(1m)};\theta^m) \\
&= \frac{1}{\mathcal{Z}(\theta^m)} \sum_{\mathbf{h}^{(1m)},\mathbf{h}^{(2m)}} \exp\Bigg(-\sum_i \frac{(v^m_i - b^m_i)^2}{2\sigma^2_i} + \sum_{ij} \frac{v^m_i}{\sigma_i} W^{(1m)}_{ij} h^{(1m)}_j + \\
&\qquad \sum_{jl} W^{(2m)}_{jl} h^{(1m)}_j h^{(2m)}_l + \sum_j b^{(1m)}_j h^{(1m)}_j + \sum_l b^{(2m)}_l h^{(2m)}_l \Bigg).
\end{aligned}
$$

Note that conditioned on the states of $\mathbf{h}^{(1m)}$, the image-specific DBM uses Gaussian distribution to model the distribution over real-valued image features. Similarly, text-specific DBM uses Replicated Softmax to model the distribution over word count vectors. The corresponding probability that the text-specific two-layer DBM assigns to $\mathbf{v}^t$ is given by

$$
\begin{aligned}
P(\mathbf{v}^t;\theta^t) &= \sum_{\mathbf{h}^{(1t)},\mathbf{h}^{(2t)}} P(\mathbf{v}^t,\mathbf{h}^{(2t)},\mathbf{h}^{(1t)};\theta^t) \\
&= \frac{1}{\mathcal{Z}_M(\theta^t)} \sum_{\mathbf{h}^{(1t)},\mathbf{h}^{(2t)}} \exp\Bigg(\sum_{jk} W^{(1t)}_{k,j} h^{(1t)}_j v^t_k + \sum_{jl} W^{(2t)}_{jl} h^{(1t)}_j h^{(2t)}_l + \\
&\qquad \sum_k b^t_k v^t_k + M \sum_j b^{(1t)}_j h^{(1t)}_j + \sum_l b^{(2t)}_l h^{(2t)}_l \Bigg),
\end{aligned}
$$

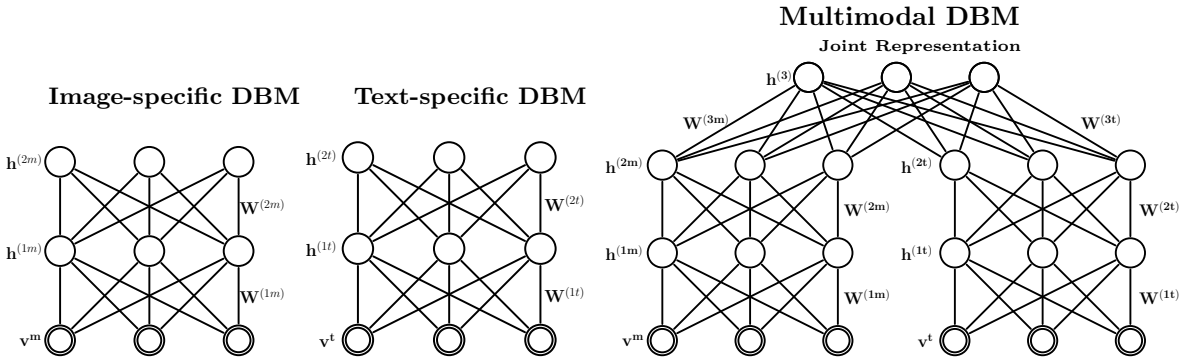


Figure 3: **Left:** Image-specific two-layer DBM that uses a Gaussian model to model the distribution over real-valued image features. **Middle:** Text-specific two-layer DBM that uses a Replicated Softmax model to model its distribution over the word count vectors. **Right:** A Multimodal DBM that models the joint distribution over image and text inputs. All layers but the first (bottom) layer use standard binary units.

where $\mathbf{h}^{(1t)} \in \{0,1\}^{F_1^t}$, $\mathbf{h}^{(2t)} \in \{0,1\}^{F_2^t}$ represent the two layers of hidden units.

To form a multimodal DBM, we combine the two models by adding an additional layer on top of them. The resulting graphical model is shown in Figure 3, right panel. The joint distribution over the multi-modal input, where $\mathbf{h} = \{\mathbf{h}^{(1m)}, \mathbf{h}^{(2m)}, \mathbf{h}^{(1t)}, \mathbf{h}^{(2t)}, \mathbf{h}^{(3)}\}$ denotes all hidden variables, can be written as

$$P(\mathbf{v}^m, \mathbf{v}^t; \theta) = \sum_{\mathbf{h}^{(2m)}, \mathbf{h}^{(2t)}, \mathbf{h}^{(3)}} P(\mathbf{h}^{(2m)}, \mathbf{h}^{(2t)}, \mathbf{h}^{(3)}) \Big(\sum_{\mathbf{h}^{(1m)}} P(\mathbf{v}_m, \mathbf{h}^{(1m)} | \mathbf{h}^{(2m)}) \Big) \Big(\sum_{\mathbf{h}^{(1t)}} P(\mathbf{v}^t, \mathbf{h}^{(1t)} | \mathbf{h}^{(2t)}) \Big)$$

$$= \frac{1}{\mathcal{Z}_M(\theta)} \sum_{\mathbf{h}} \exp \Bigg(\underbrace{\sum_{kj} W_{kj}^{(1t)} v_k^t h_j^{(1t)} + \sum_{jl} W_{jl}^{(2t)} h_j^{(1t)} h_l^{(2t)} + \sum_k b_k^t v_k^t + M \sum_j b_j^{(1t)} h_j^{(1t)} + \sum_l b_l^{(2t)} h_l^{(2t)}}_{\text{Replicated Softmax Text Pathway}}$$

$$\underbrace{- \sum_i \frac{(v_i^m - b_i^m)^2}{2\sigma_i^2} + \sum_{ij} \frac{v_i^m}{\sigma_i} W_{ij}^{(1m)} h_j^{(1m)} + \sum_{jl} W_{jl}^{(2m)} h_j^{(1m)} h_l^{(2m)} + \sum_j b_j^{(1m)} h_j^{(1m)} + \sum_l b_l^{(2m)} h_l^{(2m)}}_{\text{Gaussian Image Pathway}}$$

$$\underbrace{+ \sum_{lp} W^{(3t)} h_l^{(2t)} h_p^{(3)} + \sum_{lp} W^{(3m)} h_l^{(2m)} h_p^{(3)} + \sum_p b_p^{(3)} h_p^{(3)}}_{\text{Joint } 3^{rd} \text{ Layer}} \Bigg). \tag{7}$$

The normalizing constant depends on the number of words M in the corresponding document, since the low-level part of the text pathway contains as many softmax units as there are words in the document. Similar to the Replicated Softmax model, the multimodal DBM can be viewed as a family of different-sized DBMs that are created for documents of different lengths that share parameters.

The conditional distributions over the visible and the five sets of hidden units are given by

$$
\begin{aligned}
p(h_j^{(1m)}=1|\mathbf{v}^m,\mathbf{h}^{(2m)}) &= g\left(\sum_{i=1}^{D} W_{ij}^{(1m)}\frac{v_i^m}{\sigma_i}+\sum_{l=1}^{F_2^m} W_{jl}^{(2m)}h_l^{(2m)}+b_j^{(1m)}\right), \qquad (8)\\
p(h_l^{(2m)}=1|\mathbf{h}^{(1m)},\mathbf{h}^{(3)}) &= g\left(\sum_{j=1}^{F_1^m} W_{jl}^{(2m)}h_j^{(1m)}+\sum_{p=1}^{F_3} W_{lp}^{(3m)}h_p^{(3)}+b_l^{(2m)}\right),\\
p(h_j^{(1t)}=1|\mathbf{v}^t,\mathbf{h}^{(2t)}) &= g\left(\sum_{k=1}^{K} W_{kj}^{(1t)}v_k^t+\sum_{l=1}^{F_2^t} W_{jl}^{(2t)}h_l^{(2t)}+Mb_j^{(1t)}\right),\\
p(h_l^{(2t)}=1|\mathbf{h}^{(1t)},\mathbf{h}^{(3)}) &= g\left(\sum_{j=1}^{F_1^t} W_{jl}^{(2t)}h_j^{(1t)}+\sum_{p=1}^{F_3} W_{lp}^{(3t)}h_p^{(3)}+b_l^{(2t)}\right),\\
p(h_p^{(3)}=1|\mathbf{h}^{(2)}) &= g\left(\sum_{l=1}^{F_2^m} W_{lp}^{(3m)}h_l^{(2m)}+\sum_{l=1}^{F_2^t} W_{lp}^{(3t)}h_l^{(2t)}+b_p^{(3)}\right),\\
p(v_{ik}^t=1|\mathbf{h}^{(1t)}) &= \frac{\exp\left(\sum_{j=1}^{F_1^t} h_j^{(1t)}W_{jk}^{(1t)}+b_k^t\right)}{\sum_{q=1}^{K}\exp\left(\sum_{j=1}^{F_1^t} h_j^{(1t)}W_{jq}^{(1t)}+b_k^t\right)},\\
v_i^m|\mathbf{h}^{(1m)} &\sim \mathcal{N}\left(\sigma_i\sum_{j=1}^{F_1^m} W_{ij}^{(1m)}h_j^{(1m)}+b_i^m,\sigma_i^2\right).
\end{aligned}
$$

Extending multimodal DBMs to other data modalities requires a simple modification of the first-layer modules. For example, consider modelling video-audio bi-modal data. Unlike image-text data, video-audio data can be represented as a sequence of real-valued vector pairs. Let $\mathbf{v}^m \in \mathbb{R}^D$ denote a real-valued input from the video stream (e.g., several consecutive image frames), and $\mathbf{v}^a \in \mathbb{R}^D$ denote an associated audio input (e.g., corresponding consecutive audio frames). We can easily construct the corresponding multimodal DBM with both pathways using Gaussian RBMs as the first layer. Compared to Equation 7, the the joint distribution over the multi-modal input variables, can be written as[2]

$$
\begin{aligned}
P(\mathbf{v}^m,\mathbf{v}^a;\theta) = \frac{1}{\mathcal{Z}(\theta)}\sum_{\mathbf{h}}\exp\Bigg(&\underbrace{-\sum_i\frac{(v_i^m)^2}{2\sigma_i^2}+\sum_{ij}\frac{v_i^m}{\sigma_i}W_{ij}^{(1m)}h_j^{(1m)}+\sum_{jl}W_{jl}^{(2m)}h_j^{(1m)}h_l^{(2m)}}_{\text{Gaussian Video Pathway}} \qquad (9)\\
&\underbrace{-\sum_i\frac{(v_i^a)^2}{2\sigma_i^2}+\sum_{ij}\frac{v_i^a}{\sigma_i}W_{ij}^{(1a)}h_j^{(1a)}+\sum_{jl}W_{jl}^{(2a)}h_j^{(1a)}h_l^{(2a)}}_{\text{Gaussian Audio Pathway}}+\underbrace{\sum_{lp}W^{(3t)}h_l^{(2t)}h_p^{(3)}+\sum_{lp}W^{(3m)}h_l^{(2m)}h_p^{(3)}}_{\text{Joint } 3^{rd} \text{ Layer}}\Bigg).
\end{aligned}
$$

2. We omit the bias terms for the hidden layers for clarity of presentation.

4.1 Approximate Inference and Learning

Exact maximum likelihood learning in this model is intractable, but efficient approximate learning of DBMs can be carried out by using a variational approach, where mean-field inference is used to estimate data-dependent expectations and an MCMC based stochastic approximation procedure is used to approximate the model's expected sufficient statistics.

4.1.1 Estimating the Data-dependent Statistics

Consider any approximating distribution $Q(\mathbf{h}|\mathbf{v};\boldsymbol{\mu})$, parameterized by a vector of parameters $\boldsymbol{\mu}$, for the posterior $P(\mathbf{h}|\mathbf{v};\theta)$. Then the log-likelihood of the DBM model has the following variational lower bound,

$$\begin{aligned}\log P(\mathbf{v};\theta) &\geq \sum_{\mathbf{h}} Q(\mathbf{h}|\mathbf{v};\theta)\log P(\mathbf{v},\mathbf{h};\theta) + \mathcal{H}(Q) \\ &\geq \log P(\mathbf{v};\theta) - \mathrm{KL}(Q(\mathbf{h}|\mathbf{v};\boldsymbol{\mu})||P(\mathbf{h}|\mathbf{v};\theta)),\end{aligned} \tag{10}$$

where $\mathrm{KL}(Q||P)$ is the Kullback-–Leibler divergence between the two distributions, and $\mathcal{H}(\cdot)$ is the entropy functional. The bound becomes tight if and only if $Q(\mathbf{h}|\mathbf{v};\boldsymbol{\mu}) = P(\mathbf{h}|\mathbf{v};\theta)$.

We approximate the true posterior $P(\mathbf{h}|\mathbf{v};\theta)$, where $\mathbf{v} = \{\mathbf{v}^m, \mathbf{v}^t\}$, with a fully factorized approximating distribution over the five sets of hidden units $\{\mathbf{h}^{(1m)}, \mathbf{h}^{(2m)}, \mathbf{h}^{(1t)}, \mathbf{h}^{(2t)}, \mathbf{h}^{(3)}\}$:

$$Q(\mathbf{h}|\mathbf{v};\boldsymbol{\mu}) = \left(\prod_{j=1}^{F_1^m} q(h_j^{(1m)}|\mathbf{v}) \prod_{l=1}^{F_2^m} q(h_l^{(2m)}|\mathbf{v})\right)\left(\prod_{j=1}^{F_1^t} q(h_j^{(1t)}|\mathbf{v}) \prod_{l=1}^{F_2^t} q(h_l^{(2t)}|\mathbf{v})\right)\prod_{p=1}^{F_3} q(h_p^{(3)}|\mathbf{v}), \tag{11}$$

where $\boldsymbol{\mu} = \{\boldsymbol{\mu}^{(1m)}, \boldsymbol{\mu}^{(1t)}, \boldsymbol{\mu}^{(2m)}, \boldsymbol{\mu}^{(2t)}, \boldsymbol{\mu}^{(3)}\}$ are the mean-field parameters with $q(h_i^{(l)} = 1|\mathbf{v}) = \mu_i^{(l)}$ for $l = 1, 2, 3$.

For each training example, the variational bound of Equation 10 is maximized with respect to the variational parameters $\boldsymbol{\mu}$ for fixed parameters θ. This results in the following mean-field fixed-point equations

$$\begin{aligned}
\mu_j^{(1m)} &\leftarrow g\Big(\sum_{i=1}^{D} W_{ij}^{(1m)}\frac{v_i^m}{\sigma_i} + \sum_{l=1}^{F_2^m} W_{jl}^{(2m)}\mu_l^{(2m)}\Big), \quad \mu_l^{(2m)} \leftarrow g\Big(\sum_{j=1}^{F_1^m} W_{jl}^{(2m)}\mu_j^{(1m)} + \sum_{k=1}^{F_3} W_{lk}^{(3m)}\mu_k^{(3)}\Big), \\
\mu_j^{(1t)} &\leftarrow g\Big(\sum_{k=1}^{K} W_{kj}^{(1t)}v_k^t + \sum_{l=1}^{F_2^t} W_{jl}^{(2t)}\mu_l^{(2t)}\Big), \quad \mu_l^{(2t)} \leftarrow g\Big(\sum_{j=1}^{F_1^t} W_{jl}^{(2t)}\mu_j^{(1t)} + \sum_{k=1}^{F_3} W_{lk}^{(3t)}\mu_k^{(3)}\Big), \\
\mu_p^{(3)} &\leftarrow g\Big(\sum_{l=1}^{F_2^m} W_{lp}^{(3m)}\mu_l^{(2m)} + \sum_{l=1}^{F_2^t} W_{lp}^{(3t)}\mu_l^{(2t)}\Big),
\end{aligned} \tag{12}$$

where $g(x) = 1/(1+\exp(-x))$ is the logistic function. To solve these fixed-point equations, we simply cycle through layers, updating the mean-field parameters within a single layer. The variational parameters $\boldsymbol{\mu}$ are then used to compute the data-dependent statistics in Equation 6. For example,

$$\mathbb{E}_{P_{\text{data}}}[\mathbf{v}^m \mathbf{h}^{(1m)\top}] = \frac{1}{N}\sum_{n=1}^{N} \mathbf{v}_n^m \boldsymbol{\mu}_n^{(1m)\top}$$
$$\mathbb{E}_{P_{\text{data}}}[\mathbf{h}^{(1m)} \mathbf{h}^{(2m)\top}] = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\mu}_n^{(1m)} \boldsymbol{\mu}_n^{(2m)\top},$$

where the average on the RHS is over training cases. Note the close connection between the form of the mean-field fixed point updates and the form of the conditional distribution defined by Equation 8. In fact, implementing the mean-field updates requires no extra work beyond implementing the Gibbs sampler.

4.1.2 Estimating the Data-independent Statistics

Given the variational parameters $\boldsymbol{\mu}$, the model parameters θ are then updated to maximize the variational bound using an MCMC-based stochastic approximation (Salakhutdinov and Hinton, 2009b; Tieleman, 2008; Younes, 1998). Remember that in out setting, we are learning a whole family of different-sized DBMs that depend on the number of words, or the number of replicated softmax variables (see Equation 7). Let us first assume that the text input only contains a set of M words. Learning with stochastic approximation proceeds as follows. Let θ_t and $\mathbf{x}_t = \{\mathbf{v}_t^m, \mathbf{v}^t, \mathbf{h}_t^{(1m)}, \mathbf{h}_t^{(1t)}, \mathbf{h}_t^{(2m)}, \mathbf{h}_t^{(2t)}, \mathbf{h}_t^{(3)}\}$ be the current parameters and the state. Then $\mathbf{x}_t$ and θ_t are updated sequentially as follows:

- Given $\mathbf{x}_t$, sample a new state $\mathbf{x}_{t+1}$ from the transition operator $T_{\theta_t}(\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t)$ that leaves $P(\cdot; \theta_t)$ invariant. This can be accomplished by using Gibbs sampling (see Equation 8).
- A new parameter θ_{t+1} is then obtained by making a gradient step, where the intractable model's expectation $\mathbb{E}_{P_{\text{model}}}[\cdot]$ in the gradient is replaced by a point estimate at sample $\mathbf{x}_{t+1}$.

In practice, we typically maintain a set of S "persistent" Markov chains $X_t = \{\mathbf{x}_{t,1}, ..., \mathbf{x}_{t,S}\}$, and use an average over those particles.

The overall learning procedure for DBMs is summarized in Algorithm 1. Extensions to the variable text input is trivial. For each $m = 1, ..., M_{max}$, where M_{max} is the maximum number of words across all documents, we can create a corresponding multimodal DBM with m replicated softmax variables and shared parameters. For each model m, we simply maintain a set of S_m persistent Markov chains.[3] Learning then proceeds as discussed before.

Stochastic approximation provides asymptotic convergence guarantees and belongs to the general class of Robbins–Monro approximation algorithms (Robbins and Monro, 1951; Younes, 1998). Sufficient conditions that ensure almost sure convergence to an asymptotically stable point are given in Younes (1989, 1998); Yuille (2004). One necessary condition requires the learning rate to decrease with time, so that $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$.

3. Ideally, we would have each S_m be as large as computationally feasible. However, given a fixed budget for the total number of chains, we could choose S_m to be proportional to the number of documents containing m words.

Algorithm 1 Learning Procedure for a Multimodal Deep Boltzmann Machine.

Given: a training set of N data vectors $\mathbf{v}_n = \{\mathbf{v}_n^m, \mathbf{v}_n^t\}$, $n = 1, ..., N$, and S, the number of persistent Markov chains. Let Λ be a diagonal $D \times D$ matrix with $\Lambda_{ii} = 1/\sigma_i$.
Randomly initialize parameter vector θ_0 and S samples: $\{\tilde{\mathbf{v}}_{0,1}, \tilde{\mathbf{h}}_{0,1}\}, ..., \{\tilde{\mathbf{v}}_{0,S}, \tilde{\mathbf{h}}_{0,S}\}$, where we define $\tilde{\mathbf{h}} = \{\tilde{\mathbf{h}}^{(1m)}, \tilde{\mathbf{h}}^{(1t)}, \tilde{\mathbf{h}}^{(2m)}, \tilde{\mathbf{h}}^{(2t)}, \tilde{\mathbf{h}}^{(3)}\}$.
for $t = 0$ to T (number of iterations) **do**

  // Variational Inference:
  for each training example $\mathbf{v}_n$, $n = 1$ to N **do**
    Run the mean-field fixed-point updates until convergence using Equation 12.
    Set $\boldsymbol{\mu}_n = \boldsymbol{\mu}$.
  end for

  // Stochastic Approximation:
  for each sample $s = 1$ to S (number of persistent Markov chains) **do**
    Sample $(\tilde{\mathbf{v}}_{t+1,s}, \tilde{\mathbf{h}}_{t+1,s})$ given $(\tilde{\mathbf{v}}_{t,s}, \tilde{\mathbf{h}}_{t,s})$ by running a Gibbs sampler for one step using Equation 8.
  end for

  // Parameter Update:

  // Image Pathway:

$$\mathbf{W}_{t+1}^{(1m)} = \mathbf{W}_t^{(1m)} + \alpha_t \left(\frac{1}{N} \sum_{n=1}^{N} \mathbf{v}_n^m \Lambda (\boldsymbol{\mu}_n^{(1m)})^\top - \frac{1}{S} \sum_{s=1}^{S} \tilde{\mathbf{v}}_{t+1,s}^m \Lambda (\tilde{\mathbf{h}}_{t+1,s}^{(1m)})^\top \right).$$

$$\mathbf{W}_{t+1}^{(2m)} = \mathbf{W}_t^{(2m)} + \alpha_t \left(\frac{1}{N} \sum_{n=1}^{N} \boldsymbol{\mu}_n^{(1m)} (\boldsymbol{\mu}_n^{(2m)})^\top - \frac{1}{S} \sum_{s=1}^{S} \tilde{\mathbf{h}}_{t+1,s}^{(1m)} (\tilde{\mathbf{h}}_{t+1,s}^{(2m)})^\top \right).$$

  // Text Pathway:

$$\mathbf{W}_{t+1}^{(1t)} = \mathbf{W}_t^{(1t)} + \alpha_t \left(\frac{1}{N} \sum_{n=1}^{N} \mathbf{v}_n^t (\boldsymbol{\mu}_n^{(1t)})^\top - \frac{1}{S} \sum_{s=1}^{S} \tilde{\mathbf{v}}_{t+1,s}^t (\tilde{\mathbf{h}}_{t+1,s}^{(1t)})^\top \right).$$

$$\mathbf{W}_{t+1}^{(2t)} = \mathbf{W}_t^{(2t)} + \alpha_t \left(\frac{1}{N} \sum_{n=1}^{N} \boldsymbol{\mu}_n^{(1t)} (\boldsymbol{\mu}_n^{(2t)})^\top - \frac{1}{S} \sum_{s=1}^{S} \tilde{\mathbf{h}}_{t+1,s}^{(1t)} (\tilde{\mathbf{h}}_{t+1,s}^{(2t)})^\top \right).$$

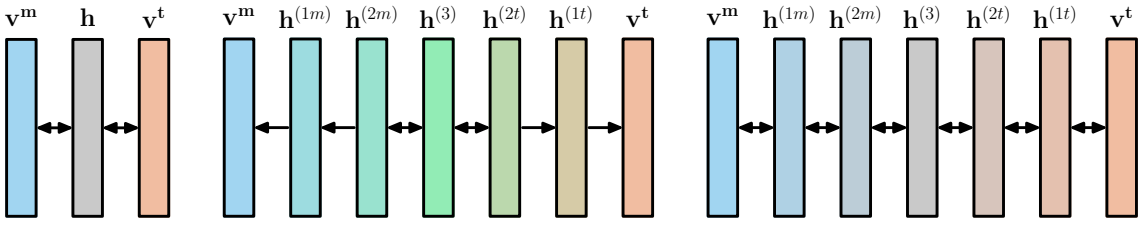
  // Joint Layer:

$$\mathbf{W}_{t+1}^{(3m)} = \mathbf{W}_t^{(3m)} + \alpha_t \left(\frac{1}{N} \sum_{n=1}^{N} \boldsymbol{\mu}_n^{(2m)} (\boldsymbol{\mu}_n^{(3)})^\top - \frac{1}{S} \sum_{s=1}^{S} \tilde{\mathbf{h}}_{t+1,s}^{(2m)} (\tilde{\mathbf{h}}_{t+1,s}^{(3)})^\top \right).$$

$$\mathbf{W}_{t+1}^{(3t)} = \mathbf{W}_t^{(3t)} + \alpha_t \left(\frac{1}{N} \sum_{n=1}^{N} \boldsymbol{\mu}_n^{(2t)} (\boldsymbol{\mu}_n^{(3)})^\top - \frac{1}{S} \sum_{s=1}^{S} \tilde{\mathbf{h}}_{t+1,s}^{(2t)} (\tilde{\mathbf{h}}_{t+1,s}^{(3)})^\top \right).$$

  Decrease α_t.
end for

This condition can, for example, be satisfied simply by setting $\alpha_t = a/(b + t)$, for positive constants $a > 0$, $b > 0$. Typically, in practice, the sequence $|\theta_t|$ is bounded, and the Markov chain, governed by the transition kernel T_θ, is ergodic. Together with the condition on the learning rate, this ensures almost sure convergence of the stochastic approximation algorithm to an asymptotically stable point (Younes, 1998; Yuille, 2004).



(a) RBM (b) Multimodal DBN (c) Multimodal DBM

Figure 4: Different ways of modeling multimodal inputs.

4.1.3 Greedy Layer-wise Pretraining

The learning procedure for Deep Boltzmann Machines described above can be used by initializing model parameters at random. However, the model performs much better if parameters are initialized sensibly. We therefore use a greedy layer-wise pretraining strategy by learning a stack of modified Restricted Boltzmann Machines (RBMs) (for details see Salakhutdinov and Hinton, 2009b). The pretraining procedure is quite similar to the pretraining procedure of Deep Belief Networks (Hinton et al., 2006), and it allows us to perform approximate inference by a single bottom-up pass. This fast approximate inference can also be used to initialize the mean-field, which then converges much faster than mean-field initialized at random.

5. Applying Multimodal DBMs to Different Tasks

There are several tasks that are of interest when working with multimodal data, such as generating missing modalities, inferring a joint representation or discriminative tasks that require classifying the multimodal input. In this section, we describe how DBMs can be used to solve these tasks. We also highlight the motivation behind the use of this approach.

5.1 Motivation

A Multimodal DBM can be viewed as a composition of unimodal undirected pathways. Each pathway can be pretrained separately in a completely unsupervised fashion, which allows us to leverage a large supply of unlabelled unimodal data. Any number of pathways each with any number of layers could potentially be used. The type of the lower-level RBMs in each pathway could be different, accounting for different input distributions. However, the hidden representations at the end of each pathway can be made to be of the same type (binary). Moreover, they can be encouraged to have nice statistical properties which we can control, such as having the same level of sparsity. These hidden representations are now much easier to combine than the low-level input representations.

The intuition behind our model is as follows. Each data modality has very different statistical properties which make it difficult for a single-layer model to directly find correlations in features across modalities. In our model, this difference is bridged by putting layers of hidden units between the modalities. The idea is illustrated in Figure 4c, which is just a different way of displaying Figure 3. Compared to the simple RBM (see Figure 4a), where the hidden layer $\mathbf{h}$ directly models the distribution over $\mathbf{v}^m$ and $\mathbf{v}^t$, the first layer of hidden

units $\mathbf{h}^{(1m)}$ in a DBM has an easier task to perform — that of modeling the distribution over $\mathbf{v}^m$ and $\mathbf{h}^{(2m)}$. Each layer of hidden units in the DBM makes a small contribution towards bridging $\mathbf{v}^m$ and $\mathbf{v}^t$. In the process, each layer learns successively higher-level representations and removes modality-specific correlations. Therefore, the middle layer in the network can be seen as a (relatively) "modality-free" representation of the input as opposed to the input layers which were "modality-full".

Another way of using a deep model to combine multimodal inputs is to use a Multimodal Deep Belief Network (DBN) (Figure 4b) which consists of an RBM at the center followed by directed belief networks leading out to the input layers. We emphasize that there is an important distinction between this model and the DBM model of Figure 4c. In a DBN model, and related autoencoder models, the responsibility for multimodal modeling falls entirely on the joint layer ($\mathbf{h}^{(2m)} \leftrightarrow \mathbf{h}^{(3)} \leftrightarrow \mathbf{h}^{(2t)}$). In the DBM, on the other hand, this responsibility is spread out over the entire network. From the generative perspective, states of low-level hidden units in one pathway can influence the states of hidden units in other pathways through the higher-level layers, which is not the case for DBNs.

5.2 Generating Missing Modalities

As argued in the introduction, many real-world applications will often have one or more modalities missing. The Multimodal DBM can be used to generate such missing data modalities by clamping the observed modalities at the inputs and sampling the hidden modalities by running the standard Gibbs sampler.

For example, consider generating text conditioned on a given image[4] $\mathbf{v}^m$. The observed modality $\mathbf{v}^m$ is clamped at the inputs and all hidden units are initialized randomly. Alternating Gibbs sampling is used to draw samples from $P(\mathbf{v}^t|\mathbf{v}^m)$ by updating each hidden layer given the states of the adjacent layers (see Equation 8). A sample drawn from this distribution describes a multinomial distribution over the text vocabulary. This distribution can then be used to sample words. This process is illustrated for a test image in Figure 5, showing the generated text after every 50 Gibbs steps. We see that not only does the sampler generate meaningful text, it shows evidence of jumping across different modes. For example, it generates *tropical*, *caribbean* and *resort* together, then moves on to *canada*, *bc*, *quebec lake*, *ice*, and then *italia*, *venizia* and *mare*. Each of these groups of words are plausible descriptions of the image. Moreover, each group is consistent within itself. This suggests that the model has been able to associate clusters of consistent descriptions with the same image. In other words, the model can capture multiple modes in the conditional distribution and access them by a Gibbs sampler.

The model can also be used to generate image features conditioned on text. Figure 6 shows examples of two such runs.

5.3 Inferring Joint Representations

The model can also be used to generate a joint representation of data by combining multiple data modalities. For inferring the joint representation, conditioned on the observed modalities, the observed modalities are clamped and Gibbs sampling is performed to sample from

4. Generating image features conditioned on text can be done in a similar way.

Step 50	Step 100	Step 150	Step 200	Step 250
travel	beach	sea	water	italy
trip	ocean	beach	canada	water
vacation	waves	island	bc	sea
africa	sea	vacation	britishcolumbia	boat
earthasia	sand	travel	reflection	italia
asia	nikon	ocean	alberta	mare
men	surf	caribbean	lake	venizia
2007	rocks	tropical	quebec	acqua
india	coast	resort	ontario	ocean
tourism	shore	trip	ice	venice

Figure 5: Text generated by the DBM conditioned on an image by running a Gibbs sampler. Ten words with the highest probability are shown at the end of every 50 sampling steps.

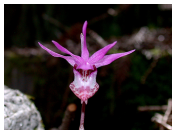
Input tags	Step 50	Step 100	Step 150	Step 200	Step 250
purple, flowers					
car, automobile					

Figure 6: Images retrieved by running a Gibbs sampler conditioned on the input tags. The images shown are those which are closest to the sampled image features. Samples were taken after every 50 steps.

$P(\mathbf{h}^{(3)}|\mathbf{v}^m, \mathbf{v}^t)$ (if both modalities are present) or from $P(\mathbf{h}^{(3)}|\mathbf{v}^m)$ (if text is missing). A faster alternative, which we adopt in our experimental results, is to use variational inference to approximate posterior $Q(\mathbf{h}^{(3)}|\mathbf{v}^m, \mathbf{v}^t)$ or $Q(\mathbf{h}^{(3)}|\mathbf{v}^m)$ (see Section 4.1). The marginals of the approximate posterior over $\mathbf{h}^{(3)}$ (variational parameters $\boldsymbol{\mu}^{(3)}$) constitute the joint representation of the inputs.

This representation can then be used to do information retrieval for multimodal or unimodal queries. Each data point in the database (whether missing some modalities or not) can be mapped to this latent space. Queries can also be mapped to this space and an appropriate distance metric can be used to retrieve results that are close to the query.

5.4 Discriminative Tasks

After learning, the Multimodal Deep Boltzmann Machine can be used to initialize a multilayer neural network by partially unrolling the lower layers (Salakhutdinov and Hinton, 2009b). We can then use the standard backpropagation algorithm to discriminatively fine-tune the model. For each multimodal input vector $\mathbf{v} = \{\mathbf{v}^m, \mathbf{v}^t\}$, mean-field inference is

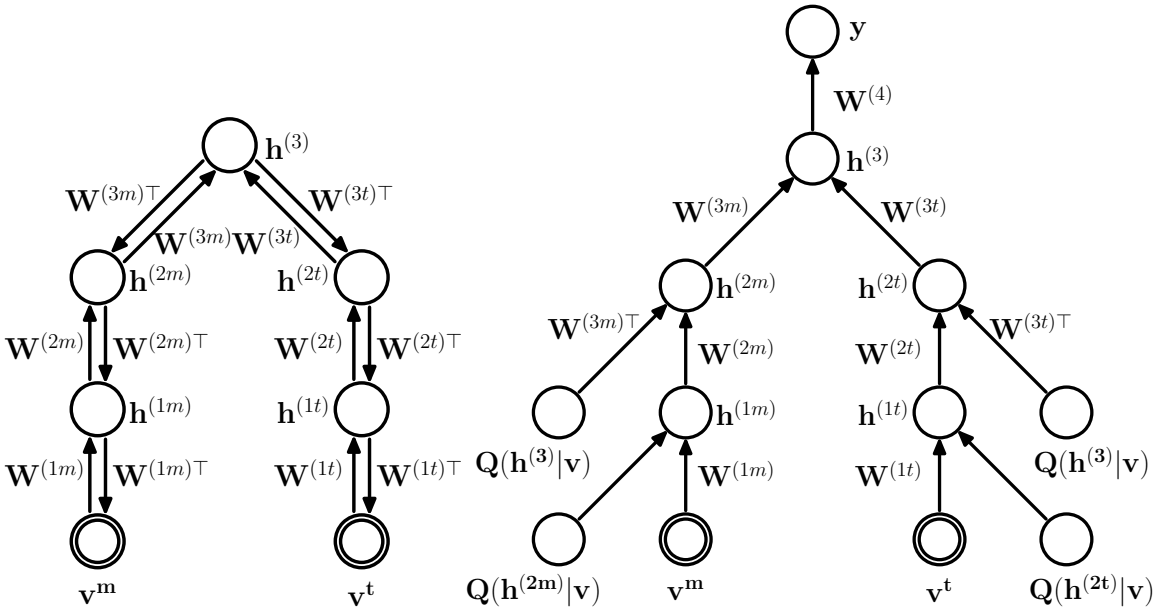


Figure 7: After learning, the DBM model as shown in (a) is used to initialize a multilayer neural network (b), where the marginals of approximate posterior $Q(h_i = 1|\mathbf{v})$ are used as additional inputs. The network is fine-tuned by backpropagation.

used to obtain an approximate posterior distribution $Q(\mathbf{h}|\mathbf{v})$. The marginals of this approximate posterior (variational parameters $\boldsymbol{\mu}$), together with the data, can be used to create an augmented input for this multimodal deep multilayer neural network, as shown in Figure 7. This augmented input is important because it helps maintain the scale of inputs that each hidden unit is expecting. For example, in Figure 7a, the conditional distribution over $\mathbf{h}^{(2m)}$, as defined by the DBM model (see Equation 8), takes the following form:

$$p(h_l^{(2m)} = 1|\mathbf{h}^{(1m)}, \mathbf{h}^{(3)}) \;=\; g\left(\sum_j W_{jl}^{(2m)} h_j^{(1m)} + \sum_p W_{lp}^{(3m)} h_p^{(3)} + b_l^{(2m)}\right).$$

Hence layer $\mathbf{h}^{(2m)}$ receives inputs from $\mathbf{h}^{(1m)}$ as well as $\mathbf{h}^{(3)}$. When this DBM is used to initialize a feed-forward network (Figure 7b), the augmented inputs $Q(\mathbf{h}^{(3)}|\mathbf{v})$ serve as a proxy for $\mathbf{h}^{(3)}$. This ensures that when the feed-forward network is fine-tuned, the hidden units in $\mathbf{h}^{(2m)}$ start off with receiving the same input as they would have received in a mean-field update during unsupervised pretraining. However, once the weights start changing during fine-tuning, the augmented inputs are no longer fixed points of the mean-field update equations and the model is free to use those inputs as it likes. The weights from $Q(\mathbf{h}^{(3)}|\mathbf{v})$ to $\mathbf{h}^{(2m)}$ are only initialized to $\mathbf{W}^{(3m)\top}$ and are not tied to the weights from $\mathbf{h}^{(2m)}$ to $\mathbf{h}^{(3)}$. This initialization scheme makes sure that the model starts fine-tuning from the same place where pretraining left off.

Note that the gradient-based fine-tuning may choose to ignore the marginals of the approximate posterior $Q(\mathbf{h}|\mathbf{v})$ by driving the corresponding weights to zero. This will result in a standard neural network, much like the neural network that is obtained from a Deep Belief Network or an autoencoder model. In practice, however, the network typically uses the entire augmented input for making predictions.

When using this model at test time, we first have to run the mean-field updates in the DBM to get the additional inputs and then use the fine-tuned feed-forward network to get the model's predictions. This creates an overhead in the running time. For all of the data sets in our experimental results, we typically used 5 mean-field updates, which was sufficient for the mean-field to settle down.

6. Experimental Results with Image-Text data

Our first data set consists of image-text pairs. Bi-modal data of this kind exemplifies a common real-world scenario where we have some image and a few words describing that image. There is a need to build representations that fuse this information into a homogeneous space, so that each data point can be represented as a single vector. This representation would be convenient for classification and retrieval problems.

6.1 Data Set and Feature Extraction

We used the MIR Flickr Data set (Huiskes and Lew, 2008) in our experiments. The data set consists of 1 million images retrieved from the social photography website Flickr along with their user assigned tags. The collection includes images released under the Creative Commons License. An example is shown in Figure 8. Among the 1 million images, 25,000 have been annotated using 24 labels including object categories such as, *bird*, *tree*, *people* and scene categories, such as *indoor*, *sky* and *night*. A stricter labeling was done for 14 of these classes where an image was annotated with a category only if that category was salient. This leads to a total of 38 classes where each image may belong to several classes. The data set also consists of an additional 975,000 unannotated images. From the 25,000 annotated images we use 10,000 images for training, 5,000 for validation and 10,000 for testing, following Huiskes et al. (2010). Mean Average Precision (MAP) is used as the performance metric. Results are averaged over 5 random splits of the 25,000 examples into train, validation and test sets.

There are more than 800,000 distinct tags in the data set. In order to keep the text representation manageable, each text input was represented using a vocabulary of the 2000 most frequent tags in the 1 million collection. After restricting to this vocabulary, the average number of tags associated with an image is 5.15 with a standard deviation of 5.13. There are 128,501 images which do not have any tags, out of which 4,551 are in the labelled 25K subset. Hence about 18% of the labelled data has images but is missing text.

Images were represented by 3857-dimensional features, that were extracted by concatenating Pyramid Histogram of Words (PHOW) features (Bosch et al., 2007), Gist (Oliva and Torralba, 2001) and MPEG-7 descriptors (EHD, HTD, CSD, CLD, SCD) (Manjunath et al., 2001). Each dimension was mean-centered and normalized to unit variance. PHOW features are bags of image words obtained by extracting dense SIFT features over multiple

Classes	baby, female, people, portrait	plant life, river, water	clouds, sea, sky, transport, water	animals, dog, food	clouds, sky, structures
Images					
Tags	claudia	⟨ no text ⟩	barco, pesca, boattosail, navegação	watermelon, hilarious, chihuahua, dog	colors, cores, centro, comercial, building

Figure 8: Some examples from the MIR-Flickr data set. Each instance in the data set is an image along with textual tags. Each image has multiple classes.

scales and clustering them. We used publicly available code (Vedaldi and Fulkerson, 2008; Bastan et al., 2010) for extracting these features.[5]

6.2 Model Architecture and Implementation Details

The image pathway consists of a Gaussian RBM with 3857 linear visible units and 1024 hidden units. This is followed by a layer of 1024 binary hidden units. The text pathway consists of a Replicated Softmax Model with 2000 visible units and 1024 hidden units, followed by another layer of 1024 hidden units. The joint layer contains 2048 hidden units. All hidden units are binary. Each Gaussian visible unit was set to have unit variance ($\sigma_i = 1$) which was kept fixed and not learned.[6] Each layer of weights was pretrained using CD-n where n was gradually increased from 1 to 20. All word count vectors were normalized so that they sum to one. This way we avoid running separate Markov chains for each document length to get the model distribution's sufficient statistics, which makes it possible to have a fast GPU implementation.

We also experimented with training a proper generative model, that is, without normalizing the data. Remember, the image-text bimodal DBM can be viewed as a family of different-sized DBMs that are created for documents of different lengths that share parameters. In this setting, we used separate MCMC chains for different sized documents. However, the results were statistically indistinct from the case when we made the simplifying assumptions. This is probably because this data set does not have a huge variance in the number of words per image (5-15 tags per image).

After training the DBM model generatively, we applied it for classification and retrieval tasks. We compared different ways of using the model for classification. The simplest method is to extract the representation at the joint hidden layer and perform 1-vs-all classification using logistic regression. We compare this to fine-tuning the model discriminatively as described in Section 5.4. We also used dropout (Hinton et al., 2012; Srivastava et al., 2014) during fine-tuning to further improve the classification performance. For dropout, we retained each unit with probability $p = 0.8$.

5. The extracted features are publicly available at `http://www.cs.toronto.edu/~nitish/multimodal`.
6. We found that learning the variance made the training unstable.

Model	MAP	Prec@50
Random	0.124	0.124
SVM (Huiskes et al., 2010)	0.475	0.758
LDA (Huiskes et al., 2010)	0.492	0.754
DBM	0.526 ± 0.007	0.791 ± 0.008
DBM (using unlabelled data)	**0.585** ± 0.004	**0.836** ± 0.004

Table 2: Multimodal Classification Results. Mean Average Precision (MAP) and precision@50 obtained by different models. A similar set of input features is used across all models.

6.3 Classification Tasks

We run two classification experiments to highlight two distinct capabilities of the proposed DBM model. In the first experiment, we train and test the model on multimodal inputs. This experiment is designed to evaluate the DBM's ability to *represent* multimodal data in a way that is useful for classification. In the second experiment, we train on multimodal inputs, but at test time we are only given images. This experiment is designed to evaluate the DBM's ability to *generate* useful text and use it as a substitute for real data.

Since examples in the data set may have multiple labels, classification accuracy is not very meaningful. Instead, we evaluate our models using Mean Average Precision (MAP) and precision at top-50 predictions (Prec@50). These are standard metrics used for multi-label classification and have been previously used to report results on this data set.

6.3.1 Multimodal Inputs

In out first experiment, the task is to assign labels to image-text pairs. Huiskes et al. (2010) provided baselines for this data set with Linear Discriminant Analysis (LDA) and RBF-kernel Support Vector Machines (SVMs) using the labelled 25K subset of the data. They represent the multimodal input as a concatenation of image features and word counts. Table 2 shows the performance of these models. The image features did not include SIFT-based features. Therefore, to make a fair comparison, our model was first trained using the same amount of data and a similar set of features (i.e., excluding our SIFT-based features). Table 2 shows that the DBM model outperforms its competitor SVM and LDA models in terms of MAP and Prec@50. The DBM achieves a MAP of 0.526, compared to 0.475 and 0.492, achieved by SVM and LDA models.

Next, we tried to see how much gain can be obtained by using the 975,000 unlabelled examples. We trained a DBM using these examples and, not surprisingly, this improved the DBM's MAP to 0.585.

Having established that DBMs outperform simple linear models, we now compare DBMs to two other deep models—Deep Belief Nets (DBNs) (Hinton et al., 2006) and Denoising Autoencoders (DAEs) (Vincent et al., 2008). We found that further improvements can be obtained by using more image features. We added PHOW features, which use dense SIFT descriptors, to learn a feature dictionary. Table 3 shows results using this extended feature set. We use unlabelled data to pretrain these models. We also closely explore the benefits of full discriminative fine-tuning, regularizers that encourage sparse activations and dropout.

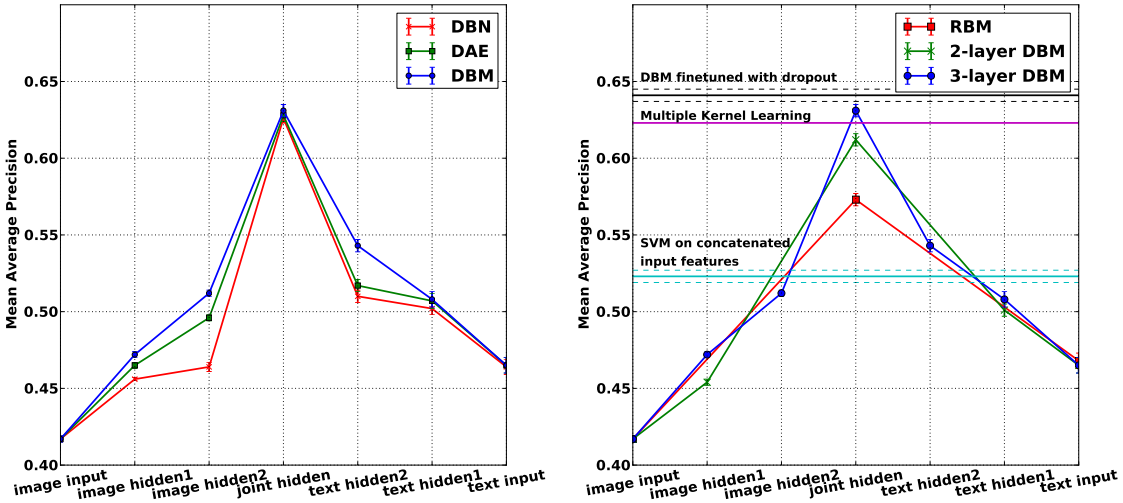
Model	No Pretraining	DBN	DAE	DBM
Logistic regression on joint layer features	-	0.599 ± 0.004	0.600 ± 0.004	0.609 ± 0.004
Sparsity + Logistic regression on joint layer features	-	0.626 ± 0.003	0.628 ± 0.004	0.631 ± 0.004
Sparsity + discriminative fine-tuning	0.482 ± 0.003	0.630 ± 0.004	0.630 ± 0.003	0.634 ± 0.004
Sparsity + discriminative fine-tuning + dropout	0.575 ± 0.004	0.638 ± 0.004	0.638 ± 0.004	**0.641 ± 0.004**

Table 3: Comparison of MAP across different deep models. Sparsity, full discriminative fine-tuning and dropout lead to improvements across all models. More input features were used compared to Table 2.

First, we apply simple logistic regression on the high-level joint representation learned by each of the three models. As shown in Table 3, the DBN and DAE obtain a MAP of 0.599 and 0.600 respectively, whereas the DBM gets 0.609. The error bars indicate that this improvement is statistically significant. The DBNs and DAEs give very similar results. Next we added a KL-sparsity regularizer (Olshausen and Field, 1996) during unsupervised pretraining of all the three models. This improved the performance across all models. In particular, the DBM achieved a MAP of 0.631. Full discriminative training further improved the DBM's MAP to 0.634. Next, we fine-tuned the model using the dropout technique proposed by Hinton et al. (2012). Using this we achieved the a MAP of **0.641**. The DBN and DAE also produce very close results. The Multiple Kernel Learning approach proposed by Guillaumin et al. (2010) obtained a MAP of 0.623 where they used a much larger set of image features (37,152 dimensions). TagProp (Verbeek et al., 2010) obtained a MAP of 0.640 which is comparable to DBMs again using a much larger set of features.

In terms of Prec@50, the DBM achieves a score of 0.888 ± 0.004. The DBN and DAE score 0.887 ± 0.003 and 0.888 ± 0.004 respectively. Therefore, all the deep models do about the same in terms of this metric.

Learning a deep hierarchy of features is widely believed to be the reason why deep models have been successful in a number of machine learning tasks. In order to better understand the properties of different layers in the network, we evaluate the quality of representation at each layer of the network. We do this by measuring MAP obtained by logistic regression classifiers on the representation at different layers of the network. We choose a simple classifier so that the MAP results represent a good measure of the representation's discriminative ability. Figure 9a compares different deep models. In all the models, MAP increases as we go from the input layer towards the joint hidden layer from either side. This shows that higher level representations become increasingly good at discovering useful features. It is interesting to note that the performance of the DBM's hidden layers increases rapidly with depth whereas that for the DBN seems to stagnate at the second layer. At the middle (joint) layer, the performance of both models increases tremendously. This behavior points to an important property of DBMs. Intuitively, the joint generative training of all the layers allows information to flow more readily between



(a) (b)

Figure 9: Mean Average Precision (MAP) obtained by applying logistic regression to representations learned at different layers. **Left :** Comparison of different deep models - Deep Belief Nets, Denoising Autoencoders and Deep Boltzmann Machines. All model have the same architecture and same number of parameters. **Right:** Comparison of DBMs of different depths with SVMs and MKL models. Observe that adding depth improves performance.

the image and text pathways. This comparison empirically verifies the intuition behind having undirected connections throughout the model as mentioned in Section 5.1.

Next, we investigate the effect of depth more closely on DBMs. The question that we try to answer here is how many intervening layers of hidden units should we put between the image and text modalities. It is useful to think of intervening layers as shown in Figure 4c. We could just have one intervening layer, creating an RBM (image input—joint hidden layer—text input). A two-layered DBM would have 3 intervening layers (image input—image hidden 1—joint hidden—text hidden 1—text input), and so on. Figure 9b compares the layer-wise performance of these models (RBM, 2-layer DBM and 3-layer DBM). The performance of other models is also indicated with horizontal lines. Comparing the performance of the joint hidden layer across the three models, we can see that having more intervening layers leads to better performance. The incremental utility of adding more layers seems to decrease.

6.3.2 Unimodal Inputs

In a multimodal data setting, it is very common for some data points to be missing some data modalities. For example, there may be images which do not have captions or tags. This raises interesting questions—Can we use a model that was trained on images and text, when we only have images at test time? Can this model do better than one that was trained on images alone? For multimodal DBMs the answer is affirmative. In this section, we describe an experiment to demonstrate this.

The task is the same as in the previous experiment. We trained a DBM using the unlabelled data and fine-tuned it for discrimination as before. The only difference is that

Model	MAP	Prec@50
Image LDA (Huiskes et al., 2010)	0.315	-
Image SVM (Huiskes et al., 2010)	0.375	-
Image DBN	0.463 ± 0.004	0.801 ± 0.005
Image DBM	0.469 ± 0.005	0.803 ± 0.005
Multimodal DBM (generated text)	**0.531 ± 0.005**	**0.832 ± 0.004**

Table 4: Unimodal Classification Results. Mean Average Precision (MAP) and precision@50 obtained by different models. A similar set of input features is used across all models.

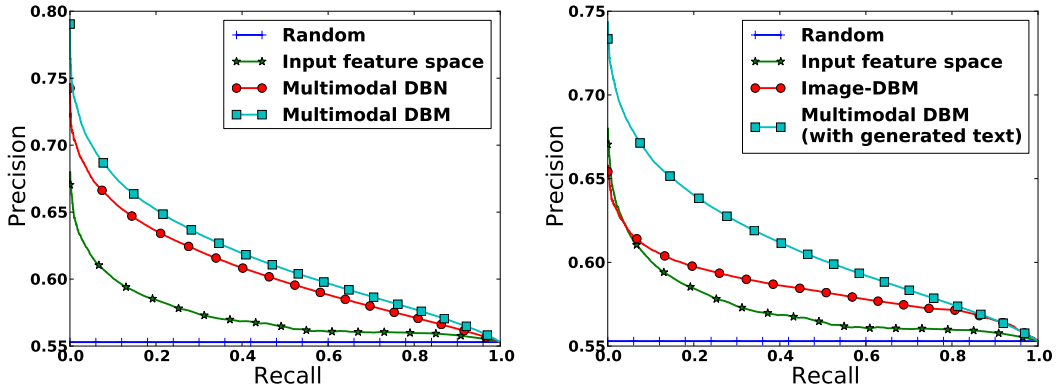
at test time, the model was given only image inputs and used the DBM to generate and fill in missing data. This was done by mean-field updates. We also tried Gibbs sampling and found that it work just as well but with more variance.

We compare the Multimodal DBM with models that were trained using only images. We compare with baseline (RBF-kernel) SVM and LDA results, using a restricted feature set which is similar to that used in Huiskes et al. (2010). Table 4 shows that the LDA and SVM models achieve a MAP of 0.315 and 0.375, respectively. A DBN trained on the similar image features improves this to 0.463. A DBM further improves this to 0.469. In both these cases, pretraining was done using images from the unlabelled set. Both models had the same number of layers and same number of hidden units in each layer. Next, we used a Multimodal DBM to infer the text input and hidden representations at each layer (using mean-field updates). At test time, these representations along with the image features were given as input to the discriminatively fine-tuned DBM. This achieved a significantly higher MAP of 0.531.

This result shows that the DBM can generate meaningful text that serves as a plausible proxy for missing data. This further suggests that *learning multimodal features helps even when some modalities are absent at test time*. The model learns much better features when it has access to multiple modalities because it is being asked to discover features that explain both modalities simultaneously. This can be interpreted as a regularization effect, where instead of the asking the model to be simple or sparse, we ask it to explain an alternative "view" of the data which lies on a very different manifold but shares essential discriminative characteristics with the original view.

6.4 Retrieval Tasks

The next set of experiments was designed to evaluate the quality of the learned joint representation for retrieval purposes. A database of images was created by randomly selecting 5000 image-text pairs from the test set. We also randomly selected a disjoint set of 1000 images to be used as queries. Each query contained both image and text modalities. Each data point has 38 labels. Using these, binary relevance labels were created by assuming that if any of the 38 labels overlapped between a query and a data point, then that data point is relevant to the query.



(a) Multimodal Queries (b) Unimodal Queries

Figure 10: Precision-Recall curves for Retrieval Tasks.

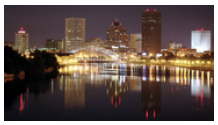
Multimodal Query	**Top 4 retrieved results**			
 hongkong, causewaybay, shoppingcentre, building, mall	 howell, bridge, genesee, river, rochester, downtown, building	 london, uk, night, skyline, river, thames, lights, bridge	 edinburgh, scotland, dusk, bank	 arcoiris, fincadehierro, lluvia, sannicolas, valencia
 me, myself, eyes, blue, hair	 urban, me, abigfave, fiveflickrfavs,	 trisha, mynewcamera, lake, field, girl	 me, ofme, self, selfportrait	 pink, prettyinpink, explored

Figure 11: Retrieval Results for Multimodal Queries from the DBM model.

6.4.1 Multimodal Queries

Figure 10a shows the precision-recall curves for the DBM, DBN, and DAE models (averaged over all queries). For each model, all queries and all points in the database were mapped to the joint hidden representation under that model. Cosine similarity function was used to match queries to data points.

The DBM model performs the best among the compared models achieving a MAP of 0.622. This is slightly better than the performance of the autoencoder and DBN models which achieve a MAP of 0.612 and 0.609 respectively. Figure 11 shows some examples of multimodal queries and the top 4 retrieved results. Note that even though there is little overlap in terms of text, the model is able to perform well.

Image				
Generated Tags	water, glass, beer, bottle, drink, wine, bubbles, splash, drops, drop	portrait, women, army, soldier, mother, postcard, soldiers	nikon, d200, nikkor, d50, tamron, d300, d90, f28, sb600, d60	obama, barackobama, election, politics, president, hope, change

Figure 12: Examples where the DBM does not work well.

6.4.2 Unimodal Queries

The DBM model can also be used to query for images alone. Figure 10b shows the precision-recall curves for the DBM model along with other unimodal models. Each model received the same set of test image queries as input. The joint hidden representation was inferred keeping the text input layer unclamped. Using this representation, the DBM model was able to achieve far better results than any unimodal method (MAP of 0.614 as compared to 0.587 for an Image-DBM and 0.578 for an Image-DBN).

6.5 When Does the Model Not Work?

In this section, we analyze the DBM model to understand when it fails to work and what exactly goes wrong. Figure 12 shows some examples where the model fails to generate meaningful text. To diagnose the problem, we looked at the Markov chains that lead to these results. By visual observation, it was clear that some of these chains got stuck in a region of space and never came out. This happened often when the text sampler reached the space of frequently occurring tags, such as those which refer to camera brands or lens specifications. These tags occur across all kinds of images and seem to take up a huge probability mass under the model independent of the image. There could be other more subtle causes of failure but they were hard to diagnose by visual inspection.

7. Experimental Results with Video-Audio Data

We next demonstrate our approach on video-audio bimodal data. We use data sets that consist of videos of lip movements along with the sound recordings of the words being spoken. This setting has been previously explored in the context of deep multimodal learning by Ngiam et al. (2011) using sparse denoising autoencoders.

7.1 Preprocessing and Data Sets

We represent the auditory information using 40 dimensional log-filter banks along with temporal derivatives to create a 120-d frame for 20 ms speech windows with a stride of 10 ms. Similar to Ngiam et al. (2011) we extract 60×80 mouth regions from the video using a simple object detector (Dalal and Triggs, 2005). The detections were cleaned by median filtering. The extracted mouth regions were compressed to 32 dimensions with

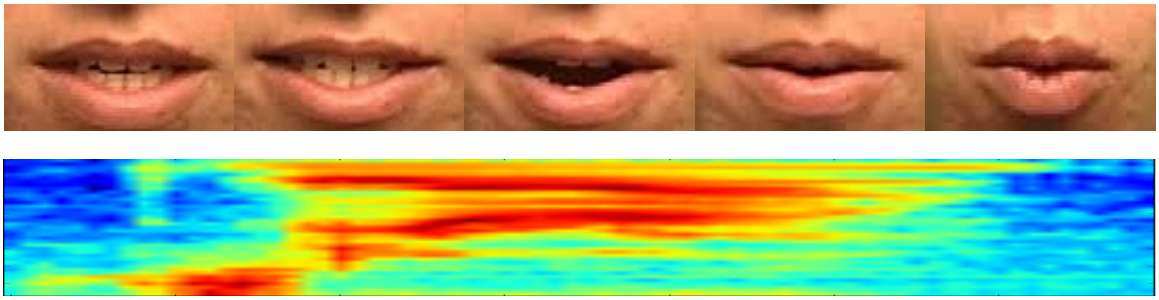


Figure 13: An example of audio-video data extracted from the CUAVE data set.

PCA. Temporal derivatives were then added to create a 96 dimensional representation for each frame. We combined several data sets in this experiment.

CUAVE (Patterson et al., 2002): This data set consists of 36 speakers speaking the digits 0 to 9. The data set has each speaker speaking with different facial orientations (front and sideways) and speaking speeds. We exclude the sideways oriented portions of the data set for simplicity. We use half the speakers for testing and the other half for training.

AVLetters (Matthews et al., 2002): This data set consists of 10 speakers speaking letters A-Z three times each. This data set does not come with raw audio and was used for unsupervised pretraining of the video pathway. We treat this data set as if it were missing audio and evaluate the DBM's ability of fill in the missing data and use it for classification.

AVLetters 2 (Cox et al., 2008): This data set consists of high resolution recordings from 5 speakers speaking letters A-Z. The videos were down-sampled. This data set was used for unsupervised training of the entire model.

TIMIT (Fisher et al., 1986): This data set consists of recordings from 680 speakers covering 8 major dialects of American English reading ten phonetically-rich sentences in a controlled environment. We used this for the unsupervised pretraining of the auditory pathway.

In addition to these, Ngiam et al. (2011) use the Stanford Data Set which consists of 23 speakers speaking the letters A-Z and digits 0-9. However, this data set is not publicly available yet and we were unable to use it. Since all of the above data sets differ in terms of video recording environments, we use PCA in the hope to ameliorate some of these difference. In all the experiments, all available data was used for unsupervised pretraining.

7.2 Model Description

A Multimodal DBM was trained with 4 consecutive image frames and 10 consecutive audio frames since they roughly correspond to same amount of time. Both pathways used Gaussian RBMs as the first layer, as defined in Equation 9. The auditory pathway consisted of 1200 input units followed by 2 layers of 1024 hidden units. The visual pathway had 384 input units followed by 2 layers of 1024 hidden units. The joint layer had 2048 hidden units.

The task was to label each utterance with the digit or letter that was being uttered. Different utterances had different lengths. We obtained a fixed length representation by applying average pooling on the features obtained from the joint layer. In addition, we divided each utterance into 3 equal splits and average pooled the features over those separately.

Method	Classification accuracy %
Concatenated video and audio features	63.5
Video RBM (Ngiam et al., 2011)	65.4 ± 0.6
Multimodal DAE (Ngiam et al., 2011)	66.7
Multimodal DBN	67.2 ± 0.9
Video DBM	67.8 ± 1.1
Video DAE (Ngiam et al., 2011)	68.7 ± 1.8
Multimodal DBM	**69.0 ± 1.5**
Discrete Cosine Transform (Gurban and Thiran, 2009)	64
Active Appearance Model (AAM) (Papandreou et al., 2007)	75.7
Fused Holistic + Patch (Lucey and Sridharan, 2006)	77.08
Visemic AAM (Papandreou et al., 2009)	83

Table 5: Classification results on the CUAVE data set.

These 4 sets of pooled features were concatenated to form the multimodal representation of the input. We then used a linear SVM to classify based on these representations. This is the same as the method used in Ngiam et al. (2011). No discriminative fine-tuning of the DBM was performed.

7.3 Classification Results

We report the results of two classification experiments. In the first experiment, we classify utterances from the CUAVE data set into 10 digit classes. We use the DBM to extract features using both video and audio inputs. We compare this to a DBN, DAE (Ngiam et al., 2011) and various other methods. The results are shown in Table 5. Linear SVM on concatenated video and audio features serves as a baseline, which achieves 63.5% accuracy. A video-only RBM achieves 65.4%, which can be improved to 67.2% with a 3-layer DBN and to 67.8% with a 3-layer DBM. The denoising autoencoder achieves an even better performance of 68.7%. Ngiam et al. (2011) showed that adding audio features seems to hurt the performance of DAEs, reducing it down to 66.7%. The Multimodal DBM does not suffer from adding audio features and improves the performance slightly to 69.0%. However, this is not a significant improvement over the Video DAE. Note that the DBM was trained on less data compared to the Video DAE of Ngiam et al. (2011). The Multimodal DBM does improve significantly on the performance of the Video DBM trained on the same amount of data.

The performance of the deep models is much worse than that of Active Appearance Models (Papandreou et al., 2007, 2009) and Patch-based methods (Lucey and Sridharan, 2006). However, these models use a different train-test split and specialized image preprocessing techniques that are specifically designed for visual speech recognition tasks.

In our second experiment, we try to classify utterances from the AVLetters data set into 26 letter classes. In this case the audio input is considered missing and we use the DBM to infer the joint hidden representation keeping the audio input unclamped. We do the same for a DBN as well as compare to DAEs and other methods. Table 6 shows the results.

The baseline model which uses the preprocessed video features achieves 46.2% accuracy. An RBM on the same features achieves 54.2%, whereas a 3-layer DBM gets 61.8% and a DAE gets 64.4%. However, the Multimodal DAE again suffers from adding audio features at

Method	Classification accuracy
Video features (Ngiam et al., 2011)	46.2
Video RBM (Ngiam et al., 2011)	54.2 ± 3.3
Multimodal DAE (Ngiam et al., 2011)	59.2
Video DBM	61.8 ± 2.5
Multimodal DBN	63.2 ± 2.1
Video DAE (Ngiam et al., 2011)	64.4 ± 2.4
Multimodal DBM	**64.7 ± 2.5**
Multiscale Spatial Analysis (Matthews et al., 2002)	44.6
Local Binary Pattern (Zhao et al., 2009)	58.85

Table 6: Classification results on the AVLetters data set.

test time compared to a Video DAE, getting an accuracy of 59.2%. The Multimodal DBM, on the other hand, improves over the Video DBM and gets 64.7%, essentially matching the performance of the Video DAE (even though it used less data).

These experiments show that the Multimodal DBM model can effectively combine features across modalities. It consistently shows improvements over training on unimodal data, even when only unimodal inputs are given at test time.

8. Conclusion

We proposed a Deep Boltzmann Machine model for learning multimodal data representations. Large amounts of unlabelled data can be effectively utilized by the model. Pathways for each modality can be pretrained independently and "plugged in" together for performing joint learning. The model fuses multiple data modalities into a unified representation, which captures features that are useful for classification and retrieval. It also performs well when some modalities are absent and improves upon models trained on only the observed modalities. Our model performs well in terms of classification results on the bi-modal MIR-Flickr data set as well as on the CUAVE and AVLetters video-audio data sets, demonstrating the usefulness of this approach.

Acknowledgments

This research was supported by Google, Samsung, and ONR Grant N00014-14-1-0232.

References

M. Bastan, H. Cam, U. Gudukbay, and O. Ulusoy. Bilvideo-7: An MPEG-7- compatible video indexing and retrieval system. *IEEE Multimedia*, 17:62–73, 2010. ISSN 1070-986X.

A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. *IEEE 11th International Conference on Computer Vision (2007)*, 23:1–8, 2007.

S. Cox, R. Harvey, Y. Lan, J. Newman, and B. Theobald. The challenge of multispeaker lip-reading. In *International Conference on Auditory-Visual Speech Processing*, pages

179–184, September 2008.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

W. M. Fisher, G. R. Doddington, and K. M. Goudie-Marshall. The DARPA speech recognition research database: Specifications and status. In *Proceedings of DARPA Workshop on Speech Recognition*, pages 93–99, 1986.

Y. Freund and D. Haussler. Unsupervised learning of distributions on binary vectors using two layer networks. Technical report, University of California at Santa Cruz, Santa Cruz, CA, USA, 1994.

M. Guillaumin, J. Verbeek, and C. Schmid. Multimodal semi-supervised learning for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 902 –909, 2010.

M. Gurban and J.-P. Thiran. Information theoretic feature extraction for audio-visual speech recognition. *IEEE Transactions on Signal Processing*, 57(12):4765–4776, 2009.

G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1711–1800, 2002.

G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

M. J. Huiskes and M. S. Lew. The MIR Flickr retrieval evaluation. In *ACM International Conference on Multimedia Information Retrieval*, 2008.

M. J. Huiskes, B. Thomee, and M. S. Lew. New trends and ideas in visual concept detection: the MIR Flickr retrieval evaluation initiative. In *11th ACM International Conference on Multimedia Information Retrieval*, pages 527–536, 2010.

H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th International Conference on Machine Learning*, pages 609–616, 2009.

P. Lucey and S. Sridharan. Patch-based representation of visual speech. In *Proceedings of the HCSNet workshop on Use of vision in human-computer interaction - Volume 56*, VisHCI '06, pages 79–85. Australian Computer Society, Inc., 2006.

B. Manjunath, J.-R. Ohm, V. Vasudevan, and A. Yamada. Color and texture descriptors. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(6):703 –715, 2001.

I. Matthews, T. F. Cootes, J. A. Bangham, S. Cox, and R. Harvey. Extraction of visual features for lipreading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):198–213, Feb. 2002.

A. Mohamed, G. Dahl, and G. Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 2011.

J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42:145–175, 2001.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

G. Papandreou, A. Katsamanis, V. Pitsikalis, and P. Maragos. Multimodal fusion and learning with uncertain features applied to audiovisual speech recognition. In *IEEE 9th Workshop on Multimedia Signal Processing*, pages 264–267, 2007.

G. Papandreou, A. Katsamanis, V. Pitsikalis, and P. Maragos. Adaptive multimodal fusion by uncertainty compensation with application to audiovisual speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(3):423–435, 2009.

E. K. Patterson, S. Gurbuz, Z. Tufekci, and J. N. Gowdy. Cuave: A new audio-visual database for multimodal human-computer interface research. In *International Conference on Audio Speech and Signal Processing*, pages 2017–2020, 2002.

H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Stat.*, 22: 400–407, 1951.

R. Salakhutdinov and G. E. Hinton. Replicated softmax: an undirected topic model. In *Advances in Neural Information Processing Systems 22*, pages 1607–1614, 2009a.

R. R. Salakhutdinov and G. E. Hinton. Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 12, 2009b.

P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pages 194–281. MIT Press, Cambridge, MA, USA, 1986.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *Eurpean Conference on Computer Vision*, 2010.

T. Tieleman. Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1064–1071, 2008.

A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms, 2008.

J. Verbeek, M. Guillaumin, T. Mensink, and C. Schmid. Image annotation with TagProp on the MIRFLICKR set. In *11th ACM International Conference on Multimedia Information Retrieval*, pages 537–546, 2010.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103, 2008.

M. Welling, M. Rosen-Zvi, and G. E. Hinton. Exponential family harmoniums with an application to information retrieval. In *Advances in Neural Information Processing Systems 18*, pages 1481–1488, 2005.

E. P. Xing, R. Yan, and A. G. Hauptmann. Mining associated text and images with dual-wing harmoniums. In *Uncertainty in Artificial Intelligence (UAI)*, pages 633–641. AUAI Press, 2005.

L. Younes. Parameter inference for imperfectly observed Gibbsian fields. *Probability Theory Rel. Fields*, 82:625–645, 1989.

L. Younes. On the convergence of Markovian stochastic algorithms with rapidly decreasing ergodicity rates. In *Stochastics and Stochastics Models*, pages 177–228, 1998.

A. L. Yuille. The convergence of contrastive divergences. In *Advances in Neural Information Processing Systems 17*, 2004.

G. Zhao, M. Barnard, and M. Pietikainen. Lipreading with local spatiotemporal descriptors. *IEEE Transactions on Multimedia*, 11(7):1254–1265, 2009.

Journal of Machine Learning Research 15 (2014) 2981-3012 Submitted 4/12; Revised 4/14; Published 10/14

Optimal Data Collection For Informative Rankings Expose Well-Connected Graphs

Braxton Osting OSTING@MATH.UTAH.EDU
Department of Mathematics
University of Utah
Salt Lake City, UT 84112, USA

Christoph Brune C.BRUNE@UTWENTE.NL
Department of Applied Mathematics
University of Twente
7500 AE Enschede, The Netherlands

Stanley J. Osher SJO@MATH.UCLA.EDU
Department of Mathematics
University of California
Los Angeles, CA 90095, USA

Editor: Gert Lanckriet

Abstract

Given a graph where vertices represent alternatives and arcs represent pairwise comparison data, the statistical ranking problem is to find a potential function, defined on the vertices, such that the gradient of the potential function agrees with the pairwise comparisons. Our goal in this paper is to develop a method for collecting data for which the least squares estimator for the ranking problem has maximal Fisher information. Our approach, based on experimental design, is to view data collection as a bi-level optimization problem where the inner problem is the ranking problem and the outer problem is to identify data which maximizes the informativeness of the ranking. Under certain assumptions, the data collection problem decouples, reducing to a problem of finding multigraphs with large algebraic connectivity. This reduction of the data collection problem to graph-theoretic questions is one of the primary contributions of this work. As an application, we study the Yahoo! Movie user rating data set and demonstrate that the addition of a small number of well-chosen pairwise comparisons can significantly increase the Fisher informativeness of the ranking. As another application, we study the 2011-12 NCAA football schedule and propose schedules with the same number of games which are significantly more informative. Using spectral clustering methods to identify highly-connected communities within the division, we argue that the NCAA could improve its notoriously poor rankings by simply scheduling more out-of-conference games.

Keywords: ranking, active learning, scheduling, optimal experimental design, graph synthesis, algebraic connectivity

1. Introduction

The problem of statistical ranking[1] arises in a variety of applications, where a collection of alternatives is ranked based on pairwise comparisons. Methods for ranking must address a number of inherent difficulties including incomplete, inconsistent, and imbalanced data. Despite and possibly as a consequence of these difficulties, although ranking from pairwise comparison data is an old problem (David, 1963), there have been several recent contributions to the subject with applications in social networking, game theory, e-commerce, and logistics (Langville and Meyer, 2012; Osting et al., 2013b; Hirani et al., 2011; Jiang et al., 2010; Callaghan et al., 2007).

The statistical ranking problem can be generally posed as finding an estimate for a ranking, ϕ, for a set of alternatives from a data set which consists of (i) a set of alternative pairs which have been queried, w, and (ii) noisy, cardinal[2] pairwise comparisons for those alternative pairs, y. We symbolically express an estimator for the ranking problem,

$$\hat{\phi}_w = \mathcal{R}(y, w), \tag{1}$$

where the dependence of the ranking, $\hat{\phi}_w$, on the queried pairs (data collected), w, is emphasized by the subscript.

Consider the dependence of a ranking, $\hat{\phi}_w$, satisfying (1), on the collected data, w. Generally speaking, for a fixed number of alternatives, the more alternative pairs which have been queried, the more informative we expect the ranking, $\hat{\phi}_w$. That is, there is a tradeoff between the amount of pairwise data collected and the informativeness of the ranking. In this paper, we consider the following question: Given a pairwise comparison data set, (w_0, y_0), and the opportunity to collect ξ additional pairwise comparisons, which pairs should be targeted to maximally improve the informativeness of a statistical ranking, $\hat{\phi}_w$, satisfying (1)?

We propose a learning algorithm for ranking from cardinal pairwise comparisons. To accomplish this, we follow the methodology of the optimal design community (Haber et al., 2008; Pukelsheim, 2006; Melas, 2006; Fedorov, 1972), and consider the *Fisher information* for the ranking estimate, $\hat{\phi}_w$, denoted F.I.$(\hat{\phi}_w)$. We are thus led to the following bilevel optimization problem:

$$\max_{w} \quad f\left(\text{F.I.}(\hat{\phi}_w)\right) \tag{2a}$$

$$\text{such that} \quad \hat{\phi}_w = \mathcal{R}(y, w) \tag{2b}$$

$$w \in \mathbb{Z}_+^N, \quad w \succeq w_0, \quad \|w - w_0\|_1 \leq \xi. \tag{2c}$$

1. We use the term ranking to indicate a numerical score for each item in a collection, which is also sometimes referred to as a rating.
2. A *cardinal* pairwise comparison data set refers to quantitative (real-valued) comparisons between items, as opposed to an *ordinal* pairwise comparison data set, where only pairwise preferences are specified.

where $N := \binom{n}{2} = \frac{n(n-1)}{2}$ and $f\colon \mathbb{S}^n_+ \to \mathbb{R}$ is a convex function. For general optimal design problems, common choices for the scalar function $f(A)$ include

$$f(A) = \min_i \ \lambda_i(A) \qquad \text{E-optimal} \tag{3a}$$

$$f(A) = -\mathrm{tr} A^{-1} = -\sum_i \lambda_i(A^{-1}) \qquad \text{A-optimal} \tag{3b}$$

$$f(A) = \det A = \prod_i \lambda_i(A) \qquad \text{D-optimal,} \tag{3c}$$

where $\{\lambda_i(A)\}_{i=1}^n$ denote the eigenvalues of A. The constraint in (2c) specifies that only a limited amount of additional data is collected.

The ranking problem can be represented on a complete directed graph, $G = (V, A)$, with vertices representing the alternatives and the pairwise comparison data, y, is a function on the arcs. The queried pairs, w, can be viewed as an integer valued function on the arcs representing the number of times a pairwise comparison has been queried for that particular pair. In Section 4, we show that for the least squares ranking estimate, $\hat{\phi}_w = \arg\min_{\langle \phi, 1\rangle = 0} \|B\phi - y\|_{2,w}$, where B is defined in Section 3, the constraint (2b) in the optimization problem (2) decouples, yielding a graph synthesis problem of finding the graph whose w-weighted graph Laplacian has desired spectral properties. For example, an E-optimal design (3a) corresponds to finding edge weights w for which the w-weighted graph Laplacian has maximal second eigenvalue (algebraic connectivity). This reduction of the data collection problem to graph-theoretic questions is one of the primary contributions of this and previous work (Osting et al., 2013a).

For the active learning problem for ranking from *ordinal* pairwise data, there has been a large amount of recent work, which we briefly discuss in Section 2. However, the analogous cardinal problem considered here has received less attention. Several recent papers have proposed using iid random sampling (corresponding to an Erdös-Rényi graph) for quality assessment algorithms and crowdsourcing experiments, see, *e.g.*, Eichhorn et al. (2010) and Xu et al. (2012). These algorithms collect pairwise comparisons from a large number of distributed sources without considering the informativeness of the resulting rankings. Like random sampling, the data collection methodology advocated here does not depend on the previous pairwise preferences to select new pairwise queries; our proposed learning algorithm is parallelizable.

In Section 5, we consider several applications of the methodology developed in Section 4 for the optimal data collection problem (2). We begin with a few constructed examples and show that graphs can be generated which have larger algebraic connectivity than Erdös-Rényi randomly generated graphs. The rankings of the data sets represented by these well-connected graphs are more informative then those represented by Erdös-Rényi graphs. We then consider the data collection problem for ranking Yahoo! movies and for the 2011-2012 NCAA Division 1 football season.

1.1 Application: Improving The Informativeness Of Yahoo! Movie Rankings

The Yahoo! Movie user rating data set consists of an incomplete user-movie matrix where entries represent a score given to the movie by the user. By considering the differences in movie reviews by each user, a pairwise comparison data set (w_0, y_0) can be constructed.

For this data set, we empirically demonstrate that the assumptions made in Section 4 are reasonable. By applying the methodology developed in Section 4, we show that the addition of a small number of well-chosen pairwise comparisons can significantly increase the Fisher informativeness of the ranking. The same number of randomly chosen additional pairs has no appreciable impact on the Fisher information.

1.2 Application: Sports Scheduling

The statistical ranking problem arises in competitive sports. Here, teams (alternatives) are ranked based on the schedule (queried alternative pairs) and the game results (pairwise comparisons). The data set is incomplete if not all teams play all other teams; inconsistent if there are teams A, B, and C, such that team A beats team B, team B beats team C, and team C beats team A; and imbalanced if the "strength of schedule" varies among the teams. In this setting, the tradeoff between the amount of data collected (number of games) and the informativeness of the ranking is especially transparent. In a single elimination tournament with n teams, there are only $n-1$ games played. Here, we expect that the "best team" wins the tournament, but it is difficult to rank the remaining teams in any reasonable way. At the other extreme, a round-robin tournament among n teams requires $\binom{n}{2}$ games which may not be possible if n is large. The optimal data collection problem (2) can be interpreted as designing the schedule so that the rankings are the most informative, and thus we refer to the optimal design problem in this context as *schedule design*. In Section 5.8, we study the 2011-12 NCAA football schedule and, using the methodology developed in Section 4, propose schedules with the same number of games which are significantly more informative. Using spectral clustering methods to identify highly-connected communities within the division, we argue that the NCAA could improve its notoriously poor rankings by simply scheduling more out-of-conference games. In Section 5.11, we continue with the graph constructed in Section 5.8 and demonstrate using synthetic data that ranking estimates obtained via active sampling are more accurate (in the sense of both the L^2-distance and the Kendall-τ rank distance) than via random sampling.

1.3 Outline

In Section 2, we review related work. In Section 3, we review properties of the eigenvalues of the graph Laplacian and establish notation used in subsequent sections. In Section 4 we study the optimal data collection problem (2) and show the reduction of (2) to a graph synthesis problem. In Section 5, we conduct a number of numerical experiments to demonstrate how the optimal data collection methodology developed in Section 4 can be employed. Finally, we conclude in Section 6 with a discussion of further directions.

2. Related Work

Our work is related to several subject areas, which we discuss in turn: active learning methods for ordinal ranking, statistics and experimental design, sports scheduling, and graph theory. This work is an extension of the conference proceeding, Osting et al. (2013a). In particular, the present article includes a more extended survey of related work, provides a comparison of Erdös-Rényi graphs and those with maximal algebraic connectivity and

a discussion of the implications of this for the optimal data collection problem, a more complete discussion of the scalarizing criterion for the Fisher information (3), and additional examples.

2.1 Active Learning Methods For Ordinal Ranking

Ailon et al. (2014) and Ailon (2012) study the problem of optimally sampling preference labels for the minimum feedback arc-set in weighted tournaments (MFAST) also known as Kemeny-Young ranking. In this work, the data set considered is ordinal, *i.e.*, only pairwise preference labels are specified, whereas in the present work, the data set is cardinal, *i.e.*, the preferences are represented as quantitative (real valued) differences between items. Jamieson and Nowak (2011b) and Jamieson and Nowak (2011a) consider the problem of actively learning the optimal permutation for a collection of alternatives under the assumption that the alternatives have additional geometric structure, namely a Euclidean embedding in a low dimensional space. Wauthier et al. (2013) propose ranking methods based on independent random sampling, which have worse theoretical complexity, but are relatively simple and easily parallelizable.

2.2 Statistics And Experimental Design

Excellent surveys of the optimal experiment design literature can be found in Haber et al. (2008); Pukelsheim (2006); Melas (2006); Fedorov (1972). Methods of optimal experiment design have been applied to ill-posed inverse problems, *e.g.*, in geophysical (Haber et al., 2008) or biomedical imaging (Horesh et al., 2011, ch. 13, p. 273-290), (Chung and Haber, 2012; Quinn and Keough, 2002; DiStefano 3rd, 1976). It is instructive to consider the analogy between these applications and the optimal data collection problem considered here. In imaging systems, there is a tradeoff between the amount of collected data and the accuracy of the reconstruction, or equivalently, the sparsity of the measurement and the uncertainty in the solution to the inverse problem. For application dependent reasons (*e.g.*, high radiation dose to a patient or the cost of collecting data), it is often desirable to place as few sensors as possible while still maintaining an acceptable accuracy in the reconstruction. In the current work, the goal is to construct the best ranking possible from a small number of pairwise comparisons. In both situations, it is desirable to take "measurements" which are maximally informative.

2.3 Methods For Scheduling From Sports

As discussed in the introduction, in the context of sports ranking, (2) is equivalent to optimal schedule design. There are large variations in the methods currently used for sports scheduling. It is convenient to distinguish between *static* and *dynamic* scheduling. In static scheduling, the schedule is determined prior to the season, independent of the performance of teams throughout the season. Examples of leagues employing static schedules include NCAA football and Major League Baseball (MLB). In dynamic scheduling, the schedule is determined by past score results. For example, in a single elimination tournament, a team advances to the next round only if they win in the current round. Leagues which partially rely on single elimination tournaments include ATP tennis and FIFA World Cup soccer.

Glickman (2005) proposes a dynamic scheduling method where games are scheduled which maximize the expected gain in information and thus one can view the resulting schedules as a greedy algorithm to learn as much as possible about the rankings. This active learning algorithm is similar to several in the machine learning community, where past observations are used to control the process of gathering future observations, see, for example, Krause et al. (2008); Seeger and Nickisch (2011); Silva and Carin (2012). While dynamic schedules utilize the results of previous games and can thus be more informative than static schedules, they have the disadvantage that they may not be completely determined prior to the season. The algorithm developed in this paper is a static scheduling method.

Another type of scheduling in sports focuses on the seeding policy of single-elimination tournaments with the objective of arranging the teams so that the outcome of the tournament agrees with a preexisting ranking (Glickman, 2008; D'Souza, 2010; Scarf and Yusof, 2011) or an arrangement which favors a particular team (Vu et al., 2009). These objectives depend on a preexisting ranking of the teams, which we do not assume to know in this paper. Another type of tournament scheme is investigated by Ben-Naim and Hengartner (2007), where a sequence of rounds of diminishing size are used to determine the best team.

2.4 Graph Theory

In this paper, we reduce the schedule design problem (2) to a graph synthesis problem. We focus on the optimality condition given in (3a), which reduces to finding graphs with maximal algebraic connectivity. There is a tremendous amount of work on the algebraic connectivity of graphs, originating with studies by Fiedler (1973). Many properties of algebraic connectivity are reviewed in (Mohar, 1991; Biyikoglu et al., 2007) and we also review some of these results in Section 3. The problems arising from the other optimality conditions, (3b) and (3c), are less well studied (Grimmett, 2010; Ghosh and Boyd, 2006a; Ghosh et al., 2008).

The robustness of a network to node/edge failures is highly dependent on the algebraic connectivity of the graph. Also, the rate of convergence of a Markov process on a graph to the uniform distribution is determined by the algebraic connectivity (Boyd et al., 2004; Sun et al., 2004). In the "chip-firing game" of Björner et al. (1991), the algebraic connectivity dictates the length of a terminating game. Consequently, algebraic connectivity is a measure of performance for the convergence rate in sensor networks, data fusion, load balancing, and consensus problems (Olfati-Saber et al., 2007).

Finally, we mention recent work of Boumal et al. (2014) on a problem of estimating a set of rotations from a set of noisy measurements. Here, bounds on synchronization are connected to the algebraic connectivity of a measurement graph, where the edge weights are proportional to the measurement quality.

3. Eigenvalues Of The Graph Laplacian And Algebraic Connectivity

In this section, we briefly survey relevant results on the eigenvalues of the graph Laplacian and algebraic connectivity. More extensive treatments are given in Fiedler (1973); Biyikoglu et al. (2007); Mohar (1991); Chung (1997). In Section 3.1, we recall algorithms for computing graphs with large algebraic connectivity (Ghosh and Boyd, 2006b; Wang and Mieghem, 2008).

Let $B \in \mathbb{R}^{N\times n}$ where $N := \binom{n}{2}$ be the arc-vertex incidence matrix for the complete directed graph $G = (V, E)$ on $|V| = n$ nodes,

$$B_{k,j} = \begin{cases} 1 & j = \text{head}(k) \\ -1 & j = \text{tail}(k) \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Here, we have used the terminology that if an arc $k = (i, j)$ is directed from node i to node j then i is the *tail* and j is the *head* of arc k. The arc orientations (heads and tails of arcs) can be chosen arbitrarily. The matrix B as defined in (4) is also sometimes referred to as the graph gradient (Hirani et al., 2011; Jiang et al., 2010). Given an edge-weight $w \in \mathbb{Z}_+^N$, the w-weighted graph Laplacian is defined

$$\Delta_w := B^t W B \qquad \text{where } W = \text{diag}(w).$$

If we consider a subset of the edges, $\tilde{E} \subset E$, and let w be the indicator function on $\tilde{E}$, then Δ_w is referred to as the un-normalized (symmetric) graph Laplacian for $(G, \tilde{E})$. One may interpret the triple (G, E, w) for $w \in \mathbb{Z}_+^N$ as a directed multigraph where w_k for $k = (i, j)$ is the number of arcs connecting vertices i and j. The w-weighted degree vector $d \in \mathbb{R}^n$ is defined by $d_i = \sum_j w_{ij}$. Let $M := \|w\|_1 = \frac{1}{2}\|d\|_1$ and d_+ and d_- denote the maximum and minimum w-weighted degrees in the graph.

Let $\lambda_i(w)$ for $i = 1, \ldots, n$ denote the eigenvalues of the w-weighted graph Laplacian, Δ_w. The eigenvalues are contained in the interval $[0, d_+]$. The first eigenvalue of Δ_w, λ_1, is zero with corresponding eigenvector $v_1 = 1$. The second eigenvalue, λ_2, is nonzero if and only if the graph is connected. The second eigenvalue is characterized by

$$\lambda_2(w) = \min_{\substack{\|v\|=1 \\ \langle v,1\rangle=0}} \|Bv\|_{2,w}. \tag{5}$$

In the case where w is the indicator function for an edge set $\tilde{E}$, $\lambda_2(w)$ is referred to as the *algebraic connectivity* of the graph $G = (V, \tilde{E})$. The eigenvector v_2 corresponding to λ_2 is sometimes called the Fiedler vector after Miroslav Fiedler for his contribution to the subject (Fiedler, 1973). For $w \in \mathbb{Z}_+^N$, $\lambda_2(w)$ is the algebraic connectivity for the multigraph with w_{ij} edges between nodes i and j.

Let $w_i \in \mathbb{Z}_+^N$ for $i = 1, 2$ be edge weights on G. It follows from (5) that $w_1 \le w_2$ implies $\lambda_2(w_1) \le \lambda_2(w_2)$. That is, the function $\lambda_2(w)$ is non-decreasing in w. In particular, if $w_i \in \{0, 1\}^N$ are the indicator functions for two edge sets E_i, $i = 1, 2$ and $w_1 \le w_2$ (component-wise), then $E_1 \subseteq E_2$ and the more connected graph has greater algebraic connectivity.

Let $U \subset V$ and $\text{cut}(U, U^c) := \sum_{i\in U, j\in U^c} w_{ij}$ measure the set of edges connecting U and $U^c := V \setminus U$. Then the algebraic connectivity is bounded by the normalized graph cut,

$$\lambda_2(w) \le \min_{U\subseteq V} \frac{n|\text{cut}(U, U^c)|}{|U||U^c|}. \tag{6}$$

In particular, if $U = \{v\}$ where $v \in V$ is the node with smallest degree, i.e., $d_v = d_-$, then $d_v \le \frac{2M}{n}$ where $M = \|w\|_1$ and we obtain

$$\lambda_2(w) \le \frac{nd_-}{n-1} \le \frac{2M}{n-1}. \tag{7}$$

Properties of graphs for which the bound in (7) is tight have been studied (Fallat et al., 2003).

If $w \in \{0,1\}^N$ is the indicator function for an incomplete edge set $\tilde{E}$ and $\tilde{G} := (V, \tilde{E})$, then the edge connectivity of a $\tilde{G}$, $C_e(\tilde{G})$, is the minimal number of edges whose removal would result in a disconnected graph,

$$C_e(\tilde{G}) = \min_{A \subset V} \sum_{i \in A, j \in A^c} w_{ij}.$$

The vertex connectivity of $\tilde{G}$, $C_v(\tilde{G})$ is the minimal number of vertices (together with adjacent edges) whose removal would result in a disconnected graph. In this case, the algebraic connectivity is bounded above by both the edge and vertex connectivities,

$$\lambda_2(w) \leq C_v(\tilde{G}) \leq C_e(\tilde{G}),$$

(Fiedler, 1973). The algebraic connectivity can also be bounded in terms of Cheeger's inequality, Buser's inequality, and the diameter of the graph (Biyikoglu et al., 2007; Mohar, 1991; Chung, 1997).

There are also a number of results for the perturbation of the eigenvalues of Δ_w under changes to the weights w. Let $\Delta_w v = \lambda v$, $\lambda > 0$, $w_0 = \min_k w_k$ and $w' = w - w_0$. Then

$$\Delta_{w'} v = (\lambda - w_0 n) v. \tag{8}$$

This follows from the fact that $B^t B = n \operatorname{Id} - 1_n 1_n^t$. Thus, adding weight w_0 to w simply increases all of the eigenvalues of Δ_w by w_0.

Consider the weight $w' = w + \delta_k$ where δ_k is the indicator function for edge k. Then using Weyl's theorem (Horn and Johnson, 1990), we obtain

$$\lambda_2(w') \leq \lambda_2(w) + \|B^t \operatorname{diag}(\delta_k) B\| = \lambda_2(w) + 2. \tag{9}$$

Consider the weight $w' = w + \delta_k$ where δ_k is the indicator function for edge k. Denote the eigenvalues of the w and w'-weighted graph Laplacians by λ_j and λ'_j respectively. Then the eigenvalues λ and λ' interlace (Mohar, 1991), *i.e.*,

$$0 = \lambda_1 = \lambda'_1 \leq \lambda_2 \leq \lambda'_2 \leq \lambda_3 \leq \ldots \leq \lambda_n \leq \lambda'_n. \tag{10}$$

3.1 Finding Graphs With Large Algebraic Connectivity

In several applications, it is useful to compute graphs with large algebraic connectivity, (5). The problem of finding weights $w \in \mathbb{R}^N$ which maximize $\lambda_2(w)$ is a convex optimization problem and can be formulated as a semidefinite program (SDP) (Ghosh and Boyd, 2006b). However, if $w \in \mathbb{Z}_+^N$, the problem is NP-hard (Mosk-Aoyama, 2008). This is the case arising in the optimal data collection problem.

The integer constrained problem may be solved by relaxing to the unconstrained problem and then rounding the solution. This is clearly a lower bound on the optimal solution and, if the values w are large, a reasonable approximation. Another approach, advocated by Ghosh and Boyd (2006b); Wang and Mieghem (2008), is to use the greedy algorithm based on the Fiedler vector described in Algorithm 1. This algorithm adds a specified number of edges to an input graph to maximize the algebraic connectivity of the resulting augmented graph. In this work, we refer to graphs produced via this method as *nearly-optimal*.

Algorithm 1 A greedy heuristic for finding integer-valued edge weights w for which the w-weighted graph Laplacian has large second eigenvalue (Ghosh and Boyd, 2006b; Wang and Mieghem, 2008). See Section 3.1.

Input: An initial edge weight $w_0 \in \mathbb{Z}_+^N$ defined on the complete graph of n nodes and an integer, ξ.

Output: An edge weight, $w \succeq w_0$, such that $\|w - w_0\|_1 = \xi$, and Δ_w has large second eigenvalue.

Set $w = w_0$ (current edge weight)
for $\ell = 1$ **to** ξ, **do**
  Compute the second eigenvector, $F = \arg \min_{\substack{\|v\|=1 \\ \langle v,1\rangle=0}} \|Bv\|_w$
  Find the edge (i,j) which maximizes $(F_i - F_j)^2$
  Set $w = w + \delta_{ij}$
end for

4. Optimal Scheduling Using A Least Squares Ranking

We assume that each alternative $j = 1, \ldots, n$ has a ranking (measure of strength) given by ϕ_j. We consider a complete graph with n nodes representing the alternatives. The edges of the graph are given an arbitrary orientation and enumerated $k = 1, \ldots, \binom{n}{2} \equiv N$. Let $B \in \mathbb{R}^{N \times n}$ denote the arc-vertex incidence matrix (4) for the complete graph. For each ordered pair $k = (i,j)$, we assume that the pairwise comparison data collected is of the form

$$y_k = (B\phi)_k + \epsilon_k, \tag{11}$$

where ϵ_k is a random variable with zero mean, *i.e.*, $\mathbb{E}\epsilon = 0$. Let $w_k \in \mathbb{Z}_+$ denote the number of pairwise comparisons between alternatives i and j. We assume that the variance of ϵ_k is given by σ^2/w_k for some constant σ. More comparisons between alternatives i and j, reduce the variance in the observed pairwise comparisons.

4.1 Ranking

There are several choices for the ranking $\mathcal{R}(y,w)$ in (1). The Gauss-Markov theorem states that the least squares estimator,

$$\hat{\phi}_w = \arg \min_{\langle \phi,1\rangle=0} \|B\phi - y\|_{2,w} \tag{12a}$$

$$= (B^t W B)^\dagger B^t W y, \tag{12b}$$

is the linear, unbiased ($\mathbb{E}[\hat{\phi}_w] = \phi$) estimator with smallest covariance. In (12b), $W := \mathrm{diag}(w)$ is the diagonal matrix with entries $W_{kk} = w_k$. Equation (12a) can be interpreted as finding a potential function, ϕ, defined on the vertices, such that the gradient of the potential function agrees with the pairwise comparisons in the least squares sense. The least-squares estimate (12) is also sometimes referred to as HodgeRank (Jiang et al., 2010) and is related to the Massey and Colley methods used in sports rankings (Langville and

Meyer, 2012). The least squares estimator has proven to have relatively good predictive power when empirically compared against a number of other ranking methods on sports data sets (Barrow et al., 2013) and is the ranking method considered in the present work.

Proposition 4.1 *Consider the data model* (11) *where* ϵ *is a random vector with* $\mathbb{E}\epsilon = 0$ *and* $\mathrm{Var}(\epsilon) = \sigma^2 W^{-1}$ *where* $W = \mathrm{diag}(w)$ *and* $w \in \mathbb{Z}_+^N$. *The Fisher information of the least squares estimator* $\hat{\phi}_w$, *as defined in* (12), *is given by*

$$\text{F.I.}(\hat{\phi}_w) = \sigma^{-2}(B^t W B) = \sigma^{-2}\Delta_w, \tag{13}$$

where Δ_w *is the* w*-weighted graph Laplacian.*

Proof Let $\hat{\phi}_w$ be the least squares estimator (12) for ϕ in (11). We first compute

$$\hat{\phi}_w = (B^t W B)^\dagger B^t W y = (B^t W B)^\dagger B^t W (B\phi + \epsilon) = \phi + (B^t W B)^\dagger B^t W \epsilon.$$

Thus,

$$\mathrm{Var}(\hat{\phi}_w) = \mathbb{E}\left[(\hat{\phi}_w - \phi)(\hat{\phi}_w - \phi)^t\right] = (B^t W B)^\dagger B^t W \mathbb{E}\left[\epsilon\epsilon^t\right] W B (B^t W B)^\dagger.$$

Assuming that $\mathbb{E}\left[\epsilon\epsilon^t\right] = \sigma^2 W^{-1}$, we obtain

$$\mathrm{Var}(\hat{\phi}_w) = \sigma^2 (B^t W B)^\dagger = \sigma^2 \Delta_w^\dagger, \tag{14}$$

which is the Moore-Penrose pseudoinverse of the w-weighted graph Laplacian. Since the least squares ranking is unbiased, *i.e.*, $\mathbb{E}\hat{\phi}_w = \phi$, the Fisher information is the pseudoinverse of the covariance matrix, $\mathrm{Var}(\hat{\phi}_w)$. ■

4.2 Optimal Data Collection

The optimal data collection problem (2) is a scalarization of maximizing $\text{F.I.}(\hat{\phi}_w)$ in the sense of the semi-definite ordering (*i.e.*, $A \geq B$ if $A - B \succeq 0$). Traditional optimality criteria are functions of the eigenvalues of $\text{F.I.}(\hat{\phi}_w)$ such as given in (3) (Haber et al., 2008; Pukelsheim, 2006; Melas, 2006; Fedorov, 1972).

Proposition 4.2 *Consider the data model* (11) *with* ϵ *as in Prop. 4.1 and let* $\hat{\phi}_w$ *be the least squares estimator* (12). *The three optimality criteria* (3) *for the bi-level optimization problem* (2) *are given by*

$$f\left(\text{F.I.}(\hat{\phi}_w)\right) = \lambda_2(w) \qquad \textit{E-optimal} \tag{15a}$$

$$f\left(\text{F.I.}(\hat{\phi}_w)\right) = -\sum_{i\geq 2} \lambda_i^{-1}(w) \qquad \textit{A-optimal} \tag{15b}$$

$$f\left(\text{F.I.}(\hat{\phi}_w)\right) = \prod_{i\geq 2} \lambda_i(w) \qquad \textit{D-optimal,} \tag{15c}$$

where $\lambda_i(w)$ *for* $i = 1, \ldots, n$ *denote the eigenvalues of the* w*-weighted graph Laplacian,* Δ_w.

Proof For a connected graph, the only zero eigenvalue of the graph Laplacian is the first one. The expressions in (15) then follow directly from $\text{F.I.}(\hat{\phi}_w) = \Delta_w$, as shown in Prop. 4.1, and the optimal criteria definitions in (3). ■

Proposition 4.1 shows that $\text{F.I.}(\hat{\phi}_w)$ doesn't depend on the scores, y. Consequently, the constraint in the optimal data collection problem (2b) decouples. Using the E-optimal criteria (15a), the bilevel optimization problem (2) reduces to the following eigenvalue optimization problem

$$\begin{aligned} &\max_{w} \quad \lambda_2(w) \\ &\text{such that} \quad w \in \mathbb{Z}_+^N, \quad w \succeq w_0, \quad \|w - w_0\|_1 \leq \xi. \end{aligned} \tag{16}$$

Equation (16) can be interpreted as the graph synthesis problem of adding ξ edges to the multigraph representing the data set to maximize the algebraic connectivity.

Remark 1 *The A- and D-optimal conditions given in Proposition 4.2 also have interesting interpretations in terms of the graph. By Kirchhoff's matrix-tree theorem, the D-optimal condition can be interpreted as the number of spanning trees within the graph (Ghosh and Boyd, 2006a). The A-optimal condition is the total effective resistance of a electric circuit constructed by identifying each edge of the graph with a resistor of equal resistance (Ghosh and Boyd, 2006a; Ghosh et al., 2008) and is related to the return time for a reversible Markov chain (Grimmett, 2010).*

We also comment that the T-optimality condition, $\text{tr}\left(\text{F.I.}(\hat{\phi}_w)\right)$, which is another criteria commonly used in optimal design, simplifies in this setting to $\text{tr}(\Delta_w) = \|w\|_1$, which is simply the total number of pairwise comparisons.

5. Numerical Experiments

In this section, we study graphs corresponding to data sets which have informative rankings, which, by Proposition 4.2, are those with large algebraic connectivity. In Section 5.1, we consider structured graphs for which the eigenvalues of the Laplacian can be analytically computed and small graphs with ≤ 5 edges. In Section 5.2, we compare the expected algebraic connectivity of Erdös-Rényi random graphs with graphs obtained using the greedy algorithm described in Section 3.1. In Section 5.3, we consider the informativeness of the ranking for the Yahoo! Movie user ratings data set. In Section 5.8, we discuss the algebraic connectivity for the graph corresponding to the 2011-12 NCAA Division I football schedule. In Section 5.11, we continue with the graph constructed in Section 5.8 and demonstrate using synthetic data that ranking estimates obtained via active sampling are more accurate (in an L^2 sense) than via random sampling.

5.1 Algebraic Connectivity For Example Graphs

In this section, we give results on the algebraic connectivity for graphs with easily computable spectra and graphs with a small number of nodes. In Table 1, we tabulate the eigenvalues, algebraic connectivity (5), edge connectivity, vertex connectivity, and diameter for 4 well-known graphs.

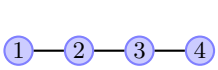
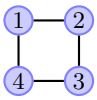
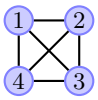
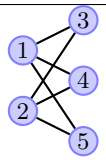
	path, P_n	cycle, C_n	complete, K_n	complete bipartite, $K_{n,\ell}$
diagram				
eigenvalues	$2-2\cos(\pi k/n)$ $k=0,\dots n-1$	$2-2\cos(2\pi k/n)$ $k=0,\dots n-1$	0_1 , n_{n-1}	0_1 , $n_{\ell-1}$, ℓ_{n-1}, $(\ell+n)_1$
alg. conn. (5)	$2-2\cos(\pi/n)$	$2-2\cos(2\pi/n)$	n	$\min(n,\ell)$
edge conn.	1	2	$n-1$	$\min(n,\ell)$
diameter	$n-1$	$\lfloor n/2 \rfloor$	1	2

Table 1: A comparison of several measures of connectivity for 4 well-known graphs. We assume $n \geq 3$. Subscripts on the eigenvalues denote multiplicity and $\lfloor \cdot \rfloor$ indicates the floor function. See Section 5.1.

The number of distinct n-node, connected, unlabeled graphs for n =1, 2, 3, ... are 1, 1, 2, 6, 21, 112, 853, 11117, 261080,....[3] In Figure 1 we plot, for $n = 4$ and $n = 5$, each of these graphs together with the algebraic connectivity, λ_2. In Figure 1, we observe that as the number of edges, m, is increased, the algebraic connectivity, λ_2, generally increases. Furthermore, for a fixed number of edges, m, the algebraic connectivity can vary significantly. For $m = 5$, 6, and 7, the value of λ_2 varies by a factor ≥ 2. For $m = 5$, the graph with smallest λ_2 has small edge connectivity (and hence small algebraic connectivity) and the graph with largest λ_2 has nodes with equal degree. These small graphs beautifully illustrate the bounds given in Section 3.

In Figure 2, we illustrate the effect of adding edges on the algebraic connectivity of a graph by studying (16) where $\|w_0\|_1 = 6$ and $\xi = 1$. Although the graphs in Figure 2 are small in size, it is already nontrivial to determine which edge should be added to maximally increase the algebraic connectivity. We observe that for graphs with low algebraic connectivity, a significant gain can be achieved, while the results for graphs with relatively high algebraic connectivity are modest. In the lowermost panel in Figure 2, the algebraic connectivity remains constant as an edge is added. This follows from the fact that the second eigenvalue for the graph on the left has multiplicity 2 and the interlacing property described in (10).

Further consideration of the algebraic connectivity for certain families of graphs is considered in Kolokolnikov (2014). Here, it is observed that the greedy algorithm (Algorithm 1) is unable to discover certain small, structured graphs with maximal algebraic connectivity.

3. This is sequence number A001349 in the The On-Line Encyclopedia of Integer Sequences, published electronically at `http://oeis.org`.

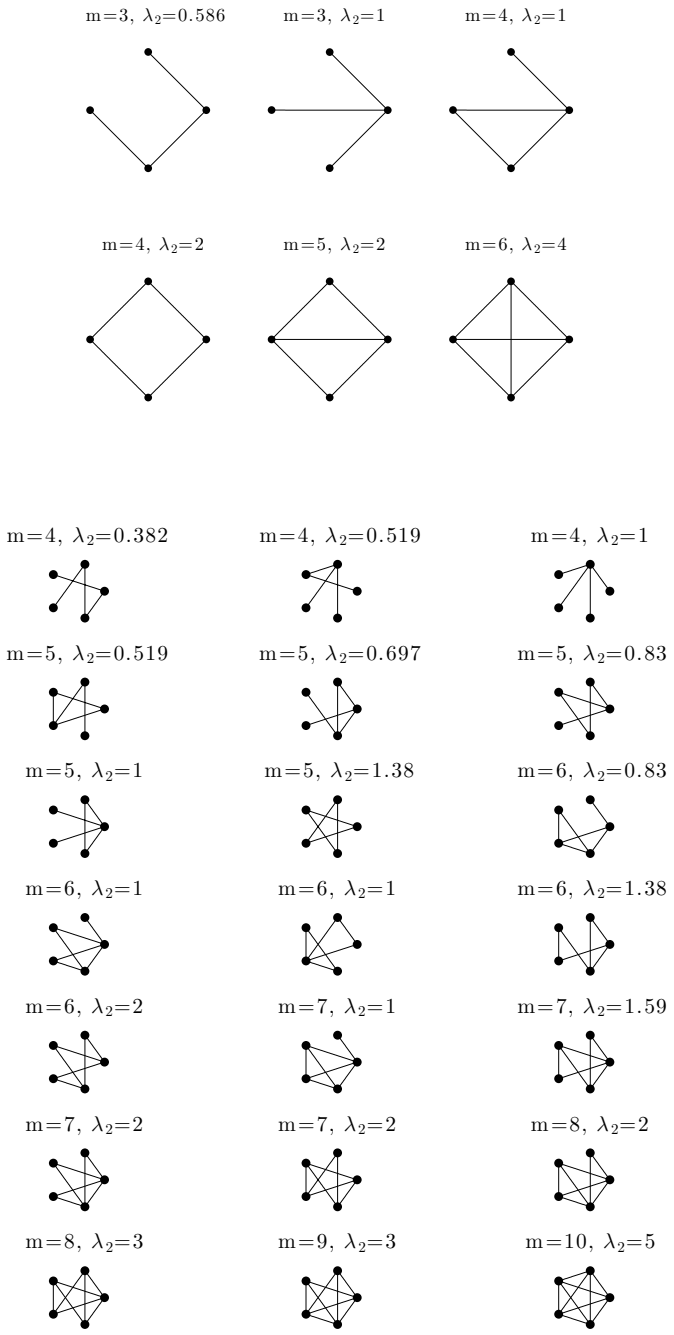


Figure 1: The 4- and 5-node connected graphs and their algebraic connectivity, λ_2. Graphs with large algebraic connectivity represent data sets with informative rankings. See Section 5.1.

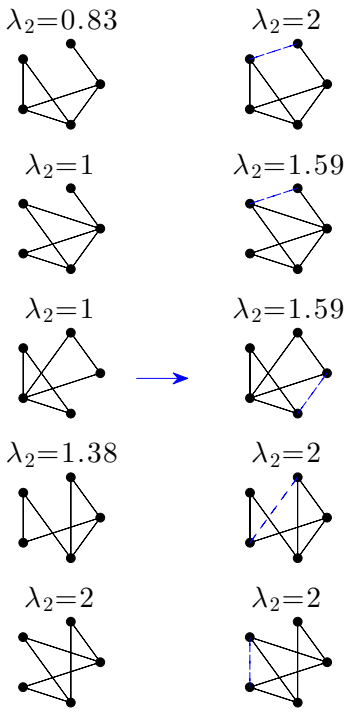


Figure 2: Targeted data collection for small graphs. **(left)** The five topologically distinct connected graphs with $n = 5$ nodes and $m = 6$ edges. **(right)** For each edgeset on the left, we select one additional edge (blue dashes) so that λ_2 for the perturbed graph is maximal. The algebraic connectivity of each graph is indicated. By Prop. 4.2, a ranking on a data set represented by a graph on the right is more informative than one from a graph on the left. See Section 5.1.

5.2 Algebraic Connectivity Of Erdös-Rényi Random Graphs And Computed Nearly-Optimal Graphs

We consider the Erdös-Rényi random graph model $G(n, p)$ containing graphs with n nodes and edges included with probability p, independent from every other edge. The expected number of edges for a graph in $G(n, p)$ is $p\binom{n}{2}$ and the threshold for connectedness is $p_c = \frac{\log n}{n}$.

There are several results on the spectrum of the graph Laplacian for Erdös-Rényi graphs, especially in the limit $n \uparrow \infty$; see, for example, Juhász (1991); Chung et al. (2003); Feige and Ofek (2005); Coja-Oghlan (2007); Jamakovic and Mieghem (2008); Oliveira (2009); Chung and Radcliffe (2011); Kolokolnikov et al. (2014). The algebraic connectivity of Erdös-Rényi, Watts-Strogatz, and Barabási-Albert random graphs has been studied numerically in Jamakovic and Uhlig (2007). The algebraic connectivity of a Watts-Strogatz graph is known to have a phase transition (Olfati-Saber et al., 2007).

We will utilize the following elementary upper bound on the algebraic connectivity, analogous to (7), derived using a concentration inequality.

Proposition 5.1 *Let $\epsilon > 0$ and assume n to be even. With probability at least $1 - \epsilon$, the algebraic connectivity, λ_2, of an Erdös-Rényi graph $G(n, p)$ satisfies*

$$\lambda_2 \leq np + 4n^{-2}\sqrt{2\log(1/\epsilon)}. \tag{17}$$

Proof Choose any subset $U \subset V$ with $|U| = \frac{n}{2}$. Equation (6) implies that $\lambda_2 \leq \frac{4C}{n}$ where $C \sim \mathcal{B}(\frac{n^2}{4}, p)$. For $a > 0$, we compute

$$\operatorname{pr}(\lambda_2 \geq np + a) \leq \operatorname{pr}(4C/n \geq np + a) = \operatorname{pr}\left(C - pn^2/4 \geq +an/4\right) \leq \exp\left(-a^2n^4/32\right),$$

where the last inequality is due to Hoeffding. Setting $a = 4n^{-2}\sqrt{2\log(1/\epsilon)}$, we find that $\operatorname{pr}(\lambda_2 \geq np + a) \leq \epsilon$ as desired. ■

For a random graph $G(n, p)$, the number of edges $m \sim \mathcal{B}(N, p)$ where $N := n(n-1)/2$. Thus, $\mathbb{E}[m] = pN$ and we may restate (17) as: with probability at least $1 - \epsilon$,

$$\lambda_2 \leq \frac{2E[m]}{n-1} + 4n^{-2}\sqrt{2\log(1/\epsilon)}. \tag{18}$$

Indeed, the first term on the right hand side of (18) matches the right hand side of (7).

In Figure 3, we plot, for $n = 50$ (left) and $n = 100$ (right) and $p = .4$ (blue), $p = .6$ (red), and $p = .8$ (green) the value of m vs. λ_2 for 5,000 randomly generated Erdös-Rényi graphs. The mean values obtained are indicated by circles. We use the greedy algorithm described in Section 3.1 (see Algorithm 1) with initial graph taken to be the path with n vertices, P_n, to compute nearly-optimal graphs with n-nodes and m-edges. The solid black line in Figure 3 represents the value of λ_2 for these graphs. Finally, the dashed blue line in Figure 3 represents the upper bound on λ_2 given in (7) (compare also to Equation 18).

We observe in Figure 3 that nearly-optimal graphs have values which are indeed close to the upper bound on the algebraic connectivity, indicating (i) the upper bound is nearly-tight and (ii) the greedy heuristic (Algorithm 1) produces graphs which are nearly-optimal. We also observe that the algebraic connectivity of nearly-optimal graphs is significantly better than the values for an average Erdös-Rényi random graph.

5.3 Informativeness Of The Ranking For The Yahoo! Movie User Ratings Data Set

In this section, we apply the methodology formulated in Section 4, to study the Fisher informativeness of the Yahoo! Movie user rating data set. We show that the addition of targeted edges can significantly improve the informativeness of the movie rating system.

5.4 The Data Set

The Yahoo! Movie user rating data set consists of a $7,642 \times 11,915$ user-movie matrix where each of the 211,197 nonzero entries (0.23% sparsity density) is a 1 to 13 rating (yah).[4] Each movie was rated by between 1 and 4,238 users (the average number of reviews per movie is 17.7). Each user rated between 10 and 1,632 movies (the average number of reviews made by each reviewer is 27.6). Of the 70,977,655 (movie) pairs (i, j) where $i > j$, there

4. 34 entries reviewing Yahoo! movie_id 0 were discarded due to absence in movie content description file.

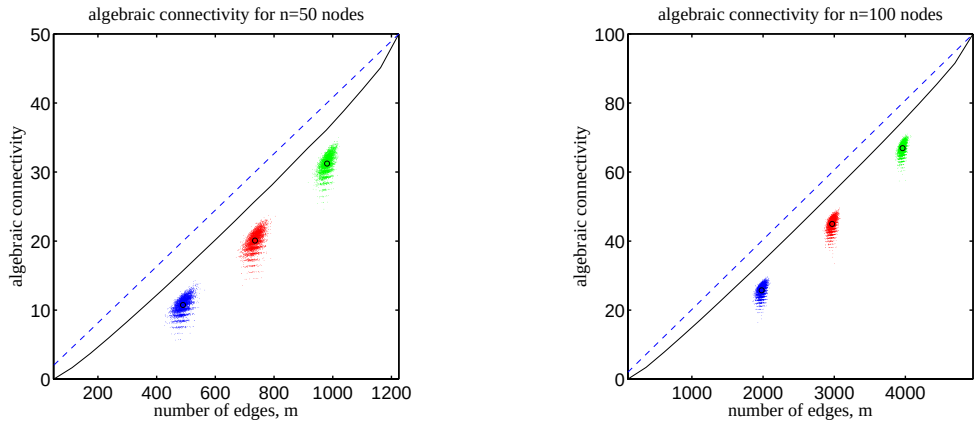


Figure 3: Algebraic connectivity, λ_2 as a function of m for 50- and 100-node graphs. The dashed blue line represents the upper bound on λ_2 given in (7). The solid black line represents the nearly-optimal value of λ_2. Finally, for $p = .4$ (blue), .6 (red), and .8 (green) we give a scatter plot of (m, λ_2) for 5,000 randomly generated Erdös-Rényi graphs. The mean values obtained are indicated by circles. See Section 5.2.

# times movie reviewed	1	2	3	4	5	6	7	8	9	≥ 10
occurrences	4,901	1,882	897	548	398	316	237	202	167	2,367

Table 2: Frequency of reviews for items in the Yahoo! Movie user rating data set. See Section 5.2.

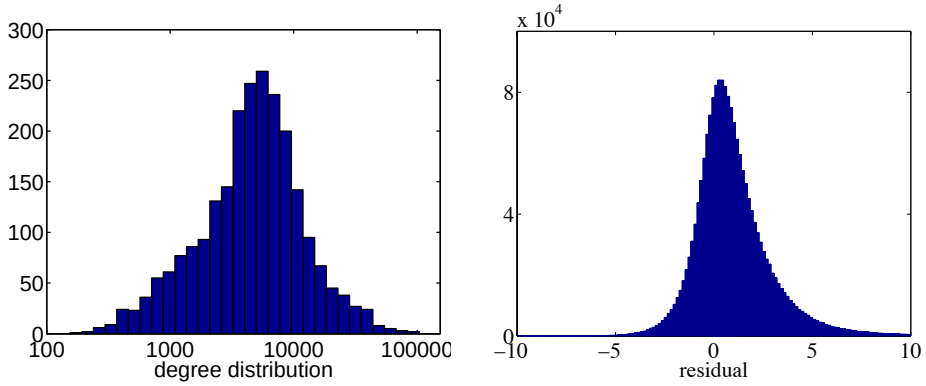
are 5,742,557 for which a user has given a rating to both movies i and j implying that the pairwise comparisons for the raw data set are 8.1% complete. The majority of movies in the data set received relatively few reviews, as reported in Table 2. The movies which received less than 10 rankings were discarded from the data set, leaving 2,367 movies, each of which were reviewed by an average of 79.8 users. We then removed 11 users who did not review any of the remaining movies. The remaining 7,631 reviewers reviewed between 1 and 1,220 movies (on average they reviewed 24.8 movies).

5.5 Construction Of Pairwise Comparison Data From Movie-User Rating Data

Let Σ be the set of Yahoo! users, V be the set of all Yahoo! movies and r_i^σ be the rating given to movie $i \in V$ by user $\sigma \in \Sigma$. For each unordered movie pair $\{i, j\} \in V^2$, we define

$$\Sigma_{ij} = \{\sigma \in \Sigma \text{ who rated both movies } i \text{ and } j\}.$$

For each movie pair $\{i, j\} \in V^2$, we define w_{ij} to be the number of users who have viewed both movies i and j, *i.e.*, $w_{ij} = |\Sigma_{ij}|$, and y_k to be the average difference in movie reviews,



$\hat{\phi}_w$	Movie Name
4.46	It's a Wonderful Life (1946)
4.45	Singin' in the Rain (1952)
4.34	Rear Window (1954)
4.11	24: Season 1 (2002)
3.96	The Longest Day (1962)
3.94	The Man Who Shot Liberty Valance (1962)
3.92	Rebecca (1940)
3.87	Friends - The Complete Fourth Season (1997)
3.79	Lady and the Tramp (1955)
3.79	It Happened One Night (1934)

Figure 4: **(top left)** A log-histogram of the w-weighted degree distribution for the graph representing the Yahoo! movie pairwise comparison data. **(top right)** A histogram of the residual, $y - B\hat{\phi}_w$, where $\hat{\phi}_w$ is the least squares ranking. **(bottom)** Top 10 movies and ranking, $\hat{\phi}_w$. See Section 5.3.

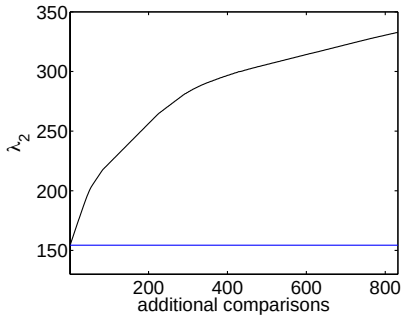
written

$$y_{ij} = \frac{1}{|\Sigma_{ij}|} \sum_{\sigma \in \Sigma_{ij}} (r_j^\sigma - r_i^\sigma), \text{ where } \{i,j\} \in V^2 \text{ and } i < j. \tag{19}$$

Note that the expression in parenthesis is anti-symmetric in the indices i and j and lies in the interval $[-12, 12]$. The choice $i < j$ corresponds to the choice in arc direction in (4). For the Yahoo! Movie user rating data set, we have $n := |V| = 2,367$, $N := \binom{n}{2} = 2,800,161$, $m := \|w\|_0 = 1,884,504$, and $M := \|w\|_1 = 8,322,538$. Thus, there exists at least one comparison for $m/N = 67\%$ of the movie pairs. The mean w-weighted degree of each node is given by $2 \cdot M/n = 3,516$. A log-histogram of the w-weighted degree distribution of the graph representing the pairwise comparison data is given in Figure 4 (top left).

5.6 The Least Squares Ranking

A ranking is obtained by solving the least squares problem, (1), using Matlab's `lsqr` function. The top ten movies found are given in Figure 4. The relative residual norm of the least squares estimator, $\hat{\phi}_w$, is $\frac{\|B\hat{\phi}_w - y\|_w}{\|y\|_w} = 0.53$. In Figure 4 (top right), we plot a histogram



method of data collection	additional comparisons original $\xi = 0$	.005% M $\xi = 416$	.01% M $\xi = 832$
random	154.38	154.38	154.38
optimal	154.38	**298.44**	**332.78**
upper bound (7)	7,035	7,035	7,036

Figure 5: **(left)** The informativeness of the ranking, $\lambda_2(w)$, as a small number ($.01\% \cdot M$) of targeted pairwise comparisons (black) and randomly selected pairwise comparisons (blue) are added. **(right)** The value of $\lambda_2(w)$ for this augmented data set and the upper bound on λ_2 given in (7). The change in informativeness for randomly added data is inappreciable compared to a 2.2 fold increase for targeted data. See Section 5.3.

of the residual, $y - B\hat{\phi}_w$. For this pairwise comparison data set, the normality assumption in Prop. 4.2 is reasonable.

The informativeness of the ranking is $\lambda_2(w) = [\mathrm{Var}(\hat{\phi}_w)]^{-1} = 154.38$. This value is small compared to the upper bound given in (7), $\lambda_2(w) \leq \frac{2M}{n-1} = 7,036$. We next demonstrate that the Fisher information can be significantly improved by the addition of a small number of targeted pairwise comparisons.

5.7 Targeted Data Collection

We apply the optimal experimental design approach developed in Section 4 to improve the Fisher information of the least squares ranking. To approximate the solution of (16), we use the greedy algorithm described in Algorithm 1. The second eigenpair of the graph Laplacian is computed using Matlab's `eigs` function, initialized using the eigenvector from the previous iteration. We choose a very modest value of pairwise comparison edges to add, $\xi = .01\% \cdot M = 832$ edges. The results are given in Figure 5. The addition of the targeted pairwise comparisons leads to an increase in the second eigenvalue of the w-weighted graph Laplacian by a factor of 2.2. The maximum increase for the addition of a single pairwise comparison is ≈ 1, less than the upper bound given in (9). We observe in Figure 5, that the rate of information increase slows as more pairwise comparisons are added. For a comparison, we also consider the addition of randomly chosen movie pairs. For this modest value of additional edges, ξ, the effect of the informativeness of the ranking is inappreciable.

Finally, we use graph visualization via spectral clustering to illustrate the pairwise comparison and targeted data. In Figure 6(top) we plot the pairwise movie comparisons obtained from the Yahoo! user-movie database. In Figure 6(bottom) we plot the proposed pairwise comparisons, targeted to improve the informativeness of the rating system. To enhance the readability of the graph representation, we plot only 15% randomly selected

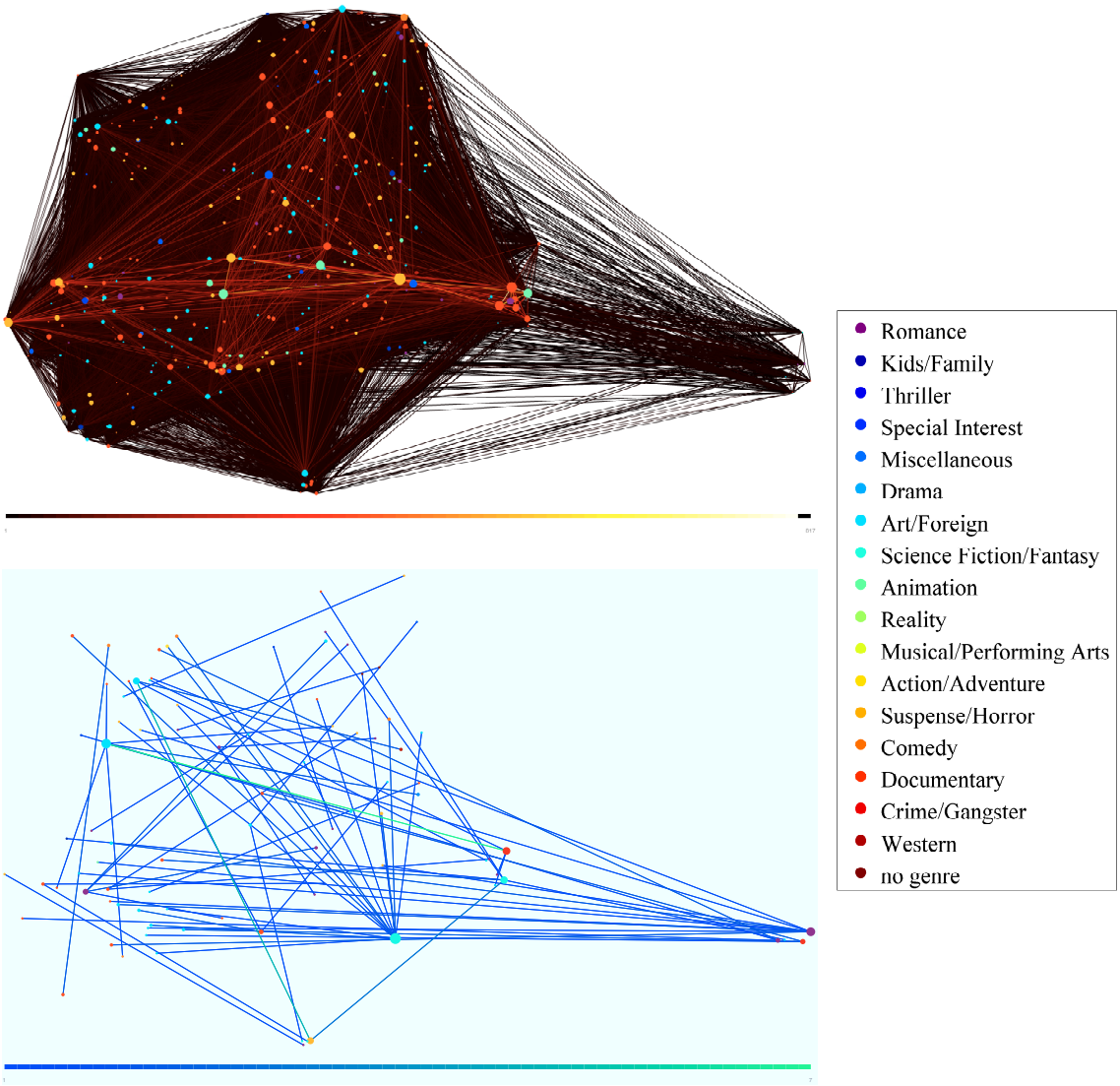


Figure 6: **Yahoo! Movie ratings and targeted data collection**. **(top)** A (15% randomly chosen) subset of the pairwise comparison graph for the Yahoo! user-movie database. Nodes represent movies, node size reflects weighted degree (*i.e.*, number of comparisons with other movies), and node color indicates genre (see legend). Edges represent weighted pairwise comparisons colored by edge weights (*i.e.*, number of comparisons). **(bottom)** Pairwise comparisons targeted for collection to improve the informativeness of the least squares ranking. Targeted comparisons are colored by weight (multiplicity). See Section 5.3.

nodes (356 of $n = 2367$) and the interconnecting edges ($45,327$ of $m = 1,884,504$). Figure 6(top) was generated as follows. First normalized spectral clustering (based on k-means) was used to detect clusters of movies. Next, the Fruchterman-Reingold algorithm was used to generate reasonable positions for the movie clusters and the Kamada-Kawai algorithm was used to place movies within the clusters (Traud et al., 2009). The node placement was obtained using the full data set. Finally, the weighted graphs were plotted using `wgPlot` (Wu, 2009). Figure 6(bottom) was then generated using the same node placements as in Figure 6(top).

A comparison of the top and bottom panels of Figure 6 shows that the primary improvement to informativeness arises from the addition of edges which connects two relatively weakly connected components of the graph. With 4 exceptions, each targeted movie pair is only incremented once; it isn't generally advantageous to add an edge multiple times.

5.8 2011-12 NCAA Division I Football Schedule

Recall from Section 1 that in sports the optimal pairwise data collection problem in equivalent to designing the schedule. In this section, we study the 2011-12 NCAA Division 1 football schedule, downloaded from Massey Ratings.[5] The NCAA Division 1 Football League is divided into the Football Bowl Subdivision (FBS) and Football Championship Subdivision (FCS).[6] The FBS is further decomposed into 12 conferences and the FCS into 15. Of the 246 teams in Division 1, 120 belong to FBS and 126 belong to FCS. Lafayette College is a member of FBS, however every opponent of Lafayette during the 2011-12 season was a member of the FCS. For our purposes, it is more convenient to reclassify Lafayette as a member of FCS and thus, in what follows, FBS has 119 teams and FCS has 127. There were $m = 1430$ games among the Division 1 teams and $m = 693$ games among the FBS teams.

For static schedules, an important statistic is the the ratio of the total number of games played to the total number of teams. For example, in Major League Baseball (MLB), there are 30 teams, divided into two leagues: the American League (14 teams) and the National League (16 teams). During the regular season, each team plays approximately 160 games, primarily against teams within the same division. Thus, within each league, teams play an average of $160/15 \approx 10$ times. With so many games and equal strength of schedule among teams, it is intuitive that the scheduling has little effect on the rankings. And, in fact, MLB simply uses win/loss percentages for ranking purposes. In the NCAA football considered here however, there are 120 teams in the NCAA Football Bowl Subdivision (FBS) and each team plays approximately 6 games per year within FBS. Thus each team only plays roughly 5% of the other teams. There are several rankings for NCAA football which are generated either mathematically or by expert opinion and then aggregated to determine official rankings and select teams to compete in the prestigious end-of-season "bowl games". The fact that these rankings generally disagree and that none of them is more reliable than the others suggest that none of them are very informative. It is this

5. These were obtained from `http://masseyratings.com/scores.php?t=11590&s=107811&all=1&mode=2&format=0`
6. These were formally known as Division 1-A and 1-AA respectively.

situation, where there are relatively few games compared to the number of teams, that the schedule has a large effect on the rankings.

5.9 Data Visualization Via Spectral Clustering

We use the data visualization method described below to demonstrate that NCAA Division 1 teams primarily play against other teams within their own conference. We then show that this clustering of teams by conference results in the graph having poor algebraic connectivity.

We first use normalized spectral clustering to detect communities within the teams (Shi and Malik, 2000). This, in turn, relies on the k-means algorithm where k is the desired number of communities (27 for Division 1 and 12 for Division 1 FBS). Then, using the Matlab toolbox described in Traud et al. (2009), the Fruchterman-Reingold algorithm finds an optimal placement of the communities and the Kamada-Kawai algorithm is used for the placement of nodes within each community. The mean within-cluster sum of point-to-centroid distances for the k-means clustering obtained for the Division 1 and Division 1 FBS data is 0.147 and 0.133 respectively.

In Figures 7 and 8, we plot the 2011-12 NCAA Division 1 and Division 1 FBS football schedules respectively. In 7(top) and 8(top), the vertices represent teams, the edges represent games, and each vertex (team) is colored by conference membership. In 7(bottom) and 8(bottom), the vertices represent the spectrally clustered communities and the edges represent the community interactions. We observe from Figures 7 and 8 that the teams primarily play within their own conference, which has implications discussed below.

We next compare the value of the algebraic connectivity for these schedules with schedules from Erdös-Rényi random graphs and proposed nearly-optimal schedules.

5.10 Comparison Of NCAA Division 1, Erdös-Rényi Random And Nearly-Optimal Schedules

In the introduction, we noted that there are several common scalar measures of $\mathrm{Var}(\hat{\phi}_w)$, three of which are given in (3). In this section, we compare these various measures for the NCAA Division 1, Erdös-Rényi random, and nearly-optimal schedules.

More concretely, let w be a given schedule (defining a graph on n vertices) and define the graph Laplacian: $\Delta_w := B^t[\mathrm{diag}(w)]B$. Define the following three functions of w:

$$J_E(w) := \lambda_2(w) \tag{20a}$$

$$J_A(w) := \left[\frac{1}{n}\mathrm{tr}(\Delta_w^\dagger)\right]^{-1} = \left[\frac{1}{n}\sum_{i\geq 2}\frac{1}{\lambda_i(w)}\right]^{-1} \tag{20b}$$

$$J_D(w) := \log[\det(\Delta_w)]^{\frac{1}{n}} = \frac{1}{n}\sum_{i\geq 2}\log[\lambda_i(w)]. \tag{20c}$$

To obtain quantities more comparable to those for the E-optimality condition, for $J_A(w)$ we have used the harmonic mean of the eigenvalues rather than the negative of the inverses as in (15b) and for $J_D(w)$, we have taken the logarithm of the determinant in (15c). An interpretation of the three quantities defined in (20) in terms of the graph is given in Remark 1.

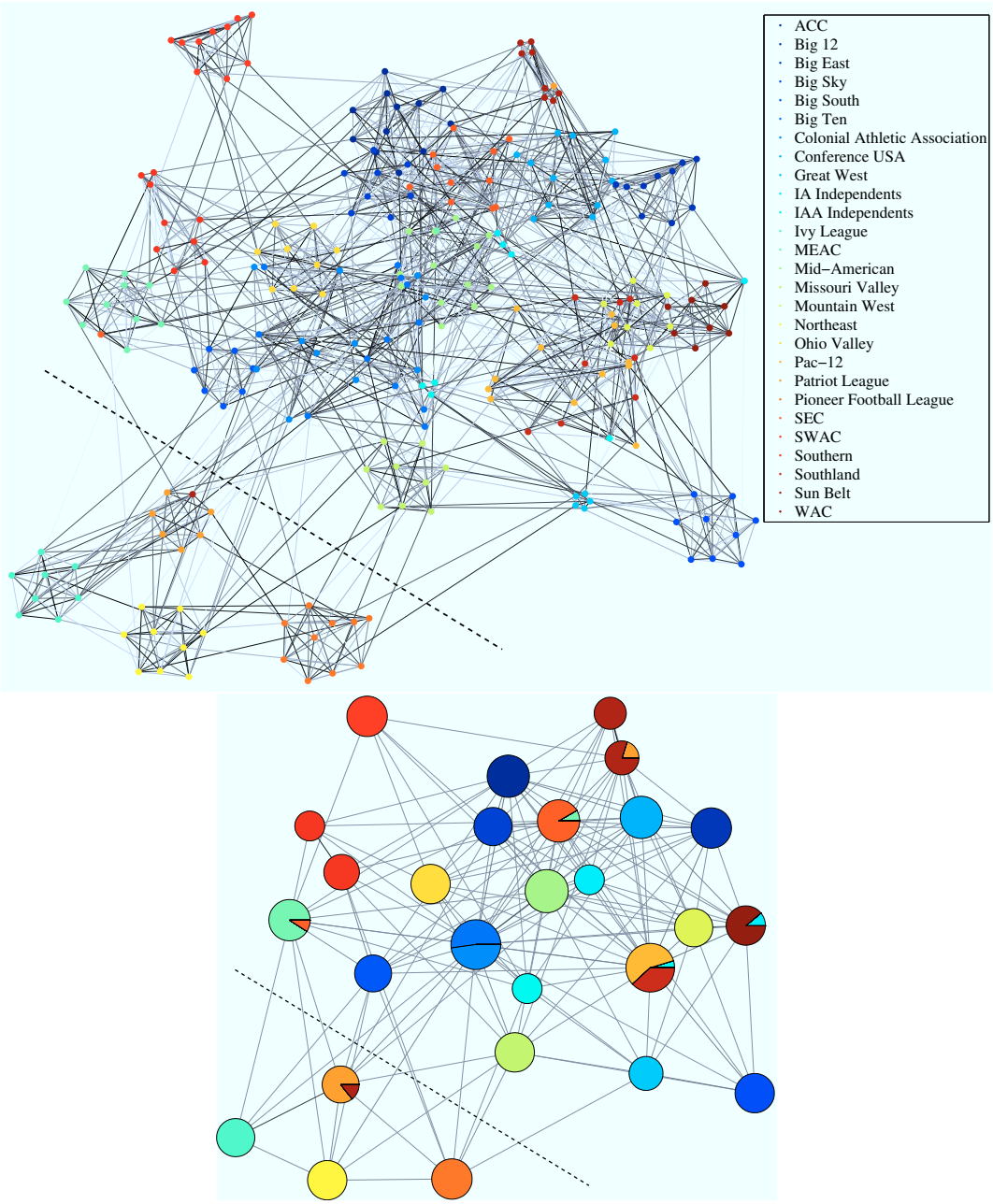


Figure 7: **2011-12 NCAA Division 1 (FBS and FCS) football schedule**. Graph representation of schedule via spectral clustering by games, *top:* vertices represent teams, edges represent games, coloring indicates conference membership. *bottom:* community detection of teams (represented using pie-graphs) reveals that teams primarily play within their own conference. The dashed lines indicate an edge cut which is discussed in the text. See Section 5.8.

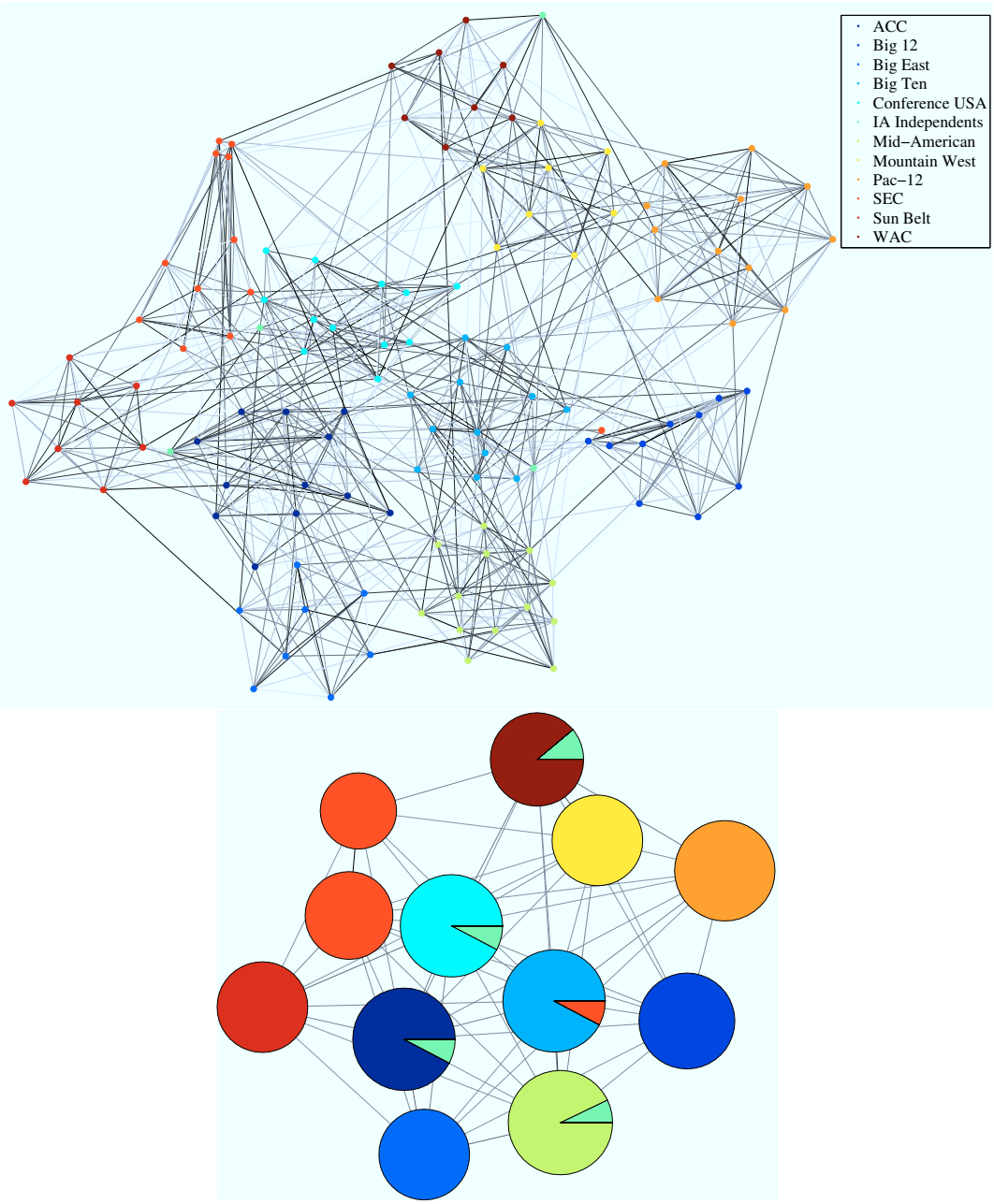


Figure 8: **2011-12 NCAA Division 1 (only FBS) football schedule**. Graph representation of schedule via spectral clustering by games, *top:* vertices represent teams, edges represent games, coloring indicates conference membership. *bottom:* community detection of teams (represented using pie-graphs) reveals that teams primarily play within their own conference. See Section 5.8.

For the Division 1 and Division 1 FBS schedules, we compute the various measures of the quality of schedule given in (20) and record them in Table 3. We also plot $J_E(w)$ given in (20a) in Figure 9 by a red diamond. We next discuss schedules for which we compare the Division 1 and Division 1 FBS schedules in Table 3 and Figure 9.

The expected number of edges for a $G(n,p)$ Erdös-Rényi random graph is pN where $N := \binom{n}{2}$. To compare to the football schedules, we take $p = m/N$ and consider the family of random graphs, $G(n, m/N)$. For $n = 119$ and $m = 693$, we choose $p = m/N \approx 0.0987$ which is approximately 2.5 times the threshold for connectivity, $p_c = \log(n)/n \approx 0.0402$. For $n = 246$ and $m = 1430$, we choose $p = m/N \approx 0.0475$ which is approximately 2.1 times the threshold for connectivity, $p_c = \log(n)/n \approx 0.0224$. In Table 3, we tabulate the expected values of the three quantities given in (20) for $G(n, m/N)$ graphs, obtained by averaging over a sample size of 1000. Similar to Section 5.2, in Figure 9, we give a scatter plot of (m, λ_2) for $G(n, m/N)$ graphs and indicate the mean values with a blue circle.

As in Section 5.2 and Section 5.3, we again use the greedy algorithm described in Section 3.1 (see Algorithm 1) to compute graphs with n nodes and m edges which nearly-maximize $J_E = \lambda_2$. We then evaluate all three quantities given in (20) for these graphs and tabulate these values in Table 3. The solid black line in Figure 9 is the best value of $J_E = \lambda_2$ obtained. Finally, the dashed blue line in Figure 9 represents the upper bound on λ_2 given in (7).

We observe in Figure 9 and Table 3 that the schedules which nearly-maximize $J_E(w) = \lambda_2$ have significantly larger values of J_E than the NCAA Division 1 and Division 1 FBS schedules. In fact, the NCAA schedules have worse values than schedules associated with Erdös-Rényi random graphs of the same size. Furthermore, we show in Table 3 that schedules which maximize J_E also have larger values of J_A and J_D. That is, the schedules which are good in the sense of E-optimality are also good schedules in the sense of D- and E-optimality as defined in (3).

The reason for the relatively poor value of $J_E(w) = \lambda_2$ for the NCAA Division 1 and Division 1 FBS schedules can be understood from Figures 7 and 8. Figures 7 and 8 reveal that teams primarily play within their own conference. This results in a small edge cut between a conference (or set of conferences) and its vertex complement, which, by (6), implies a small algebraic connectivity. For example, the edge cut indicated by the dashed line in Figure 7 (entire NCAA Division 1 schedule) results in an upper bound on the algebraic connectivity of 1.297. The edge cut obtained by considering the set consisting of teams in the SWAC conference yields an upper bound equal to 1.043. Both of these bounds are already less than the expected value of λ_2 for Erdös-Rényi random graphs of comparable size (compare with the top part of the first column in Table 3). To summarize, the NCAA primarily schedules games among teams within the same conferences and this reduces the informativeness of the rankings.

The schedule design methodology advocated in (16) is flexible in the following two senses: (i) The optimal schedules contain symmetry with respect to permutations in the seeding of the teams. This problem has been studied previously for tournaments; see the discussion in Section 2. (ii) The optimal schedule is *not* time dependent and thus the scheduling of future games does *not* depend on past game performances, *i.e.*, the schedule is completely known before the season begins and the games may be played in *any* order. These properties can be exploited in the further design of the schedule.

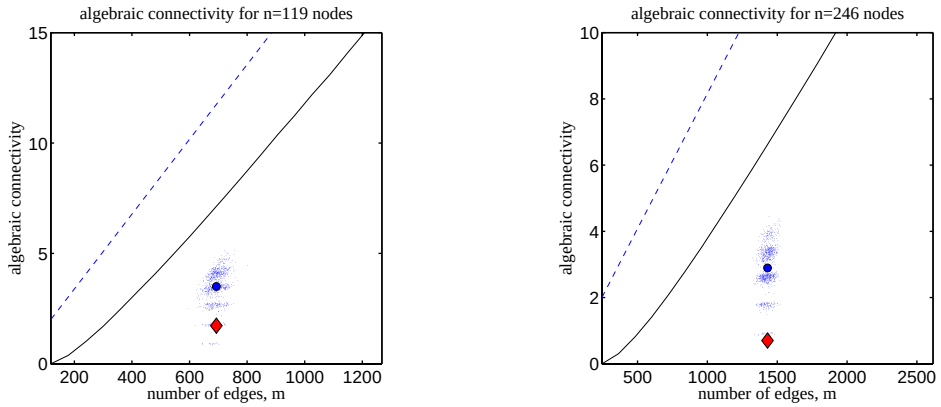


Figure 9: A comparison of $J_E(w) = \lambda_2$ defined in (20a) for the Division 1 and Division 1 FBS schedules, Erdös-Rényi random schedules, and schedules which nearly-maximize λ_2. The red diamonds represents the 2011 NCAA Division 1 (right) and Division 1 FBS (left) football schedule. The solid black lines represent the nearly-optimal values of λ_2 obtained for $n = 119$ (left) and $n = 246$ (right). The dashed blue lines represent the upper bound on λ_2 given in (7). The blue dots represent a scatter plot of (m, λ_2) for 1,000 randomly generated Erdös-Rényi graphs, $G(n, m/N)$. The mean values are indicated by blue circles. See Section 5.8.

	$J_E(w)$ in (20a)	$J_A(w)$ in (20b)	$J_D(w)$ in (20c)
Div. 1 FBS and FCS	0.7015	8.780	2.363
Erdös-Rényi, $n = 246$	2.892	9.681	2.358
E-optimal design, $n = 246$	**6.630**	**10.71**	**2.403**
Div. 1 FBS	1.725	9.634	2.372
Erdös-Rényi, $n = 119$	3.497	9.911	2.361
E-optimal design, $n = 119$	**7.142**	**10.92**	**2.402**

Table 3: A comparison of the three objective functions defined in (20) for the Division 1 and Division 1 FBS schedules, Erdös-Rényi random schedules, and schedules which nearly-maximize $J_E(w) = \lambda_2$. Schedules which nearly-maximize $J_E(w) = \lambda_2$ also have larger values of J_A and J_D than the comparison schedules. See Section 5.8

5.11 Synthetic Data Experiment On The 2011-2012 NCAA Division 1 FBS Graph

To further illustrate and test our proposed active learning method, we again consider the graph generated in Section 5.8 from the 2011-12 NCAA Division I Football Bowl Subdivision (FBS) schedule with $n = 119$ nodes and $m = 693$ edges, as shown in Figure 8. We take as ground truth rating, ϕ, a normally distributed vector with mean zero and variance, $\sigma^2 = 1$. The ground truth rating, ϕ, is used to generate new data according to the normal model

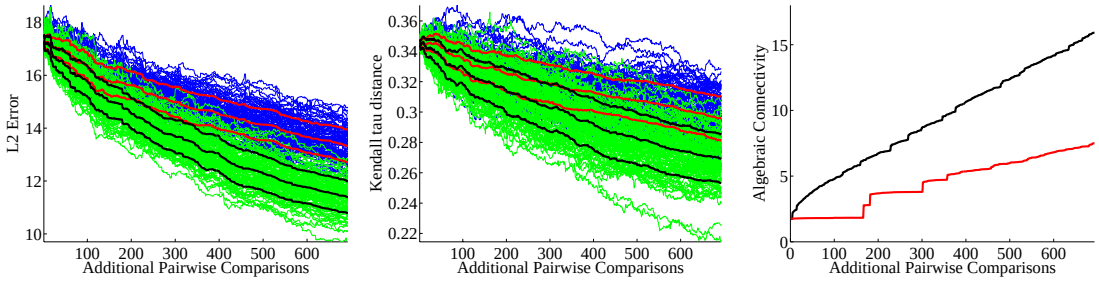


Figure 10: A comparison of ranking errors and algebraic connectivity, a measure of the informativeness of the ranking, for two data collection strategies: the proposed active sampling method (black and green) and random sampling (red and blue). **(left)** The L^2-error, $\|\hat{\phi}_\xi - \phi\|_2$ between the estimated and ground truth rankings. **(center)** The Kendall-τ rank distance, (21), between the estimated and ground truth rankings. **(right)** The algebraic connectivity of the graph representing the data set. See Section 5.11.

(11) with $\sigma^2 = 5$. With this data, we solve (12) to obtain a least squares estimate, $\hat{\phi}_{w_0}$. We compute $\|\hat{\phi}_{w_0} - \phi\|_2 = 17.31$ and $K(\phi_{w_0}, \phi) = 0.35$. Here, the *Kendall-$\tau$ rank distance* between two rankings ϕ_1 and ϕ_2 is defined as the fraction of pairwise disagreements between the rankings,

$$K(\phi_1, \phi_2) := \frac{\#\{(i,j) \colon i > j,\ \phi_1(i) < \phi_1(j),\ \text{and } \phi_2(i) > \phi_2(j)\}}{n(n-1)/2}. \tag{21}$$

We then consider enhancing the data set by adding ξ more pairwise comparisons. Using the enhanced data set, we compute an estimate of the ranking, $\hat{\phi}_\xi$, and, as $\hat{\phi}_\xi$ is an unbiased estimate of ϕ, expect $\|\hat{\phi}_\xi - \phi\|_2$ to diminish as $\xi \to \infty$. We choose $\xi = 693$, so that the number of pairwise comparisons (games played) is doubled. As in Section 5.3, we add pairwise comparisons either by the greedy algorithm (Algorithm 1) or by random selection. As with the data collected on the initial graph, the new data are collected according to the normal model (11) with $\sigma^2 = 5$. In Figure 10(left), we plot the number of additional pairwise comparisons vs. the L^2-error, $\|\hat{\phi}_\xi - \phi\|_2$, for an ensemble of ranking estimates determined using the two data collection strategies. The (thin) blue and green lines represent the error for 100 instances of data collection using the random and greedy methods respectively. The (thick) red and black lines represent the mean and mean plus/minus one standard deviation for each of the two data collection strategies. For $\xi = 693$, the mean L^2-error for the proposed data collection strategy is 11.38 while the mean error for the random data collection strategy is 13.32, representing a reduction in error of 34% and 23% respectively. In Figure 10(center), we plot the number of additional pairwise comparisons vs. the Kendall-τ rank distance, $K(\phi, \phi_\xi)$, for these two data collection strategies. For $\xi = 693$, the mean distance for the proposed data collection strategy is .27 while the mean error for the random data collection strategy is 0.30, representing a reduction in distance of 22% and 14% respectively. In Figure 10(right), we plot the algebraic connectivity, a measure of the informativeness of the ranking, vs. the number of additional pairwise comparisons. The black (red) line is the

algebraic connectivity for the graph representing the data set where edges are added using the greedy algorithm (random sampling). Supported by Propositions 4.1 and 4.2, the data set represented by a graph with larger algebraic connectivity is more informative and thus produces a ranking estimate with greater fidelity to the ground truth estimate.

6. Discussion And Future Directions

We have applied methods from optimal experiment design to provide a new framework for data collection for more informative statistical rankings. At the heart of this framework is a bi-level optimization problem (2) where the inner problem is to determine the unbiased ranking for a given schedule and the outer problem is to identify data which maximizes the Fisher information of the ranking. For the least-squares estimate, the outer problem decouples from the inner problem and reduces to an eigenvalue optimization problem. For the E-optimality criterion for the Fisher information, this is the problem of finding an edge weight $w \in \mathbb{Z}_+^N$, such that the w-weighted graph Laplacian has large second eigenvalue (16). This can be interpreted as finding a multigraph with large algebraic connectivity, a problem which has been well-studied in graph theory. In the case of NCAA Division 1 football, we demonstrated in Section 5.8 and Table 3 that the nearly-optimal data collection strategy in the sense of E-optimality is also a good strategy in the sense of D- and A-optimality; the choice of scalar function $f\colon \mathbb{S}_+^n \to \mathbb{R}$ as defined in (2) does *not* strongly effect the optimal data collection strategy (see Remark 1 for a further discussion of these optimality criteria). Furthermore, in Section 5.11, we demonstrate using a synthetically constructed data set on this graph that the ranking estimate obtained via active sampling has greater fidelity to ground truth than the ranking estimate obtained via random sampling.

There are several applications in, *e.g.*, social networking, game theory, and e-commerce, where improved data collection could potentially benefit ranking. In particular, for the Yahoo! Movie user ratings data set (considered in Section 5.3), we have shown that the informativeness of ranking can be increased by a factor of 2.2 if just .01% of additional optimally-targeted pairwise comparisons are added to the data set. In contrast, if the same amount of random data is added, there is an inappreciable effect on the informativeness of the ranking. For this application, the data collection problem could be more carefully modeled. Here, the pairwise comparison data is constructed from user rating data and thus any targeted pairwise comparison addition must be solicited from a user. Since the number of pairwise comparisons for which a particular reviewer adds when a new movie is reviewed is equal to the number of previous reviews that user has contributed, it may make sense to solicit additional reviews from users with many previous reviews. That is, the propagation of information from the user reviews to the pairwise comparison data in (19) should also be considered.

We have focused on optimal data collection for improved rankings, neglecting several important factors including the cost of data collection and potential constraints on what data may be collected. There are two simple extensions to our method which may be employed to accommodate these additional factors. The cost of data collection could be incorporated by either adding a penalization term in (16) or by incorporating additional weights into the norm used to compute λ_2 in (16). Data collection constraints may be handled by explicitly

forbidding certain edge weights to be incremented in the greedy Algorithm 1 for targeting data collection.

The least-squares ranking estimate (1) is referred to as HodgeRank by some authors (Jiang et al., 2010; Xu et al., 2011), since the Hodge decomposition implies that the residual in (1), $r = B\phi - y$, can be further decomposed into two orthogonal components: (1) a divergence-free component which consists of 3-cycles and (2) a harmonic component which consists of longer cycles (Jiang et al., 2010; Hirani et al., 2011). In fact, Jiang et al. (2010) argues that a data set which has a large harmonic component is inherently inconsistent and does not have a reasonable ranking. The harmonic component lies in the kernel of the graph Helmholtzian with dimension given by the first Betti number of the associated simplical complex. Optimal reduction of the first Betti number may provide an alternative approach to improving the informativeness of the least squares ranking.

Recently, Masuda et al. (2013) developed an algorithm for removing nodes from a graph to increase the algebraic connectivity. This algorithm could be used to prune the alternatives in a data set to increase the informativeness of a ranking.

Finally, we are interested in extending this work to nonlinear ranking methods, including robust estimators (Osting et al., 2013b), random walker methods (Callaghan et al., 2007), Perron-Frobenius eigenvalue methods (Keener, 1993; Langville and Meyer, 2012), and Elo methods (Elo, 1978; Glickman, 1995; Langville and Meyer, 2012).

Acknowledgments

We thank Lawrence Carin, Jérôme Darbon, Mark L. Green, and Yuan Yao for useful discussions. B. Osting is supported by NSF DMS-1103959. C. Brune is supported by ONR grants N00014-10-10221 and N00014-12-10040. S. Osher is supported by ONR N00014-08-1-1119, N00014-10-10221, and NSF DMS-0914561.

References

Yahoo! Webscope dataset: ydata-ymovies-user-movie-ratings-content-v1_0. `http://webscope.sandbox.yahoo.com`. accessed: 10/5/2011.

N. 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.

N. Ailon, R. Begleiter, and E. Ezra. Active learning using smooth relative regret approximations with applications. *Journal of Machine Learning Research*, 15:885–920, 2014.

D. Barrow, I. Drayer, P. Elliott, G. Gaut, and B. Osting. Ranking rankings: an empirical comparison of the predictive power of sports ranking methods. *Journal of Quantitative Analysis in Sports*, 9(2):187–202, 2013.

E. Ben-Naim and N. W. Hengartner. Efficiency of competitions. *Physical Review E*, 76: 026106, 2007.

T. Biyikoglu, J. Leydold, and P. F. Stadler. *Laplacian Eigenvectors of Graphs*. Springer, 2007.

A. Björner, L. Lovász, and P. W. Shor. Chip-firing games on graphs. *European Journal of Combinatorics*, 12(4), 1991.

N. Boumal, A. Singer, P.-A. Absil, and V. D. Blondel. Cramér-Rao bounds for synchronization of rotations. *Information and Inference*, 3(1):1–39, 2014.

S. Boyd, P. Diaconis, and L. Xiao. Fastest mixing Markov chain on a graph. *SIAM Review*, 46(4):667–689, 2004.

T. Callaghan, P. J. Mucha, and M. A. Porter. Random walker ranking for NCAA Division I-A football. *American Mathematical Monthly*, 114(9):761–777, 2007.

F. R. K. Chung. *Spectral Graph Theory*. AMS, 1997.

F. R. K. Chung and M. Radcliffe. On the spectra of general random graphs. *Electronic Journal of Combinatorics*, 18(1):215–229, 2011.

F. R. K. Chung, L. Lu, and V. H. Vu. The spectra of random graphs with given expected degrees. *Internet Mathematics*, 1:257–275, 2003.

M. Chung and E. Haber. Experimental design for biological systems. *SIAM Journal on Control and Optimization*, 50(1):471–489, 2012.

A. Coja-Oghlan. On the Laplacian eigenvalues of $G_{n,p}$. *Combinatorics, Probability, & Computing*, 16(6):923–946, 2007.

H. A. David. *The Method of Paired Comparisons*. Charles Griffin & Co., 1963.

J. J. DiStefano 3rd. Tracer experiment design for unique identification of nonlinear physiological systems. *American Journal of Physiology*, 230(2):476–485, 1976.

C. D'Souza. Optimising a tournament for use with ranking algorithms. preprint, 2010.

A. Eichhorn, P. Ni, and R. Eg. Randomised pair comparison: an economic and robust method for audiovisual quality assessment. In *Proceedings of the 20th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 63–68. ACM, 2010.

A. Elo. *The Rating of Chessplayers, Past and Present*. Arco Pub., 1978.

S. M. Fallat, S. Kirkland, and S. Pati. On graphs with algebraic connectivity equal to minimum edge density. *Linear Algebra and its Applications*, 373:31–50, 2003.

V. V. Fedorov. *Theory of Optimal Experiments*. Academic Press, 1972.

U. Feige and E. Ofek. Spectral techniques applied to sparse random graphs. *Random Structures & Algorithms*, 27(2):251–275, 2005.

M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23:298–305, 1973.

A. Ghosh and S. Boyd. Upper bounds on algebraic connectivity via convex optimization. *Linear Algebra and its Applications*, 418:693–707, 2006a.

A. Ghosh and S. Boyd. Growing well-connected graphs. *Proc. IEEE Conf. Decision & Control*, 2006b.

A. Ghosh, S. Boyd, and A. Saberi. Minimizing effective resistance of a graph. *SIAM Review*, 50(1):37–66, 2008.

M. E. Glickman. A comprehensive guide to chess ratings. *American Chess Journal*, 3, 1995.

M. E. Glickman. Adaptive paired comparison design. *Journal of Statistical Planning and Inference*, 127:279–293, 2005.

M. E. Glickman. Bayesian locally optimal design of knockout tournaments. *Journal of Statistical Planning and Inference*, 138(7):2117 – 2127, 2008.

G. Grimmett. *Probability on Graphs: Random Processes on Graphs and Lattices*. Cambridge University Press, 2010.

E. Haber, L. Horesh, and L. Tenorio. Numerical methods for experimental design of large-scale linear ill-posed inverse problems. *Inverse Problems*, 24:055012, 2008.

A. N. Hirani, K. Kalyanaraman, and S. Watts. Least squares ranking on graphs. arXiv:1011.1716v4, 2011.

L. Horesh, E. Haber, and L. Tenorio. Optimal experimental design for the large-scale nonlinear ill-posed problem of impedance imaging. In L. Biegler et al, editor, *in: Large-Scale Inverse Problems and Quantification of Uncertainty*. Wiley Ser. Comp. Stat., 2011.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

A. Jamakovic and P. Van Mieghem. On the robustness of complex networks by using the algebraic connectivity. In *Networking 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, pages 183–194. Springer, 2008.

A. Jamakovic and S. Uhlig. On the relationship between the algebraic connectivity and graph's robustness to node and link failures. In *Proceedings of the 3rd EURO-NGI Conference on Next Generation Internet Network*, 2007.

K. G. Jamieson and R. D. Nowak. Active ranking in practice: General ranking functions with sample complexity bounds. NIPS Workshop, 2011a.

K. G. Jamieson and R. D. Nowak. Active ranking using pairwise comparisons. In *Neural Information Processing Systems (NIPS)*, pages 2240–2248, 2011b.

X. Jiang, L.-H. Lim, Y. Yao, and Y. Ye. Statistical ranking and combinatorial Hodge theory. *Math. Program. Ser. B*, 127(1):203–244, 2010.

F. Juhász. The asymptotic behaviour of Fiedler's algebraic connectivity for random graphs. *Discrete Mathematics*, 96(1):59–63, 1991.

J. P. Keener. The Perron-Frobenius theorem and the ranking of football teams. *SIAM Review*, 35(1):80–93, 1993.

T. Kolokolnikov. Maximizing algebraic connectivity for certain families of graphs. Manuscript in preparation, 2014.

T. Kolokolnikov, B. Osting, and J. Von Brecht. Algebraic connectivity of Erdös-Rényi graphs near the connectivity threshold. Manuscript in preparation, 2014.

A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *The Journal of Machine Learning Research*, 9:235–284, 2008.

A. N. Langville and C. D. Meyer. *Who's #1?: The Science of Rating and Ranking*. Princeton University Press, 2012.

N. Masuda, T. Fujie, and K. Murota. Application of semidefinite programming to maximize the spectral gap produced by node removal. In *Complex Networks IV*, pages 155–163. Springer, 2013.

V. B. Melas. *Functional approach to optimal experimental design*. Springer, 2006.

B. Mohar. The Laplacian spectrum of graphs. In *Graph Theory, Combinatorics, and Applications*, volume 2, pages 871–898. Wiley, 1991.

D. Mosk-Aoyama. Maximum algebraic connectivity augmentation is NP-hard. *Operations Research Letters*, 36(6):677–679, 2008.

R. Olfati-Saber, A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

R. I. Oliveira. Concentration of the adjacency matrix and of the Laplacian in random graphs with independent edges. arXiv:0911.0600v2, 2009.

B. Osting, C. Brune, and S. Osher. Enhanced statistical rankings via targeted data collection. In *Proceedings of The 30th International Conference on Machine Learning (ICML)*, volume 28, pages 489–497. JMLR, W&CP, 2013a.

B. Osting, J. Darbon, and S. Osher. Statistical ranking using the ℓ^1-norm on graphs. *AIMS J. Inverse Problems and Imaging*, 7(3):907–926, 2013b.

F. Pukelsheim. *Optimal Design of Experiments*. SIAM, 2006.

G. P. Quinn and M. J. Keough. *Experimental Design and Analysis for Biologists*. Cambridge University Press, 2002.

P. A. Scarf and M. M. Yusof. A numerical study of tournament structure and seeding policy for the soccer world cup finals. *Statistica Neerlandica*, 65(1):43–57, 2011.

M. Seeger and H. Nickisch. Large scale Bayesian inference and experimental design for sparse linear models. *SIAM Journal of Imaging Sciences*, 4(1):166–199, 2011.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

J. Silva and L. Carin. Active learning for online Bayesian matrix factorization. In *18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2012.

J. Sun, S. Boyd, L. Xiao, and P. Diaconis. The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem. *SIAM Review*, 48:2006, 2004.

A. L. Traud, C. Frost, P. J. Mucha, and M. A. Porter. Visualization of communities in networks. *Chaos*, 19:041104, 2009.

T. Vu, A. Altman, and Y. Shoham. On the complexity of schedule control problems for knockout tournaments. In *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AA-MAS 2009) in Budapest, Hungary*, 2009.

H. Wang and P. Van Mieghem. Algebraic connectivity optimization via link addition. In *Bionetics 2008, Hyogo, Japan*, 2008.

F. L. Wauthier, M. I. Jordan, and N. Jojic. Efficient ranking from pairwise comparisons. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

M. Wu. wgPlot. `http://www.mathworks.com/matlabcentral/fileexchange/24035`, 2009.

Q. Xu, Y. Yao, T. Jiang, Q. Huang, B. Yan, and W. Lin. Random partial paired comparison for subjective video quality assessment via HodgeRank. In *ACM Multimedia*, 2011.

Q. Xu, Q. Huang, and Y. Yao. Online crowdsourcing subjective image quality assessment. In *Proceedings of the 20th ACM International Conference on Multimedia*, pages 359–368. ACM, 2012.

Journal of Machine Learning Research 15 (2014) 3013-3036 Submitted 2/14; Revised 5/14; Published 10/14

Bayesian Co-Boosting for Multi-modal Gesture Recognition

Jiaxiang Wu JIAXIANG.WU@NLPR.IA.AC.CN
Jian Cheng* JCHENG@NLPR.IA.AC.CN
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
Beijing, 100190, China

Editor: Isabelle Guyon, Vassilis Athitsos, Sergio Escalera

Abstract

With the development of data acquisition equipment, more and more modalities become available for gesture recognition. However, there still exist two critical issues for multi-modal gesture recognition: how to select discriminative features for recognition and how to fuse features from different modalities. In this paper, we propose a novel Bayesian Co-Boosting framework for multi-modal gesture recognition. Inspired by boosting learning and co-training method, our proposed framework combines multiple collaboratively trained weak classifiers to construct the final strong classifier for the recognition task. During each iteration round, we randomly sample a number of feature subsets and estimate weak classifier's parameters for each subset. The optimal weak classifier and its corresponding feature subset are retained for strong classifier construction. Furthermore, we define an upper bound of training error and derive the update rule of instance's weight, which guarantees the error upper bound to be minimized through iterations. For demonstration, we present an implementation of our framework using hidden Markov models as weak classifiers. We perform extensive experiments using the ChaLearn MMGR and ChAirGest data sets, in which our approach achieves 97.63% and 96.53% accuracy respectively on each publicly available data set.

Keywords: gesture recognition, Bayesian co-boosting, hidden Markov model, multi-modal fusion, feature selection

1. Introduction

As one of the most natural and intuitive ways for human computer interaction, gesture recognition has been attracting more and more attention from academe and industry. With automatic gesture recognition techniques, one can use his/her hands to freely interact with computers. It has been widely applied to sign language recognition (Zafrulla et al., 2011; Oz and Leu, 2011), robot control (Raheja et al., 2010), games (Roccetti et al., 2011), etc. In the early days, accelerometer-based approaches were especially popular for gesture recognition, due to their simpleness and accuracy in data acquirement (Mantyla et al., 2000; Chambers et al., 2002; Pylvänäinen, 2005; Liu et al., 2009). As an extension to the accelerometer, the inertial measurement unit (IMU) can be adopted to collect more information, such as linear acceleration and angular acceleration. There are also several IMU-based gesture recognition methods proposed recently (Zhang et al., 2013; Yin and

*. Corresponding author.

Davis, 2013). Nevertheless, the requirement of wearing accelerometers or IMUs limits the applicability of the above approaches. Vision-based approaches, which do not need to wear any extra devices, offer an appealing approach to gesture recognition. However, vision-based approaches are vulnerable to illumination, self-occlusion, and variation of gesture. Moreover, visual feature representation is still an open problem.

As an alternative, depth-aware camera (e.g., Microsoft® Kinect™) can capture RGB image, depth image, and audio, which makes gesture recognition less sensitive to illumination changes, self-occlusion, and can offer strong information for background removal, object detection, and localization in 3D space. With the prevalence of depth-aware camera, the study of gesture recognition is extremely stimulated and multi-modal based approaches are becoming a hot topic. Recently, there are many research works to utilize multiple modalities acquired by depth-aware camera for gesture recognition (Wu et al., 2012; Lui, 2012a; Malgireddy et al., 2012; Bayer and Silbermann, 2013; Nandakumar et al., 2013; Chen and Koskela, 2013). Since 2011, ChaLearn has organized a series of competitions based on the multi-modal gesture data captured by Kinect™. The tasks include one-shot-learning of gestures (Guyon et al., 2012) and continuous gesture spotting and recognition (Escalera et al., 2013). Many of participants achieved satisfactory performances on gesture recognition. However, for multi-modal based approaches, there still exist two critical issues for gesture recognition: how to select discriminative features for recognition, and how to fuse features from different modalities.

In the context of dynamic gesture recognition, an instance is represented by a time series sequence. Most of existing feature extraction methods for time series are mainly based on the self-defined criterion functions to evaluate each feature dimension's contribution (Kashyap, 1978; Mörchen, 2003; Yoon et al., 2005). For face detection, Viola and Jones (2001, 2004) constructed a strong classifier by selecting a small number of important features using AdaBoost. Foo et al. (2004) and Zhang et al. (2005) employed boosting learning for the single-modal gesture recognition task. However, boosting learning could be prone to be overfitting in practice when training data is rather small. As a late fusion strategy, co-training alternately uses the most confident unlabeled data instance(s) in one modality to assist the model training of another modality, to overcome the problem of insufficient training samples (Blum and Mitchell, 1998). Furthermore, Yu et al. (2008, 2011) proposed a Bayesian undirected graphical model interpretation for co-training methods in the context of semi-supervised multi-view learning. These two publications clarified several fundamental assumptions underlying these models and can automatically estimate how much trust should be given to each view so as to accommodate noisy views.

Inspired by boosting and Bayesian co-training methods, we present a novel Bayesian Co-Boosting training framework to realize effectively the multi-modal fusion for gesture recognition task.[1] In our framework, weak classifiers are trained with weighted data instances through multiple iterations. In each iteration round, several feature subsets are randomly generated and weak classifiers are trained on different feature groups. Only the weak classifier, which achieves the minimal training error, together with the corresponding feature subset is retained. Instance's weight is updated according to the classification result given by the weak classifiers of two modalities, so that the difficult instances will

1. Our preliminary work of multi-modal fusion on ChaLearn MMGR challenge 2013 achieved the 1st prize on gesture recognition (Wu et al., 2013).

gain more focus in the subsequent iterations. The strong classifier is constructed with all retained weak classifiers, and the classification decision is determined by the voting result of all weak classifiers. The weak classifier's voting weight is related to its prediction error on the training set.

The main contributions of this paper are concluded as follows:

1. The proposed framework is illuminated in a Bayesian perspective, and its error upper bound is minimized through iterations, which is guaranteed in theory.
2. Feature selection and multi-modal fusion are naturally embedded into the training process of weak classifiers in each Co-Boosting iteration round and bring significant improvement to the recognition performance.
3. A novel parameter estimation method is presented to address the training problem of hidden Markov model on the weighted data set.

This paper is organized as follows. In Section 2, commonly used approaches for gesture recognition is reviewed. We describe our proposed approach and related theoretical derivation in Section 3. Section 4 presents the experimental result of our method, comparing with several state-of-the-art methods. Finally, we conclude our work in Section 5.

2. Related Work

Gesture recognition has been an important research topic in human computer interaction and computer vision field. There already exist a few published surveys in this area, such as Gavrila (1999), Mitra and Acharya (2007), Weinland et al. (2011), and Suarez and Murphy (2012). As concluded in these literatures, classifiers commonly used in gesture recognition include k-nearest neighbours (Malassiotis et al., 2002), hidden Markov model (Eickeler et al., 1998), finite state machine (Yeasin and Chaudhuri, 2000), neural network (Yang and Ahuja, 2001), and support vector machine (Biswas and Basu, 2011).

Gesture recognition based on accelerometers has been investigated by many researchers (Mantyla et al., 2000; Chambers et al., 2002; Pylvänäinen, 2005; Liu et al., 2009). As an extension to the accelerometer sensors, the applications of inertial measurement unit (IMU) have also been explored recently. Ruffieux et al. (2013) collected a benchmark data set with Kinect™ and XSens IMU sensors for the development and evaluation of multi-modal gesture spotting and recognition algorithms. With this data set, Yin and Davis (2013) presented a hand tracking method based on gesture salience, and concatenated hidden Markov models were applied to perform gesture spotting and recognition.

Considering the inconvenience of wearing accelerometers or IMUs while performing gestures, it is more natural to develop vision-based gesture recognition systems. Single or stereo camera is mostly widely used in research, but Kinect™ sensor has been attracting increasing interest, due to its ability to capture both color and depth images simultaneously. ChaLearn has organized several competitions focused on the Kinect™-based gesture recognition ever since 2011 (Guyon et al., 2012; Escalera et al., 2013).

Approaches based on hidden Markov model (HMM) are widely adopted in vision-based gesture recognition. Elmezain et al. (2008) applied HMM to recognize isolated and continuous gestures in real-time. Spatio-temporal trajectories were converted to orientation dynamic features and then quantized to one of the codewords. The quantized observation

sequence was then used to inference the hidden gesture label. Gaus et al. (2013) compared the recognition performance given by both fixed state HMM and variable state HMM. In Nandakumar et al. (2013), gesture instances in the continuous data stream were segmented using both audio and hand joint information. Three modalities were used for classification: HMM classifier for MFCC feature extracted from audio signal, and SVM (support vector machine) classifier for both RGB (STIP feature) and skeleton (covariance descriptor).[2] Wu et al. (2013) performed automatic gesture detection based on the endpoint detection result in the audio data stream. HMM classifiers were then applied to both audio and skeleton features, and a late fusion strategy was employed to make the final classification decision.

In order to enhance the recognition performance of HMM-based approaches, ensemble learning, especially AdaBoost, has been embedded into the training process of hidden Markov models in a few researches. Adaptive boosting (Freund and Schapire, 1995; F. and E.S., 1997) is a training framework to generate multiple weak classifiers with different training instances' weight distribution, and construct a strong classifier with these weak classifiers to achieve a better classification performance. Foo et al. (2004) proposed a novel AdaBoost-HMM classifier to boost the recognition of visual speech elements. Weak classifiers were trained using biased Baum-Welch algorithm under the AdaBoost framework to cover different groups of training instances. Their decisions on the unlabeled instance were combined following a novel probability synthesis rule to obtain the final decision. In Zhang et al. (2005), a similar approach was applied in the application of sign language recognition. However, both researches neglected the potential noisy dimensions in the feature space, which could cause the deterioration of recognition performance.

Besides HMM-based approaches, there are also many other methods proposed in the context of vision-based gesture recognition. In Lui et al. (2010) and Lui (2012b), action videos were factorized using higher order singular value decomposition (HOSVD) and the classification was performed based on the geodesic distance on the product manifold. Boyali and Kavakli (2012) proposed a variant version of sparse representation based classification (innovated by Wright et al., 2009; Wagner et al., 2009) for gesture recognition. For a more complete overview of commonly used approaches in gesture recognition, we recommend the survey papers mentioned at the beginning of this section.

3. Bayesian Co-Boosting with Hidden Markov Model

For multi-modal gesture recognition task, fusion of features from different modalities is one of the most vital problems. Many existing approaches use a simple weighted-based fusion strategy (Bayer and Silbermann, 2013; Nandakumar et al., 2013). However, this weight coefficient usually needs to be empirically tuned, which is rather difficult if not impossible on large-scale data set. As we mentioned before, Bayesian co-training (Yu et al., 2008, 2011) can automatically determine each view's confidence score, which inspired us to adopt a similar approach to fuse multiple modalities. Boosting learning can perform feature

2. MFCC: Mel-Frequency Cepstral Coefficients (Zheng et al., 2001), a common used audio feature for speech recognition. The feature extraction process is as follows: a) the signal segment is turned into frequency domain using Discrete Fourier Transform; b) the short-term power spectrum is warped into the Mel-frequency; c) the warped power spectrum is convolved with the triangular band-pass filter; d) the MFCC feature is the Discrete Cosine Transform result of the convolved power spectrum.

selection through training multiple weak classifiers, and can be used in gesture recognition to select optimal feature dimensions for the classification problem.

In this section, we introduce a novel Bayesian Co-Boosting training framework for combining multiple hidden Markov model classifiers for multi-modal gesture recognition. Based on the proposed Bayesian Co-Boosting framework, different modalities are naturally combined together and can provide complementary information for each other. We also analyze the minimization of the error upper bound so as to derive the update rule of instance's weight in Co-Boosting process.

3.1 Model Learning

In the task of multi-modal gesture recognition, two or more modalities (in this paper, we constraint the amount of modalities to be two) are simultaneously available for describing gesture instances. Based on the raw data of each modality, a time series sequence of feature vectors can be extracted according to certain feature extraction procedures. This time series sequence data is then used as the input to the pre-trained classifier for model training and evaluation.

The most straightforward approach to this problem is to separately train a classifier for each modality, and then combine their classification results in a late fusion style. However, this approach will bring the following issues. First, feature vectors may contain noisy data dimensions, which will lead to deterioration of classification performance. Second, one classifier for one modality may not be sufficient to achieve a satisfying classification accuracy level. Third, the fusion weights of different classifiers, which have significant impact on the final classification result, are difficult to be tuned manually.

In this paper, we propose an approach to solve all these problems together. Under the Co-Boosting framework, multiple weak classifiers of each modality are trained through a number of iterations. The final strong classifier is a linear combination of these weak classifiers, and each classifier's weight is determined by its prediction error on the training data set. Figure 1 depicts the work flow of our proposed method, and Algorithm 1 describes the detailed procedures in the model training process.

The aim of our proposed Bayesian Co-Boosting framework is to generate a strong classifier for the multi-modal gesture recognition task. As we can see in Figure 1, the resulting strong classifier $H(x_i)$ is the combination of multiple weak classifiers trained on V different modalities through T iterations. In each iteration round, M_v candidate weak classifiers are trained on the v-th modality using different feature dimension subsets, and the best candidate among them is selected as the optimal weak classifier $h^*_{t,v}(x_i)$. The optimal weak classifier is the one which achieves the minimal training error among all candidate weak classifiers for modality v. Then we use all these selected weak classifiers (one weak classifier per modality) obtained at this iteration round to update each training instance's weight.

In the rest of this section, we firstly introduce the training process of a single weak classifier with weighted instances. Secondly, we derive the update rule of the instance's weight to minimize the training error's upper bound from a Bayesian perspective. The construction of the strong classifier $H(x_i)$ is described at the end of this section.

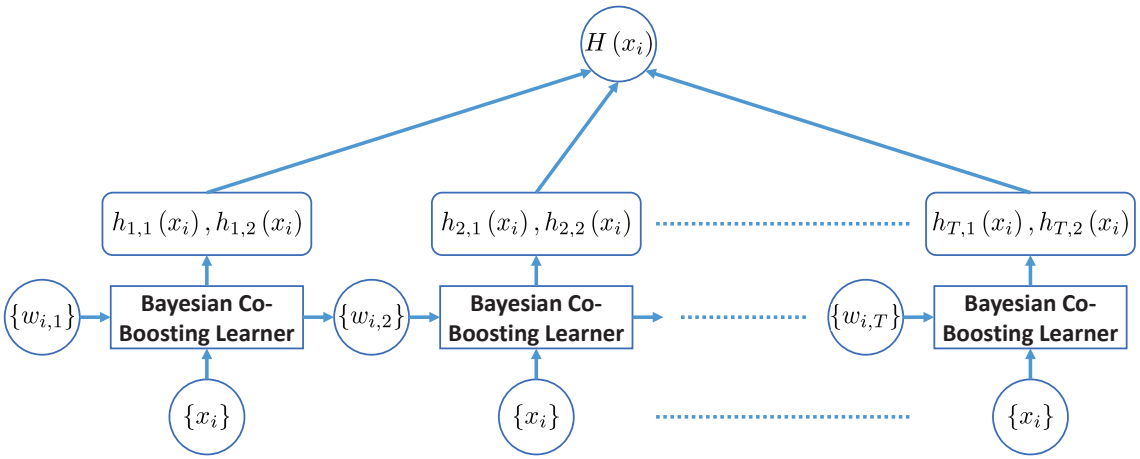


Figure 1: Work flow of Bayesian Co-Boosting training framework.
x_i: training instance; $w_{i,t}$: training instance x_i's weight at the t-th iteration; $h_{t,v}(x_i)$: weak classifier learnt from modality v at the t-th iteration; $H(x_i)$: final strong classifier.

Algorithm 1 Bayesian Co-Boosting Training Framework.[3]

Input: training instances $\{x_i\}$
Output: strong classifier $H(x_i)$

initialize data weight distribution $\{w_i\}$
for $t = 1, \ldots, T$ **do**
  for $v = 1, \ldots, V$ **do**
    for $m = 1, \ldots, M_v$ **do**
      randomly generate feature subset $\tilde{F}_{t,v,m} \subset F_v, |\tilde{F}_{t,v,m}| = \lambda_v \cdot |F_v|$
      generate training data set $\{(\tilde{x}_i, w_i)\}$ with feature dimensions in $\tilde{F}_{t,v,m}$
      train candidate weak classifier $h_{t,v,m}(x_i)$ (refer to Algorithm 2)
      calculate classifier's training error $\varepsilon_{t,v,m}$
    end for
    select optimal candidate weak classifier $h^*_{t,v}(x_i)$ and feature subset $\tilde{F}^*_{t,v}$
    calculate weak classifier's voting weight $\alpha^*_{t,v}$
  end for
  update instances' weights $\{w_i\}$ (refer to Algorithm 3)
end for
construct strong classifier $H(x_i)$

3. T: the number of Co-Boosting iteration rounds; V: the number of modalities; M_v: the number of candidate weak classifiers for modality v; F_v: all available feature dimensions for modality v; λ_v: the feature dimension selection ratio for modality v.

3.1.1 Weak Classifier Training

As we concluded in Section 2, hidden Markov model is one of the most commonly used classifiers in gesture recognition. Therefore, in this paper, we implement the Bayesian Co-Boosting training framework with HMM-based weak classifiers embedded. However, other weak classifiers can also be easily adopted in our framework.

Hidden Markov model is a statistical model based on Markov process, in which the generation of an observation sequence is modeled as the result of a series of unobserved state transitions (Rabiner, 1989). In order to deal with continuous observation vectors, a multi-variate Gaussian distribution is adopted to determine the observation probability of each observation-state pair. To simplify the subsequent analysis, we define the following symbols:

$x_{i,1:T_i}$: observation sequence of length T_i, composed of feature vectors $x_{i,t}$.

$z_{i,1:T_i}$: state transition sequence; $z_{i,t} \in \{1, \ldots, K\}$, K is the number of states.

$\mathcal{D}$: the training data set consists of N observation sequences $x_{i,1:T_i}$.

π_k: initial state probability, $\pi_k = P(z_{i,1} = k)$.

$A_{j,k}$: state transition probability, $A_{j,k} = P(z_{i,t+1} = k | z_{i,t} = j)$.

μ_k, Σ_k: mean vector and covariance matrix, $P(x_{i,t} | z_{i,t} = k) = \mathcal{N}(x_{i,t} | \mu_k, \Sigma_k)$.

For multiple-class classification problem in gesture recognition, a hidden Markov model is trained for each gesture class, with its parameters denoted as θ_c. The resulting classifier is denoted as

$$\hat{y}_i = \arg\max_c P(x_i | \theta_c),$$

where $x_i = x_{i,1:T_i}$ is the unlabeled gesture instance. $P(x_i|\theta_c)$ measures the probability for model θ_c generating observation sequence x_i and can be rewritten as

$$P(x_i | \theta_c) = \sum_{z_i} P(x_i, z_i | \theta_c),$$

where the full data probability $P(x_i, z_i|\theta_c)$ is given by

$$\begin{aligned} P(x_i, z_i | \theta_c) &= P(z_i | \theta_c) P(x_i | z_i, \theta_c) \\ &= \pi_{z_{i,1}} \prod_{t=1}^{T-1} A_{z_{i,t}, z_{i,t+1}} \prod_{t=1}^{T} \mathcal{N}\left(x_{i,t} | \mu_{z_{i,t}}, \Sigma_{z_{i,t}}\right). \end{aligned}$$

For the parameter estimation problem of HMM, commonly used Baum-Welch algorithm (a variation of EM algorithm) can only deal with unweighted training instances. In boosting learning, however, instances are assigned with different weights, which are adjusted at the end of each iteration round to guide the subsequent weak classifiers focus on more difficult instances. Hence, we need to extend the standard Baum-Welch algorithm (Murphy, 2012) to accommodate the weighted instances' training problem in our approach. Our proposed parameter estimation method is also based on the EM algorithm.

Given the weighted training data set $\{(x_i, w_i)\}$, parameter estimation problem is to find the optimal parameters that maximize the log likelihood of the observed data, which is defined as

$$\ell(\theta) = \sum_{i=1}^{N} w_i \log P(x_i|\theta) = \sum_{i=1}^{N} w_i \log \left[\sum_{z_i} P(x_i, z_i|\theta) \right].$$

But this is difficult to optimize, since the log cannot be pushed inside the sum. To get around this problem, we define the complete data log likelihood as

$$\ell_c(\theta) = \sum_{i=1}^{N} w_i \log P(x_i, z_i^*|\theta),$$

where z_i^* is the optimal state transition sequence, and is inferred with Viterbi algorithm.

Therefore, the expected complete data log likelihood for data set $\mathcal{D}$ is given by

$$Q(\theta, \theta_{\text{old}}) = \mathbb{E}\left[\ell_c(\theta) | \mathcal{D}, \theta_{\text{old}}\right], \tag{1}$$

and the optimal parameters are estimated by maximizing this.

On the basis of the definition of $P(x_i, z_i|\theta_c)$, Equation (1) can be rewritten as

$$\begin{aligned}
Q(\theta, \theta_{\text{old}}) &= \mathbb{E}\left[\sum_{i=1}^{N} w_i \log P(x_i, z_i^*|\theta)\right] \\
&= \sum_{i=1}^{N} w_i \mathbb{E}\left[\log \prod_{z_i} P(x_i, z_i|\theta)^{\mathbb{I}(z_i^* = z_i)}\right] \\
&= \sum_{i=1}^{N} w_i \sum_{z_i} \mathbb{E}\left[\mathbb{I}(z_i^* = z_i)\right] \log P(x_i, z_i|\theta) \\
&= \sum_{i=1}^{N} \sum_{k=1}^{K} w_i P\left(z_{i,1}^* = k | x_i, \theta_{t-1}\right) \log \pi_k \\
&\quad + \sum_{i=1}^{N} \sum_{j=1}^{K} \sum_{k=1}^{K} \sum_{t=1}^{T_i - 1} w_i P\left(z_{i,t}^* = j, z_{i,t+1}^* = k | x_i, \theta_{t-1}\right) \log A_{j,k} \\
&\quad + \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{t=1}^{T_i} w_i P\left(z_{i,t}^* = k | x_i, \theta_{t-1}\right) \log P(x_{i,t}|z_{i,t} = k).
\end{aligned}$$

In the E step of EM algorithm, we firstly compute two groups of probabilities with forward-backward algorithm, as describe in Murphy (2012)

$$\begin{aligned}
\gamma_{i,t}(k) &= P(z_{i,t} = k | x_i, \theta_{t-1}) \\
\xi_{i,t}(j,k) &= P(z_{i,t} = j, z_{i,t+1} = k | x_i, \theta_{t-1}),
\end{aligned} \tag{2}$$

where $\gamma_{i,t}(k)$ indicates the probability of the hidden state at time t being state k, and $\xi_{i,t}(j,k)$ represents the probability of the hidden state being state j at time t and state k

at time $(t+1)$. Based on these probabilities, we compute the following expectation items

$$\begin{aligned}
\mathbb{E}\left[N_k^1\right] &= \sum_{i=1}^{N} w_i \gamma_{i,1}(k) \\
\mathbb{E}\left[N_{j,k}\right] &= \sum_{i=1}^{N} \sum_{t=1}^{T_i-1} w_i \xi_{i,t}(j,k) \\
\mathbb{E}\left[N_k\right] &= \sum_{i=1}^{N} \sum_{t=1}^{T_i} w_i \gamma_{i,t}(k) \\
\mathbb{E}\left[\bar{x}_k\right] &= \sum_{i=1}^{N} \sum_{t=1}^{T_i} w_i \gamma_{i,t}(k)\, x_{i,t} \\
\mathbb{E}\left[\bar{x}_k \bar{x}_k^T\right] &= \sum_{i=1}^{N} \sum_{t=1}^{T_i} w_i \gamma_{i,t}(k)\, x_{i,t} x_{i,t}^T.
\end{aligned} \tag{3}$$

In the M step, parameters are updated so that $Q(\theta, \theta_{\text{old}})$ is maximized. Here, we only present the final update rule for each parameter, due to the limitation of space

$$\begin{aligned}
\hat{\pi}_k &= \frac{\mathbb{E}\left[N_k^1\right]}{\sum_{k'=1}^{K} \mathbb{E}\left[N_{k'}^1\right]} \\
\hat{A}_{j,k} &= \frac{\mathbb{E}\left[N_{j,k}\right]}{\sum_{k'=1}^{K} \mathbb{E}\left[N_{j,k'}\right]} \\
\hat{\mu}_k &= \frac{\mathbb{E}\left[\bar{x}_k\right]}{\mathbb{E}\left[N_k\right]} \\
\hat{\Sigma}_k &= \frac{\mathbb{E}\left[\bar{x}_k \bar{x}_k^T\right]}{\mathbb{E}\left[N_k\right]} - \hat{\mu}_k \hat{\mu}_k^T.
\end{aligned} \tag{4}$$

The training procedure of weak classifier is demonstrated in Algorithm 2.

3.1.2 Instance's Weight Updating

In this sub-section, we define the training error for instances in each class, together with its upper bound to simplify the error minimization formulation. Based on this formulation, we derive the update rule for instance's weight in our proposed framework.

In the t-th iteration round of Bayesian Co-Boosting training process, the training error for class c is denoted by $E_{t,c}$, and the corresponding error upper bound is denoted by $B_{t,c}$.

We define the random variable $z_i \in \{1, \ldots, C\}$ to represent the hidden label for observation x_i. The binary prediction value for each candidate class of the strong classifier is determined by

$$H_{t,c}(x_i) = sgn\left(P_{t,c,i} > \bar{P}_{t,c,i}\right) = \begin{cases} +1, & P_{t,c,i} > \bar{P}_{t,c,i} \\ -1, & P_{t,c,i} \leq \bar{P}_{t,c,i} \end{cases},$$

where

$$\begin{aligned}
P_{t,c,i} &= P\left(z_i = c | h_{1,1}(x_i), h_{1,2}(x_i), \ldots, h_{t,1}(x_i), h_{t,2}(x_i)\right) \\
\bar{P}_{t,c,i} &= P\left(z_i \neq c | h_{1,1}(x_i), h_{1,2}(x_i), \ldots, h_{t,1}(x_i), h_{t,2}(x_i)\right),
\end{aligned}$$

Algorithm 2 Weak Classifier Training

Input: weighted training instances $\{(x_i, w_i)\}$
Output: weak classifier $h(x_i)$

for $c = 1, \ldots, C$ **do**
  initialize model parameters θ_c
  for $t = 1, \ldots, T$ **do**
    initialize expectation items
    for $i = 1, \ldots, N$ **do**
      compute $\gamma_{i,t}(k), \xi_{i,t}(j,k)$ according to Equation (2)
      update expectation items according to Equation (3)
    end for
    compute $\theta_c = \left\{\hat{\pi}_k, \hat{A}_{j,k}, \hat{\mu}_k, \hat{\Sigma}_k\right\}$ according to Equation (4)
  end for
end for
construct weak classifier $h(x_i) = \arg\max_c P(x_i|\theta_c)$

and $h_{*,*}(x_i) \in \{1, \ldots, C\}$ represents the predicted class label of weak classifier.

The training error $E_{t,c}$ is defined as the sum of $0-1$ loss of classifier's binary predictions for the c-th class, which is

$$E_{t,c} = \sum_{i:y_i=c} \mathbb{1}(H_{t,c}(x_i) \neq 1) + \sum_{i:y_i \neq c} \mathbb{1}(H_{t,c}(x_i) = 1), \tag{5}$$

where function $\mathbb{1}(\cdot)$ equals to 1 when the inner expression is true; otherwise, its value is 0.

The error upper bound $B_{t,c}$ is given by

$$B_{t,c} = \sum_{i=1}^{N} \left(\frac{\bar{P}_{t,c,i}}{P_{t,c,i}}\right)^{sgn(y_i=c)} = \sum_{i:y_i=c} \frac{\bar{P}_{t,c,i}}{P_{t,c,i}} + \sum_{i:y_i \neq c} \frac{P_{t,c,i}}{\bar{P}_{t,c,i}}. \tag{6}$$

Theorem 1 *$E_{t,c} \leq B_{t,c}$ always holds with definitions in Equation (5) and (6).*

Proof For each training instance x_i, we consider its training error $E_{t,c,i}$ and the corresponding upper bound $B_{t,c,i}$. It surely falls into one of the following conditions:

(1) $H_{t,c}(x_i) = 1, y_i = c$:
Based on the definition of $H_{t,c}(x_i)$, we have $P_{t,i,c} > \bar{P}_{t,i,c}$.
Since $E_{t,c,i} = 0, B_{t,c,i} = \bar{P}_{t,i,c}/P_{t,i,c} \in [0, 1)$, thus $E_{t,c,i} \leq B_{t,c,i}$.

(2) $H_{t,c}(x_i) = 1, y_i \neq c$:
Based on the definition of $H_{t,c}(x_i)$, we have $P_{t,i,c} > \bar{P}_{t,i,c}$.
Since $E_{t,c,i} = 1, B_{t,c,i} = P_{t,i,c}/\bar{P}_{t,i,c} \in [1, +\infty)$, thus $E_{t,c,i} \leq B_{t,c,i}$.

(3) $H_{t,c}(x_i) \neq 1, y_i = c$:
Based on the definition of $H_{t,c}(x_i)$, we have $P_{t,i,c} \leq \bar{P}_{t,i,c}$.
Since $E_{t,c,i} = 1, B_{t,c,i} = \bar{P}_{t,i,c}/P_{t,i,c} \in [1, +\infty)$, thus $E_{t,c,i} \leq B_{t,c,i}$.

(4) $H_{t,c}(x_i) \neq 1, y_i \neq c$:
Based on the definition of $H_{t,c}(x_i)$, we have $P_{t,i,c} \leq \bar{P}_{t,i,c}$.
Since $E_{t,c,i} = 0, B_{t,c,i} = P_{t,i,c}/\bar{P}_{t,i,c} \in [0,1)$, thus $E_{t,c,i} \leq B_{t,c,i}$.

Therefore, $E_{t,c,i} \leq B_{t,c,i}$ holds for every instance x_i; hence, $E_{t,c} \leq B_{t,c}$ is proved. ■

In the Co-Boosting training process, the weight of each training instance should reflect the difficulty for current weak classifiers to correctly classify it. Hence, instance's weight can be determined by

$$w_i = \frac{\bar{P}_{t,y_i,i}}{P_{t,y_i,i}}. \tag{7}$$

Now we derive the update rule of training instance's weight so as to minimize the error upper bound $B_{t,c}$ through iterations, from a Bayesian perspective.

Based on the definition of $P_{t,c,i}$, we have

$$\begin{aligned}
P_{t,c,i} &= P\left(z_i = c | h_{1,1}, h_{1,2}, \ldots, h_{t,1}, h_{t,2}\right) \\
&= \frac{P\left(z_i = c, h_{1,1}, h_{1,2}, \ldots, h_{t,1}, h_{t,2}\right)}{P\left(h_{1,1}, h_{1,2}, \ldots, h_{t,1}, h_{t,2}\right)} \\
&= \frac{P\left(z_i = c, h_{1,1}, h_{1,2}, \ldots, h_{t-1,1}, h_{t-1,2}\right)}{P\left(h_{1,1}, h_{1,2}, \ldots, h_{t-1,1}, h_{t-1,2}\right)} \frac{P\left(h_{t,1} | z_i = c\right) P\left(h_{t,2} | z_i = c\right)}{P\left(h_{t,1}, h_{t,2} | h_{1,1}, h_{1,2}, \ldots, h_{t-1,1}, h_{t-1,2}\right)} \\
&= P_{t-1,c,i} \cdot \frac{P\left(h_{t,1} | z_i = c\right) P\left(h_{t,2} | z_i = c\right)}{P\left(h_{t,1}, h_{t,2} | h_{1,1}, h_{1,2}, \ldots, h_{t-1,1}, h_{t-1,2}\right)},
\end{aligned}$$

in which $h_{*,*} = h_{*,*}(x_i)$ is the predicted class label given by the weak classifier.

Similarly, we can derive the update equation for $\bar{P}_{t,c,i}$

$$\bar{P}_{t,c,i} = \bar{P}_{t-1,c,i} \cdot \frac{P\left(h_{t,1} | z_i \neq c\right) P\left(h_{t,2} | z_i \neq c\right)}{P\left(h_{t,1}, h_{t,2} | h_{1,1}, h_{1,2}, \ldots, h_{t-1,1}, h_{t-1,2}\right)}.$$

Therefore, the ratio between $\bar{P}_{t,c,i}$ and $P_{t,c,i}$ can be rewritten as

$$\frac{\bar{P}_{t,c,i}}{P_{t,c,i}} = \frac{\bar{P}_{t-1,c,i} \cdot P\left(h_{t,1} | z_i \neq c\right) P\left(h_{t,2} | z_i \neq c\right)}{P_{t-1,c,i} \cdot P\left(h_{t,1} | z_i = c\right) P\left(h_{t,2} | z_i = c\right)}. \tag{8}$$

In order to simplify the following theoretical derivation, we define these symbols

$$\begin{aligned}
&P_{c,1} = P\left(h_{t,1} = c | z_i = c\right), P_{c,2} = P\left(h_{t,1} = c | z_i \neq c\right) \\
&P_{c,3} = P\left(h_{t,1} \neq c | z_i = c\right), P_{c,4} = P\left(h_{t,1} \neq c | z_i \neq c\right) \\
&Q_{c,1} = P\left(h_{t,2} = c | z_i = c\right), Q_{c,2} = P\left(h_{t,2} = c | z_i \neq c\right) \\
&Q_{c,3} = P\left(h_{t,2} \neq c | z_i = c\right), Q_{c,4} = P\left(h_{t,2} \neq c | z_i \neq c\right).
\end{aligned} \tag{9}$$

For each instance x_i, considering whether its ground-truth label y_i and predicted label $h_{t,1}, h_{t,2}$ is equal to c or not, we can assign it into one of the following subsets

$$\begin{aligned}
&\mathcal{D}_1 = \{x_i | h_{t,1} = c, h_{t,2} = c, y_i = c\}, \mathcal{D}_2 = \{x_i | h_{t,1} = c, h_{t,2} = c, y_i \neq c\} \\
&\mathcal{D}_3 = \{x_i | h_{t,1} = c, h_{t,2} \neq c, y_i = c\}, \mathcal{D}_4 = \{x_i | h_{t,1} = c, h_{t,2} \neq c, y_i \neq c\} \\
&\mathcal{D}_5 = \{x_i | h_{t,1} \neq c, h_{t,2} = c, y_i = c\}, \mathcal{D}_6 = \{x_i | h_{t,1} \neq c, h_{t,2} = c, y_i \neq c\} \\
&\mathcal{D}_7 = \{x_i | h_{t,1} \neq c, h_{t,2} \neq c, y_i = c\}, \mathcal{D}_8 = \{x_i | h_{t,1} \neq c, h_{t,2} \neq c, y_i \neq c\}.
\end{aligned} \tag{10}$$

On the basis of the above data partitioning, $B_{t,c}$ can be expanded as

$$\begin{aligned}
B_{t,c} &= \sum_{i:y_i=c} \frac{\bar{P}_{t,c,i}}{P_{t,c,i}} + \sum_{i:y_i\neq c} \frac{P_{t,c,i}}{\bar{P}_{t,c,i}} \\
&= \sum_{i:x_i\in\mathcal{D}_1} \frac{\bar{P}_{t-1,c,i}P_{c,2}Q_{c,2}}{P_{t-1,c,i}P_{c,1}Q_{c,1}} + \sum_{i:x_i\in\mathcal{D}_2} \frac{P_{t-1,c,i}P_{c,1}Q_{c,1}}{\bar{P}_{t-1,c,i}P_{c,2}Q_{c,2}} \\
&\quad + \sum_{i:x_i\in\mathcal{D}_3} \frac{\bar{P}_{t-1,c,i}P_{c,2}Q_{c,4}}{P_{t-1,c,i}P_{c,1}Q_{c,3}} + \sum_{i:x_i\in\mathcal{D}_4} \frac{P_{t-1,c,i}P_{c,1}Q_{c,3}}{\bar{P}_{t-1,c,i}P_{c,2}Q_{c,4}} \\
&\quad + \sum_{i:x_i\in\mathcal{D}_5} \frac{\bar{P}_{t-1,c,i}P_{c,4}Q_{c,2}}{P_{t-1,c,i}P_{c,3}Q_{c,1}} + \sum_{i:x_i\in\mathcal{D}_6} \frac{P_{t-1,c,i}P_{c,3}Q_{c,1}}{\bar{P}_{t-1,c,i}P_{c,4}Q_{c,2}} \\
&\quad + \sum_{i:x_i\in\mathcal{D}_7} \frac{\bar{P}_{t-1,c,i}P_{c,4}Q_{c,4}}{P_{t-1,c,i}P_{c,3}Q_{c,3}} + \sum_{i:x_i\in\mathcal{D}_8} \frac{P_{t-1,c,i}P_{c,3}Q_{c,3}}{\bar{P}_{t-1,c,i}P_{c,4}Q_{c,4}}.
\end{aligned}$$

To simplify the expression, we define

$$\alpha_1 = \frac{P_{c,1}}{P_{c,2}}, \alpha_2 = \frac{P_{c,3}}{P_{c,4}}, \alpha_3 = \frac{Q_{c,1}}{Q_{c,2}}, \alpha_4 = \frac{Q_{c,3}}{Q_{c,4}}, \tag{11}$$

$$\begin{aligned}
&S_1 = \sum_{i:x_i\in\mathcal{D}_1} \frac{\bar{P}_{t-1,c,i}}{P_{t-1,c,i}},\ S_2 = \sum_{i:x_i\in\mathcal{D}_2} \frac{P_{t-1,c,i}}{\bar{P}_{t-1,c,i}},\ S_3 = \sum_{i:x_i\in\mathcal{D}_3} \frac{\bar{P}_{t-1,c,i}}{P_{t-1,c,i}},\ S_4 = \sum_{i:x_i\in\mathcal{D}_4} \frac{P_{t-1,c,i}}{\bar{P}_{t-1,c,i}} \\
&S_5 = \sum_{i:x_i\in\mathcal{D}_5} \frac{\bar{P}_{t-1,c,i}}{P_{t-1,c,i}},\ S_6 = \sum_{i:x_i\in\mathcal{D}_6} \frac{P_{t-1,c,i}}{\bar{P}_{t-1,c,i}},\ S_7 = \sum_{i:x_i\in\mathcal{D}_7} \frac{\bar{P}_{t-1,c,i}}{P_{t-1,c,i}},\ S_8 = \sum_{i:x_i\in\mathcal{D}_8} \frac{P_{t-1,c,i}}{\bar{P}_{t-1,c,i}}.
\end{aligned} \tag{12}$$

where $\alpha_k, k = 1,\dots 4$ are unknown variables and $S_k, k = 1,\dots,8$ can be computed with weak classifier's prediction. Then we rewrite $B_{t,c}$ as

$$\begin{aligned}
B_{t,c} &= \frac{S_1}{\alpha_1\alpha_3} + S_2\cdot\alpha_1\alpha_3 + \frac{S_3}{\alpha_1\alpha_4} + S_4\cdot\alpha_1\alpha_4 \\
&\quad + \frac{S_5}{\alpha_2\alpha_3} + S_6\cdot\alpha_2\alpha_3 + \frac{S_7}{\alpha_2\alpha_4} + S_8\cdot\alpha_2\alpha_4.
\end{aligned}$$

The partial derivatives of $B_{t,c}$ for the unknown variables $\alpha_{1:4}$ are

$$\begin{aligned}
\frac{\partial B_{t,c}}{\partial\alpha_1} &= -\frac{S_1}{\alpha_1^2\alpha_3} + S_2\cdot\alpha_3 - \frac{S_3}{\alpha_1^2\alpha_4} + S_4\cdot\alpha_4 \\
\frac{\partial B_{t,c}}{\partial\alpha_2} &= -\frac{S_5}{\alpha_2^2\alpha_3} + S_6\cdot\alpha_3 - \frac{S_7}{\alpha_2^2\alpha_4} + S_8\cdot\alpha_4 \\
\frac{\partial B_{t,c}}{\partial\alpha_3} &= -\frac{S_1}{\alpha_1\alpha_3^2} + S_2\cdot\alpha_1 - \frac{S_5}{\alpha_2\alpha_3^2} + S_6\cdot\alpha_2 \\
\frac{\partial B_{t,c}}{\partial\alpha_4} &= -\frac{S_3}{\alpha_1\alpha_4^2} + S_4\cdot\alpha_1 - \frac{S_7}{\alpha_2\alpha_4^2} + S_8\cdot\alpha_2.
\end{aligned} \tag{13}$$

The optimal values of α_k should ensure that all partial derivatives in Equation (13) are equal to 0. Therefore, we obtain the following equations

$$\begin{aligned}\alpha_1 &= \sqrt{\frac{S_1/\alpha_3 + S_3/\alpha_4}{S_2 \cdot \alpha_3 + S_4 \cdot \alpha_4}}, \alpha_2 = \sqrt{\frac{S_5/\alpha_3 + S_7/\alpha_4}{S_6 \cdot \alpha_3 + S_8 \cdot \alpha_4}} \\ \alpha_3 &= \sqrt{\frac{S_1/\alpha_1 + S_5/\alpha_2}{S_2 \cdot \alpha_1 + S_6 \cdot \alpha_2}}, \alpha_4 = \sqrt{\frac{S_3/\alpha_1 + S_7/\alpha_2}{S_4 \cdot \alpha_1 + S_8 \cdot \alpha_2}},\end{aligned} \tag{14}$$

and α_k can be solved within a few iterations (less than 10 rounds for most conditions, according to our experimental results).

Based on the definitions in Equation (9), it is obvious that

$$\begin{aligned}P_{c,1} + P_{c,3} = 1, P_{c,2} + P_{c,4} = 1 \\ Q_{c,1} + Q_{c,3} = 1, Q_{c,2} + Q_{c,4} = 1,\end{aligned} \tag{15}$$

and these eight variables can be solved after all α_k are obtained.

Based on the above analysis for training error minimization, the detailed algorithm for multiple weak classifiers training is concluded in Algorithm 3.

Algorithm 3 Instance's Weight Updating

Input: training instances $\{x_i\}$
Input: instances' weight $\{w_{i,t-1}\}$
Input: weak classifiers $h_{t,1}(x_i), h_{t,2}(x_i)$
Output: updated instances' weight $\{w_{i,t}\}$

```
for c = 1, ..., C do
    assign instances into D_k according to Equation (10)
    compute S_k according to Equation (12)
    compute α_k according to Equation (14)
    compute P_{c,k}, Q_{c,k} according to Equation (11) and (15)
    for instance x_i in the c-th class do
        compute P_{t,c,i}, P̄_{t,c,i} according to Equation (8)
        compute w_{i,t} according to Equation (7)
    end for
end for
```

3.2 Class Label Inference

In our multi-modal gesture recognition system, the predicted class label of unclassified instance is determined by the voting result of all weak classifiers.

For the optimal weak classifier $h^*_{t,v}(x_i)$ with training error $\varepsilon^*_{t,v}$, the classifier weight is defined as

$$\alpha^*_{t,v} = \log \frac{1 - \varepsilon^*_{t,v}}{\varepsilon^*_{t,v}},$$

where the training error is calculated by

$$\varepsilon_{t,v}^{*}=\sum_{c=1}^{C}\sum_{i:y_i=c}w_i\cdot\mathbb{1}\left\{h_{t,v}^{*}\left(x_i\right)\neq c\right\}.$$

The final prediction of instance's class label is determined by

$$H\left(x_i\right)=\arg\max_{c}\sum_{t=1}^{T}\sum_{v=1}^{2}\alpha_{t,v}^{*}\mathbb{1}\{h_{t,v}^{*}\left(x_i\right)=c\}.$$

4. Experimental Results

In this section, experiments are carried out on two multi-modal gesture recognition data sets, to prove the effectiveness of our proposed Bayesian Co-Boosting training framework. On the basis of comparative results of different training algorithms, the main contributing elements to our improvement on classification accuracy are also analyzed.

4.1 Baseline Methods Description

The training framework we propose in this paper is a general model, and some state-of-the-art methods can be considered as the special cases of our framework. The key parameters controlling the complexity of training process are T (number of iterations), V (number of modalities), and M_v (number of feature subset candidates). Various approaches can be obtained with different combinations of these three parameters.

If we set $T=1$, then model is trained without boosting learning. Many approaches using a single HMM to model instances from one gesture class can be categorized into this case.

If we set $V=1$, then the classifier is actually trained with only one feature modality. During iterations, feature selection procedure remains unchanged, but the update rule of instance's weight no longer applies. In this case, an instance's weight can be updated in a similar way as described in Viola and Jones (2004).

If we set $M_v=1$ for each modality, the feature selection procedure is removed from training process. In this case, there is no need to generate feature subset, since it may cause unnecessary information loss. All feature dimensions are used during training.

Now we define 7 baseline approaches listed as follows, each of which is a special case of our framework. Through this comparison, we can discover which part of the framework is really contributing to the improvement in classification accuracy.

(1) M1: training a classifier with the 1st modality:
Parameters setup: $T=1, V=1, M_1=1$.
Classifier: $H\left(x_i\right)=\arg\max_c P\left(x_i|\theta_{1,c}\right)$.
x_i is the unlabeled instance, and $\theta_{1,c}$ are the parameters of hidden Markov model for instances in the c-th class, trained on the 1st modality.

(2) M2: training a classifier with the 2nd modality:
Parameters setup: $T=1, V=1, M_2=1$.
Classifier: $H\left(x_i\right)=\arg\max_c P\left(x_i|\theta_{2,c}\right)$.

x_i is the unlabeled instance, and $\theta_{2,c}$ are the parameters of hidden Markov model for instances in the c-th class, trained on the 2nd modality.

(3) M1+M2: training classifiers with the 1st and 2nd modality:
Parameters setup: $T = 1, V = 2, M_1 = M_2 = 1$.
Classifier: $H(x_i) = \arg\max_c [\alpha P(x_i|\theta_{1,c}) + (1-\alpha) P(x_i|\theta_{2,c})]$.
x_i is the unlabeled instance, and $\theta_{1,c}$ and $\theta_{2,c}$ are respectively the parameters of hidden Markov model for instances in the c-th class, trained on the 1st and 2nd modality.

(4) Boost.M1: training boosted classifiers with the 1st modality:
Parameters setup: $T > 1, V = 1, M_1 = 1$.
Classifier: $H(x_i) = \arg\max_c \sum_{t=1}^{T} \alpha_{t,1} \mathbb{1}\{h_{t,1}(x_i) = c\}$.
$h_{t,1}(x_i) = \arg\max_c P(x_i|\theta_{t,1,c})$ is the weak classifier learnt at the t-th boosting iteration, and $\alpha_{t,1}$ is the corresponding classifier's weight.

(5) Boost.M2: training boosted classifiers with the 2nd modality:
Parameters setup: $T > 1, V = 1, M_2 = 1$.
Classifier: $H(x_i) = \arg\max_c \sum_{t=1}^{T} \alpha_{t,2} \mathbb{1}\{h_{t,2}(x_i) = c\}$.
$h_{t,2}(x_i) = \arg\max_c P(x_i|\theta_{t,2,c})$ is the weak classifier learnt at the t-th boosting iteration, and $\alpha_{t,2}$ is the corresponding classifier's weight.

(6) Boost.Sel.M1: training boosted classifiers with selected features of the 1st modality:
Parameters setup: $T > 1, V = 1, M_1 > 1$.
Classifier: $H(x_i) = \arg\max_c \sum_{t=1}^{T} \alpha_{t,1} \mathbb{1}\{h_{t,1}(x_i) = c\}$.
$h_{t,1}(x_i) = \arg\max_c P(x_i|\theta_{t,1,c})$ is the weak classifier learnt at the t-th boosting iteration, and $\alpha_{t,1}$ is the corresponding classifier's weight. Unlike "Boost.M1", feature selection is performed in the training process of weak classifier $h_{t,1}(x_i)$.

(7) Boost.Sel.M2: training boosted classifiers with selected features of the 2nd modality:
Parameters setup: $T > 1, V = 1, M_2 > 1$.
Classifier: $H(x_i) = \arg\max_c \sum_{t=1}^{T} \alpha_{t,2} \mathbb{1}\{h_{t,2}(x_i) = c\}$.
$h_{t,2}(x_i) = \arg\max_c P(x_i|\theta_{t,2,c})$ is the weak classifier learnt at the t-th boosting iteration, and $\alpha_{t,2}$ is the corresponding classifier's weight. Unlike "Boost.M2", feature selection is performed in the training process of weak classifier $h_{t,2}(x_i)$.

For convenience, we denote our proposed approach as "BayCoBoost". Its corresponding parameter setup is $T > 1, V = 2, M_1 > 1, M_2 > 1$.

"M1" and "M2" are two naive methods for single-modal gesture recognition, and many HMM-based recognizers can be categorized into one of these. "M1+M2" is the late fusion result of "M1" and "M2". Considering the weight coefficient α, we evaluate 11 candidate values from 0 to 1 with equal step length on the training set using cross validation, and select the optimal α which reaches the minimal error. The approach used in Wu et al. (2013) can be regarded as a variation of the "M1+M2" method.

In "Boost.M1" and "Boost.M2", boosting learning is applied to enhance the recognition performance. Multiple HMM-based weak classifiers are trained through iterations. Foo et al. (2004); Zhang et al. (2005) respectively used this type of approach for the recognition of visual speech element and sign language. "Boost.Sel.M1" and "Boost.Sel.M2" are similar to them, but feature selection is embedded into the training process of each weak classifier.

Finally, our proposed method "BayCoBoost" integrates both modalities under the Bayesian Co-Boosting framework.

4.2 Experiment 1: ChaLearn MMGR data set

In 2013, ChaLearn organized a challenge on multi-modal gesture recognition with motion data captured by the Kinect™ sensor. This challenge provides a benchmark data set on the topic of multi-modal gesture recognition. Detailed information about this data set can be found in Escalera et al. (2013).

This data set contains 20 gesture categories, each of which is an Italian cultural or anthropological sign. Gestures in the data set are performed with one or two hands by 27 users, along with the corresponding word/phase spoken out. Data modalities provided in this data set include color image, depth image, skeletal model, user mask, and audio data.

The data set has been divided into three subsets already, namely *Development*, *Validation*, and *Evaluation*. In our experiment, *Development* and *Validation* subsets are used respectively for model training and testing. Based on the labeled data, we can segment out $7,205$ gesture instances from *Development* subset and $3,280$ instances from *Validation*. These two numbers are slightly smaller than the amount ($7,754$ and $3,362$) announced in Escalera et al. (2013), since we filter out those gesture instances which contain invalid skeleton data (when Kinect™ fails to track the skeleton and outputs all-zero skeleton data).

Among all feature modalities offered in this data set, we choose audio and skeleton feature to perform our proposed Bayesian Co-Boosting training process. We extract 39-dimension MFCC feature (Martin et al., 2001) from audio data stream and denote it as the first feature modality. The second modality is the 138-dimension skeleton feature extracted from 3D coordinates of 20 tracked joint points. The detailed extraction process of skeleton feature is described in the appendix.

In this experiment, parameters in Algorithm 1 are chosen as follows: $T = 20, V = 2, M_1 = 5$, and $M_2 = 10$. For MFCC feature, the size of feature subset is set to be 50% of all feature dimensions. The skeleton feature subset consists of 15% dimensions from the original feature space. Therefore, the number of feature dimensions used to train weak classifiers is respectively 20 for audio and 21 for skeleton. The number of iterations to estimate parameters of hidden Markov models for weak classifiers is set to 20. All these parameters are selected roughly using a grid search based on the cross validation result on the training subset.

We report the recognition accuracy of each gesture category in Figure 2. Also, several statistics are computed to provide a quantitative comparison between different methods' average recognition performance across all categories, which are reported in Table 1. The recognition accuracy is defined as the ratio of the number of correctly classified gestures against the number of all existing gestures in each class.

4.3 Experiment 2: ChAirGest data set

In Ruffieux et al. (2013), a multi-modal data set was collected to provide a benchmark for the development and evaluation of gesture recognition methods. This data set is captured with a Kinect™ sensor and four Xsens inertial motion units. Three data streams are provided by the Kinect™ sensor: color image, depth image, and 3D positions of upper-body joint points.

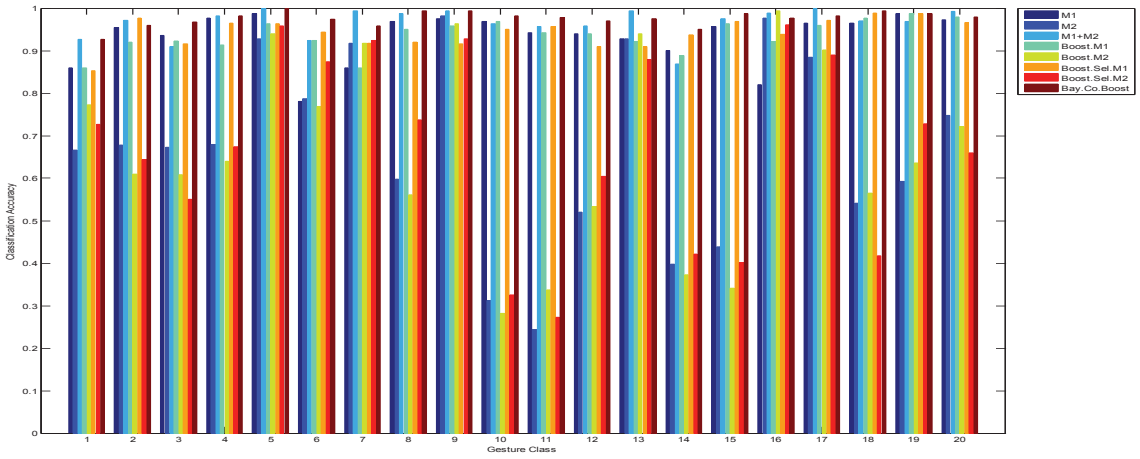


Figure 2: Recognition accuracy of each gesture category on ChaLearn MMGR data sets.

Method	Mean	Std	Conf	[Mean-Conf, Mean+Conf]
M1	0.9326	0.0584	0.0273	[0.9052, 0.9599]
M2	0.6749	0.2223	0.1040	[0.5709, 0.7790]
M1+M2	0.9666	0.0345	0.0162	[0.9504, 0.9827]
Boost.M1	0.9364	0.0366	0.0171	[0.9192, 0.9535]
Boost.M2	0.6705	0.2276	0.1065	[0.5640, 0.7770]
Boost.Sel.M1	0.9432	0.0334	0.0156	[0.9275, 0.9588]
Boost.Sel.M2	0.6793	0.2219	0.1038	[0.5754, 0.7831]
BayCoBoost	**0.9763**	**0.0173**	**0.0081**	**[0.9682, 0.9844]**

Table 1: Recognition accuracy on ChaLearn MMGR data sets.

Each Xsens IMU sensor can provide linear acceleration, angular acceleration, magnetometer, Euler orientation, orientation quaternion, and barometer data with a frequency of 50Hz.

This data set contains a vocabulary of 10 one-hand gestures commonly used in close human-computer interaction. Gestures are performed by 10 subjects, and each gesture is repeated 12 times, including 2 lighting conditions and 3 resting postures. The total number of gesture instances is 1200.

Similar to the previous experiment, two feature modalities are chosen to perform our Bayesian Co-Boosting training process. The first feature modality is based on the data captured by Xsens sensors. We use the raw data collected by four Xsens sensors as feature vector, which is of 68-dimension. Skeleton data captured by the Kinect™ is used as the second modality, and a 120-dimension feature vector is extracted per frame (see the appendix for details). The number of skeleton feature dimensions is smaller than the previous one, because the position of two joint points (hip-center and spine) cannot be tracked since all users were performing gestures while sitting.

The parameters in this experiment are almost identical with previous experiment. In Algorithm 1, parameters are: $T = 20, V = 2$, and $M_1 = M_2 = 10$. The feature selection ratio of Xsens and skeleton are respectively 20% and 15%. Under this setup, the feature

dimension of Xsens data for weak classifier training is 14, and this number is 18 for skeleton feature. The number of iterations for weak classifier training is also set to 20. Similar to the previous experiment, these parameters are also determined by cross-validation.

Since no division of training and testing subset is specified in this data set, we perform leave-one-out cross validation. In each round, gesture instances of one subject are used for model evaluation, and other instances are used to train the model. We compute the average recognition accuracy for each gesture class and report them in Figure 3 and Table 2.

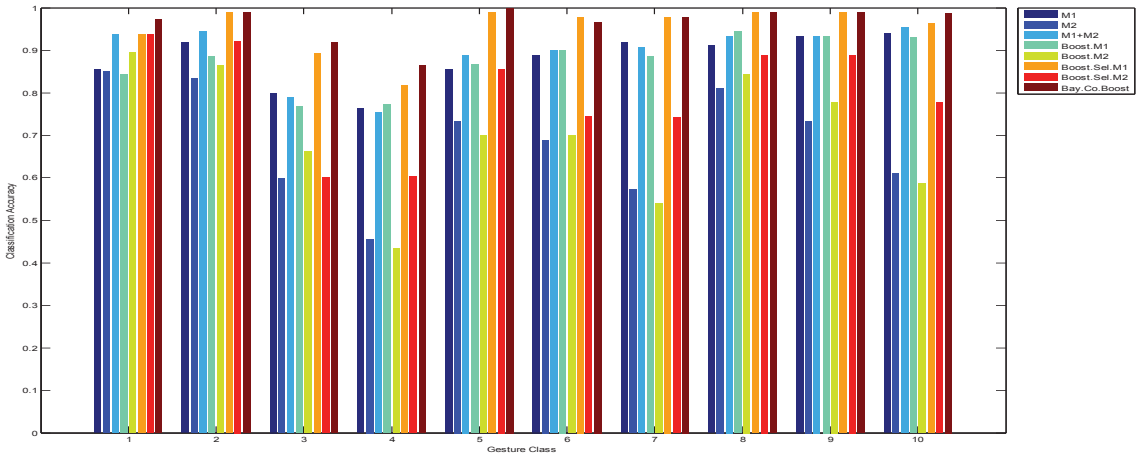


Figure 3: Recognition accuracy of each gesture category on ChAirGest data sets.

Method	Mean	Std	Conf	[Mean-Conf, Mean+Conf]
M1	0.8782	0.0598	0.0427	[0.8355, 0.9210]
M2	0.6884	0.1283	0.0918	[0.5966, 0.7801]
M1+M2	0.8940	0.0685	0.0490	[0.8450, 0.9430]
Boost.M1	0.8728	0.0623	0.0445	[0.8283, 0.9174]
Boost.M2	0.7003	0.1501	0.1074	[0.5929, 0.8077]
Boost.Sel.M1	0.9522	0.0564	0.0403	[0.9119, 0.9925]
Boost.Sel.M2	0.7958	0.1242	0.0889	[0.7070, 0.8847]
BayCoBoost	**0.9653**	**0.0420**	**0.0300**	**[0.9353, 0.9953]**

Table 2: Recognition accuracy on ChAirGest data sets.

4.4 Result Analysis

From the above experimental results, it is obvious that our proposed Bayesian Co-Boosting training algorithm achieves the best recognition accuracy in both data sets. Our approach's recognition accuracy ranks first in 14 out of 20 classes on ChaLearn MMGR data set and 9 out of 10 classes on ChAirGest data set. The average recognition accuracy of our method is also superior to any other baseline methods, as shown in Table 1 and Table 2. This improvement of our method mainly benefits from two aspects: multi-modal fusion under Bayesian Co-Boosting framework, and boosting learning with feature selection.

The improvement brought by multi-modal fusion is inevitable, since different modalities surely can provide complementary information for each other. "M1+M2" implements late fusion using a weight coefficient α, which requires more training time to determine its optimal value through cross-validation. On the other hand, in our approach, each classifier's weight is determined during boosting process, which avoids extra parameter tuning and is more reasonable and explainable based on the above theoretical analysis.

Comparing the result of "M1", "M2", "Boost.M1", and "Boost.M2", we can see that boosting learning could not necessarily improve the recognition accuracy. This may due to the overfitting caused by the small amount of available training instances. The overfitting problem of boosting methods has been discussed in several literatures (Zhang and Yu, 2005; Reyzin and Schapire, 2006; Vezhnevets and Barinova, 2007; Yao and Doretto, 2010). Considering the high feature dimension of instances, the weak classifier may be too complex to be well trained on such few instances.

Based on the above observation, we tackle the overfitting problem from two aspects. Firstly, feature selection is used to reduce the number of feature dimensions while preserving enough discriminative information, which alleviates overfitting brought by the small size sample problem. Secondly, Bayesian Co-Boosting is employed to combine two weak classifiers together with collaborative training strategy, and each modality can provide complementary information for the other modality. Therefore, the amount of available training information for classifiers is actually increased to avoid overfitting problem to some extent.

As demonstrated in Table 1 and Table 2, "Boost.Sel.M1" and "Boost.Sel.M2" outperform their corresponding training methods without feature selection. On this basis, after applying Co-Boosting method to fuse two modalities, our proposed "BayCoBoost" achieves superior recognition accuracy than all baseline methods.

As for the computation complexity, we compare the average classification time for each method. It takes around 0.31s/0.11s for our proposed "BayCoBoost" method to label an instance in ChaLearn MMGR and ChAirGest data set, respectively. Although non-boosting methods can operate at higher speed (for "M1+M2", the time is about 0.037s/0.013s), we think it is worthy to spend more time since our method's performance is superior to these methods, especially for the second data set. Another remarkable comparison is that by using feature selection strategy, "Boost.Sel.M1" and "Boost.Sel.M2" not only run twice as fast as "Boost.M1" and "Boost.M2", due to the lower classifier's complexity, but also outperform them in the classification performance. This also proves that the effectiveness of the feature selection strategy in our "BayCoBoost" method.

5. Conclusion

In this paper, a novel Bayesian Co-Boosting training framework for multi-modal gesture recognition is proposed. The merits of our work are three-fold: first, the collaborative training between multiple modalities provides complementary information for each modality; second, the boosting learning combines weak classifiers to construct a strong classifier of higher accuracy; third, the Bayesian perspective theoretically ensures that the training error of our method is minimized through iterations. Feature selection and multi-modal fusion are naturally embedded into the training process, which bring significant improvement to the recognition accuracy. Experimental results on two multi-modal gesture recognition

data sets prove the effectiveness of our proposed approach. Moreover, our proposed framework can be easily extended to other related tasks in multi-modal scenarios, such as object detection and tracking.

Acknowledgments

This work was supported in part by 973 Program (Grant No. 2010CB327905), National Natural Science Foundation of China (Grant No. 61332016, 61202325), and Key Project of Chinese Academy of Sciences (Grant No. KGZD-EW-103-5).

Appendix A. Skeleton Feature Extraction

The Kinect™ sensor is able to provide 3D position information for 20 joint points of human body. We denote the original 3D coordinates of these joints as $(x_i, y_i, z_i), i = 1, \ldots, 20$.

In order to extract the skeleton feature which is invariant to user's position, orientation, and body size, we perform the following transformations:

1. Select one joint point as the origin of the normalized coordinate system.
   Translate all joint points to move the selected point to the origin.
2. Select three joint points to construct the reference plane.
   Rotate the reference plane so that it is orthogonal to the z-axis.
3. Calculate the distance sum of 19 directly connected joint pairs.
   Normalize all coordinates so that the sum is equal to 1.

After above transformations, we can obtain the normalized 3D coordinates (x_i^*, y_i^*, z_i^*), which are invariant to the user's position, orientation, and body size.

Since most gestures are performed with upper body, and the lower body's movement may interfere the recognition of gestures, we only select joint points in the upper body for feature extraction. The final feature vector consists of four parts:

1. Absolute 3D position of joint points.
2. Relative 3D position of joint points, defined on directly connected joint pairs.
3. First order difference in time of part 1 in the feature vector.
4. First order difference in time of part 2 in the feature vector.

References

I. Bayer and T. Silbermann. A multi modal approach to gesture recognition from audio and video data. In *Proceedings of the 15th ACM International Conference on Multimodal Interaction*, pages 461–466, 2013.

K.K. Biswas and S.K. Basu. Gesture recognition using Microsoft Kinect. In *the 5th International Conference on Automation, Robotics and Applications*, pages 100–103, 2011.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100, 1998.

A. Boyali and M. Kavakli. A robust gesture recognition algorithm based on sparse representation, random projections and compressed sensing. In *IEEE Conference on Industrial Electronics and Applications*, pages 243–249, 2012.

G.S. Chambers, S. Venkatesh, G.A.W. West, and H.H. Bui. Hierarchical recognition of intentional human gestures for sports video annotation. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 2, pages 1082–1085, 2002.

X. Chen and M. Koskela. Online RGB-D gesture recognition with extreme learning machines. In *Proceedings of the 15th ACM International Conference on Multimodal Interaction*, pages 467–474, 2013.

S. Eickeler, A. Kosmala, and G. Rigoll. Hidden Markov model based continuous online gesture recognition. In *Proceedings of the 14th International Conference on Pattern Recognition*, volume 2, pages 1206–1208, 1998.

M. Elmezain, A. Al-Hamadi, J. Appenrodt, and B. Michaelis. A hidden Markov model-based continuous gesture recognition system for hand motion trajectory. In *Proceedings of the 19th International Conference on Pattern Recognition*, pages 1–4, 2008.

S. Escalera, J. Gonzàlez, X. Baró, M. Reyes, O. Lopes, I. Guyon, V. Athitsos, and H. Escalante. Multi-modal gesture recognition challenge 2013: Dataset and results. In *Proceedings of the 15th ACM International Conference on Multimodal Interaction*, pages 445–452, 2013.

Yoav F. and Robert E.S. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

S.W. Foo, Y. Lian, and L. Dong. Recognition of visual speech elements using adaptively boosted hidden Markov models. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(5):693–705, 2004.

Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory*, volume 904 of *Lecture Notes in Computer Science*, pages 23–37. Springer Berlin Heidelberg, 1995.

Y.F.A. Gaus, F. Wong, K. Teo, R. Chin, R.R. Porle, L.P. Yi, and A. Chekima. Comparison study of hidden Markov model gesture recognition using fixed state and variable state. In *IEEE International Conference on Signal and Image Processing Applications*, pages 150–155, 2013.

D.M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

I. Guyon, V. Athitsos, P. Jangyodsuk, B. Hamner, and H.J. Escalante. ChaLearn gesture challenge: Design and first results. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–6, 2012.

R.L. Kashyap. Optimal feature selection and decision rules in classification problems with time series. *IEEE Transactions on Information Theory*, 24(3):281–288, 1978.

J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

Y. M. Lui. A least squares regression framework on manifolds and its application to gesture recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 13–18, 2012a.

Y.M. Lui. Human gesture recognition on product manifolds. *Journal of Machine Learning Research*, 13(1):3297–3321, 2012b.

Y.M. Lui, J.R. Beveridge, and M. Kirby. Action classification on product manifolds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 833–839, 2010.

S. Malassiotis, N. Aifanti, and M.G. Strintzis. A gesture recognition system using 3D data. In *Proceedings of the 1st International Symposium on 3D Data Processing Visualization and Transmission*, pages 190–193, 2002.

M.R. Malgireddy, I. Inwogu, and V. Govindaraju. A temporal Bayesian model for classifying, detecting and localizing activities in video sequences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 43–48, 2012.

V. Mantyla, J. Mantyjarvi, T. Seppanen, and E. Tuulari. Hand gesture recognition of a mobile device user. In *IEEE International Conference on Multimedia and Expo*, volume 1, pages 281–284, 2000.

A. Martin, D. Charlet, and L. Mauuary. Robust speech/non-speech detection using LDA applied to MFCC. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 237–240, 2001.

S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(3):311–324, 2007.

F. Mörchen. Time series feature extraction for data mining using DWT and DFT. Technical report, Philipps-University Marburg, 2003.

K.P. Murphy. *Machine learning: A probabilistic perspective*. MIT Press, 2012.

K. Nandakumar, K.W. Wan, S.M.A. Chan, W.Z.T. Ng, J.G. Wang, and W.Y. Yau. A multi-modal gesture recognition system using audio, video, and skeletal joint data. In *Proceedings of the 15th ACM International Conference on Multimodal Interaction*, pages 475–482, 2013.

C. Oz and M.C. Leu. American sign language word recognition with a sensory glove using artificial neural networks. *Engineering Applications of Artificial Intelligence*, 24(7):1204–1213, 2011.

T. Pylvänäinen. Accelerometer based gesture recognition using continuous HMMs. In *Pattern Recognition and Image Analysis*, volume 3522 of *Lecture Notes in Computer Science*, pages 639–646. Springer Berlin Heidelberg, 2005.

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

J.L. Raheja, R. Shyam, U. Kumar, and P.B. Prasad. Real-time robotic hand control using hand gestures. In *the 2nd International Conference on Machine Learning and Computing*, pages 12–16, 2010.

L. Reyzin and R.E. Schapire. How boosting the margin can also boost classifier complexity. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 753–760, 2006.

M. Roccetti, G. Marfia, and A. Semeraro. A fast and robust gesture recognition system for exhibit gaming scenarios. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, pages 343–350, 2011.

S. Ruffieux, D. Lalanne, and E. Mugellini. ChAirGest - a challenge for multimodal mid-air gesture recognition for close HCI. In *Proceedings of the 15th ACM International Conference on Multimodal Interaction*, pages 483–488, 2013.

J. Suarez and R.R. Murphy. Hand gesture recognition with depth images: A review. In *IEEE RO-MAN*, pages 411–417, 2012.

A. Vezhnevets and O. Barinova. Avoiding boosting overfitting by removing confusing samples. In *Proceedings of the 18th European Conference on Machine Learning*, volume 4701 of *Lecture Notes in Computer Science*, pages 430–441. Springer Berlin Heidelberg, 2007.

P. Viola and M. Jones. Robust real-time face detection. In *Proceedings of the 8th IEEE International Conference on Computer Vision*, volume 2, pages 747–747, 2001.

P. Viola and M.J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

A. Wagner, J. Wright, A. Ganesh, Z. Zhou, and Y. Ma. Towards a practical face recognition system: Robust registration and illumination by sparse representation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 597–604, 2009.

D. Weinland, R. Ronfard, and E. Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 115(2):224–241, 2011.

J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009.

D. Wu, F. Zhu, and L. Shao. One shot learning gesture recognition from RGBD images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 7–12, 2012.

J. Wu, J. Cheng, C. Zhao, and H. Lu. Fusing multi-modal features for gesture recognition. In *Proceedings of the 15th ACM International Conference on Multimodal Interaction*, pages 453–460, 2013.

M. Yang and N. Ahuja. Recognizing hand gestures using motion trajectories. In *Face Detection and Gesture Recognition for Human-Computer Interaction*, volume 1 of *The International Series in Video Computing*, pages 53–81. Springer US, 2001.

Y. Yao and G. Doretto. Boosting for transfer learning with multiple sources. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1855–1862, 2010.

M. Yeasin and S. Chaudhuri. Visual understanding of dynamic hand gestures. *Pattern Recognition*, 33(11):1805–1817, 2000.

Y. Yin and R. Davis. Gesture spotting and recognition using salience detection and concatenated hidden Markov models. In *Proceedings of the 15th ACM International Conference on Multimodal Interaction*, pages 489–494, 2013.

H. Yoon, K. Yang, and C. Shahabi. Feature subset selection and feature ranking for multivariate time series. *IEEE Transactions on Knowledge and Data Engineering*, 17(9): 1186–1198, 2005.

S. Yu, B. Krishnapuram, R. Rosales, H. Steck, and R.B. Rao. Bayesian co-training. In *Advances in Neural Information Processing Systems*, volume 20, pages 1665–1672. MIT Press, 2008.

S. Yu, B. Krishnapuram, R. Rosales, and R.B. Rao. Bayesian co-training. *Journal of Machine Learning Research*, 12:2649–2680, 2011.

Z. Zafrulla, H. Brashear, T. Starner, H. Hamilton, and P. Presti. American sign language recognition with the Kinect. In *Proceedings of the 13th International Conference on Multimodal Interfaces*, pages 279–286, 2011.

L. Zhang, X. Chen, C. Wang, Y. Chen, and W. Gao. Recognition of sign language subwords based on boosted hidden Markov models. In *Proceedings of the 7th International Conference on Multimodal Interfaces*, pages 282–287, 2005.

T. Zhang and B. Yu. Boosting with early stopping: Convergence and consistency. *Annals of Statistics*, pages 1538–1579, 2005.

Y. Zhang, W. Liang, J. Tan, Y. Li, and Z. Zeng. PCA & HMM based arm gesture recognition using inertial measurement unit. In *Proceedings of the 8th International Conference on Body Area Networks*, pages 193–196, 2013.

F. Zheng, G. Zhang, and Z. Song. Comparison of different implementations of MFCC. *Journal of Computer Science and Technology*, 16(6):582–589, 2001.

Journal of Machine Learning Research 15 (2014) 3037-3064 Submitted 3/14; Published 10/14

Effective String Processing and Matching for Author Disambiguation

Wei-Sheng Chin D01944006@NTU.EDU.TW
Yong Zhuang R01922139@NTU.EDU.TW
Yu-Chin Juan R01922136@NTU.EDU.TW
Felix Wu B99902090@NTU.EDU.TW
Hsiao-Yu Tung B98901044@NTU.EDU.TW
Tong Yu R01922141@NTU.EDU.TW
Jui-Pin Wang R01922165@NTU.EDU.TW
Cheng-Xia Chang R01944041@NTU.EDU.TW
Chun-Pai Yang R99902109@NTU.EDU.TW
Wei-Cheng Chang B99902019@NTU.EDU.TW
Kuan-Hao Huang B99902059@NTU.EDU.TW
Tzu-Ming Kuo B99902073@NTU.EDU.TW
Shan-Wei Lin B99902023@NTU.EDU.TW
Young-San Lin B97902055@NTU.EDU.TW
Yu-Chen Lu B98902105@NTU.EDU.TW
Yu-Chuan Su R01922159@NTU.EDU.TW
Cheng-Kuang Wei B98901037@NTU.EDU.TW
Tu-Chun Yin D00922023@NTU.EDU.TW
Chun-Liang Li R01922001@NTU.EDU.TW
Ting-Wei Lin R01944011@NTU.EDU.TW
Cheng-Hao Tsai R01922025@NTU.EDU.TW
Shou-De Lin SDLIN@CSIE.NTU.EDU.TW
Hsuan-Tien Lin HTLIN@CSIE.NTU.EDU.TW
Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
Department of Computer Science and Information Engineering
National Taiwan University
Taipei 106, Taiwan

Editors: Senjuti Basu Roy, Vani Mandava, Martine De Cock

Abstract

Track 2 of KDD Cup 2013 aims at determining duplicated authors in a data set from Microsoft Academic Search. This type of problems appears in many large-scale applications that compile information from different sources. This paper describes our solution developed at National Taiwan University to win the first prize of the competition. We propose an effective name matching framework and realize two implementations. An important strategy in our approach is to consider Chinese and non-Chinese names separately because of their different naming conventions. Post-processing including merging results of two predictions further boosts the performance. Our approach achieves F1-score 0.99202 on the private leader board, while 0.99195 on the public leader board.

Keywords: deduplication, author disambiguation, name matching

1. Introduction

Track 2 in KDD Cup 2013 is a task of name disambiguation. Ideally, a name uniquely identifies an entity (e.g., an author), but practically an entity may correspond to different names. For example, once two data sets assigning an entity two or more names are integrated, the entity may become associated with different names. In this article, we call these names as *duplicates* of the original entity.

The data set is offered by Microsoft Academic Search (MAS). As a search engine, MAS integrates information of authors and their publications from different sources. We have mentioned that duplicates more frequently occur when data sets are integrated to a larger one, so MAS naturally suffers from this issue. The aim of this competition is to find which of around 250,000 authors are duplicates. We are given seven files, `Author.csv`, `Paper.csv`, `PaperAuthor.csv`, `Conference.csv`, `Journal.csv`, `Train.csv`, and `Valid.csv`. The two more important ones are `Author.csv` and `PaperAuthor.csv`, where the former stores author information (e.g., names and identifiers), and the latter gives authorships for around two million publications. Each line, a record, in both `Author.csv` and `PaperAuthor.csv` includes an author identifier and a name. The task is to upload duplicated identifiers in `Author.csv`. Other details of data sets and the competition can be found in Roy et al. (2013). Because no training information is given, the problem is an unsupervised learning task. The evaluation measure is the average of F1-scores over all authors in `Author.csv`. The definition of F1-score is

$$\text{F1-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}},$$

where

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \text{ and}$$
$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negative}}.$$

For example, to find author A's duplicates, if our algorithm returns A, B, C, F, and the true duplicates are A, B, C, D, E, then

$$\text{precision} = \frac{|\{A,B,C\}|}{|\{A,B,C\}| + |\{F\}|} = \frac{3}{4}, \text{ and}$$
$$\text{recall} = \frac{|\{A,B,C\}|}{|\{A,B,C\}| + |\{D,E\}|} = \frac{3}{5}.$$

In the competition, 20% of authors in `Author.csv` are used to evaluate duplicates submitted by participants. We refer to them as results on the public leader board. For the final evaluation, the remaining 80% authors in `Author.csv` are used and the F1-scores are called results on the private leader board. The baseline F1-score on the public leader board is 0.94411 by assuming no duplicates (i.e., all author names are considered as different entities).

Author disambiguation is a difficult problem that is also known as entity resolution (Whang et al., 2013; Köpcke and Rahm, 2010), duplication elimination/detection (Ananthakrishna et al., 2002), object matching (Köpcke et al., 2010), and record linkage (Brizan and Tansel, 2006). Generally speaking, this problem includes two fundamental phases, similarity measurement and record grouping; if the similarity is high enough, then two records should be assigned to a group corresponding to one real entity. Note that a record in a data set refers to a real-world object; for example, in the case of author disambiguation, a record refers to an author of a paper. It has been pointed out by, for example, Köpcke et al. (2010) and Bilenko et al. (2003), that algorithms well-tuned for some data sets may not perform well on other data sets. Thus, we should carefully design suitable measurements and tune parameters for distinct applications. Some data sets intrinsically have a hierarchical structure; for example, customer data sets may contain detailed addresses so selling records can be divided into several subsets according to the countries, states, and cities. Ananthakrishna et al. (2002) proposed a disambiguation method to find duplicates by using hierarchical information. That is, two records are referred to one real-world object if they are considered as duplicates at each level of the hierarchy. In Bhattacharya and Getoor (2004), the authors iteratively merge a pair of highly similar records to be duplicates for reducing the false positive rate using more conservative strategies (e.g., two records are merged if their similarity is larger than a threshold). In each iteration, the duplicates found are used to evaluate the distance between two entries; i.e., the distance between entries depends on the overlap between duplicates found in previous iterations of the two entries. In Bilenko et al. (2003), support vector machine (SVM) is used to integrate measurements on multiple attributes. For example, similarity between two authors can be the weighting sum of Jaro distances between their first names, middle names, last names, and addresses, where the weights correspond to the coefficients of SVM. They report that with the existence of typos the string-based distances (e.g., Levenshtein distance and Jaro distances) are superior to the token-based distances (e.g., Jaccard similarity and cosine distance with TF-IDF). Because typos and other types of noise exist in KDD Cup 2013 track 2 (see Section 2.2), we can expect the difficulty of using token-based techniques. Whang and Garcia-Molina (2012) proposed an iterative framework to solve disambiguation problems when there are different types of entries in the data set. Their framework allows users to set the logical relations between these types, and automatically find the best order to conduct disambiguation for different types. Note that different types of disambiguation can be handled in parallel if they are logically independent (i.e., the result of one type of disambiguation does not provide additional information for the disambiguation of another type). Take the KDD Cup 2013 as an example. The types of entries include author, publication, conference, journal, etc. Obviously, a parallel disambiguation for conferences and journals is possible, but the disambiguation of publications and conferences should be ordered. Treeratpituk and Giles (2009) provide rich measurements on many attributes (e.g., the last name of authors, authors' affiliation and coauthorship) to judge whether two authors correspond to one entity. In KDD Cup 2013, due to missing values, noises, and insufficient information we can not directly generate reliable features using their measurements. Consequently, similar to Torvik and Smalheiser (2009); Bilenko et al. (2003), we consider author-name strings as the major sources to measure the similarity between two authors. Thus all strategies in our algorithm are highly related to string processing. Furthermore, our approach is related to the iterative

Filename	Fields in each line	#Entries
`Author.csv`	AuthorID, Name, Affiliation	247,203
`PaperAuthor.csv`	PaperID, AuthorID, AuthorName, AuthorAffiliation	12,776,670
`Paper.csv`	PaperID, Title, ConferenceID, JournalID, Keyword	2,267,542
`Conference.csv`	ConferenceID, ShortName, FullName, Homepage	4,544
`Journal.csv`	JournalID, ShortName, FullName, Homepage	15,150

Table 1: Brief description of each file.

scheme proposed by Bhattacharya and Getoor (2004) because we also iteratively identify reliable duplicates and then refine results.

The next section summarizes the noise sources so that we can further understand the difficulties in the competition. We describe our approach in Section 3. It is realized in our two implementations described in Sections 4 and 5, respectively. Section 6 proposes a strategy to ensemble the predictions from these two implementations. We handle typographical errors (typos) in Section 7. Our results are summarized in Section 8, and the last section provides a detailed comparison on several successful approaches in this competition. Section 10 concludes this paper and gives some possible directions. Besides, our implementations are available at

`https://github.com/kdd-cup-2013-ntu/track2`.

A preliminary version of this work appears in a conference paper (Chin et al., 2013). The main extensions include three new sections, Section 2, Section 9, and Section 10, and the provision of more details in many places.

2. An Overview of Data sets

In this section, we firstly give a brief introduction about the content in the data set. Then we discuss various types of noise observed in the competition data.

2.1 Data Set Description

The organizers of KDD Cup 2013 release seven files, `Author.csv`, `PaperAuthor.csv`, `Conference.csv`, `Journal.csv`, `Paper.csv`, `Train.csv`, and `Valid.csv`. We do not use `Train.csv` and `Valid.csv` because they are for Track 1. Details of other files are shown in Table 1.

2.2 Noisy Patterns in the Competition Data

A search engine like MAS integrates relationships between authors and papers from different sources. What MAS must do is parsing and merging, but this process of integration results in some unexpected errors. By careful analysis, we observe the following types of noisy data from the sets given by MAS.

- **Alleviation**: Two entries in `Author.csv` are "2071403, Yuri Gurevichy" and "926979, Yuri Gurevichyand." At first glance, they are different persons. However, if we eliminate the last 3 characters "and" from the second author, the resulting author "Yuri Gurevichy"

may be the same as the first one. It is hard to decide because the two may actually be different.

- **Unusual Name**: The entry "894651, 560263, A Case Study," may provide incorrect author information because "A Case Study" is not a common name. However, we can not rule out the possibility that it is someone's name. Besides, in `Author.csv`, an entry "1813899, Ming-Wei Chang Dan Roth" appears. Apparently, this entry concatenates two author names.
- **Inconsistent Information**: In `Author.csv`, we get an entry "1791054, Steven L. Salzberg," which means the 1791054th author is "Steven L. Salzberg." However, in `PaperAuthor.csv`, the entry "310678, 1791054, David Saunders" indicates that the 1791054th author is "David Saunders." Therefore, the information in `Author.csv` and PaperAuthor.csv is not consistent.
- **Typo**: Some names in `Author` may contain typos. For example, "Chih-Jen Lin" becomes "Chi-Jen Lin."
- **Incomplete Name**: For the names "Michael Jordan," "M. I. Jordan," and "Michael I. Jordan," some authors have middle names, while others do not. Besides, for the name "M. I. Jordan," all words except the last name are abbreviated. Then it is difficult to decide if "Michael Jordan" and "Michael I. Jordan" correspond to the same author.
- **Empty Entry**: For example, the 14143th and 56473th entries in `Author.csv` are empty, so we cannot obtain any information.
- **Missing Value**: Some information of an entry is missing. In `PaperAuthor.csv`, each entry contains "PaperID, AuthorID, AuthorName, AuthorAffiliation." However, for the entry "127675, 1360414, Chih-Jen Lin" in `PaperAuthor.csv`, it only tells us that the 1360414th author named Chih-Jen Lin writes the 127675th paper. The affiliation of the author is missing.
- **Nickname**: Instead of using real names, some authors use nicknames in their publications. For example, "532490, 1060199, William R. Gates" and "717206, 1060199, Bill Gates" may be the same author, although the latter uses the nickname "Bill."
- **Wrong matching between authors and papers**: Errors of matching authors and papers exist in `PaperAuthor.csv`. From the entry "1360414, Chih-Jen Lin, National Taiwan University" in `Author.csv`, we check papers written by him in `PaperAuthor.csv`. Although most obtained entries are his papers, some are not. For example, a record "674977, 1360414, Chih-jen Lin" indicates that "Chih-jen Lin" writes the 674977th paper. However, the 674977th paper has the title "Built-In Current Sensor for IDDQ Test in CMOS," and is not written by the "Chih-Jen Lin" from National Taiwan University.
- **Non-English characters**: Some non-English characters like "43416, J. Jürjens" cause difficulties to compare names.

We carefully take these noisy patterns into consideration in designing our approach.

3. Our Approach

In this section, we discuss three key strategies of our approach, and then introduce a framework based on these strategies. Two implementations of the framework were finished by two groups within the team. They are respectively described in Sections 4 and 5.

3.1 The Main Strategies

The first strategy is that we identify duplicates mainly based on string matching; in particular, name matching. Academic authors often use their real names. If two authors are duplicates, then their name strings are similar. Therefore, name matching is very effective for this problem.

The second strategy is that if an author in `Author.csv` has no publication records in `PaperAuthor.csv`, then we assume that this author has no duplicates. In our earlier experiments, this strategy significantly improves the F1-score by around 0.02. Apparently MAS implements this rule internally, so we admit that this strategy may not be useful for other data sets.

The third strategy is to classify an author as Chinese or non-Chinese before any string matching. This strategy is useful because Chinese and non-Chinese names have some crucial differences. First, a western name may be written without the middle name. For example, "Vladimir Naumovich Vapnik" may write his name as "Vladimir Vapnik." In contrast, Chinese generally do not omit any part of their name. For instance, "Chih Lin" and "Chih Jen Lin" are almost surely different authors. Second, Chinese last names provide much less information than non-Chinese ones because some Chinese last names are very common (e.g., "Wang" and "Lin") and the romanization process may map different last names to the same English word (e.g., "林" and "藺" are romanized to "Lin"). The common last names cause difficulties to analyze a shortened Chinese name. For example, there are much more names that can be shortened as "C. J. Lin" than those as "E. W. Dijkstra." Besides, Korean, Vietnamese, and Singaporean names have a similar structure (i.e., two-word first name and a shortened last name), so we include them along with Chinese names. Examples include "Chi Minh Ho" (Vietnamese), "Yong Jun Bae" (Korean), and "Hsien Loong Lee" (Singaporean). Interestingly, we do not consider Japanese names because they often have a longer last name and a one-word first name (e.g., "Ichiro Suzuki").

For easy description, in this paper, we categorize words to two classes, shortened words and full words. A word is considered shortened if it is one single character or includes a period. For example, "C" and "Chris." are shortened words. A word is a full word if it is not a shortened word. Similarly, we consider two types of names. A name should be a shortened name if it consists of at least one shortened word; otherwise, the name is a full name.

3.2 The Framework

Our framework can be divided into six stages. Here we briefly introduce each stage, but leave details in Sections 4 and 5. Note that if not specified, in the sequel "an author" refers to an unique identifier (rather than a name and a human).

1. **Chinese-or-not**. We classify each author as Chinese or non-Chinese.
2. **Cleaning**. To efficiently compare author names, we remove redundant information. For example, "CHIH JEN LIN" is likely to be a duplicate of "chih jen lin," so we lowercase all strings.
3. **Selection**. For each author we select some candidates of possible duplicates. Naively, an author should be compared with all other authors, but the computational cost is high.

Therefore, we select only some candidates for subsequent analysis. At this stage, recall is the main concern because any missed names cannot be recovered in a later stage.

4. **Identification**. For each author, we check if those in the candidate set are duplicates or not.
5. **Splitting**. Because some names are wrongly grouped together after the identification stage, we make corrections by splitting some of them.
6. **Linking**. This stage maps author names back to author identifiers. This step is needed only for our second implementation; see more explanation below.

We illustrate an important difference between our two approaches by the following example.

Author identifier	1001
Name in `Author.csv`	"Chih Jen Lin"
Names in `PaperAuthor.csv`	"C. J. Lin" "Chih Jen Peter Lin" "C. J. P. Lin"

This table shows that, in the given data, one identifier may correspond to different names. Following the competition requirement to find duplicates of each author identifier, in the first implementation, each author is referred to as an author identifier. However, the second implementation treats each name string as an author. That is, the four names in the above examples are considered different. The algorithm must group them together among all name strings. Therefore, the second implementation needs a linking stage in the end.

4. The First Implementation

In this section, we discuss details of the first implementation. Each sub-section corresponds to a stage of the framework. Note that the linking stage is not performed in this implementation.

4.1 Chinese-or-not

An author is classified to be either Chinese or non-Chinese by the flowchart in Figure 1. The process relies on information including the romanization of common Chinese last names and Chinese syllables. We build two dictionaries, which differ in the coverage of Chinese words. The first dictionary contains 2,381 English words that are the romanization of top 100 Chinese last names and 410 syllables. Each Chinese last name is romanized to four English words because there are four different romanization systems in Taiwan,[1] and the list of Chinese syllables is publicly available on Wikipedia.[2] Note that romanization of several Chinese words can be the same. The second dictionary extends from the first by including 853 additional words of the romanization of 406 Chinese last names and 20 Korean last names.[3] Moreover, each dictionary consists of two sub-dictionaries storing last names

1. The romanization tool we used can be found at `http://www.boca.gov.tw/content?CuItem=5609&mp=1`.
2. These Chinese syllables can be found at `http://en.wikipedia.org/wiki/Comparison_of_Chinese_romanization_systems` and `http://www.pinyin.info/romanization/compare/gwoyeu_romatzyh.html`.
3. The additional Chinese last names and the Korea last names are available at `https://en.wikipedia.org/wiki/List_of_common_Chinese_surnames` and `http://mirror.enha.kr/wiki`, respectively

and syllables respectively. Some words in our Chinese dictionary also commonly appear in non-Chinese names (e.g., "van," "den," and "ben"), so we construct a "banned list" according our own knowledge.

The flow to determine if an input name is Chinese or not is in Figure 1. In this figure, hexagons are the final decision of Chinese or non-Chinese, ellipses stand for variables, and rectangles are operations. Every counter directly counts the number of inputs. For example, Counter_2 gives the total number of Chinese words in a name after checking the dictionary. To begin, a name is cleaned and then tokenized. Note that this cleaning step only removes non-alphabet characters and lowercase all remaining letters because we assume that punctuation provides no useful information and the matching between two words should be case-insensitive. Next, we consider three cases according to the number of full words (0, 1, or more) in an author name.

1. If there is no full word in a name (upper sub-tree in Figure 1), words in our Chinese dictionary cannot be used and hence the author is classified as non-Chinese. For example, "C J L" is considered as non-Chinese.
2. If an author has only one full word (right sub-tree in Figure 1), then we consider the author as Chinese if the full word is one of the last names in our dictionary but not in the banned list. This case is mainly for detecting Chinese authors with abbreviated first name such as "C. J. Lin," "C.-J. Lin," and "C. J. P. Lin."
3. For a name with more than one full word (left sub-tree in Figure 1), if it has more than one full word not in our Chinese dictionary, then the name is considered as non-Chinese. For example, "Chih J. Peter Lin" is Chinese if "Peter" is the only full word not in our dictionary. In contrast, if our dictionary does not contain "Chih," then the name becomes non-Chinese because of having two non-Chinese full words. The banned list plays a similar role as in the previous rule; if a word is on the list, then the word is not considered as a Chinese word. However, an exception of not applying the banned list is when the name contains a full Chinese word. This exception helps to recognize "Dan Wang" as a Chinese name and "Dan Roth" as non-Chinese. In detail, "Dan" is regarded as a Chinese word because "Wang" is a Chinese word. However, the other "Dan" is not a Chinese word because "Roth" is not. This idea can be found in the blocking mechanism of Counter_3; that is, if there is at least one Chinese word, the output of Counter_3 is the number of banned words. Otherwise, the output of Counter_3 is 0.

We illustrate our Chinese classification using an author with four full words. Given "Chih Jen Dean H. Lin," "H." is removed first, and then four words "Chih," "Jen," "Dean," and "Lin" are obtained. Counter_1 in Figure 1 returns 4 because of four full words. Then we use the left sub-tree in Figure 1. Assume that

1. our Chinese dictionary contains common last names {lin, wang} and common Chinese words {wang, chih, dean, and jen}, and
2. the banned list is {dean}.

Counter_2 returns 3 for one match of the last name "lin" and two matches of syllables "chih" and "jen." Because the output of Counter_2 is larger than 0 (some Chinese full words are found), Counter_3 is activated and returns 1 for the match of "dean." Then the adder returns 4. Finally, the subtracter, which subtracts the total number of full words from the result returned by the adder, returns 0 to be the number of non-Chinese full words, so "Chih Jen Dean H. Lin" is classified as Chinese.

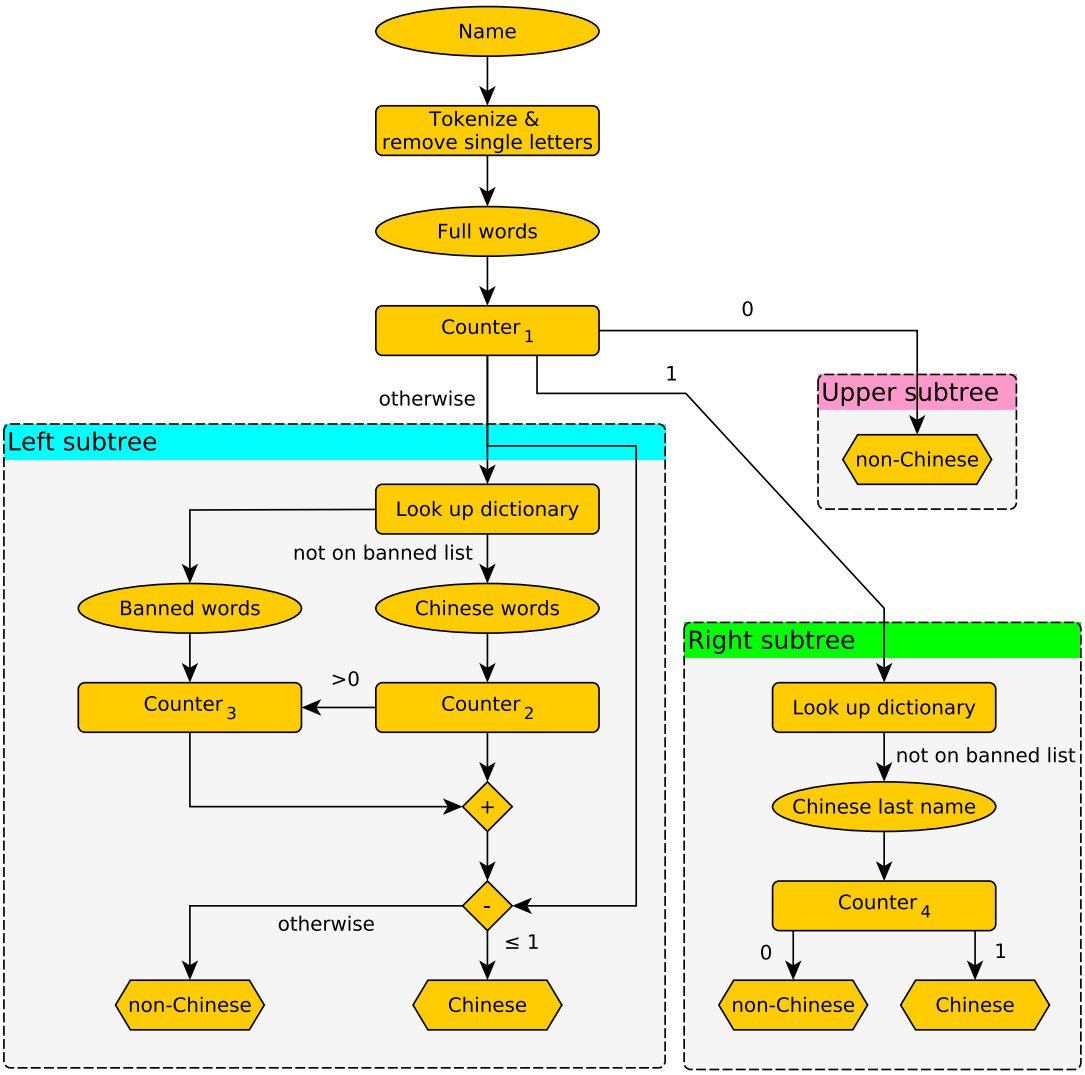


Figure 1: Flow of finding if a name is Chinese or not.

4.2 Cleaning

The purpose of data cleaning is to make name comparisons more accurate.[4] This process consists of the following steps.

1. Split two consecutive uppercase characters because we suspect that each character is an abbreviation of a word. For instance, we replace "CJ" with "C J."
2. Remove English honorifics (e.g., "Mr." and "Dr."), and some suffixes such as "Jr." and "III."
3. Transform uppercase to lowercase.

4. Note that in Section 4.1 a simpler cleaning step was conducted before identifying Chinese names.

4. Remove apostrophes and replace punctuation. For example, "o'relly" becomes "orelly." We then replace punctuation except periods with blanks. For example, "Chih-Jen" becomes "Chih Jen." We keep the period because it is useful to determine if a word is shortened or not.
5. Replace European alphabets with similar English alphabets. For example, we replace "ä" with "a."
6. Replace common English nicknames.[5] For example, we replace "bill" with "william."

4.3 Selection

In this stage, for each author, a set of possible duplicates are obtained. The purpose is to reduce the quadratic complexity of subsequent string matchings to linear. Our method includes two phases. In the first phase, we build a dictionary of (key, value) pairs. Each key is a set of words, while each value is a set of authors containing the key. To generate keys, we consider all word combinations of an author name. For example, "Chih Jen Lin" leads to the following keys.

"Chih"	"Jen"	"Lin"
"Chih Jen"	"Jen Lin"	"Chih Lin"
"Chih Jen Lin"		

Note that the order does not matter, so "Chih Lin" is considered the same as "Lin Chih." To avoid too many keys, we do not consider the combination of more than four words. In the second phase, for each given author, we examine his/her (key, value) pairs. If the "value" contains no more than 17 authors, then they are included as possible duplicates. The reason to discard a "value" with too many authors is because the corresponding key is too common. For example, two authors shall not be suspected as duplicates merely because their first name is "Lin." Including these names does not improve the recall much, but significantly increases the running time. In our experience, changing the threshold from 17 to 100 results in a three-fold increase of the running time. Moreover, a higher recall may not lead to a better final result.

4.4 Identification

Because the criteria to select candidates in the previous stage is loose, many authors were wrongly selected. In this stage, we find a subset as duplicates by applying a main procedure and two additional procedures. The main procedure, described in Section 4.4.1, uses many matching functions listed in Section 4.4.2. The two additional procedures are described in Section 4.4.3.

4.4.1 The Main Procedure for Identification

We iteratively apply matching functions to identify duplicates from the candidate set. Each matching function checks if two given authors are similar to each other. Criteria used in matching functions here should be stricter than those in the previous stage because the aim

5. The list of all nicknames we considered is available at `http://www.cc.kyoto-su.ac.jp/~trobb/nicklist.html` and `http://mentalfloss.com/article/24761/origins-10-nicknames`.

is to remove authors that were wrongly selected. However, although a strict criterion gives a high precision, it many cause a low recall. Therefore, we sequentially apply matching functions (listed in Section 4.4.2) from the strictest to the loosest.

Each iteration consists of two steps. First, between an author and any member of its candidate set, a matching function returns if the two names are duplicates or not. For a name "Chih Jen Lin," if his candidate set includes

"Chih Jen Lin," "Lin Chih Jen," "Chih Jen M. Lin," and "Chih Jen K. Lin,"

and the matching function requires that two names have the same word set, then "Chih Jen Lin" and "Lin Chih Jen" are considered as duplicates.

In the second step of an iteration, we make some corrections because duplicates obtained after applying matching functions may be wrong. For example, assume the following duplicates have been identified.

"C. J. Lin," "Chih Jen Lin," and "Chen Ju Lin."

Obviously they are not the same author because "Chih Jen Lin" and "Chen Ju Lin" are very different. We design a dry-run procedure to detect if a set of selected duplicates contains some very different names. If such names exist, then the set is discarded. For describing the dry-run function, we say that two author names are *loosely identical* if one of the following conditions holds.

1. One author has at least a shortened word and one author's first-character set of words is a subset of the other.
2. Both authors contain full words only and one author's first-three-character set of words is a subset of the other.

For example, "C J Lin" and "Chih Jen Lion" are loosely identical, while "Chih Jen Lin" and "Chen Ju Lin" are not.

In the dry-run procedure, we divide the selected set of duplicates to two sets: the longest-name set and the shorter-name set, where the former contains names with the largest number of words, while the latter contains the rest. The dry-run procedure is passed if the following conditions hold.

1. In the longest-name set, any two names are loosely identical.
2. In the shorter-name set, any name is loosely identical to at least one name in the longest-name set.

The identification procedure is shown in Algorithm 1.

4.4.2 Matching Functions

For easy description, we define the following relationship between two author names.

1. **The same name:** Two names share the same set of words.
   Example: "Chih Jen Lin" and "Lin Chih Jen."
2. **A shortened name:** The first author is a shortened name of the second one if the following conditions hold.
   - The full-word set of the first is a subset of the second.
   - Each shortened word in the first name is the prefix of a word in the second.

   Example: "Ch. J. Lin" and "Chih Jen Lin."

```
Data: Each author has a set of candidates.
Result: All authors are split to groups of duplicates.
begin
    for a ∈ all authors do
        set {a} as a's set of duplicates
    end
    for f ∈ matching functions do
        for a ∈ all authors do
            P ← {duplicates already identified for a}
            for c ∈ {candidates of a} do
                if f(a, c) then
                    P ← P ∪ {c's identified duplicates}
                end
            end
            if P passes the dry-run procedure then
                all authors in P are considered as duplicates
            end
        end
    end
end
```

Algorithm 1: The main procedure in the identification stage of the first implementation.

3. **A partially shortened name:** The first author is a shortened name of the second and each word in the first name is the prefix of a word in the second.
   Example: "C. J. Lin" and "C. J. Lint."
4. **Alias:** Name A in `PaperAuthor.csv` is an alias of name B in `Author.csv` if A and B have the same author identifiers, and B is a shortened name of A.
   Example: "C. J. Lin" is in `Author.csv`, while "C. Jen Lin" and "Chih Jen Lin" are in `PaperAuthor.csv`. If they have the same author identifiers, then "C. Jen Lin" and "Chih Jen Lin" are aliases of "C. J. Lin."

Now we introduce the following matching functions, each of which checks two input names.

1. Two authors have the same words in their names.
   Example: "Chih Jen Lin," and "Lin Chih Jen."
2. One is a shortened name of the other and both have the same initial-character set of words. However, this rule is not applied if
   - both authors are Chinese and each has at least one shortened word, or
   - one of the authors is Chinese and contains no more than two words.

   Example: "John S. Nash" and "John Smith Nash."
3. (Non-Chinese only) One author name is a shortened name of the other.
   Example: "Michael Jordan" and "Michael I. Jordan."
4. (Non-Chinese only) One author name is a partially shortened name of the other.
   Example: "Marek J. Druzdze" and "Marek J. Druzduzel."

5. Two authors have at least one identical alias.
   Example: Suppose that the name "1273890, Thomas J. Webb" in `PaperAuthor.csv` is an alias of the author "1273890, Thomas Webb" in `Author.csv`, while "207564, Thomas J. Webb" is an alias of the author "207564, Thomas J. Webb." Then "207564, Thomas J. Webb" and "1273890, Thomas Webb" are considered as duplicates.
6. (Non-Chinese only) Two author names are loosely identical and both have at least one identical paper or affiliation.
   Example: "571595, Alex Pentland" and "993869, Alex Pentland Perceptual" are loosely identical and both have the same affiliation "MIT Media Laboratory."
7. The two authors satisfy the following conditions.
   - Each author name has at least three words.
   - Two author names are the same after removing their respective last word.
   - The last word of each name should be the same after removing the last character of the longer word.

   Example: "Chih Jen Linu" and "Chih Jen Lin."
8. (Non-Chinese only) Only one author name has middle name and their last names differ in the last two characters.
   Example: "Michael I. Jordan" and "Michael Jordann."
9. (Chinese only) Two authors have at least one identical alias and one identical affiliation.
   Example: Assume "Chih Jen Lin" in `Author.csv` has the same identifier with "Chih Jen Lin" and "C. J. Lin" in `PaperAuthor.csv`. Similarly, "Chih J. Lin" has "Chih Jen Lin" and "C. J. Lin" in `PaperAuthor.csv`.
   The two authors have the same alias "C. J. Lin." If both have the same affiliation, then they are considered as duplicates.
10. (Chinese only) Each author has at least three words, but has no shortened words. Moreover, one author's word set is a subset of the other.
    Example: "Michael Chih Jen Lin" and "Chih Jen Lin."
11. (Chinese only) Both author names have more than three words and their lists of initial characters are the same.
    Example: "Michael Chih Jen Lin" and "M. Chih Jen Lin." Their lists of initial characters are both "m c j l."
12. (Chinese only) Both author names have more than three words, neither has a shortened word, and the full-word set of one's name is a subset of the other.
    Example: "Michael Chih Jen Lin" and "Michael Jackson Chih Jen Lin."
13. (Chinese only) The two authors satisfy one of the following conditions.
    - Each has three words and both have the same list of initial characters.
    - Neither has a shortened word and one author's full-word set is a subset of the other.

    Example: "Lin Chih Jen" and "C. Jen Lin."

4.4.3 Additional Procedures for Identification

We conduct two additional procedures to improve the identification of duplicates. Instead of fitting them to the main procedure, we find that separately considering them is more suitable.

Same paper title: Because data are collected from different sources, some papers have an identical title after removing non-alphabet characters. For any two such papers, if an author of one paper is a partially shortened name of an author of the other paper, then the two authors are considered as duplicates. For example, assume two papers are shown in Table 2. "C. C. Chang" is a partially shortened name of "Chih-Chung Chang," while "C. J. Lin" is that of "Chih J. Lin" and "Chih Jen Lin." Therefore, authors 1 and 2 are regarded as duplicates, and so are authors 3 and 5.

Paper IDs	1	2
PaperName	LIBSVM	lib-svm
AuthorList	1, C. C. Chang 3, Chih Jen Lin 5, C. J. Lin	2, Chih-Chung Chang 3, Chih J. Lin

Table 2: An example of using papers with the same title to identify duplicates.

Parsing alleviation: Because of incorrect string parsing, some names such as "Chih JenLin," and "Chih Jen LinAN" may appear although the correct one is "Chih Jen Lin." To find duplicates for these ill-formed names, we map each name to some keys and group names with the same key as duplicates.

To obtain keys and identify duplicates, we run two separate steps. The first one uses two keys. After some duplicates have been identified, the second uses three keys to get more duplicates.

The step of using two keys generates keys for each author. Words in a name are concatenated and lowercased to get the first key. By removing the last character of this key, we generate the second key if one of the following conditions holds.

1. The name has one shortened and two full words.
2. The name has more than two full words and the length of the name is greater than 12.
3. The length of each word in the name is greater than four.

For example, "Chih Jen Lin" has a key "chihjenlin," while "Chih J. Lint" has keys "chihjlint" and "chihjlin."

In the step of using three keys, the first key is the same as that in the previous step. The second key is also by removing the last character of the first key, but it is generated if one of the four conditions holds. These four conditions are similar to the three conditions used in generating the second key in the case of using two keys except that the first condition is modified to the following two rules:

1. The name has one shortened and two full words. Moreover, the length of the last word is greater than four.
2. The name has more than two full words and the length of the last name is greater than four.

Then we remove the last two characters of the first key to get a new key if one of two conditions holds.

1. The length of the name is greater than 15 and that of the last word is greater than five.
2. The number of full word in one's name is greater than two and the length of the last word is greater than five.

For example, "Petra QuillfeldtA" has keys "petraquillfeldta," "petraquillfeldt," and "petraquillfeld."

4.5 Splitting

In some groups of duplicates that have been identified, we still observe very different author names. For example, the following authors are considered as duplicates after the identification stage.

"k. kobayashi" "keven w. kobayashi"
"kazuo kobayashi" "kunikazu kobayashi"

When the third matching function mentioned in Section 4.4.2 is applied to the author "k. kobayashi," the above four authors are considered as duplicates. Then because "keven w. kobayashi" is the longest name in the set, and "kazuo kobayashi" and "kunikazu kobayashi" are loosely identical to "keven w. kobayashi," they pass the dry-run function. Obviously the grouping is incorrect. Therefore, in this splitting stage, we check the number of incorrectly identified pairs (details described below) in every set of duplicates. If the number exceeds three, then we dissolve the group and each element goes back to be an individual. We say two authors are incorrectly identified if they satisfy that
1. Each author name is a full name with two words.
2. Neither author name is a partially shortened name of the other.
For example, "kazuo kobayashi" and "kunikazu kobayashi" satisfy the above criteria because "kazno" is not a prefix of "kunikazu" and vice versa. We do not consider names with more than two words because "Alex Pentland Perceptual" and "Alexander Pentland" may be unexpectedly treated as incorrect duplicates.

5. The Second Implementation

In this section, we introduce the second implementation following the framework in Section 3. As mentioned in Section 3.2, we treat all names in `PaperAuthor.csv` and `Author.csv` as individuals and find groups of duplicates. Only in the end we obtain groups of author identifiers as requested by the competition.

5.1 Chinese-or-not

In contrast to Section 4.1, the implementation is simpler here. We build a Chinese dictionary that consists of 694 words of romanized Chinese syllables[6] and a banned list including some manually-selected words such as "ben", "dan", and "long."[7] For each author name after tokenization, we check if any word is on the Chinese list but not on the banned list. If such a word exists, we classify the name as Chinese. Our Chinese-or-not techniques were developed by two groups independently, so the dictionaries and the banned lists are slightly different while their designing spirits are identical. Note that the dictionary in the second

6. Related information can be found at `http://www.chineseinla.com/lastname/key_ng.html` and `http://irw.ncut.edu.tw/general/chen813`.
7. For the full list, see `https://github.com/kdd-cup-2013-ntu/track2/blob/master/main2/chinese_name_list_v4.py`.

implementation collects less Chinese words. Because of using less Chinese information and simpler rules, results here may be worse than those obtained by the Chinese-or-not procedure in the first implementation.

5.2 Cleaning

This stage goes through three phases: character-based filtering, word-based filtering, and parsing alleviation.

1. **Character-based filtering:** This phase replaces European alphabets to English alphabets and we remove all punctuation except the blank.
2. **Word-based filtering:** This phase splits two consecutive uppercase characters (e.g., "CJ" → "C J"), removes English titles and some suffixes, transforms uppercase to lowercase, and replaces common English nicknames with formal names.[8] In addition, the nobility particles (e.g., "von" and "de") are removed, and we split each two-Chinese-character word to two words (e.g., "ChihJen" → "Chih Jen").
3. **Parsing alleviation:** This phase addresses incorrect string parsing by going through two steps. First, among names that are the same after blank removal, we keep only the longest one. For instance, if "Joseph Douglas Horton," "Josephdouglas Horton," and "JosephDouglasHorton" appear in the database, we keep only "Joseph Douglas Horton." The second step handles typos caused by incorrect string parsing; see examples in Section 4.4.3. For any pair of two names, if the longer one differs from the shorter one in less than four characters, then we remove the longer one. The threshold four is chosen by the scores on the leader board.

5.3 Selection

Recall that in this stage for each author name we obtain a candidate set. One author name is a candidate of another (and vice versa) if one of the following conditions holds.

1. Both names are exactly the same regardless of the order of words. **Example:** "Chih Jen Lin" and "Lin Chih Jen."
2. Both names are Chinese with more than two words or neither is Chinese. Further, the set of initial characters of words in a name is a subset of the other and so is the set of full words. **Example:** "Chih Jen Lin" and "Chih Jen K. Lin."
3. The set of initial characters of words in a name is a subset of the other. The shorter name has only one full word not in the longer, but the word is the prefix of a word in the longer name.
   Example: "Ch Jen Lin" and "Chih Jen Lin."

5.4 Identification

In contrast to the first implementation, we believe that the candidates selected in Section 5.3 are of high credibility, so this stage is not performed. That is, the candidates selected in Section 5.3 are identified as duplicates.

8. The nickname list is available at `https://code.google.com/p/author-dedupe/`.

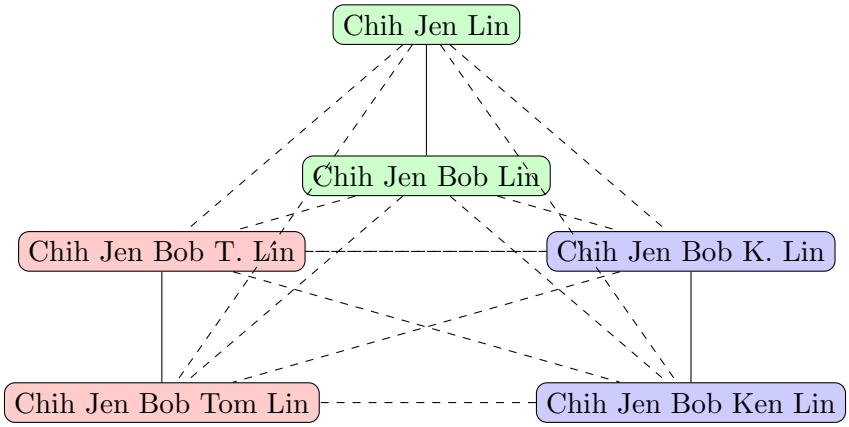


Figure 2: An example to illustrate the splitting procedure. Dashed edges are removed from the figure. In the end, the graph is split to three sub-graphs, each of which is considered as a set of duplicates.

5.5 Splitting

After Section 5.4, some names are still wrongly grouped as duplicates (e.g., "Chih Jen Lin," "Chih K. Lin," and "Chih H. Lin"). In this stage, we split such groups to improve precision. For easy explanation, we define the following terms.

1. **An extended/abbreviated name:** For two names satisfying the conditions in Section 5.3, if A is not shorter (not longer) than B, then A is an extended (abbreviated) name of B.
   Example: "Chih Jen Lin" is an abbreviated name of "Chih Jen Bob Lin," while "Chih Jen Bob Lin" is an extended name of "Chih Jen Lin."
2. **Common extended names (CEN):** In a set of duplicates, B is a CEN of A if B is an extended name of A and every A's extended name is either B's abbreviated or extended name. Note that any name is a common extended name of itself.
   Example: In Figure 2, assume "Chih Jen Bob Lin" and five other names are grouped as duplicates. All names except "Chih Jen Lin" are its extended names because of equal or longer length. We see that "Chih Jen Bob T. Lin" is not a CEN of "Chih Jen Bob Lin" because "Chih Jen Bob K. Lin" is an extended name of the latter, but is neither an extended nor abbreviated name of the former. Therefore, "Chih Jen Bob Lin" is the only CEN of "Chih Jen Bob Lin." Another example is in Figure 3. Further, Table 3 and Table 4 respectively list CENs of some authors in Figures 2 and 3.
3. **Longest common extended name (LCEN)**: A name B is an LCEN of A if B is a CEN of A and has the largest number of abbreviated names among A's CENs.
   Example: In Figure 3, assume "Chih Jen Bob Lin" and five other names are grouped as duplicates. For "Chih Jen Lin," whose CENs include "Chih Jen Lin," "Chih Jen Bob Lin," and "Chih Jen Bob Tom Ken Lin." In these CENs, "Chih Jen Bob Ken Lin" has three abbreviated names, more than the other two CENs. Therefore, "Chih Jen Bob Tom Ken Lin" is an LCEN of "Chih Jen Lin."

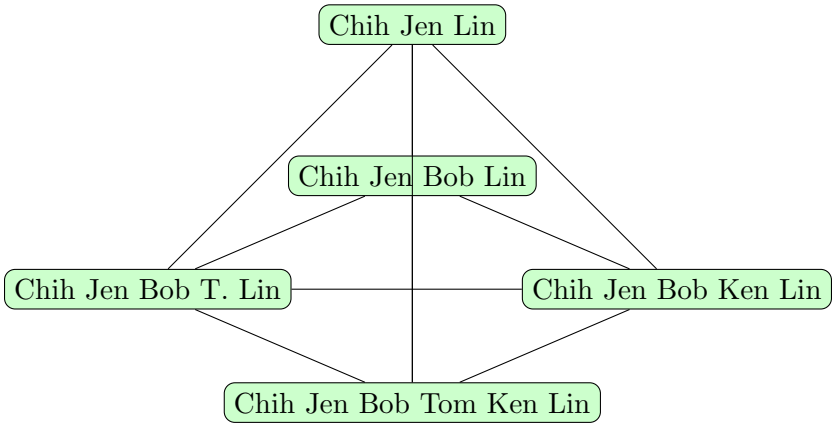


Figure 3: An example to illustrate the splitting procedure. All edges are preserved, so all names in this figure are considered as duplicates.

Author Name	CEN	LCEN
Chih Jen Bob Lin	Chih Jen Bob Lin	✓
Chih Jen Bob T. Lin	Chih Jen Bob T. Lin Chih Jen Bob Tom Lin	✓

Table 3: An part of examples for CEN and LCEN for Figure 2.

With the above definition, we describe the splitting procedure. For each group of duplicates, we construct an undirected graph so that nodes are names and any two names are connected by an edge. For each name, we get the corresponding LCEN. We then split any edge whose two nodes have different LCENs. Next, names in each connected sub-graph are considered as a set of duplicates.

Consider an example in Figure 2, where six names are considered as duplicates after the procedure in Section 5.4. The following example shows how we split edges. The LCEN of "Chih Jen Bob T. Lin" is "Chih Jen Bob Tom Lin," which differs from "Chih Jen Bob Lin" of "Chih Jen Bob Lin." Therefore, the link between "Chih Jen Bob T. Lin" and "Chih Jen Bob Lin" is removed. We remove many other edges by the similar reason. In the end only three edges remain.

We give another example in Figure 3, in which any two names have the same LCEN "Chih Jen Bob Tom Ken Lin." Therefore, we keep all edges. All names in this figure are then considered as duplicates.

5.6 Linking

As mentioned in Section 3.2, previous stages group duplicated names rather than identifiers. However, the competition task is to group duplicated identifiers, so some transformation is needed. Recall that each record in `Author.csv` and `PaperAuthor.csv` is a (name, identifier) pair. Our procedure starts from removing pair in `PaperAuthor.csv` that conflict with pairs in `Author.csv`. For example if "C J Lin" in `PaperAuthor.csv` and "C C Lin" in `Author.csv`

Author Name	CEN	LCEN
Chih Jen Lin	Chih Jen Lin	
	Chih Jen Bob Lin	
	Chih Jen Bob Tom Ken Lin	✓
Chih Jen Bob T. Lin	Chih Jen Bob T. Lin	
	Chih Jen Bob Tom Ken Lin	✓

Table 4: An part of examples for CEN and LCEN for Figure 3.

have the same identifier, we remove "C J Lin" because of the name mismatching. Next, for any group of names considered as duplicates, we construct an undirected graph so that each node is a name. For any node, we link it to all nodes satisfying that their (name, identifier) pairs appear in either `Author.csv` and or `PaperAuthor.csv`. In the end, if one identifier appears in two connected components of the graph, then the two groups are put together as duplicates. We consider the following example

Group_1		Group_2	
name	identifier	name	identifier
"C J Lin"	9_A, 41_A	"Chih Lin"	9_A, 41_A
"Chihjen Lin"	75_A	"C Lin"	8_{PA}, 10_{PA}
"Chih Jen Lin"	12_{PA}, 28_{PA}		

Assume that each group corresponds to a connected component of the constructed undirected graph. The subscript of an identifier indicates the source of the (name, identifier) pair, where "A" and "PA" denotes `Author.csv` and `PaperAuthor.csv`, respectively. Because the two groups share identifiers 9 and 41, all author identifiers in this table are considered as duplicates.

6. Ensemble

Because the two implementations in Sections 4 and 5 detect different sets of duplicates, an ensemble of their results may improve the performance. In this section, we propose a conservative setting to accurately find more duplicates by using background information such as an author's affiliation and field of study. The main idea is that if two authors have a similar background, then we are more confident in saying that they are duplicates.

For the two predictions generated by our implementations, we denote the prediction by the first implementation as the *major prediction*, while the other as the *auxiliary prediction*. This naming comes from the fact that the first implementation generally gives a better prediction; see Table 6. Given an author a, we say a' is an additional duplicate if a and a' are considered as duplicates only in the auxiliary prediction. We use a *filter* to check if a and a' have a similar background. If they pass the filter, then we consider a' as a possible duplicate of a. By an approach similar to that in Section 4.4.1, we choose duplicates from these possible duplicates by a dry-run function. That is, possible duplicates becomes real duplicates only if they pass the dry-run function. Moreover, in our practical experience, if a high-frequency word such as "Lin" exists in names considered as duplicates, the precision

file	author identifier	duplicates
major	10	10,11
auxiliary	10	10,11,12,13,14
ensembled	10	10,11,12,14

Table 5: An example of ensembling duplicates.

is often low. The reason is that people with a common last name are in general different. Therefore, we discard names having high-frequency words.[9]

Table 5 shows an example, where the additional duplicates of author 10 are 12, 13, and 14. Therefore, we apply the filter to pairs (10, 12), (10, 13), and (10, 14). Assume 12 and 14 pass the filtering. We then check if 10, 11, 12, and 14 could be duplicates by the dry-run function, and examine if high-frequency words exist in the names of these authors. In this example, we assume that the two checks are passed, so 12 and 14 are added as duplicates of 10.

In Section 6.1, we discuss the collection of background information, while in Section 6.2, we describe the filter. The ensemble procedure is summarized in Algorithm 2.

6.1 Collection of Background Information

For each author, we collect two sets of words: the affiliation-word set and the field-word set. The affiliation-word set is collected from affiliation information in `Author.csv` and `PaperAuthor.csv`; the field-word set is collected from paper titles and keywords in `Paper.csv`. The procedure can be divided into three stages: cleaning, stop-word removal, and collection.

In the cleaning stage, we remove common punctuation and handle several synonyms in the sources. From our statistics, some frequent words in affiliation sources are synonyms. For example, "univ" and "universidade" frequently appear in the data set, but they are equivalent to "university." For each set of synonyms, we replace all words with the most frequent one. Totally we consider three sets of synonyms, which are respectively transformed to words "university," "center," and "department."

In the stop-word removal stage, we generate two stop-word lists for affiliation and fields, respectively. Each includes a stop-word list and some high-frequency words (words occurred more than 1,704 and 32,000 in affiliation and field sources, respectively). In addition, for affiliation, we include several common country names. For fields, we include words that appear only once because such words are not very informative. After the two lists are generated, we remove all stop words.

Finally, in the collection stage, for each author all collected strings are split by space. The resulting two sets of words on affiliations and fields are then used by the filter in the ensemble process.

9. We call a word as a high-frequency one if it appears in `Author.csv` more than 1,200 times.

Data: A major prediction and an auxiliary prediction denoted by P_m and P_s, respectively.
Result: P_m and P_s are ensembled.
begin
  background information collection
  for $a \in$ all authors **do**
    $D_m \leftarrow$ duplicates of a from P_m
    $D_s \leftarrow$ duplicates of a from P_s
    $P \leftarrow D_m$
    for $a' \in D_s - D_m$ **do**
      if `filter` (a,a') **then**
        $P \leftarrow P \cup \{a'$ and its duplicates in $P_m\}$
      end
    end
    if any author in P has high-frequency words in its name **then**
      continue
    end
    if P passes the dry-run procedure **then**
      authors in P are duplicates
    end
  end
end

Algorithm 2: Ensemble of two results.

6.2 Filter

The filter considers that two authors have a similar background if the following conditions hold.

1. Two authors have at least two common words in their affiliation-word sets and at least one common word in their field-word sets, or they have at least one empty affiliation-word set and at least two common field words.
2. The two authors' field-word sets have no more than 75 common words.

The first condition implies that authors must share some words on affiliations or fields for having a similar background. The second condition addresses some special situations where two authors have papers in various fields. For such cases data tend to be more noisy.

7. Typo Correction

In this competition, typos occur in many places such as author names and paper titles. We focus on typos in author names because they are directly related to author disambiguation. From our observation, names in `PaperAuthor.csv` are too noisy, so we only handle typos in author names of `Author.csv`. To begin, we pre-process data by replacing all non-word characters with blanks and converting strings to lowercase. We also remove 11 manually selected

method	F1-score on leader board public	private	submitted
baseline	0.94411	0.94352	yes
implementation 1	0.99186	0.99198	yes
implementation 2	0.99071	0.99083	no
ensemble	0.99192	0.99201	no
ensemble + typo correction	0.99195	0.99202	yes

Table 6: F1-scores by our approach. Baseline means that we assume no duplicates at all. A submitted result means that it was uploaded during the competition. For unsubmitted results, we obtain F1-scores through the help of the competition organizers.

common words of author affiliations in `Author.csv` such as "department," "university," and "institute."

Because typos rarely occur, we assume that a word which appears at least twice in all author names is not a typo. Based on this principle, we split all words of author names in `Author.csv` to two sets. The first one includes words that appear only once as typo candidates, while the second includes all others. Next, for any typo candidate in the first set, we find their corrections from the second set. Specifically, a word in the second set is called a correction of a typo candidate if they differ in only one character. Note that a typo may have several corrections. For example, corrections of "cocn" may include "coin," "corn," and "conn."

After obtaining (typo, correction) pairs, the remaining task is to find duplicates. Two author names are considered as duplicates if

1. their word sets are the same after treating each typo the same as after its correction, and
2. their affiliations share at least one common word.

The first rule identifies "Lin Chih Jen" and "Litn Chih Jen" as duplicates if ("lint", "lin") is a (typo, correction) pair. However, the same rule also identifies "Lin C J" and "Litn C J" as duplicates, though the two names are likely different. Therefore, we impose the second rule. In the end, about 10 pairs of duplicates are obtained.

Finally, we merge the newly founded duplicates with results obtained in Section 6. Two author groups are combined if they share at least one author name.

8. Results

In our experiments, the used platform includes 96GB memory and two Intel Xeon E5-2620 2.0 Hz processors of which each has 6 physical cores. The running time of the first and the second implementations is around 15 minutes and 3 hours, respectively. The significant time gap is caused by the difference between the selection steps in the two implementations. The first implementation restricts the maximum number of authors in a group of possible duplicates to be less than a constant 17, but the other implementation does not put any

	F1-score on leader board		
method	public	private	submitted
implementation 1	0.99186	0.99198	yes
without Chinese-or-not	0.99109	0.99125	no
without dry-run	0.99097	0.99112	no
without both	0.98891	0.98934	no

Table 7: Evaluations of the first implementation with/without Chinese-or-not and/or dry-run.

limitation. The ensemble of the two implementations takes about 10 minutes, while the typo correction discussed in Section 7 needs about 7 minutes.

Table 6 presents the results (F1-score) on both public and private leader boards. Our first implementation gives slightly higher F1-scores than the second. After the post-processing procedure in Sections 6 and 7, the result is further boosted. Our approach gives the best F1-score in this competition, while the first implementation gives the second best (see the submitted results in Table 6). In Table 6 we see that some results were not uploaded during the competition because each team is not allowed to make more than two submissions per day. Therefore, we carefully submit results that may lead to the largest improvement. Several factors attribute to the success of our approach. Important ones include the identification of Chinese/non-Chinese names and effective string matching procedures to find duplicates with few of ad hoc parameters. Taking Chinese-or-not and dry-run procedure as examples, Table 7 shows the degeneration of implementation 1 if we do not include them into our approach. The degeneration is significant because the rank is lowered to the third if either the Chinese-or-not or the dry-run procedure is not applied. The rank is further lowered to the fourth if neither is applied.

9. Comparison on Approaches in KDD Cup 2013

At the KDD Cup workshop, four of the top 10 teams presented their work. These teams are

- 1st place: Algorithm @ National Taiwan University (Chin et al., 2013)
- 2nd place: SmallData @ UIUC (Liu et al., 2013)
- 4rd place: BS Man&Dmitry&Leustagos (Solecki et al., 2013)
- 7th place: SEU_WIP_AD (Zhao et al., 2013)

In this section, we conduct a comparison on them. Results are summarized in Table 9. Except the 7th that uses a semi-supervised strategy, all others are unsupervised and consider many heuristic rules. Compared with other approaches, ours highly relies on the information provided by `Author.csv`; in other words, we believe that `Author.csv` is trustworthy. The three unsupervised approaches can be described by the following four stages: pre-processing, finding candidates of duplicates for all authors, determining if the candidates are trustworthy, and post-processing. Based on Table 9, subsequently we highlight important differences between the four teams.

Aspect Considered	1st	2nd	4th	7th
Use `Author.csv`	√	√	√	√
Use `Paper.csv`	√	√	√	√
Use `PaperAuthor.csv`	√	√	√	√
Use `Conference.csv` and `Journal.csv`		√		√
Remove English honorifics	√	√	√	
Consider stop words	√	√	√	√
Split two consecutive uppercase characters	√		√	
Treat different types of punctuation using different strategies	√		√	
Transform European alphabets into English	√	√	√	
Remove nobility particles	√	√	√	
Replace nicknames using a public list	√	√	√	
Correct typos and accidentally added letters within every name	√	√	√	
Identify the last and the first name in a name	√	√		
Identify Asian authors	√	√		
Perform name disambiguation on authors' IDs (i.e., identify duplicated IDs)	√	√	√	√
Perform name disambiguation on authors' names (i.e., identify duplicated names)	√			
Consider name frequencies	√	√	√	√
Calculate similarity between authors using string distances		√	√	√
Use topic model			√	
Calculate similarity between authors using an bibliographic graph		√	√	
Compare authors using heuristic rules	√			√
Consider different naming conventions	√	√		
Use support vector machine/logistic regression				√
Select possible duplicates to reduce computational complexity	√	√	√	√
Check conflict names after duplicates are selected	√	√		
Use duplicated papers	√		√	
Refine result iteratively		√		√
Ensemble results generated by different algorithms	√			√

Table 8: Comparison on four of the top 10 teams

The pre-processing stage includes tasks such as cleaning data and correcting strings. This step is not directly related to author disambiguation but significantly improves the final results. However, some pre-processing procedures may bring additional noise into the data set or remove useful information. Therefore, in our approach we merely applied few conservative rules, so the data is changed in only a minor way. The most risky rule we

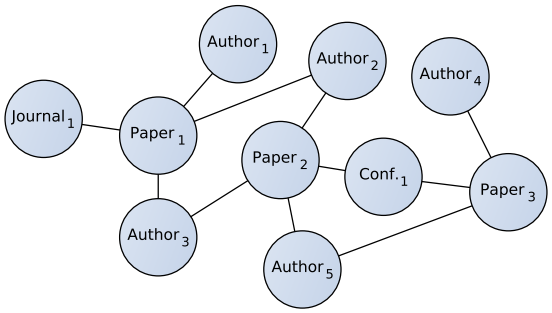


Figure 4: An example of bibliographic graph.

used is the nickname replacement. The 2nd-place team used more aggressive strategies than us because they tried to replace a name unit with a similar word which appears more frequently in the data set. The most aggressive pre-processing was conducted by the 4th-place team. They consider 16 steps including reversing the order of name units, replacing nickname, removing single letters within a name, etc. Therefore, ours is the most restrictive one among the three teams using purely unsupervised strategies. For the 7th-place team, they do not conduct any cleaning technique for pre-processing.

In the second stage, all teams select some possible candidates of duplicates. The aim is to reduce the computational complexity. The $O(n^2)$ cost of pairwise comparisons for n authors is prohibitively expensive. The selection stage begins with binning authors in the data set into different groups using some keys generated from their names. Similar to our strategies shown in Section 4.3, the 2nd-place team assign the authors with same keys (e.g., initial letters and identical name units) to the same bin. Furthermore, these two teams consider set comparisons, so a name's order of tokens does not effect the comparison result. The other two teams generate keys of an author using its first name and its last name. If we permute the tokens in a name, different results may be generated. The reason is that they generate keys in according to the location of the first name and the last name of an author. In addition, all these approaches find candidates for every author ID except our 2nd implementation that searches for possible candidates for every author name.

After we collect possible candidates for every author, the next step is to determine if these candidates are duplicates. In general, all proposed methods compare an author with all its candidates. The 4th place team's comparison and ours are purely string matching. In contrast, a bibliographic graph is often used to describe the complicated relation between authors, publications, and other information; see an illustration in Figure 4. Interestingly, these two teams did not build any bibliographic graph probably because data in `PaperAuthor.csv` are too noisy. That is, the links between authors and publications are unreliable. Note that Figure 2 and Figure 3 are not bibliographic graphs because they are built using our name comparison rules for checking conflict names. The other two teams measure differences between authors using meta-paths in their bibliographic graphs. If two authors are considered as duplicates then all their duplicates are merged. The operation is risky because the merge of two groups is determined by only a pair of authors without considering their duplicates. A simple remedy is to check the similarities between authors

in the merged group. If they are not similar enough, then the merged group would be split. Only the 2nd-place team and our approach conduct such a check. See the dry-run procedure in our 1st implementation, the splitting procedure in our 2nd implementation, and the ranking-based merging stage in the 2nd place team's solution for details. As mentioned before, semi-supervised learning techniques only appear in the 4th-place team's solution. They generate training data using some heuristic rules, and train SVM to decide if two authors are duplicates. Unfortunately, it seems that the data set is too noisy to provide reliable training data. A possible solution is to refine the result generated by SVM using hand-made heuristics (e.g., rules in Section 4.4).

Post-processing techniques mainly include identifying confident information and ensembling models. The 2nd-place team identifies confident information to avoid error propagation in their iterative framework. Ensemble is naturally considered by teams which have multiple approaches. We carefully designed rules to merge our two implementations and conduct typo corrections, while the 4th-place team applied a rule, called transient assumption, to aggressively merge their results.

10. Conclusion and Future Works

We can make the following observations and conclusions following the description of our approaches in this paper and the comparison in Section 9. First, we try our best to keep all information and delay the modification on the data set because the provided data set is noisy and incomplete. For instance, our typo correction is the last step of the whole procedure. In contrast, the 4th-place team conducts a similar process in their pre-processing stage. Second, an important advantage of using rule-based approaches is that we can easily trace the change of results after a rule is added. We can check the effectiveness of rules and identify useful information. For example, we find that different name conventions play an important role in author disambiguation when we add the matching function "one is a shortened name of the other and both have the same initial-character set of words." In contrast, some complicated methods like LDA are hard to analyze. Furthermore, human brain is still very powerful to analyze large-scale data sets if we have systematic schemes to filter out redundant information.

Overall, our experiments show that proposed techniques may be useful for other applications of author disambiguation. For example, others can apply our Chinese-or-not procedure when the separation of Chinese and non-Chinese authors is needed. Further, they can employ our rules designed for each of the two groups. Outputs of rules can then be used to train a binary classifier for determining whether two authors are duplicates. The parallelization of our algorithm is another interesting issue because the sizes of available data sets are growing. Besides, we are also interested in the behavior of graph-based approaches on noisy bibliographic data sets.

Acknowledgments

We thank the organizers for holding this interesting competition. We also thank the College of Electrical Engineering and Computer Science as well as the Department of Computer

Science and Information Engineering at National Taiwan University for their supports and for providing a stimulating research environment. The work was also supported by National Taiwan University under Grants NTU 102R7827, 102R7828, 102R7829, and by National Science Council under Grants NSC 101-2221-E002-199-MY3, 101-2628-E002-028-MY2, 101-2628-E002-029-MY2.

References

Rohit Ananthakrishna, Surajit Chaudhuri, and Venkatesh Ganti. Eliminating fuzzy duplicates in data warehouses. In *Proceedings of the VLDB Endowment*, pages 586–597, 2002.

Indrajit Bhattacharya and Lise Getoor. Iterative record linkage for cleaning and integration. In *Proceedings of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 11–18, 2004.

Mikhail Bilenko, Raymond Mooney, William Cohen, Pradeep Ravikumar, and Stephen Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23, 2003.

David Guy Brizan and Abdullah Uz Tansel. A survey of entity resolution and record linkage methodologies. *Communications of the IIMA*, 6(3):41–50, 2006.

Wei-Sheng Chin, Yu-Chin Juan, Yong Zhuang, Felix Wu, Hsiao-Yu Tung, Tong Yu, Jui-Pin Wang, Cheng-Xia Chang, Chun-Pai Yang, Wei-Cheng Chang, Kuan-Hao Huang, Tzu-Ming Kuo, Shan-Wei Lin, Young-San Lin, Yu-Chen Lu, Yu-Chuan Su, Cheng-Kuang Wei, Tu-Chun Yin, Chun-Liang Li, Ting-Wei Lin, Cheng-Hao Tsai, Shou-De Lin, Hsuan-Tien Lin, and Chih-Jen Lin. Effective string processing and matching for author disambiguation. In *Proceedings of the KDD Cup 2013 Workshop*, pages 7:1–7:9. ACM, 2013.

Hanna Köpcke and Erhard Rahm. Frameworks for entity matching: a comparison. *Data and Knowledge Engineering*, 69(2):197–210, 2010.

Hanna Köpcke, Andreas Thor, and Erhard Rahm. Evaluation of entity resolution approaches on real-world match problems. In *Proceedings of the VLDB Endowment*, pages 484–493, 2010.

Jialu Liu, Kin Hou Lei, Jeffery Yufei Liu, Chi Wang, and Jiawei Han. Ranking-based name matching for author disambiguation in bibliographic data. In *Proceedings of the KDD Cup 2013 Workshop*, pages 8:1–8:8. ACM, 2013.

Senjuti B. Roy, Martine D. Cock, Vani Mandava, Brian Dalessandro, Claudia Perlich, William Cukierski, and Ben Hamner. The Microsoft academic search dataset and KDD Cup 2013. In *Proceedings of the KDD Cup 2013 Workshop*, pages 1:1–1:6. ACM, 2013.

Benjamin Solecki, Lucas Silva, and Dmitry Efimov. KDD Cup 2013: author disambiguation. In *Proceedings of the KDD Cup 2013 Workshop*, pages 9:1–9:3. ACM, 2013.

Vetle I. Torvik and Neil R. Smalheiser. Author name disambiguation in MEDLINE. *ACM Transactions on Knowledge Discovery from Data*, 3(3):11:1–11:29, 2009.

Pucktada Treeratpituk and C. Lee Giles. Disambiguating authors in academic publications using random forests. In *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 39–48, 2009.

Steven Euijong Whang and Hector Garcia-Molina. Joint entity resolution. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, pages 294–305, 2012.

Steven Euijong Whang, David Marmaros, and Hector Garcia-Molina. Pay-as-you-go entity resolution. *IEEE Transactions on Knowledge and Data Engineering*, 25(5):1111–1124, 2013.

Jianyu Zhao, Peng Wang, and Kai Huang. A semi-supervised approach for author disambiguation in KDD CUP 2013. In *Proceedings of the KDD Cup 2013 Workshop*, pages 10:1–10:8. ACM, 2013.

Journal of Machine Learning Research 15 (2014) 3065-3105 Submitted 11/13; Revised 5/14; Published 10/14

High-Dimensional Learning of Linear Causal Networks via Inverse Covariance Estimation

Po-Ling Loh LOH@WHARTON.UPENN.EDU
Department of Statistics
The Wharton School
466 Jon M. Huntsman Hall
3730 Walnut Street
Philadelphia, PA 19104, USA

Peter Bühlmann BUHLMANN@STAT.MATH.ETHZ.CH
Seminar für Statistik
ETH Zürich
Rämistrasse 101, HG G17
8092 Zürich, Switzerland

Editor: Hui Zou

Abstract

We establish a new framework for statistical estimation of directed acyclic graphs (DAGs) when data are generated from a linear, possibly non-Gaussian structural equation model. Our framework consists of two parts: (1) inferring the moralized graph from the support of the inverse covariance matrix; and (2) selecting the best-scoring graph amongst DAGs that are consistent with the moralized graph. We show that when the error variances are known or estimated to close enough precision, the true DAG is the unique minimizer of the score computed using the reweighted squared ℓ_2-loss. Our population-level results have implications for the identifiability of linear SEMs when the error covariances are specified up to a constant multiple. On the statistical side, we establish rigorous conditions for high-dimensional consistency of our two-part algorithm, defined in terms of a "gap" between the true DAG and the next best candidate. Finally, we demonstrate that dynamic programming may be used to select the optimal DAG in linear time when the treewidth of the moralized graph is bounded.

Keywords: causal inference, dynamic programming, identifiability, inverse covariance matrix estimation, linear structural equation models

1. Introduction

Causal networks arise naturally in a wide variety of application domains, including genetics, epidemiology, and time series analysis (Hughes et al., 2000; Stekhoven et al., 2012; Aalen et al., 2012). However, inferring the graph structure of a causal network from joint observations is a rather challenging problem. Whereas undirected graphical structures may be estimated via pairwise conditional independence testing, with worst-case time scaling as the square of the number of nodes, estimation methods for directed acyclic graphs (DAGs) first require learning an appropriate permutation order of the vertices, leading to computational complexity that scales exponentially in the graph size. Greedy algorithms present an

attractive computationally efficient alternative, but such methods are not generally guaranteed to produce the correct graph (Chickering, 2002). In contrast, exact methods for causal inference that search exhaustively over the entire DAG space may only be tractable for relatively small graphs (Silander and Myllymaki, 2006).

1.1 Restricted Search Space

In practice, knowing prior information about the structure of the underlying DAG may lead to vast computational savings. For example, if a natural ordering of the nodes is known, inference may be performed by regressing each node upon its predecessors and selecting the best functional fit for each node. This yields an algorithm with runtime linear in the number of nodes and overall quadratic complexity. In the linear high-dimensional Gaussian setting, one could apply a version of the graphical Lasso, where the feasible set is restricted to matrices that are upper-triangular with respect to the known ordering (Shojaie and Michailidis, 2010). However, knowing the node order is unrealistic for many applications. If instead a conditional independence graph or superset of the skeleton is specified a priori, the number of required conditional independence tests may also be reduced dramatically. This appears to be a more reasonable assumption, and various authors have devised algorithms to compute the optimal DAG efficiently in settings where the input graph has bounded degree and/or bounded treewidth (Perrier et al., 2008; Ordyniak and Szeider, 2012; Korhonen and Parviainen, 2013).

Unfortunately, appropriate tools for inferring such superstructures are rather limited, and the usual method of using the graphical Lasso to estimate a conditional independence graph is rigorously justified only in the linear Gaussian setting (Yuan and Lin, 2007). Recent results have established that a version of the graphical Lasso may also be used to learn a conditional independence graph for variables taking values in a discrete alphabet when the graph has bounded treewidth (Loh and Wainwright, 2013), but results for more general distributions are absent from the literature. Bühlmann et al. (2014) isolate sufficient conditions under which Lasso-based linear regression could be used to recover a conditional independence graph for general distributions, and use the Lasso as a prescreening step for nonparametric causal inference in additive noise models; however, it is unclear which non-Gaussian distributions satisfy the prescribed conditions.

1.2 Our Contributions

We propose a new algorithmic strategy for inferring the DAG structure of a linear, potentially non-Gaussian structural equation model (SEM). Deviating slightly from the literature, we use the term *non-Gaussian* to refer to the fact that the variables are not jointly Gaussian; however, we do *not* require non-Gaussianity of all exogenous noise variables, as assumed by Shimizu et al. (2011). We proceed in two steps, where each step is of independent interest: First, we infer the moralized graph by estimating the inverse covariance matrix of the joint distribution. The novelty is that we justify this approach for non-Gaussian linear SEMs. Second, we find the optimal causal network structure by searching over the space of DAGs that are consistent with the moralized graph and selecting the DAG that minimizes an appropriate score function. When the score function is decomposable and the moralized graph has bounded treewidth, the second step may be performed via dynamic program-

ming in time linear in the number of nodes (Ordyniak and Szeider, 2012). Our algorithm is also applicable in a high-dimensional setting when the moralized graph is sparse, where we estimate the support of the inverse covariance matrix using a method such as the graphical Lasso (Ravikumar et al., 2011). Our algorithmic framework is summarized in Algorithm 1:

Algorithm 1 Framework for DAG estimation

1: **Input:** Data samples $\{x_i\}_{i=1}^n$ from a linear SEM
2: Obtain estimate $\widehat{\Theta}$ of inverse covariance matrix (e.g., using graphical Lasso)
3: Construct moralized graph $\widehat{\mathcal{M}}$ with edge set defined by $\mathrm{supp}(\widehat{\Theta})$
4: Compute scores for DAGs that are consistent with $\widehat{\mathcal{M}}$ (e.g., using squared ℓ_2-error)
5: Find minimal-scoring $\widehat{G}$ (using dynamic programming when score is decomposable and $\widehat{\mathcal{M}}$ has bounded treewidth)
6: **Output:** Estimated DAG $\widehat{G}$

We prove the correctness of our graph estimation algorithm by deriving new results about the theory of linear SEMs. We present a novel result showing that for almost every choice of linear coefficients, the support of the inverse covariance matrix of the joint distribution is identical to the edge structure of the moralized graph. Although a similar relationship between the support of the inverse covariance matrix and the edge structure of an undirected conditional independence graph has long been established for multivariate Gaussian models (Lauritzen, 1996), our core result in Theorem 2 does not exploit Gaussianity, and the proof technique is entirely new.

Since we do not impose constraints on the error distribution of our SEM, standard parametric maximum likelihood methods are *not* applicable to score and compare candidate DAGs. Consequently, we use the squared ℓ_2-error to score DAGs, and prove that in the case of homoscedastic errors, the true DAG uniquely minimizes this score function. As a side corollary, we establish that the DAG structure of a linear SEM is identifiable whenever the additive errors are homoscedastic, which generalizes a recent result derived only for Gaussian variables (Peters and Bühlmann, 2013). In addition, our result covers cases with Gaussian and non-Gaussian errors, whereas Shimizu et al. (2011) require all errors to be non-Gaussian (see Section 4.2). A similar result is implicitly contained under some assumptions in van de Geer and Bühlmann (2013), but we provide a more general statement and additionally quantify a regime where the errors may exhibit a certain degree of heteroscedasticity. Thus, when errors are not too heteroscedastic, the much more complicated ICA algorithm (Shimizu et al., 2006, 2011) may be replaced by a simple scoring method using squared ℓ_2-loss.

On the statistical side, we show that our method produces consistent estimates of the true DAG by invoking results from high-dimensional statistics. We note that our theoretical results only require a condition on the gap between squared ℓ_2-scores for various DAGs in the restricted search space and eigenvalue conditions on the true covariance matrix, which is a much weaker assumption than the restrictive beta-min condition from previous work (van de Geer and Bühlmann, 2013). Furthermore, the size of the gap is *not* required to scale linearly with the number of nodes in the graph, unlike similar conditions in van de Geer and Bühlmann (2013) and Peters and Bühlmann (2013), leading to genuinely high-dimensional results. Although the precise size of the gap relies heavily on the structure

of the true DAG, we include several examples providing intuition for when our condition could be expected to hold (see Sections 4.4 and 5.2 below). Finally, since inverse covariance matrix estimation and computing scores based on linear regression are both easily modified to deal with systematically corrupted data (Loh and Wainwright, 2012), we show that our methods are also applicable for learning the DAG structure of a linear SEM when data are observed subject to corruptions such as missing data and additive noise.

The remainder of the paper is organized as follows: In Section 2, we review the general theory of probabilistic graphical models and linear SEMs. Section 3 describes our results on the relationship between the inverse covariance matrix and conditional independence graph of a linear SEM. In Section 4, we discuss the use of the squared ℓ_2-loss for scoring candidate DAGs. Section 5 establishes results for the statistical consistency of our proposed inference algorithms and explores the gap condition for various graphs. Finally, Section 6 describes how dynamic programming may be used to identify the optimal DAG in linear time, when the moralized graph has bounded treewidth. Proofs of supporting results are contained in the Appendix.

2. Background

We begin by reviewing some basic background material and introducing notation for the graph estimation problems studied in this paper.

2.1 Graphical Models

In this section, we briefly review the theory of directed and undirected graphical models, also known as conditional independence graphs (CIGs). For a more in-depth exposition, see Lauritzen (1996) or Koller and Friedman (2009) and the references cited therein.

2.1.1 Undirected Graphs

Consider a probability distribution $q(x_1, \ldots, x_p)$ and an undirected graph $G = (V, E)$, where $V = \{1, \ldots, p\}$ and $E \subseteq V \times V$. We say that G is a *conditional independence graph* (CIG) for q if the following *Markov condition* holds: For all disjoint triples $(A, B, S) \subseteq V$ such that S separates A from B in G, we have $X_A \perp\!\!\!\perp X_B \mid X_S$. Here, $X_C := \{X_j : j \in C\}$ for any subset $C \subseteq V$. We also say that G *represents* the distribution q.

By the well-known Hammersley-Clifford theorem, if q is a strictly positive distribution (i.e., $q(x_1, \ldots, x_p) > 0$ for all $(x_1, \ldots, x_p)$), then G is a CIG for q if and only if we may write

$$q(x_1, \ldots, x_p) = \prod_{C \in \mathcal{C}} \psi_C(x_C), \tag{1}$$

for some potential functions $\{\psi_C : C \in \mathcal{C}\}$ defined over the set of cliques $\mathcal{C}$ of G. In particular, note that the complete graph on p nodes is always a CIG for q, but CIGs with fewer edges may exist.

2.1.2 Directed Acyclic Graphs (DAGs)

Changing notation slightly, consider a *directed* graph $G = (V, E)$, where we now distinguish between edges (j, k) and (k, j). We say that G is a *directed acyclic graph* (DAG) if there

are no directed paths starting and ending at the same node. For each node $j \in V$, let $\mathrm{Pa}(j) := \{k \in V : (k,j) \in E\}$ denote the *parent set* of j, where we sometimes write $\mathrm{Pa}_G(j)$ to emphasize the dependence on G. A DAG G *represents* a distribution $q(x_1, \ldots, x_p)$ if q factorizes as

$$q(x_1, \ldots, x_p) \propto \prod_{j=1}^{p} q(x_j \mid x_{\mathrm{Pa}(j)}). \tag{2}$$

Finally, a permutation π of the vertex set $V = \{1, \ldots, p\}$ is a *topological order* for G if $\pi(j) < \pi(k)$ whenever $(j,k) \in E$. Such a topological order exists for any DAG, but it may not be unique. The factorization (2) implies that $X_j \perp\!\!\!\perp X_{\nu(j)} \mid X_{\mathrm{Pa}(j)}$ for all j, where $\nu(j)$ is the set of all nondescendants of j (nodes that cannot be reached via a directed path from j) excluding $\mathrm{Pa}(j)$.

Given a DAG G, we may form the *moralized graph* $\mathcal{M}(G)$ by fully connecting all nodes within each parent set $\mathrm{Pa}(j)$ and dropping the orientations of directed edges. Note that moralization is a purely graph-theoretic operation that transforms a directed graph into an undirected graph. However, if the DAG G represents a distribution q, then $\mathcal{M}(G)$ is also a CIG for q. This is because each set $\{j\} \cup \mathrm{Pa}(j)$ forms a clique C_j in $\mathcal{M}(G)$, and we may define the potential functions $\psi_{C_j}(x_{C_j}) := q(x_j \mid x_{\mathrm{Pa}(j)})$ to obtain the factorization (1) from the factorization (2).

Finally, we define the *skeleton* of a DAG G to be the undirected graph formed by dropping orientations of edges in G. Note that the edge set of the skeleton is a subset of the edge set of the moralized graph, but the latter set is generally much larger. The skeleton is not in general a CIG.

2.2 Linear Structural Equation Models

We now specialize to the framework of linear structural equation models.

We say that a random vector $X = (X_1, \ldots, X_p) \in \mathbb{R}^p$ follows a *linear structural equation model* (SEM) if

$$X = B^T X + \epsilon, \tag{3}$$

where B is a strictly upper triangular matrix known as the *autoregression matrix*. We assume $\mathrm{E}[X] = \mathrm{E}[\epsilon] = 0$ and $\epsilon_j \perp\!\!\!\perp (X_1, \ldots, X_{j-1})$ for all j.

In particular, observe that the DAG G with vertex set $V = \{1, \ldots, p\}$ and edge set $E = \{(j,k) : B_{jk} \neq 0\}$ represents the joint distribution q on X. Indeed, Equation (3) implies that

$$q(X_j \mid X_1, \ldots, X_{j-1}) = q(X_j \mid X_{\mathrm{Pa}_G(j)}),$$

so we may factorize

$$q(X_1, \ldots, X_p) = \prod_{j=1}^{p} q(X_j \mid X_1, \ldots, X_{j-1}) = \prod_{j=1}^{p} q(X_j \mid X_{\mathrm{Pa}_G(j)}).$$

Given samples $\{X^i\}_{i=1}^n$, our goal is to infer the unknown matrix B, from which we may recover G (or vice versa).

3. Moralized Graphs and Inverse Covariance Matrices

In this section, we describe our main result concerning inverse covariance matrices of linear SEMs. It generalizes a result for multivariate Gaussians, and states that the inverse covariance matrix of the joint distribution of a linear SEM reflects the structure of a conditional independence graph.

We begin by noting that

$$\mathrm{E}[X_j \mid X_1, \ldots, X_{j-1}] = b_j^T X,$$

where b_j is the j^{th} column of B, and

$$b_j = \left(\Sigma_{j,1:(j-1)} \left(\Sigma_{1:(j-1),1:(j-1)}\right)^{-1}, 0, \ldots, 0\right)^T.$$

Here, $\Sigma := \mathrm{cov}[X]$. We call $b_j^T X$ the *best linear predictor* for X_j amongst linear combinations of $\{X_1, \ldots, X_{j-1}\}$. Defining $\Omega := \mathrm{cov}[\epsilon]$ and $\Theta := \Sigma^{-1}$, we see from Equation (3) that

$$\Sigma = (I - B)^{-T} \Omega (I - B)^{-1}. \tag{4}$$

(Note that $(I - B)$ is always invertible because B is strictly upper triangular.) Furthermore, we have the following lemma, proved in Appendix C.1:

Lemma 1 *The matrix of error covariances is diagonal:* $\Omega = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_p^2)$ *for some* $\sigma_i > 0$. *The entries of* Θ *are given by*

$$\Theta_{jk} = -\sigma_k^{-2} B_{jk} + \sum_{\ell > k} \sigma_\ell^{-2} B_{j\ell} B_{k\ell}, \qquad \forall j < k, \tag{5}$$

$$\Theta_{jj} = \sigma_j^{-2} + \sum_{\ell > j} \sigma_\ell^{-2} B_{j\ell}^2, \qquad \forall j. \tag{6}$$

In particular, Equation (5) has an important implication for causal inference, which we state in the following theorem. Recalling the notation of Section 2.1.2, the graph $\mathcal{M}(G)$ denotes the moralized DAG.

Theorem 2 *Suppose* X *is generated from the linear structural equation model* (3). *Then* Θ *reflects the graph structure of the moralized DAG; i.e., for* $j \neq k$, *we have* $\Theta_{jk} = 0$ *if* (j, k) *is not an edge in* $\mathcal{M}(G)$.

Proof Suppose $j \neq k$ and (j, k) is not an edge in $\mathcal{M}(G)$, and assume without loss of generality that $j < k$. Certainly, $(j, k) \notin E$, implying that $B_{jk} = 0$. Furthermore, j and k cannot share a common child, or else (j, k) would be an edge in $\mathcal{M}(G)$. This implies that either $B_{j\ell} = 0$ or $B_{k\ell} = 0$ for each $\ell > k$. The desired result then follows from Equation (5). ■

Note that Theorem 2 may be viewed as an extension of the canonical result for Gaussian graphical models, although we do *not* require ϵ to follow a Gaussian distribution, so the class of linear SEMs covered by Theorem 2 is much broader. A multivariate Gaussian distribution may be written as a linear SEM with respect to any permutation order π of the

variables, giving rise to a DAG G^π. In that case, Theorem 2 states that $\mathrm{supp}(\Theta)$ is always a subset of the edge set of $\mathcal{M}(G^\pi)$.

In the results to follow, we will assume that the converse of Theorem 2 holds, as well. This is stated in the following Assumption:

Assumption 1 *Let (B, Ω) be the matrices of the underlying linear SEM. For every $j < k$, we have*

$$-\sigma_k^{-2} B_{jk} + \sum_{\ell > k} \sigma_\ell^{-2} B_{j\ell} B_{k\ell} = 0$$

only if $B_{jk} = 0$ and $B_{j\ell} B_{k\ell} = 0$ for all $\ell > k$.

Combined with Theorem 2, Assumption 1 implies that $\Theta_{jk} = 0$ if and only if (j, k) is not an edge in $\mathcal{M}(G)$. (Since $\Theta \succ 0$, the diagonal entries of Θ are always strictly positive.) Note that when the nonzero entries of B are independently sampled continuous random variables, Assumption 1 holds for all choices of B except on a set of Lebesgue measure zero.

Remark 3 *Assumption 1 is a type of* faithfulness *assumption (Koller and Friedman, 2009; Spirtes et al., 2000). By selecting different topological orders π, one may then derive the familiar result that $X_j \perp\!\!\!\perp X_k \mid X_{\setminus\{j,k\}}$ if and only if $\Theta_{jk} = 0$, in the Gaussian setting. Note that this conditional independence assertion may not always hold for linear SEMs, however, since non-Gaussian distributions are not necessarily expressible as a linear SEM with respect to an arbitrary permutation order. Indeed, we only require Assumption 1 to hold with respect to a single (fixed) order.*

4. Score Functions for DAGs

Having established a method for reducing the search space of DAGs based on estimating the moralized graph, we now move to the more general problem of scoring candidate DAGs. As before, we assume the setting of a linear SEM.

Parametric maximum likelihood is often used as a score function for statistical estimation of DAG structure, since the likelihood enjoys the nice property that the population-level version is maximized only under a correct parameterization of the model class. This follows from the relationship between maximum likelihood and KL divergence:

$$\arg\max_\theta \mathrm{E}_{\theta_0}[\log p_\theta(X)] = \arg\min_\theta \mathrm{E}_{\theta_0}\left[\log\left(\frac{p_{\theta_0}(X)}{p_\theta(X)}\right)\right] = \arg\min_\theta D_{KL}(p_{\theta_0} \| p_\theta),$$

and the latter quantity is minimized exactly when $p_{\theta_0} \equiv p_\theta$, almost everywhere. If the model is identifiable, this happens if and only if $\theta = \theta_0$.

However, such maximum likelihood methods presuppose a fixed parameterization for the model class. In the case of linear SEMs, this translates into an appropriate parameterization of the error vector ϵ. For comparison, note that minimizing the squared ℓ_2-error for ordinary linear regression may be viewed as a maximum likelihood approach when errors are Gaussian, but the ℓ_2-minimizer is still statistically consistent for estimation of the regression vector when errors are *not* Gaussian. When our goal is recovery of the autoregression matrix B of the DAG, it is therefore natural to ask whether squared ℓ_2-error could be used in place of maximum likelihood as an appropriate metric for evaluating DAGs.

We will show that in settings where the noise variances $\{\sigma_j\}_{j=1}^p$ are specified up to a constant (e.g., homoscedastic error), the answer is affirmative. In such cases, the true DAG uniquely minimizes the ℓ_2-loss. As a side result, we will also show that the true linear SEM is identifiable.

Remark 4 *Nowzohour and Bühlmann (2014) study the use of nonparametric maximum likelihood methods for scoring candidate DAGs. We remark that such methods could also be combined with the framework of Sections 3 and 6 to select the optimal DAG for linear SEMs with nonparametric error distributions: First, estimate the moralized graph via the inverse covariance matrix, and then find the DAG with minimal score using a method such as dynamic programming. Similar statistical guarantees would hold in that case, with parametric rates replaced by nonparametric rates. However, our results in this section imply that in settings where the error variances are known or may be estimated accurately, the much simpler method of squared ℓ_2-loss may be used in place of a more complicated nonparametric approach.*

4.1 Weighted Squared ℓ_2-Loss

Suppose X is drawn from a linear SEM (3), where we now use B_0 to denote the true autoregression matrix and Ω_0 to denote the true error covariance matrix. For a fixed diagonal matrix $\Omega = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_p^2)$ and a candidate matrix B with columns $\{b_j\}_{j=1}^p$, define the *score of B with respect to Ω* according to

$$\mathrm{score}_\Omega(B) = \mathrm{E}\left[\|\Omega^{-1/2}(I-B)^T X\|_2^2\right] = \sum_{j=1}^p \frac{1}{\sigma_j^2} \cdot \mathrm{E}[(X_j - b_j^T X)^2]. \tag{7}$$

This is a weighted squared ℓ_2-loss, where the prediction error for the j^{th} coordinate is weighted by the diagonal entry σ_j^2 coming from Ω, and expectations are taken with respect to the true distribution on X.

It is instructive to compare the score function (7) to the usual parametric maximum likelihood when $X \sim N(0, \Sigma)$. For a distribution parameterized by the pair (B, Ω), the inverse covariance matrix is $\Theta = (I-B)\Omega^{-1}(I-B)^T$, using Equation (4), so the expected log likelihood is

$$\begin{aligned}
\mathrm{E}_{X\sim N(0,\Sigma)}[\log p_{B,\Omega}(X)] &= -\operatorname{tr}[(I-B)\Omega^{-1}(I-B)^T\Sigma] + \log\det[(I-B)\Omega^{-1}(I-B)^T] \\
&= -\operatorname{tr}[(I-B)\Omega^{-1}(I-B)^T\Sigma] + \log\det(\Omega^{-1}) \\
&= -\,\mathrm{score}_\Omega(B) + \log\det(\Omega^{-1}).
\end{aligned}$$

Hence, minimizing the score over B for a fixed Ω is identical to maximizing the likelihood. For non-Gaussians, however, the convenient relationship between minimum score and maximum likelihood no longer holds.

Now let $\mathcal{D}$ denote the class of DAGs. For $G \in \mathcal{D}$, define the score of G to be

$$\mathrm{score}_\Omega(G) := \min_{B \in \mathcal{U}_G} \{\mathrm{score}_\Omega(B)\}, \tag{8}$$

where

$$\mathcal{U}_G := \{B \in \mathbb{R}^{p\times p} : B_{jk} = 0 \text{ when } (j,k) \notin E(G)\}$$

is the set of matrices that are consistent with the structure of G.

Remark 5 *Examining the form of the score function* (7)*, we see that if* $\{\mathrm{Pa}_G(j)\}_{j=1}^p$ *denotes the parent sets of nodes in* G*, then the matrix*

$$B_G := \arg\min_{B \in \mathcal{U}_G} \{\mathrm{score}_\Omega(B)\}$$

is unique, and the columns of B_G *are equal to the coefficients of the best linear predictor of* X_j *regressed upon* $X_{\mathrm{Pa}_G(j)}$*. Furthermore, the value of* B_G *does not depend on* Ω*, since the minimizing value of* b_j *in the argument of Equation* (7) *is unaffected by the weighting factor* σ_j^2*.*

The following lemma relates the score of the underlying DAG G_0 to the score of the true autoregression matrix B_0. In fact, the score of any DAG containing G_0 has the same score. The proof is contained in Appendix C.2.

Lemma 6 *Suppose* X *follows a linear SEM with autoregression matrix* B_0*, and let* G_0 *denote the underlying DAG. Consider any* $G \in \mathcal{D}$ *such that* $G_0 \subseteq G$*. Then for any diagonal weight matrix* Ω*, we have*

$$\mathrm{score}_\Omega(G) = \mathrm{score}_\Omega(B_0),$$

and B_0 *is the unique minimizer of* $\mathrm{score}_\Omega(B)$ *over* $\mathcal{U}_G$*.*

We now turn to the main theorem of this section, in which we consider the problem of minimizing $\mathrm{score}_\Omega(B)$ with respect to all matrices B that are permutation similar to upper triangular matrices. Such a result is needed to validate our choice of score function, since when the DAG structure is not known a priori, the space of possible autoregression matrices must include all $\mathcal{U} := \bigcup_{G \in \mathcal{D}} \mathcal{U}_G$. Note that $\mathcal{U}$ may be equivalently defined as the set of all matrices that are permutation similar to upper triangular matrices. We have the following vital result:

Theorem 7 *Given a linear SEM* (3) *with error covariance matrix* $\alpha\Omega_0$ *and autoregression matrix* B_0*, where* $\alpha > 0$*, we have*

$$\mathrm{score}_{\Omega_0}(B) \geq \mathrm{score}_{\Omega_0}(B_0) = \alpha p, \qquad \forall B \in \mathcal{U}, \tag{9}$$

with equality if and only if $B = B_0$*.*

The proof of Theorem 7, which is based on matrix algebra, is contained in Section 4.5. In particular, Theorem 7 implies that the squared ℓ_2-loss function (7) is indeed an appropriate measure of model fit when the components are correctly weighted by the diagonal entries of Ω_0.

Note, however, that Theorem 7 requires the score to be taken with respect to (a multiple of) the true error covariance matrix Ω_0. The following example gives a cautionary message that if the weights Ω are chosen incorrectly, minimizing $\mathrm{score}_\Omega(B)$ may produce a structure that is *inconsistent* with the true model:

Example 1 *Suppose* (X_1, X_2) *is distributed according to the following linear SEM:*

$$X_1 = \epsilon_1, \quad \textit{and} \quad X_2 = -\frac{X_1}{2} + \epsilon_2,$$

so the autoregression matrix is given by $B_0 = \begin{pmatrix} 0 & -\frac{1}{2} \\ 0 & 0 \end{pmatrix}$. *Let* $\Omega_0 = \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{4} \end{pmatrix}$. *Consider* $B_1 = \begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix}$. *Then*

$$\text{score}_I(B_1) < \text{score}_I(B_0), \tag{10}$$

so using squared ℓ_2*-loss weighted by the identity will select an inappropriate model.*

Proof To verify Equation (10), we first compute

$$\Sigma = (I - B_0)^{-T}\Omega_0(I - B_0) = \begin{pmatrix} 1 & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

Then

$$\mathrm{E}[\|X - B_1^T X\|_2^2] = \mathrm{tr}\left[(I - B_1)^T\Sigma(I - B_1)\right] = \mathrm{tr}\left[\begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}\right] = 1,$$
$$\mathrm{E}[\|X - B_0^T X\|_2^2] = \mathrm{tr}\left[(I - B_0)^T\Sigma(I - B_0)\right] = \mathrm{tr}\left[\begin{pmatrix} 1 & 0 \\ 0 & 1/4 \end{pmatrix}\right] = \frac{5}{4},$$

implying Inequality (10). ■

4.2 Identifiability of Linear SEMs

Theorem 7 also has a useful consequence in terms of identifiability of a linear SEM, which we state in the following corollary:

Corollary 8 *Consider a fixed diagonal covariance matrix* Ω_0*, and consider the class of linear SEMs parameterized by the pair* $(B, \alpha\Omega_0)$*, where* $B \in \mathcal{U}$ *and* $\alpha > 0$ *is a scale factor. Then the true model* $(B_0, \alpha_0\Omega_0)$ *is identifiable. In particular, the class of homoscedastic linear SEMs is identifiable.*

Proof By Theorem 7, the matrix B_0 is the unique minimizer of $\text{score}_{\Omega_0}(B)$. Since $\alpha_0 \cdot (\Omega_0)_{11} = \mathrm{var}[X_1]$, the scale factor α_0 is also uniquely identifiable. The statement about homoscedasticity follows by taking $\Omega_0 = I$. ■

Corollary 8 should be viewed in comparison to previous results in the literature regarding identifiability of linear SEMs. Theorem 1 of Peters and Bühlmann (2013) states that when X is Gaussian and ϵ is an i.i.d. Gaussian vector with $\mathrm{cov}[\epsilon] = \alpha\Omega_0$, the model is identifiable. Indeed, our Corollary 8 implies that result as a special case, but it does not impose any additional conditions concerning Gaussianity. Shimizu et al. (2006) establish identifiability

of a linear SEM when ϵ is a vector of independent, non-Gaussian errors, by reducing to ICA, but our result does not require errors to be non-Gaussian.

The significance of Corollary 8 is that it supplies an elegant proof showing that the model is still identifiable even in the presence of both Gaussian and non-Gaussian components, provided the error variances are specified up to a scalar multiple. Since any multivariate Gaussian distribution may be written as a linear SEM with respect to an arbitrary ordering, some constraint such as variance scaling or non-Gaussianity is necessary in order to guarantee identifiability.

4.3 Misspecification of Variances

Theorem 7 implies that when the diagonal variances of Ω_0 are known up to a scalar factor, the weighted ℓ_2-loss (7) may be used as a score function for linear SEMs. Example 1 shows that when Ω is misspecified, we may have $B_0 \notin \arg\min_{B \in \mathcal{U}} \{\text{score}_\Omega(B)\}$. In this section, we further study the effect when Ω is misspecified. Intuitively, provided Ω is close enough to Ω_0 (or a multiple thereof), minimizing $\text{score}_\Omega(B)$ with respect to B should still yield the correct B_0.

Consider an arbitrary diagonal weight matrix Ω_1. We first provide bounds on the ratio between entries of Ω_0 and Ω_1 which ensure that $B_0 = \arg\min_{B \in \mathcal{U}} \{\text{score}_{\Omega_1}(B)\}$, even though the model is misspecified. Let

$$a_{\max} := \lambda_{\max}(\Omega_0 \Omega_1^{-1}) \quad \text{and} \quad a_{\min} := \lambda_{\min}(\Omega_0 \Omega_1^{-1})$$

denote the maximum and minimum ratios between corresponding diagonal entries of Ω_1 and Ω_0. Now define the additive gap between the score of G_0 and the next best DAG, given by

$$\xi := \min_{G \in \mathcal{D},\, G \not\supseteq G_0} \{\text{score}_{\Omega_0}(G) - \text{score}_{\Omega_0}(G_0)\} = \min_{G \in \mathcal{D}, G \not\supseteq G_0} \{\text{score}_{\Omega_0}(G)\} - p. \tag{11}$$

By Theorem 7, we know that $\xi > 0$. The following theorem provides a sufficient condition for correct model selection in terms of the gap ξ and the ratio $\frac{a_{\max}}{a_{\min}}$, which are both invariant to the scale factor α. It is a measure of robustness for how roughly the entries of Ω_0 may be approximated and still produce B_0 as the unique minimizer. The proof of the theorem is contained in Appendix C.3.

Theorem 9 *Suppose*

$$\frac{a_{\max}}{a_{\min}} \leq 1 + \frac{\xi}{p}. \tag{12}$$

Then $B_0 \in \arg\min_{B \in \mathcal{U}} \{\text{score}_{\Omega_1}(B)\}$. *If Inequality* (12) *is strict, then* B_0 *is the unique minimizer of* $\text{score}_{\Omega_1}(B)$.

Remark 10 *Theorem 9 provides an error allowance concerning the accuracy to which we may specify the error covariances and still recover the correct autoregression matrix* B_0 *from an improperly weighted score function. In the case* $\Omega_1 = \alpha\Omega_0$, *we have* $a_{\max} = a_{\min} = 1$, *so the condition* (12) *is always strictly satisfied, which is consistent with our earlier result in Theorem 7 that* $B_0 = \arg\min_{B \in \mathcal{U}} \{\text{score}_{\alpha\Omega_0}(B)\}$.

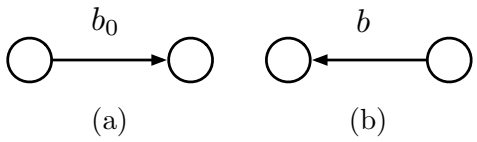


Figure 1: Two-variable DAG. (a) The forward model. (b) The backward model.

Naturally, the error tolerance specified by Theorem 9 is a function of the gap ξ between the true DAG G_0 and the next best candidate: If ξ is larger, the score is more robust to misspecification of the weights Ω. Note that if we restrict our search space from the full set of DAGs $\mathcal{D}$ to some smaller space $\mathcal{D}'$, so $B \in \bigcup_{G \in \mathcal{D}'} \mathcal{U}_G$, we may restate the condition in Theorem 9 in terms of the gap

$$\xi(\mathcal{D}') := \min_{G \in \mathcal{D}',\, G \not\supseteq G_0} \{\text{score}_{\Omega_0}(G) - \text{score}_{\Omega_0}(G_0)\}, \tag{13}$$

which may be considerably larger than ξ when $\mathcal{D}'$ is much smaller than $\mathcal{D}$. See Equation (22) and Section 5.2 on weakening the gap condition below.

Specializing to the case when $\Omega_1 = I$, we may interpret Theorem 9 as providing a window of variances around which we may treat a heteroscedastic model as homoscedastic, and use the simple (unweighted) squared ℓ_2-score to recover the correct model. See Lemma 11 in the next section for a concrete example.

4.4 Example: 2- and 3-Variable Models

In this section, we develop examples illustrating the gap ξ introduced in Section 4.3. We study two cases, involving two and three variables, respectively.

4.4.1 Two Variables

We first consider the simplest example with a two-variable directed graph. Suppose the forward model is defined by

$$B_0 = \begin{pmatrix} 0 & b_0 \\ 0 & 0 \end{pmatrix}, \qquad \Omega_0 = \begin{pmatrix} d_1^2 & 0 \\ 0 & d_2^2 \end{pmatrix},$$

and consider the backward matrix defined by the autoregression matrix

$$B = \begin{pmatrix} 0 & 0 \\ b & 0 \end{pmatrix}. \tag{14}$$

The forward and backward models are illustrated in Figure 1.

A straightforward calculation shows that

$$\text{score}_{\Omega_0}(B) = 2 + b^2 b_0^2 + \left(b \frac{d_2}{d_1} - b_0 \frac{d_1}{d_2} \right)^2,$$

which is minimized for $b = \frac{b_0}{b_0^2 + \frac{d_2^2}{d_1^2}}$, implying that

$$\xi = \min_b \{\text{score}_{\Omega_0}(B) - \text{score}_{\Omega_0}(B_0)\} = \frac{b_0^4}{\frac{d_2^4}{d_1^4} + b_0^2 \frac{d_2^2}{d_1^2}}. \tag{15}$$

We see that the gap ξ grows with the strength of the true edge b_0, when $|b_0| > 1$, and is symmetric with respect to the sign of b_0. The gap also grows with the magnitude of the ratio $\frac{d_1}{d_2}$.

To gain intuition for the interplay between b_0 and $\frac{d_1}{d_2}$, we derive the following lemma, a corollary of Theorem 9 specialized to the case of the two-variable DAG:

Lemma 11 *Consider the two-variable DAG defined by Equation* (14)*. Let* $\Omega_1 = I_2$ *and define* $r := \frac{d_2}{d_1}$*. Suppose the following conditions hold:*

$$b_0^2 \geq \begin{cases} r^2 \left((r^2 - 1) + \sqrt{r^4 - 1} \right), & \text{if } r \geq 1, \\ (1 - r^2) + \sqrt{1 - r^4}, & \text{if } r \leq 1. \end{cases} \tag{16}$$

Then $B_0 = \arg\min_{B \in \mathcal{U}} \{\text{score}_{\Omega_1}(B)\}$*; i.e.,* B_0 *is the unique minimizer of the score function under the unweighted squared-*ℓ_2 *loss.*

Lemma 11 is proved in Appendix C.4.

Remark 12 *Note that the two right-hand expressions in Inequality* (16) *are similar, although the expression in the case* $r^2 \geq 1$ *contains an extra factor of* r^2*, so the sufficient condition is stronger. Both lower bounds in Equation* (16) *increase with* $|r-1|$*, which agrees with intuition: If the true model is more non-homoscedastic, the strength of the true edge must be stronger in order for the unweighted squared-*ℓ_2 *score to correctly identify the model. When* $r = 1$*, we have the vacuous condition* $b_0^2 \geq 0$*, since* $\Omega_1 = \alpha\Omega_0$ *and the variances are correctly specified, so Theorem 7 implies* $B_0 = \arg\min_{B \in \mathcal{U}} \{\text{score}_{\Omega_1}(B)\}$ *for any choice of* b_0*.*

4.4.2 v-Structure

We now examine a three-variable graph. Suppose the actual graph involves a v-structure, as depicted in Figure 2, and is parameterized by the matrices

$$B_0 = \begin{pmatrix} 0 & 0 & b_{13} \\ 0 & 0 & b_{23} \\ 0 & 0 & 0 \end{pmatrix}, \qquad \Omega = \begin{pmatrix} d_1^2 & 0 & 0 \\ 0 & d_2^2 & 0 \\ 0 & 0 & d_3^2 \end{pmatrix}. \tag{17}$$

We have the following lemma, proved in Appendix C.5:

Lemma 13 *Consider the three-variable DAG characterized by Figure 2 and Equations* (17)*. The gap* ξ *defined by Equation* (11) *is given by*

$$\xi = \min \left\{ \frac{b_{23}^4}{\frac{d_3^4}{d_2^4} + b_{23}^2 \frac{d_3^2}{d_2^2}}, \ \frac{b_{13}^4}{\frac{d_3^4}{d_1^4} + b_{13}^2 \frac{d_3^2}{d_1^2}} \right\}.$$

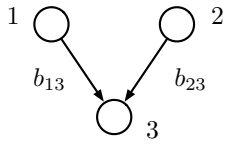


Figure 2: Three-variable DAG with v-structure.

A key reduction in the proof of Lemma 13 is to note that we only need to consider a relatively small number of DAGs, given by Figure 3 in Appendix C.5. Indeed, for $G_1 \subseteq G_2$, we have $\text{score}_{\Omega_0}(G_2) \leq \text{score}_{\Omega_0}(G_1)$, so it suffices to consider maximal elements in the poset of DAGs not containing the true DAG G_0.

Remark 14 *Note that the form of the gap in Lemma 13 is very similar to the form for the two-variable model, and the individual ratios scale with the strength of the edge and the ratio of the corresponding error variances. Indeed, we could derive a version of Lemma 11 for the three-variable model, giving lower bounds on the edge strengths b_{23}^2 and b_{13}^2 that guarantee the accuracy of the unweighted squared ℓ_2-loss; however, the conditions would be more complicated. It is interesting to note from our calculations in Appendix C.5 that the gap between models accumulates according to the number of edge reversals from the misspecified model: Reversing the directions of edges $(2,3)$ and $(1,3)$ in succession leads to an additional term in the expressions for ξ_1 and ξ_2 in Equations (43) and (44) below. We will revisit these observations in Section 5.2, where we define a version of the gap function rescaled by the number of nodes that differ in their parents sets.*

4.5 Proof of Theorem 7

First note that for a constant $\alpha > 0$, we have

$$\text{score}_{\alpha\Omega}(B) = \frac{1}{\alpha} \cdot \text{score}_{\Omega}(B),$$

so minimizing $\text{score}_{\alpha\Omega}(B)$ is equivalent to minimizing $\text{score}_{\Omega}(B)$. Furthermore, it suffices to prove the statement for $\alpha = 1$; the statement for general $\alpha > 0$ follows by a simple rescaling.

Recalling Equation (4), we may write

$$\begin{aligned}
\text{score}_{\Omega_0}(B) &= \mathrm{E}_{B_0}[\|\Omega_0^{-1/2}(I-B)^T X\|_2^2] \\
&= \text{tr}\left[\Omega_0^{-1/2}(I-B)^T \cdot \text{cov}_{B_0}[X] \cdot (I-B)\Omega_0^{-1/2}\right] \\
&= \text{tr}\left[\Omega_0^{-1/2}(I-B)^T \cdot (I-B_0)^{-T}\Omega_0(I-B_0)^{-1} \cdot (I-B)\Omega_0^{-1/2}\right].
\end{aligned}$$

Now note that

$$\begin{aligned}
(I-B)\Omega_0^{-1/2} &= \Omega_0^{-1/2}(I - \Omega_0^{1/2}B\Omega_0^{-1/2}) := \Omega_0^{-1/2}(I-\widetilde{B}), \\
\Omega_0^{1/2}(I-B_0)^{-1} &= (I - \Omega_0^{1/2}B_0\Omega_0^{-1/2})^{-1}\Omega_0^{1/2} := (I-\widetilde{B}_0)^{-1}\Omega_0^{1/2},
\end{aligned}$$

where $\widetilde{B}, \widetilde{B}_0 \in \mathcal{U}$. Hence, we may rewrite

$$\begin{aligned}\text{score}_{\Omega_0}(B) &= \text{tr}\left[(I-\widetilde{B})^T\Omega_0^{-1/2}\cdot\Omega_0^{1/2}(I-\widetilde{B}_0)^{-T}\cdot(I-\widetilde{B}_0)^{-1}\Omega_0^{1/2}\cdot\Omega_0^{-1/2}(I-\widetilde{B})\right]\\ &= \text{tr}\left[(I-\widetilde{B})^T(I-\widetilde{B}_0)^{-T}(I-\widetilde{B}_0)^{-1}(I-\widetilde{B})\right]\\ &= \text{tr}\left[(I-\widetilde{B})(I-\widetilde{B})^T(I-\widetilde{B}_0)^{-T}(I-\widetilde{B}_0)^{-1}\right].\end{aligned}$$

Since $\widetilde{B}, \widetilde{B}_0 \in \mathcal{U}$, the matrices $I-\widetilde{B}$ and $I-\widetilde{B}_0$ are both permutation similar to lower triangular matrices with 1's on the diagonal. Hence, Lemma 27 in Appendix B implies

$$\text{score}_{\Omega_0}(B) \geq p,$$

with equality if and only if

$$I-\widetilde{B} = I-\widetilde{B}_0,$$

or equivalently, $B = B_0$, as claimed.

5. Consequences for Statistical Estimation

The population-level results in Theorems 2 and 7 provide a natural avenue for estimating the DAG of a linear SEM from data. In this section, we outline how the true DAG may be estimated in the presence of fully-observed or systematically corrupted data. Our method is applicable also in the high-dimensional setting, assuming the moralized DAG is sufficiently sparse.

Our inference algorithm consists of two main components:

1. Estimate the moralized DAG $\mathcal{M}(G_0)$ using the inverse covariance matrix of X.
2. Search through the space of DAGs consistent with $\mathcal{M}(G_0)$, and find the DAG that minimizes $\text{score}_\Omega(B)$.

Theorem 2 and Assumption 1 ensure that for almost every choice of autoregression matrix B_0, the support of the true inverse covariance matrix Θ_0 exactly corresponds to the edge set of the moralized graph. Theorem 7 ensures that when the weight matrix Ω is chosen appropriately, B_0 will be the unique minimizer of $\text{score}_\Omega(B)$.

5.1 Fully-Observed Data

We now present concrete statistical guarantees for the correctness of our algorithm in the usual setting when $\{x_i\}_{i=1}^n$ are fully-observed and i.i.d. Recall that a random variable X is *sub-Gaussian* with parameter σ^2 if

$$\text{E}[\exp(\lambda(X-\text{E}[X]))] \leq \exp\left(\frac{\sigma^2\lambda^2}{2}\right), \qquad \forall\lambda\in\mathbb{R}.$$

If $X \in \mathbb{R}^p$ is a random vector, it is sub-Gaussian with parameter σ^2 if v^TX is a sub-Gaussian random variable with parameter σ^2 for all unit vectors $v \in \mathbb{R}^p$.

5.1.1 Estimating the Inverse Covariance Matrix

We first consider the problem of inferring Θ_0. Let

$$\Theta_0^{\min} := \min_{j,k} \left\{ |(\Theta_0)_{jk}| : (\Theta_0)_{jk} \neq 0 \right\}$$

denote the magnitude of the minimum nonzero element of Θ_0. We consider the following two scenarios:

Low-dimensional setting. If $n \geq p$, the sample covariance matrix $\widehat{\Sigma} = \frac{1}{n} \sum_{i=1}^n x_i x_i^T$ is invertible with high probability, and we use the estimator

$$\widehat{\Theta} = (\widehat{\Sigma})^{-1}.$$

We have the following lemma, which follows from standard bounds in random matrix theory:

Lemma 15 *Suppose the x_i's are i.i.d. sub-Gaussian vectors with parameter σ^2. With probability at least $1 - c_1 \exp(-c_2 p)$, we have*

$$\|\widehat{\Theta} - \Theta_0\|_{\max} \leq c_0 \sigma^2 \sqrt{\frac{p}{n}},$$

and thresholding $\widehat{\Theta}$ at level $\tau = c_0 \sigma^2 \sqrt{\frac{p}{n}}$ succeeds in recovering $\operatorname{supp}(\Theta_0)$*, if $\Theta_0^{\min} > 2\tau$.*

For the proof, see Appendix D.1. Here, we use to $\|\cdot\|_{\max}$ denote the elementwise ℓ_∞-norm of a matrix.

High-dimensional setting. If $p > n$, we assume each row of the true inverse covariance matrix Θ_0 is d-sparse. Then we use the graphical Lasso:

$$\widehat{\Theta} \in \arg\min_{\Theta \succeq 0} \left\{ \operatorname{tr}(\Theta \widehat{\Sigma}) - \log\det(\Theta) + \lambda \sum_{j \neq k} |\Theta_{jk}| \right\}. \tag{18}$$

Standard results (Ravikumar et al., 2011) establish the statistical consistency of the graphical Lasso (18) as an estimator for the inverse covariance matrix in the setting of sub-Gaussian observations; consequently, we omit the proof of the following lemma.

Lemma 16 *Suppose the x_i's are i.i.d. sub-Gaussian vectors with parameter σ^2. Suppose the sample size satisfies $n \geq Cd \log p$. With probability at least $1 - c_1 \exp(-c_2 \log p)$, we have*

$$\|\widehat{\Theta} - \Theta_0\|_{\max} \leq c_0 \sigma^2 \sqrt{\frac{\log p}{n}},$$

and thresholding $\widehat{\Theta}$ at level $\tau = c_0 \sigma^2 \sqrt{\frac{\log p}{n}}$ succeeds in recovering $\operatorname{supp}(\Theta_0)$*, if $\Theta_0^{\min} > 2\tau$.*

Alternatively, we may perform nodewise regression with the ordinary Lasso (Meinshausen and Bühlmann, 2006) to recover the support of Θ_0, with similar rates for statistical consistency.

5.1.2 Scoring Candidate DAGs

Moving on to the second step of the algorithm, we need to estimate the score functions $\text{score}_\Omega(B)$ of candidate DAGs and choose the minimally scoring candidate. In this section, we focus on methods for estimating an empirical version of the score function and derive rates for statistical estimation under certain models. If the space of candidate DAGs is sufficiently small, we may evaluate the empirical score function for every candidate DAG and select the optimum. In Section 6, we describe computationally efficient procedures based on dynamic programming to choose the optimal DAG when the candidate space is too large for naive search.

The input of our algorithm is the sparsely estimated inverse covariance matrix $\widehat{\Theta}$ from Section 5.1.1. For a matrix Θ, define the candidate neighborhood sets

$$N_\Theta(j) := \{k : k \neq j \text{ and } \Theta_{jk} \neq 0\}, \qquad \forall j,$$

and let

$$\mathcal{D}_\Theta := \{G \in \mathcal{D} : \text{Pa}_G(j) \subseteq N_\Theta(j), \quad \forall j\}$$

denote the set of DAGs with skeleton contained in the graph defined by $\text{supp}(\Theta)$. By Theorem 2 and Assumption 1, we have $G_0 \in \mathcal{D}_{\Theta_0}$, so if $\text{supp}(\widehat{\Theta}) \supseteq \text{supp}(\Theta_0)$, which occurs with high probability under the conditions of Section 5.1.1, it suffices to search over the reduced DAG space $\mathcal{D}_{\widehat{\Theta}}$.

Remark 17 *In fact, we could reduce the search space even further to only include DAGs with moralized graph equal to the undirected graph defined by* $\text{supp}(\Theta)$*. The dynamic programming algorithm to be described in Section 6 only requires as input a superset of the skeleton; for alternative versions of the dynamic programming algorithm taking as input a superset of the moralized graph, we would indeed restrict* $\mathcal{D}_\Theta$ *to DAGs with the correct moral structure.*

We now consider an arbitrary d-sparse matrix Θ, with $d \leq n$, and take $G \in \mathcal{D}_\Theta$. By Remark 5, we have

$$\text{score}_\Omega(G) = \sum_{j=1}^{p} f_{\sigma_j}(\text{Pa}_G(j)), \tag{19}$$

where

$$f_{\sigma_j}(S) := \frac{1}{\sigma_j^2} \cdot \text{E}[(x_j - b_j^T x_S)^2],$$

and $b_j^T x_S$ is the best linear predictor for x_j regressed upon x_S. In order to estimate $\text{score}_\Omega(G)$, we use the empirical functions

$$\widehat{f}_{\sigma_j}(S) := \frac{1}{\sigma_j^2} \cdot \frac{1}{n} \sum_{i=1}^{n} (x_{ij} - x_{i,S}^T \widehat{b}_j)^2 = \frac{1}{\sigma_j^2} \cdot \frac{1}{n} \|X_j - X_S \widehat{b}_j\|_2^2, \tag{20}$$

where

$$\widehat{b}_j := (X_S^T X_S)^{-1} X_S^T X_j$$

is the usual ordinary least squares solution for linear regression of X_j upon X_S. We will take $S \subseteq N_\Theta(j)$, so since $|N_\Theta(j)| \leq d \leq n$, the matrix $X_S^T X_S$ is invertible with high probability. The following lemma, proved in Appendix D.2, provides rates of convergence for the empirical score function:

Lemma 18 *Suppose the x_i's are i.i.d. sub-Gaussian vectors with parameter σ^2. Suppose $d \leq n$ is a parameter such that $|N_\Theta(j)| \leq d$ for all j. Then*

$$|\widehat{f}_{\sigma_j}(S) - f_{\sigma_j}(S)| \leq \frac{c_0 \sigma^2}{\sigma_j^2} \sqrt{\frac{\log p}{n}}, \qquad \forall j \text{ and } S \subseteq N_\Theta(j), \tag{21}$$

with probability at least $1 - c_1 \exp(-c_2 \log p)$.

In particular, we have the following result, proved in Appendix D.3, which provides a sufficient condition for the empirical score functions to succeed in selecting the true DAG. Here,

$$\xi_\Omega(\mathcal{D}_\Theta) := \min_{G \in \mathcal{D}_\Theta, G \not\supseteq G_0} \{\text{score}_\Omega(G) - \text{score}_\Omega(G_0)\} \tag{22}$$

is the gap between G_0 and the next best DAG in $\mathcal{D}_\Theta$. Note that the expression in Equation (22) is reminiscent of Equation (13) in Section 4.3, but we now allow Ω to be arbitrary.

Lemma 19 *Suppose Inequality* (21) *holds, and suppose*

$$c_0 \sigma^2 \sqrt{\frac{\log p}{n}} \cdot \sum_{j=1}^{p} \frac{1}{\sigma_j^2} < \frac{\xi_\Omega(\mathcal{D}_\Theta)}{2}. \tag{23}$$

Then

$$\widehat{\text{score}}_\Omega(G_0) < \widehat{\text{score}}_\Omega(G), \qquad \forall G \in \mathcal{D}_\Theta : G \not\supseteq G_0. \tag{24}$$

Remark 20 *Lemma 19 does not explicitly assume that Ω is equal to Ω_0, the true matrix of error variances. However, Inequality* (23) *can only be satisfied when $\xi_\Omega(\mathcal{D}_G) > 0$; hence, Ω should be chosen such that $G_0 = \arg\min_{G \in \mathcal{D}_\Theta, G \not\supseteq G_0} \{\text{score}_\Omega(G)\}$. As discussed in Section 4.3, this condition holds for a wider range of choices for Ω.*

Note that the conclusion (24) in Lemma 19 is not quite the same as the condition

$$G_0 = \arg \min_{G \in \mathcal{D}_\Theta, G \not\supseteq G_0} \left\{\widehat{\text{score}}_\Omega(G)\right\}, \tag{25}$$

which is what we would need for exact recovery of our score-minimizing algorithm. The issue is that $\text{score}_\Omega(G)$ is equal for all $G \supseteq G_0$; however the empirical scores $\widehat{\text{score}}_\Omega(G)$ may differ among this class, so Equation (25) may not be satisfied. However, it is easily seen from the proof of Lemma 19 that in fact,

$$\arg \min_{G \in \mathcal{D}_\Theta} \left\{\widehat{\text{score}}_\Omega(G)\right\} \subseteq \{G \in \mathcal{D}_\Theta : G \supseteq G_0\}. \tag{26}$$

By applying a thresholding procedure to the empirical score minimizer $\widehat{G} \supseteq G_0$ selected by our algorithm, we could then recover the true G_0. In other words, since $\text{Pa}_{G_0}(j) \subseteq \text{Pa}_{\widehat{G}}(j)$

for each j, we could use standard sparse regression techniques to recover the parent set of each node in the true DAG.

To gain some intuition for the condition (23), consider the case when $\sigma_j^2 = 1$ for all j. Then the condition becomes

$$c_0\sigma^2\sqrt{\frac{\log p}{n}} < \frac{\xi(\mathcal{D}_\Theta)}{2p}. \tag{27}$$

If $\xi(\mathcal{D}_\Theta) = \Omega(1)$, which might be expected based on our calculations in Section 4.4, we require $n \geq Cp^2 \log p$ in order to guarantee statistical consistency, which is not a truly high-dimensional result. On the other hand, if $\xi(\mathcal{D}_\Theta) = \Omega(p)$, as is assumed in similar work on score-based DAG learning (van de Geer and Bühlmann, 2013; Bühlmann et al., 2014), our method is consistent provided $\frac{\log p}{n} \to 0$. In Section 5.2, we relax the condition (27) to a slightly weaker condition that is more likely to hold in settings of interest.

5.2 Weakening the Gap Condition

Motivated by our comments from the previous section, we establish a sufficient condition for statistical consistency that is slightly weaker than the condition (23), which still guarantees that Equation (26) holds.

For two DAGs $G, G' \in \mathcal{D}$, define

$$H(G, G') := \{j : \mathrm{Pa}_G(j) \neq \mathrm{Pa}_{G'}(j)\}$$

to be the set of nodes on which the parent sets differ between graphs G and G', and define the ratio

$$\gamma_\Omega(G, G') := \frac{\mathrm{score}_\Omega(G) - \mathrm{score}_\Omega(G')}{|H(G, G')|},$$

a rescaled version of the gap between the score functions. Consider the following condition:

Assumption 2 *There exists $\xi' > 0$ such that*

$$\gamma_\Omega(G_0) := \min_{G \in \mathcal{D}_\Theta, G \not\supseteq G_0} \left\{ \max_{G_1 \supseteq G_0} \{\gamma_\Omega(G, G_1)\} \right\} \geq \xi'. \tag{28}$$

Note that in addition to minimizing over DAGs in the class $\mathcal{D}_\Theta$, the expression (28) defined in Assumption 2 takes an inner maximization over DAGs containing G_0. As established in Lemma 6, we have $\mathrm{score}_\Omega(G_1) = \mathrm{score}_\Omega(G_0)$ whenever $G_0 \subseteq G_1$. However, $|H(G, G_1)|$ may be appreciably different from $|H(G, G_0)|$, and we are only interested in computing the gap ratio between a DAG $G \not\supseteq G_0$ and the closest DAG containing G_0.

We then have the following result, proved in Appendix D.4:

Lemma 21 *Under Assumption 2, suppose*

$$|\widehat{f}_{\sigma_j}(S) - f_{\sigma_j}(S)| \leq \frac{\xi'}{2}, \qquad \forall j \text{ and } S \subseteq N_\Theta(j). \tag{29}$$

Then the containment (26) *holds.*

Combining with Lemma 18, we have the following corollary:

Corollary 22 *Suppose the x_i's are i.i.d. sub-Gaussian with parameter σ^2, and $|N_\Theta(j)| \leq d$ for all j. Also suppose Assumption 2 holds. Then with probability at least $1-c_1\exp(-c_2\log p)$, the condition* (26) *is satisfied.*

We now turn to the question of what values of ξ' might be expected to give condition (28) for various DAGs. Certainly, we have

$$\gamma_\Omega(G, G') \geq \frac{\text{score}_\Omega(G) - \text{score}_\Omega(G')}{p},$$

so the condition holds when

$$p \cdot \xi' < \xi(\mathcal{D}_\Theta).$$

However, for $\xi' = \mathcal{O}(\xi(\mathcal{D}_\Theta)/p)$, Corollary 22 yields a scaling condition similar to Inequality (27), which we wish to avoid. As motivated by our computations of the score functions for small DAGs (see Remark 14 in Section 4.4), the difference $\{\text{score}_\Omega(G) - \text{score}_\Omega(G_0)\}$ seems to increase linearly with the number of edge reversals needed to transform G_0 to G. Hence, we might expect $\gamma_\Omega(G, G_0)$ to remain roughly constant, rather than decreasing linearly with p. The following lemma verifies this intuition in a special case. For a review of junction tree terminology, see Appendix A.1.

Lemma 23 *Suppose the moralized graph $\mathcal{M}(G_0)$ admits a junction tree representation with only singleton separator sets. Let $C_1, \ldots, C_k$ denote the maximal cliques in $\mathcal{M}(G_0)$, and let $\{G_0^\ell\}_{\ell=1}^k$ denote the corresponding restrictions of G_0 to the cliques. Then*

$$\gamma_\Omega(G_0) \geq \min_{1 \leq \ell \leq k} \gamma_\Omega(G_0^\ell),$$

where

$$\gamma_\Omega(G_0^\ell) := \min_{G^\ell \in \mathcal{D}_\Theta|_{C_\ell}, G^\ell \not\supseteq G_0^\ell} \left\{ \max_{G_1^\ell \supseteq G_0^\ell} \left\{ \frac{\text{score}_\Omega(G^\ell) - \text{score}_\Omega(G_1^\ell)}{|H(G^\ell, G_1^\ell)|} \right\} \right\}$$

is the gap ratio computed over DAGs restricted to clique C_ℓ that are consistent with the moralized graph.

The proof is contained in Appendix D.5.

We might expect the gap ratio $\gamma_\Omega(G_0^\ell)$ to be a function of the size of the clique. In particular, if the treewidth of $\mathcal{M}(G_0)$ is bounded by w and we have $\gamma_\Omega(G_0^\ell) \geq \xi_w$ for all ℓ, Lemma 23 implies that

$$\gamma_\Omega(G_0) \geq \xi_w,$$

and we only need the parameter ξ' appearing in Assumption 2 to be larger than ξ_w, rather than scaling as the inverse of p. We expect a version of Lemma 23 to hold for graphs with bounded treewidth even when the separator sets have larger cardinality, but a full generalization of Lemma 23 and a more accurate characterization of $\gamma_\Omega(G_0)$ for arbitrary graphs is beyond the scope of this paper.

5.3 Systematically Corrupted Data

We now describe how our algorithm for DAG structure estimation in linear SEMs may be extended easily to accommodate systematically corrupted data. This refers to the setting where we observe noisy surrogates $\{z_i\}_{i=1}^n$ in place of $\{x_i\}_{i=1}^n$. Two common examples include the following:

(a) *Additive noise.* We have $z_i = x_i + w_i$, where $w_i \perp\!\!\!\perp x_i$ is additive noise with known covariance Σ_w.

(b) *Missing data.* This is one instance of the more general setting of multiplicative noise. For each $1 \le j \le p$, and independently over coordinates, we have

$$z_{ij} = \begin{cases} x_{ij}, & \text{with probability} \quad 1-\alpha, \\ \star, & \text{with probability} \quad \alpha, \end{cases}$$

where the missing data probability α is either estimated or known.

We again divide our discussion into two parts: estimating Θ_0 and computing score functions based on corrupted data.

5.3.1 Inverse Covariance Estimation

Following the observation of Loh and Wainwright (2013), the graphical Lasso (18) may still be used to estimate the inverse covariance matrix Θ_0 in the high-dimensional setting, where we plug in a suitable estimator $\widehat{\Gamma}$ for the covariance matrix $\Sigma = \mathrm{cov}[x_i]$, based on the corrupted observations $\{z_i\}_{i=1}^n$. For instance, in the additive noise scenario, we may take

$$\widehat{\Gamma} = \frac{Z^T Z}{n} - \Sigma_w, \tag{30}$$

and in the missing data setting, we may take

$$\widehat{\Gamma} = \frac{Z^T Z}{n} \odot M, \tag{31}$$

where $\odot$ denotes the Hadamard product and M is the matrix with diagonal entries equal to $\frac{1}{1-\alpha}$ and off-diagonal entries equal to $\frac{1}{(1-\alpha)^2}$.

Assuming conditions such as sub-Gaussianity, the output $\widehat{\Theta}$ of the modified graphical Lasso (18) is statistically consistent under similar scaling as in the uncorrupted setting (Loh and Wainwright, 2013). For instance, in the additive noise setting, where the z_i's are sub-Gaussian with parameter σ_z^2, Lemma 16 holds with σ^2 replaced by σ_z^2. Analogous results hold in the low-dimensional setting, when the expressions for $\widehat{\Gamma}$ in Equations (30) and (31) are invertible with high probability, and we may simply use $\widehat{\Theta} = (\widehat{\Gamma})^{-1}$.

5.3.2 Computing DAG Scores

We now describe how to estimate score functions for DAGs based on corrupted data. By Equation (19), this reduces to estimating

$$f_{\sigma_j}(S) = \frac{1}{\sigma_j^2} \cdot \mathrm{E}[(x_j - b_j^T x_S)^2],$$

for a subset $S \subseteq \{1, \ldots, p\} \backslash \{j\}$, with $|S| \leq n$. Note that

$$\sigma_j^2 \cdot f_{\sigma_j}(S) = \Sigma_{jj} - 2b_j^T \Sigma_{S,j} + b_j^T \Sigma_{SS} b_j = \Sigma_{jj} - \Sigma_{j,S} \Sigma_{SS}^{-1} \Sigma_{S,j},$$

since $b_j = \Sigma_{SS}^{-1} \Sigma_{S,j}$.

Let $\widehat{\Gamma}$ be the estimator for Σ based on corrupted data used in the graphical Lasso; e.g., Equations (30) and (31). We then use the estimator

$$\widetilde{f}_{\sigma_j}(S) = \frac{1}{\sigma_j^2} \cdot \left(\widehat{\Gamma}_{jj} - \widehat{\Gamma}_{j,S} \widehat{\Gamma}_{SS}^{-1} \widehat{\Gamma}_{S,j} \right). \tag{32}$$

Note in particular that Equation (32) reduces to the expression in Equation (20) in the fully-observed setting. We establish consistency of the estimator in Equation (32), under the following deviation condition on $\widehat{\Gamma}$:

$$\mathbb{P}\left(\left| \left| \left| \widehat{\Gamma}_{SS} - \Sigma_{SS} \right| \right| \right|_2 \geq \sigma^2 \left(\sqrt{\frac{d}{n}} + t \right) \right) \leq c_1 \exp(-c_2 n t^2), \quad \text{for any } |S| \leq d. \tag{33}$$

For instance, such a condition holds in the case of the sub-Gaussian additive noise model (cf. Lemma 29 in Appendix E), with $\widehat{\Gamma}$ given by Equation (30), where $\sigma^2 = \sigma_z^2$.

We have the following result, an extension of Lemma 18 applicable also for corrupted variables:

Lemma 24 *Suppose $\widehat{\Gamma}$ satisfies the deviation condition (33). Suppose $|N_\Theta(j)| \leq d$ for all j. Then*

$$|\widetilde{f}_{\sigma_j}(S) - f_{\sigma_j}(S)| \leq \frac{c_0 \sigma^2}{\sigma_j^2} \sqrt{\frac{\log p}{n}}, \qquad \forall j \text{ and } S \subseteq N_\Theta(j),$$

with probability at least $1 - c_1 \exp(-c_2 \log p)$.

The proof is contained in Appendix D.6. In particular, Corollary 22, providing guarantees for statistical consistency, also holds.

6. Computational Considerations

In practice, the main computational bottleneck in inferring the DAG structure comes from having to compute score functions over a large number of DAGs. The simplest approach of searching over all possible permutation orderings of indices gives rise to $p!$ candidate DAGs, which scales exponentially with p. In this section, we describe how the result of Theorem 2 provides a general framework for achieving vast computational savings for finding the best-scoring DAG when data are generated from a linear SEM. We begin by reviewing existing methods, and describe how our results may be used in conjunction with dynamic programming to produce accurate and efficient DAG learning.

6.1 Decomposable Score Functions

Following the literature, we call a score function over DAGs *decomposable* if it may be written as a sum of score functions over individual nodes, each of which is a function of

only the node and its parents:

$$\text{score}(G) = \sum_{j=1}^{p} \text{score}_j(\text{Pa}_G(j)).$$

Note that we allow the score functions to differ across nodes. Consistent with our earlier notation, the goal is to find the DAG $G \in \mathcal{D}$ that minimizes score(G).

Some common examples of decomposable scores that are used for DAG inference include maximum likelihood, BDe, BIC, and AIC (Chickering, 1995). By Equation (19), the squared ℓ_2-score is clearly decomposable, and it gives an example where score$_j$ differs over nodes. Another interesting example is the nonparametric maximum likelihood, which extends the ordinary likelihood method for scoring DAGs (Nowzohour and Bühlmann, 2014).

Various recent results have focused on methods for optimizing a decomposable score function over the space of candidate DAGs in an efficient manner. Some methods include exhaustive search (Silander and Myllymaki, 2006), greedy methods (Chickering, 2002), and dynamic programming (Ordyniak and Szeider, 2012; Korhonen and Parviainen, 2013). We will focus here on a dynamic programming method that takes as input an undirected graph and outputs the best-scoring DAG with skeleton contained in the input graph.

6.2 Dynamic Programming

In this section, we detail a method due to Ordyniak and Szeider (2012) that will be useful for our purposes. Given an input undirected graph G_I and a decomposable score function, the dynamic programming algorithm finds a DAG with minimal score that has skeleton contained in G_I. Let $\{N_I(j)\}_{j=1}^p$ denote the neighborhood sets of G_I. The runtime of the dynamic programming algorithm is exponential in the treewidth w of G_I; hence, the algorithm is only tractable for bounded-treewidth graphs. For further characterizations of graphs with bounded treewidth, including some relevant examples, see Bodlaender (1998).

The main steps of the dynamic programming algorithm are as follows. For a review of basic terminology of graph theory, including treewidth and tree decompositions, see Appendix A; for further details and a proof of correctness, see Ordyniak and Szeider (2012).

1. Construct a *tree decomposition* of G_I with minimal treewidth.

2. Construct a *nice tree decomposition* of the graph. Let $\chi(t)$ denote the subset of $\{1, \ldots, p\}$ associated to a node t in the nice tree decomposition.

3. Starting from the leaves of the nice tree decomposition up to the root, compute the *record* for each node t. The record $\mathcal{R}(t)$ is the set of tuples (a, p, s) corresponding to minimal-scoring DAGs defined on the vertices $\chi^*(t)$ in the subtree attached to t, with skeleton contained in G_I. For each such DAG, s is the score, a lists the parent sets of vertices in $\chi(t)$, such that $a(v) \subseteq N_I(v)$ for each $v \in \chi(t)$, and $a(v)$ restricted to $\chi^*(t)$ agrees with the partial DAG; and p lists the directed paths between vertices in $\chi(t)$. The records $\mathcal{R}(t)$ may computed recursively over the nice tree decomposition as follows:

   - **Join node:** Suppose t has children t_1 and t_2. Then $\mathcal{R}(t)$ is the union of tuples (a, p, s) formed by tuples $(a_1, p_1, s_1) \in \mathcal{R}(t_1)$ and $(a_2, p_2, s_2) \in \mathcal{R}(t_2)$, where (1)

$a = a_1 = a_2$; (2) p is the transitive closure of $p_1 \cup p_2$; (3) p contains no cycles; and (4) $s = s_1 + s_2$.

- **Introduce node:** Suppose t is an introduce node with child t', such that $\chi(t) = \chi(t') \cup \{v_0\}$. Then $\mathcal{R}(t)$ is the set of tuples (a, p, s) formed by pairs $P \subseteq N_I(v_0)$ and $(a', p', s') \in \mathcal{R}(t')$, such that (1) $a(v_0) = P$; (2) for every $v \in \chi(t')$, we have $a(v) = a'(v)$; (3) p is the transitive closure of $p' \cup \{(u, v_0) : u \in P\} \cup \{(v_0, u) : v_0 \in a'(u), u \in \chi(t')\}$; (4) p contains no cycles; and (5) $s = s'$.
- **Forget node:** Suppose t is a forget node with child t', such that $\chi(t') = \chi(t) \cup \{v_0\}$. Then $\mathcal{R}(t)$ is the set of tuples (a, p, s) formed from tuples $(a', p', s') \in \mathcal{R}(t')$, such that (1) $a(u) = a'(u),\ \forall u \in \chi(t)$; (2) $p = \{(u, v) \in p' : u, v \in \chi(t)\}$; and (3) $s = s' + \text{score}_{v_0}(a'(v_0))$.

Note that Korhonen and Parviainen (2013) present a variant of this dynamic programming method, also using a nice tree decomposition, which is applicable even for graphs with unbounded degree but bounded treewidth. They assume that the starting undirected graph G_I is a superset of the moralized DAG. Their algorithm runs in time linear in p and exponential in w. Since $\text{supp}(\Theta_0)$ exactly corresponds to the edge set of $\mathcal{M}(G_0)$, the alternative method will also lead to correct recovery. In practice, the relative efficiency of the two dynamic programming algorithms will rely heavily on the structure of $\mathcal{M}(G_0)$, and it is an interesting direction of future work to investigate the behavior of the two dynamic programming algorithms for different graph structures.

6.3 Runtime

We first review the runtime of various components of the dynamic programming algorithm described in Section 6.2. This is mentioned briefly in Ordyniak and Szeider (2012), but we include details for completeness before comparing the runtime of our overall procedure with other causal inference methods. In our calculations, we assume the treewidth w of G is bounded and treat w as a constant.

The first step involves constructing a tree decomposition of minimal treewidth w, which may be done in time $\mathcal{O}(p)$. The second step involves constructing a nice tree decomposition. Given a tree decomposition of width w, a nice tree decomposition with $\mathcal{O}(p)$ nodes and treewidth w may be constructed in $\mathcal{O}(p)$ time (see Appendix A.2). Finally, the third step involves computing records for nodes in the nice tree decomposition. We consider the three different types of nodes in succession. Note that

$$|\mathcal{R}(t)| \leq 2^{(w+1)(w+d)}, \qquad \forall t, \tag{34}$$

where $d = \max_j |N_I(j)|$. This is because the number of choices of parent sets of any vertex in $\chi(t)$ is bounded by 2^d, leading to a factor of $2^{d(w+1)}$, and the number of possible pairs that are connected by a path is bounded by $2^{(w+1)w}$.

- If t is a join node with children t_1 and t_2, we may compute $\mathcal{R}(t)$ by comparing pairs of records in $\mathcal{R}(t_1)$ and $\mathcal{R}(t_2)$; by Inequality (34), this may be done in time $\mathcal{O}(2^{2(w+1)(w+d)})$.

- If t is an introduce node with child t', we may compute $\mathcal{R}(t)$ by considering records in $\mathcal{R}(t')$ and parent sets of the introduced node v_0. Since the number of choices for the latter is bounded by 2^d, we conclude that $\mathcal{R}(t)$ may be computed in time $\mathcal{O}(2^{(w+1)(w+d)+d})$.

- Clearly, if t is a forget node, then $\mathcal{R}(t)$ may be computed in time $\mathcal{O}(2^{(w+1)(w+d)})$.

Altogether, we conclude that all records of nodes in the nice tree decomposition may be computed in time $\mathcal{O}(p \cdot 2^{2(w+1)(w+d)})$. Combined with the graphical Lasso preprocessing step for estimating $\mathcal{M}(G_0)$, this leads to an overall complexity of $\mathcal{O}(p^2)$. This may be compared to the runtime of other standard methods for causal inference, including the PC algorithm (Spirtes et al., 2000), which has computational complexity $\mathcal{O}(p^w)$, and (direct) LiNGAM (Shimizu et al., 2006, 2011), which requires time $\mathcal{O}(p^4)$. It has been noted that both the PC and LiNGAM algorithms may be expedited when prior knowledge about the DAG space is available, further highlighting the power of Theorem 2 as a preprocessing step for any causal inference algorithm.

7. Discussion

We have provided a new framework for estimating the DAG corresponding to a linear SEM. We have shown that the inverse covariance matrix of linear SEMs always reflects the edge structure of the moralized graph, even in non-Gaussian settings, and the reverse statement also holds under a mild faithfulness assumption. Furthermore, we have shown that when the error variances are known up to close precision, a simple weighted squared ℓ_2-loss may be used to select the correct DAG. As a corollary, we have established identifiability for the class of linear SEMs with error variances specified up to a constant multiple. We have proved that our methods are statistically consistent, under reasonable assumptions on the gap between the score of the true DAG and the next best DAG in the model class. A characterization of this gap parameter for various graphical structures is the topic of future work.

We have also shown how dynamic programming may be used to select the best-scoring DAG in an efficient manner, assuming the treewidth of the moralized graph is small. Our results relating the inverse covariance matrix to the moralized DAG provide a powerful method for reducing the DAG search space as a preprocessing step for dynamic programming, and are the first to provide rigorous guarantees for when the graphical Lasso may be used in non-Gaussian settings. Note that the dynamic programming algorithm discussed in this paper only uses the information that the true DAG has skeleton lying in the input graph, and does *not* incorporate any information about (a) the fact that the data comes from a linear SEM; or (b) the fact that the input graph exactly equals the moralized DAG. Intuitively, both types of information should place significant constraints on the restricted DAG space, leading to further speedups in the dynamic programming algorithm. Perhaps these restrictions would make it possible to establish a version of dynamic programming for DAGs where the moralized graph has bounded degree but large treewidth.

An important open question concerns scoring candidate DAGs when the diagonal matrix Ω_0 of error variances is unknown. As we have seen, using the weighted squared ℓ_2-loss to score DAGs may produce a graph that is far from the true DAG when Ω_0 is misspecified.

Alternatively, it would be useful to have a checkable condition that would allow us to verify whether a given matrix Ω will correctly select the true DAG, or to be able to select the true Ω_0 from among a finite collection of candidate matrices.

Acknowledgments

We acknowledge all the members of the Seminar für Statistik for providing an immensely hospitable and fruitful environment when PL was visiting ETH, and the Forschungsinstitut für Mathematik at ETH Zürich for financial support. We also thank Varun Jog for helpful discussions. PL was additionally supported by a Hertz Fellowship and an NSF Graduate Research Fellowship. We thank the AE and anonymous reviewers for helpful feedback.

Appendix A. Graph-Theoretic Concepts

In this Appendix, we review some fundamental concepts in graph theory that we use in our exposition. We begin by discussing junction trees, and then move to the related notion of tree decompositions. Note that these are purely graph-theoretic operations that may be performed on an arbitrary undirected graph.

A.1 Junction Trees

We begin with the basic junction tree framework. For more details, see Lauritzen (1996) or Koller and Friedman (2009).

For an undirected graph $G = (V, E)$, a *triangulation* is an augmented graph $\widetilde{G} = (V, \widetilde{E})$ that contains no chordless cycles of length greater than three. By classical graph theory, any triangulation $\widetilde{G}$ gives rise to a *junction tree* representation of G, where nodes in the junction tree are subsets of V corresponding to maximal cliques of $\widetilde{G}$, and the intersection of any two adjacent cliques C_1 and C_2 in the junction tree is referred to as a *separator set* $S = C_1 \cap C_2$. Furthermore, any junction tree must satisfy the *running intersection property*, meaning that for any two nodes in the junction tree, say corresponding to cliques C_j and C_k, the intersection $C_j \cap C_k$ must belong to every separator set on the unique path between C_j and C_k in the junction tree. The *treewidth* of G is defined to be one less than the size of the largest clique in any triangulation $\widetilde{G}$ of G, minimized over all triangulations.

As a classic example, note that if G is a tree, then G is already triangulated, and the junction tree parallels the tree structure of G. The maximal cliques in the junction tree are equal to the edges of G and the separator sets correspond to singleton vertices. The treewidth of G is consequently equal to 1.

A.2 Tree Decompositions

We now highlight some basic concepts of tree decompositions and nice tree decompositions used in our dynamic programming framework. Our exposition follows Kloks (1994).

Let $G = (V, E)$ be an undirected graph. A *tree decomposition* of G is a tree T with node set W such that each node $t \in W$ is associated with a subset $V_t \subseteq V$, and the following properties are satisfied:

(a) $\bigcup_{t \in T} V_t = V$;

(b) for all $(u, v) \in E$, there exists a node $t \in W$ such that $u, v \in V_t$;

(c) for each $v \in V$, the set of nodes $\{t : v \in V_t\}$ forms a subtree of T.

The *width* of the tree decomposition is $\max_{t \in T} |V_t| - 1$. The *treewidth* of G is the minimal width of any tree decomposition of G; this quantity agrees with the treewidth defined in terms of junction trees in the previous section. If G has bounded treewidth, a tree decomposition with minimum width may be constructed in time $\mathcal{O}(|V|)$ (cf. Chapter 15 of Kloks 1994).

A *nice tree decomposition* is rooted tree decomposition satisfying the following properties:

(a) every node has at most two children;

(b) if a node t has two children r and s, then $V_t = V_r = V_s$;

(c) if a node t has one child s, then either

(i) $|V_t| = |V_s| + 1$ and $V_s \subseteq V_t$, or

(ii) $|V_s| = |V_t| + 1$ and $V_t \subseteq V_s$.

Nodes of the form (b), (c)(i), and (c)(ii) are called *join* nodes, *introduce* nodes, and *forget* nodes, respectively. Given a tree decomposition of G with width w, a nice tree decomposition with width w and at most $4|V|$ nodes may be computed in time $\mathcal{O}(|V|)$ (cf. Lemma 13.1.3 of Kloks 1994).

Appendix B. Matrix Derivations

In this section, we present a few matrix results that are used to prove Theorem 7.

Define a *unit lower triangular (LT)* matrix to be a lower triangular matrix with 1's on the diagonal. Recall that matrices A and B are *permutation similar* if there exists a permutation matrix P such that $A = PBP^T$. Call a matrix *permutation unit LT* if it is permutation similar to a unit lower triangular matrix. We have the following lemma:

Lemma 25 *Suppose A and B are permutation unit LT matrices, and suppose $AA^T = BB^T$. Then $A = B$.*

Proof Under the appropriate relabeling, we assume without loss of generality that A is unit LT. There exists a permutation matrix P such that $C := PBP^T$ is also unit LT. We have

$$PAA^TP^T = CC^T. \tag{35}$$

Let π be the permutation on $\{1, \ldots, n\}$ such that $P_{i,\pi(i)} = 1$ for all i, and P has 0's everywhere else. Define the notation

$$\widetilde{a}_{ij} := (PA)_{ij}, \qquad \text{and} \qquad m_{ij} := (CC^T)_{ij},$$

and let $\{c_{ij}\}$ denote the entries of C. We will make use of the following equalities, which follow from Equation (35) and the fact that C is unit LT:

$$\sum_k \widetilde{a}_{ik}\widetilde{a}_{jk} \;=\; m_{ij} \;=\; \sum_{k<j} c_{ik}c_{jk} + c_{ij}, \qquad \forall i > j, \tag{36}$$

and

$$\sum_k \widetilde{a}_{ik}^2 \;=\; m_{ii} \;=\; 1 + \sum_{k<i} c_{ik}^2. \tag{37}$$

We now derive the following equality:

$$\widetilde{a}_{i,\pi(j)} = c_{ij}, \qquad \forall i, j. \tag{38}$$

Note that Equation (38) implies $(PA)P^T = C$, from which it follows that $A = B$.

If $j = i$, we have $\widetilde{a}_{i,\pi(i)} = 1$ trivially, since A has 1's on the diagonal. For the remaining cases, we induct on i. When $i = 1$, we need to show that $\widetilde{a}_{1,\pi(1)} = 1$ and all other entries in the first row are 0. By Equation (37), we have

$$\sum_k \widetilde{a}_{1k}^2 = m_{11} = 1.$$

Since $\widetilde{a}_{1,\pi(1)} = 1$, it is clear that $\widetilde{a}_{1k} = 0$ for all $k \neq \pi(1)$, establishing the base case.

For the induction step, consider $i > 1$. We first show that $\widetilde{a}_{i,\pi(j)} = c_{ij}$ for all $j < i$ by a sub-induction on j. For $j = 1$, we have by Equation (36) and the base result for $i = 1$ that

$$\widetilde{a}_{i,\pi(1)} = m_{i,1} = c_{i,1},$$

which is exactly what we want. For the sub-induction step, consider $1 < j < i$, and suppose $\widetilde{a}_{i,\pi(\ell)} = c_{i\ell}$ for all $\ell < j$. Note that $\widetilde{a}_{j,\pi(\ell)} = 0$ for all $\ell > j$ by the outer induction hypothesis. Hence, Equation (36) and the fact that $\widetilde{a}_{j,\pi(j)} = 1$ gives

$$\sum_{\ell<j} \widetilde{a}_{i,\pi(\ell)}\widetilde{a}_{j,\pi(\ell)} + \widetilde{a}_{i,\pi(j)} \;=\; m_{ij} \;=\; \sum_{k<j} c_{ik}c_{jk} + c_{ij}. \tag{39}$$

Since also $\widetilde{a}_{j,\pi(\ell)} = c_{j\ell}$ for $\ell < j$ by the outer induction hypothesis, Equation (39) condenses to

$$\sum_{\ell<j} c_{i\ell}c_{j\ell} + \widetilde{a}_{i,\pi(j)} = \sum_{k<j} c_{ik}c_{jk} + c_{ij},$$

from which it follows that $\widetilde{a}_{i,\pi(j)} = c_{ij}$, as wanted. This completes the inner induction and shows that $\widetilde{a}_{i,\pi(j)} = c_{ij}$, for all $j < i$. Finally, note that by Equation (37), we have

$$m_{ii} = \sum_k \widetilde{a}_{ik}^2 = 1 + \sum_{j\neq i} \widetilde{a}_{i,\pi(j)}^2 \geq 1 + \sum_{j<i} \widetilde{a}_{i,\pi(j)}^2 = 1 + \sum_{j<i} c_{ij}^2 = m_{ii},$$

implying that we must have $\widetilde{a}_{i,\pi(j)} = 0$, for all $j > i$. This establishes Equation (38).

■

We also need the following known result (cf. Exercise 7.8.19 in Horn and Johnson 1990). We include a proof for completeness.

Lemma 26 *Suppose $A \in \mathbb{R}^{n\times n}$ is positive definite with $\det(A) = 1$. Then*

$$\min\{\mathrm{tr}(AB) : B \succ 0 \text{ and } \det(B) = 1\} = n.$$

Proof Consider the singular value decomposition $A = U\Lambda U^*$, and note that $\mathrm{tr}(AB) = \mathrm{tr}(\Lambda(U^*BU))$. Denote $b_{ij} := (U^*BU)_{ij}$ and $\lambda_i := \Lambda_{ii}$. Then by the AM-GM inequality and Hadamard's inequality, we have

$$\frac{1}{n}\cdot \mathrm{tr}(\Lambda(U^*BU)) \geq \left(\prod_i \lambda_i b_{ii}\right)^{1/n} = \left(\det(A)\cdot \prod_i b_{ii}\right)^{1/n} \geq (\det(U^*BU))^{1/n} = 1,$$

implying the result. ■

Building upon Lemmas 25 and 26, we obtain the following result:

Lemma 27 *Suppose A and B are $n \times n$ permutation unit LT matrices. Then*

$$\min_B \mathrm{tr}(AA^TB^TB) \geq n, \tag{40}$$

with equality achieved if and only if $B = A^{-1}$.

Proof Write $A' = AA^T$ and $B' = B^TB$, and note that since $\det(A) = \det(B) = 1$, we also have $\det(A') = \det(B') = 1$. Then Inequality (40) holds by Lemma 26.

To recover the conditions for equality, note that equality holds in Hadamard's inequality if and only if some $b_{ii} = 0$ or the matrix U^*BU is diagonal. Note that the first case is not possible, since $U^*BU \succ 0$. In the second case, we see that in addition, we need $b_{ii} = \frac{1}{\lambda_i}$ for all i in order to achieve equality in the AM-GM inequality. It follows that $U^*BU = \Lambda^{-1}$, so $AA^T = B^{-1}B^{-T}$.

Since A and B are permutation unit LT, Lemma 25 implies that the last equality can only hold when $B = A^{-1}$. ■

Appendix C. Proofs for Population-Level Results

In this section, we provide proofs for the remaining results in Sections 3 and 4.

C.1 Proof of Lemma 1

We first show that Ω is a diagonal matrix. Consider $j < k$; we will show that $\epsilon_j \perp\!\!\!\perp \epsilon_k$, from which we conclude that

$$\mathrm{E}[\epsilon_j\epsilon_k] = \mathrm{E}[\epsilon_j]\cdot \mathrm{E}[\epsilon_k] = 0.$$

Indeed, we have $\epsilon_k \perp\!\!\!\perp (X_1, \ldots, X_{k-1})$ by assumption. Since $\epsilon_j = X_j - b_j^TX$ is a deterministic function of $(X_1, \ldots, X_j)$, it follows that $\epsilon_k \perp\!\!\!\perp \epsilon_j$, as claimed.

Turning to Equations (5) and (6), note that Equation (4) implies

$$\Theta = \Sigma^{-1} = (I - B)\Omega^{-1}(I - B)^T.$$

Then expanding and using the fact that B is upper triangular and Ω is diagonal, we obtain Equations (5) and (6).

C.2 Proof of Lemma 6

Since $G_0 \subseteq G$, we have $\mathrm{Pa}_{G_0}(j) \subseteq \mathrm{Pa}_G(j)$, for each j. Furthermore, no element of $\mathrm{Pa}_G(j)$ may be a descendant of j in G_0, since this would contradict the fact that G contains no cycles. By the Markov property of G_0, we therefore have

$$X_j \perp\!\!\!\perp X_{\mathrm{Pa}_G(j) \setminus \mathrm{Pa}_{G_0}(j)} \mid X_{\mathrm{Pa}_{G_0}(j)}.$$

Thus, the linear regression coefficients for X_j regressed upon $X_{\mathrm{Pa}_G(j)}$ are simply the linear regression coefficients for X_j regressed upon $X_{\mathrm{Pa}_{G_0}(j)}$ (and the remaining coefficients for $X_{\mathrm{Pa}_G(j) \setminus \mathrm{Pa}_{G_0}(j)}$ are zero). By Remark 5, we conclude that

$$B_0 = B_G = \arg\min_{B \in \mathcal{U}_G} \{\mathrm{score}_\Omega(B)\},$$

and the uniqueness of B_0 follows from the uniqueness of B_G.

C.3 Proof of Theorem 9

From the decomposition (7), it is easy to see that for any $B \in \mathcal{U}$, we have

$$a_{\min} \leq \frac{\mathrm{score}_{\Omega_1}(B)}{\mathrm{score}_{\Omega_0}(B)} \leq a_{\max}, \tag{41}$$

simply by comparing individual terms; e.g.,

$$\begin{aligned}\frac{1}{(\Omega_1)_{jj}} \cdot \mathrm{E}[(X_j - b_j^T X)^2] &\leq \max_j \left\{ \frac{1/(\Omega_1)_{jj}}{1/(\Omega_0)_{jj}} \right\} \cdot \frac{1}{(\Omega_0)_{jj}} \cdot \mathrm{E}[(X_j - b_j^T X)^2] \\ &= a_{\max} \left(\frac{1}{(\Omega_0)_{jj}} \cdot \mathrm{E}[(X_j - B_j^T X)^2] \right).\end{aligned}$$

Note that if $G \supseteq G_0$, then by Lemma 6, the matrix B_0 is the unique minimizer of $\mathrm{score}_{\Omega_1}(B)$ among the class $\mathcal{U}_G$. Now consider $G \not\supseteq G_0$ and $B \in \mathcal{U}_G$. We have

$$\left(1 + \frac{\xi}{p}\right) \cdot \mathrm{score}_{\Omega_0}(B_0) = \min_{G' \in \mathcal{D},\, G' \not\supseteq G_0} \{\mathrm{score}_{\Omega_0}(G')\} \leq \mathrm{score}_{\Omega_0}(G) \leq \mathrm{score}_{\Omega_0}(B), \tag{42}$$

where we have used the definition of the gap (11) and the fact that $\mathrm{score}_{\Omega_0}(B_0) = p$ by Theorem 7 in the first inequality. Hence,

$$\mathrm{score}_{\Omega_1}(B_0) \leq a_{\max} \cdot \mathrm{score}_{\Omega_0}(B_0) \leq \frac{a_{\max}}{1 + \xi/p} \cdot \mathrm{score}_{\Omega_0}(B) \leq \frac{a_{\max}}{a_{\min}(1 + \xi/p)} \cdot \mathrm{score}_{\Omega_1}(B),$$

where the first and third inequalities use Inequality (41), and the second inequality uses Inequality (42). By the assumption (12), it follows that

$$\mathrm{score}_{\Omega_1}(B_0) \leq \mathrm{score}_{\Omega_1}(B),$$

as wanted. The statement regarding strict inequality is clear.

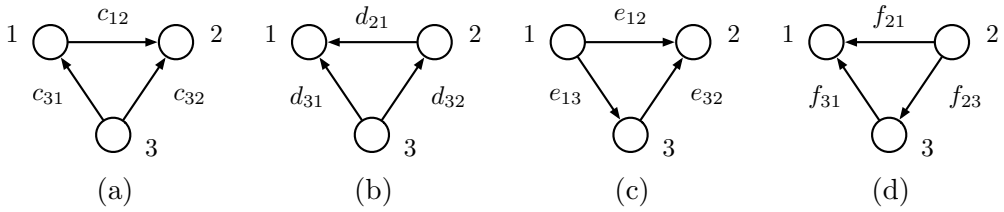


Figure 3: Alternative DAGs.

C.4 Proof of Lemma 11

We first consider the case when $r \geq 1$. Then $\frac{a_{\max}}{a_{\min}} = r^2$, so combining Theorem 9 with the expression (15), we have the sufficient condition

$$r^2 \leq 1 + \frac{b_0^4}{2(r^4 + b_0^2 r^2)}.$$

Rearranging gives

$$b_0^4 - 2r^2(r^2-1)b_0^2 - 2r^4(r^2-1) \geq 0,$$

which is equivalent to

$$b_0^2 \geq \frac{2r^2(r^2-1) + \sqrt{(4r^4(r^2-1)^2 + 8r^4(r^2-1))}}{2}.$$

Simplifying yields the desired expression.

If instead $r \leq 1$, we have $\frac{a_{\max}}{a_{\min}} = \frac{1}{r^2}$, so the sufficient condition becomes

$$\frac{1}{r^2} \leq 1 + \frac{b_0^4}{2(r^4 + b_0^2 r^2)},$$

which is equivalent to

$$b_0^4 - 2(1-r^2)b_0^2 - 2r^2(1-r^2) \geq 0,$$

or

$$b_0^2 \geq \frac{2(1-r)^2 + \sqrt{4(1-r^2)^2 + 8r^2(1-r^2)}}{2}.$$

Simplifying further yields the expression.

C.5 Proof of Lemma 13

To compute the gap ξ, it is sufficient to consider the graphs in Figure 3. Indeed, we have $\text{score}_{\Omega_0}(G) \leq \text{score}_{\Omega_0}(G')$ whenever $G' \subseteq G$, so we only need to consider maximal elements in the poset of DAGs not containing G_0. Consider the graphs given by autoregression matrices

$$C = \begin{pmatrix} 0 & c_{12} & 0 \\ 0 & 0 & 0 \\ c_{31} & c_{32} & 0 \end{pmatrix}, \qquad E = \begin{pmatrix} 0 & e_{12} & e_{13} \\ 0 & 0 & 0 \\ 0 & e_{32} & 0 \end{pmatrix},$$

corresponding to the DAGs in panels (a) and (c) of Figure 3. A simple calculation shows that

$$\text{score}_{\Omega_0}(C) = 3 + c_{31}^2 b_{13}^2 + c_{32}^2 b_{23}^2 + c_{31}^2 b_{23}^2 \frac{d_2^2}{d_1^2} + \left(c_{12}\frac{d_1}{d_2} + c_{32} b_{13}\frac{d_1}{d_2}\right)^2 + \left(c_{31}\frac{d_3}{d_1} - b_{13}\frac{d_1}{d_3}\right)^2 + \left(c_{32}\frac{d_3}{d_2} - b_{23}\frac{d_2}{d_3}\right)^2,$$

which is minimized for

$$c_{12} = -b_{13}c_{32}, \qquad c_{31} = \frac{b_{13}}{\frac{d_3^2}{d_1^2} + b_{23}^2\frac{d_2^2}{d_1^2} + b_{13}^2}, \qquad c_{32} = \frac{b_{23}}{\frac{d_3^2}{d_2^2} + b_{23}^2},$$

leading to

$$\xi_1 = \min_{c_{12},c_{31},c_{32}} \{\text{score}_{\Omega_0}(C) - \text{score}_{\Omega_0}(B_0)\} = \frac{b_{23}^4}{\frac{d_3^4}{d_2^4} + b_{23}^2\frac{d_3^2}{d_2^2}} + \frac{b_{13}^4 + b_{13}^2 b_{23}^2\frac{d_2^2}{d_1^2}}{\frac{d_3^4}{d_1^4} + b_{13}^2\frac{d_3^2}{d_1^2} + b_{23}^2\frac{d_2^2 d_3^2}{d_1^4}}. \tag{43}$$

Similarly, we may compute

$$\text{score}_{\Omega_0}(E) = 3 + \left(e_{12}\frac{d_1}{d_2} + e_{32}b_{13}\frac{d_1}{d_2}\right)^2 + \left(e_{13}\frac{d_1}{d_3} - b_{13}\frac{d_1}{d_3}\right)^2 + \left(e_{32}\frac{d_3}{d_2} - b_{23}\frac{d_2}{d_3}\right)^2,$$

which is minimized for

$$e_{12} = -b_{13}, \qquad e_{13} = b_{13}, \qquad e_{32} = \frac{b_{23}}{\frac{d_3^2}{d_2^2} + b_{23}^2},$$

leading to

$$\xi_2 = \min_{e_{12},e_{13},e_{32}} \{\text{score}_{\Omega_0}(E) - \text{score}_{\Omega_0}(B_0)\} = \frac{b_{23}^4}{\frac{d_3^4}{d_2^4} + b_{23}^2\frac{d_3^2}{d_2^2}}. \tag{44}$$

Finally, note that the graphs in panels (b) and (d) of Figure 3 are mirror images of the graphs in panels (a) and (c), respectively. Hence, we obtain

$$\xi_3 = \min_{d_{21},d_{31},d_{32}} \{\text{score}_{\Omega_0}(D) - \text{score}_{\Omega_0}(B_0)\} = \frac{b_{13}^4}{\frac{d_3^4}{d_1^4} + b_{13}^2\frac{d_3^2}{d_1^2}} + \frac{b_{23}^4 + b_{13}^2 b_{23}^2\frac{d_1^2}{d_2^2}}{\frac{d_3^4}{d_2^4} + b_{23}^2\frac{d_3^2}{d_2^2} + b_{13}^2\frac{d_1^2 d_3^2}{d_2^4}},$$

$$\xi_4 = \min_{f_{21},f_{23},f_{31}} \{\text{score}_{\Omega_0}(F) - \text{score}_{\Omega_0}(B_0)\} = \frac{b_{13}^4}{\frac{d_3^4}{d_1^4} + b_{13}^2\frac{d_3^2}{d_1^2}},$$

simply by swapping the roles of nodes 1 and 2. Taking $\xi = \min\{\xi_1, \xi_2, \xi_3, \xi_4\}$ then yields the desired result.

Appendix D. Proofs for Statistical Consistency

In this Appendix, we provide the proofs for the lemmas on statistical consistency stated in Section 5.

D.1 Proof of Lemma 15

This result follows from the fact that

$$\|\widehat{\Theta} - \Theta_0\|_{\max} \leq \left|\!\left|\!\left| \widehat{\Theta} - \Theta_0 \right|\!\right|\!\right|_2,$$

together with results on the spectral norm of sub-Gaussian covariances and their inverses (see Lemma 29 in Appendix E).

D.2 Proof of Lemma 18

First consider a fixed pair (j, S) such that $S \subseteq N_\Theta(j)$. We may write

$$x_j = b_j^T x_S + e_j, \tag{45}$$

where e_j has zero mean and is uncorrelated with x_S (and also depends on the choice of S). In matrix notation, we have

$$\widehat{b}_j = (X_S^T X_S)^{-1}(X_S^T X_j) = (X_S^T X_S)^{-1} X_S^T (X_S b_j + E_j) = b_j + (X_S^T X_S)^{-1} X_S^T E_j,$$

where the second equality follows from Equation (45). Hence,

$$\begin{aligned}
\sigma_j^2 \cdot \widehat{f}_{\sigma_j}(S) &= \frac{1}{n}\|X_j - X_S \widehat{b}_j\|_2^2 \\
&= \frac{1}{n}\|X_S(b_j - \widehat{b}_j) + E_j\|_2^2 \\
&= \frac{1}{n}\|(I - (X_S^T X_S)^{-1} X_S^T) E_j\|_2^2. \qquad (46)
\end{aligned}$$

By the triangle inequality, we have

$$\left| \|(I - (X_S^T X_S)^{-1} X_S^T) E_j\|_2 - \|E_j\|_2 \right| \leq \|(X_S^T X_S)^{-1} X_S^T E_j\|_2 \leq \left|\!\left|\!\left| (X_S^T X_S)^{-1} X_S^T \right|\!\right|\!\right|_2 \cdot \|E_j\|_2. \tag{47}$$

Furthermore,

$$\begin{aligned}
\left|\!\left|\!\left| (X_S^T X_S)^{-1} X_S^T \right|\!\right|\!\right|_2 &= \left|\!\left|\!\left| X_S (X_S^T X_S)^{-1} \right|\!\right|\!\right|_2 \\
&= \sup_{\|v\|_2 \leq 1} \left\{ v^T (X_S^T X_S)^{-1} X_S^T X_S (X_S^T X_S)^{-1} v \right\}^{1/2} \\
&= \sup_{\|v\|_2 \leq 1} \left\{ v^T (X_S^T X_S)^{-1} v \right\}^{1/2} \\
&= \frac{1}{\sqrt{n}} \left|\!\left|\!\left| \left(\frac{X_S^T X_S}{n} \right)^{-1} \right|\!\right|\!\right|_2^{1/2} \\
&\leq \frac{C}{\sqrt{n}} \left(\left|\!\left|\!\left| \Sigma_{SS}^{-1} \right|\!\right|\!\right|_2 + 2\sigma^2 \left|\!\left|\!\left| \Sigma_{SS}^{-1} \right|\!\right|\!\right|_2^2 \cdot \max\{\delta, \delta^2\} \right)^{1/2},
\end{aligned}$$

with probability at least $1 - 2\exp(-cnt^2)$, where $\delta = c'\sqrt{\frac{|S|}{n}} + c''t$, by Lemma 29. Taking a union bound over all 2^d choices for S and p choices for j, and setting $t = c\sqrt{\frac{d + \log p}{n}}$, we have

$$\left|\!\left|\!\left| (X_S^T X_S)^{-1} X_S^T \right|\!\right|\!\right|_2 \leq \frac{C'}{\sqrt{n}}, \qquad \forall S \text{ s.t. } S \subseteq N_\Theta(j) \text{ for some } j, \tag{48}$$

with probability at least

$$1 - c_1 \exp(-c_2 n t^2) = 1 - c_1 \exp(-c_2 n t^2 + d \log 2 + \log p) \geq 1 - c_1 \exp(-c_2'(d + \log p)).$$

Combining Inequalities (47) and (48), we have the uniform bound

$$\left(1 - \frac{C'}{\sqrt{n}}\right)^2 \|E_j\|_2^2 \leq \|I - (X_S^T X_S)^{-1} X_S^T) E_j\|_2^2 \leq \left(1 + \frac{C'}{\sqrt{n}}\right)^2 \|E_j\|_2^2,$$

with high probability, which together with Equation (46) implies that

$$\left| \sigma_j^2 \cdot \widehat{f}_{\sigma_j}(S) - \frac{1}{n} \|E_j\|_2^2 \right| \leq \frac{3C'}{\sqrt{n}} \cdot \frac{1}{n} \|E_j\|_2^2, \tag{49}$$

using the fact that

$$\max\{1 - (1-a)^2, (1+a)^2 - 1\} = \max\{2a - a^2, 2a + a^2\} = 2a + a^2 \leq 3a,$$

for $a = \frac{C'}{\sqrt{n}}$ sufficiently small. Furthermore,

$$\frac{1}{n} \operatorname{E}[\|E_j\|_2^2] = \sigma_j^2 \cdot f_{\sigma_j}(S).$$

Note that the e_j's are i.i.d. sub-Gaussians with parameter at most $c\sigma^2$, since we may write $e_j = \widetilde{b}_j^T x$ for the appropriate $\widetilde{b}_j \in \mathbb{R}^p$, and $\|\widetilde{b}\|_2$ is bounded in terms of the eigenvalues of Σ. Applying the usual sub-Gaussian tail bounds, we then have

$$\mathbb{P}\left(\left| \frac{1}{n} \|E_j\|_2^2 - \frac{1}{n} \operatorname{E}[\|E_j\|_2^2] \right| \geq c\sigma^2 t \right) \leq c_1 \exp(-c_2 n t^2), \qquad \forall j,$$

and taking a union bound over j and setting $t = c' \sqrt{\frac{\log p}{n}}$ gives

$$\max_j \left| \frac{1}{n} \|E_j\|_2^2 - \frac{1}{n} \operatorname{E}[\|E_j\|_2^2] \right| \leq c_0 \sigma^2 \sqrt{\frac{\log p}{n}}, \tag{50}$$

with probability at least $1 - c_1 \exp(-c_2 \log p)$. Combining Inequalities (49) and (50), it follows that

$$\begin{aligned}
\sigma_j^2 |\widehat{f}_{\sigma_j}(S) - f_{\sigma_j}(S)| &\leq \left| \sigma_j^2 \cdot \widehat{f}_{\sigma_j}(S) - \frac{1}{n} \|E_j\|_2^2 \right| + \left| \frac{1}{n} \|E_j\|_2^2 - \frac{1}{n} \operatorname{E}[\|E_j\|_2^2] \right| \\
&\leq \frac{3C'}{\sqrt{n}} \left(\frac{1}{n} \operatorname{E}[\|E_j\|_2^2] + c_0 \sigma^2 \sqrt{\frac{\log p}{n}} \right) + c_0 \sigma^2 \sqrt{\frac{\log p}{n}} \\
&\leq c_0' \sigma^2 \sqrt{\frac{\log p}{n}},
\end{aligned}$$

with probability at least $1 - c_1 \exp(-c_2 \log p)$.

D.3 Proof of Lemma 19

Combining Inequalities (21) and (23) and using the triangle inequality, we have

$$\left|\widehat{\text{score}}_\Omega(G) - \text{score}_\Omega(G)\right| \leq \sum_{j=1}^{p} \left|\widehat{\text{score}}_{\sigma_j}(\text{Pa}_G(j)) - \text{score}_{\sigma_j}(\text{Pa}_G(j))\right| < \frac{\xi(\mathcal{D}_\Theta)}{2}, \tag{51}$$

for all $G \in \mathcal{D}_\Theta$. In particular, for $G_1 \in \mathcal{D}_\Theta$ such that $G_1 \not\supseteq G_0$, we have

$$\begin{aligned}\widehat{\text{score}}_\Omega(G_0) &< \text{score}_\Omega(G_0) + \frac{\xi(\mathcal{D}_\Theta)}{2} \\ &\leq (\text{score}_\Omega(G_1) - \xi(\mathcal{D}_\Theta)) + \frac{\xi(\mathcal{D}_\Theta)}{2} \\ &< \widehat{\text{score}}_\Omega(G_1),\end{aligned}$$

where the first and third inequalities use Inequality (51) and the second inequality uses the definition of the gap $\xi(\mathcal{D}_\Theta)$. This implies Inequality (24).

D.4 Proof of Lemma 21

Consider $G \in \mathcal{D}_\Theta$ with $G \not\supseteq G_0$, and consider $G_1 \supseteq G_0$ such that $\gamma_\Omega(G, G_1)$ is maximized. Note that if $\text{Pa}_G(j) = \text{Pa}_{G_1}(j)$, then certainly,

$$\widehat{f}_{\sigma_j}(\text{Pa}_G(j)) - \widehat{f}_{\sigma_j}(\text{Pa}_{G_1}(j)) = 0 = f_{\sigma_j}(\text{Pa}_G(j)) - f_{\sigma_j}(\text{Pa}_{G_1}(j)).$$

Hence,

$$\left|\left(\widehat{\text{score}}_\Omega(G) - \widehat{\text{score}}_\Omega(G_1)\right) - \left(\text{score}_\Omega(G) - \text{score}_\Omega(G_1)\right)\right| \leq |H(G, G_1)| \cdot \frac{\xi'}{2}, \tag{52}$$

using Inequality (29) and the triangle inequality. Furthermore, by Inequality (28),

$$|H(G, G_1)| \cdot \frac{\xi'}{2} \leq \frac{\text{score}_\Omega(G) - \text{score}_\Omega(G_1)}{\xi'} \cdot \frac{\xi'}{2} = \frac{\text{score}_\Omega(G) - \text{score}_\Omega(G_1)}{2}. \tag{53}$$

Combining Inequalities (52) and (53) gives

$$\widehat{\text{score}}_\Omega(G) - \widehat{\text{score}}_\Omega(G_1) \geq \frac{\text{score}_\Omega(G) - \text{score}_\Omega(G_1)}{2} = \frac{\text{score}_\Omega(G) - \text{score}_\Omega(G_0)}{2} > 0,$$

where the last inequality holds because of the assumption $\xi' > 0$. Hence,

$$G \notin \arg\min_{G \in \mathcal{D}_\Theta} \{\widehat{\text{score}}_\Omega(G)\},$$

implying the desired result.

D.5 Proof of Lemma 23

We begin with a simple lemma:

Lemma 28 *Suppose $\mathcal{M}(G)$ admits a junction tree representation with only singleton separators, and let $C_1, \ldots, C_k$ denote the maximal cliques. If X follows a linear SEM over G, then the marginal distribution of X over the nodes in any clique C_ℓ also follows a linear SEM over C_ℓ, with DAG structure specified by G_ℓ, the restriction of G to C_ℓ. In addition, the autoregression matrix for the marginal SEM is simply the autoregression matrix for the full SEM restricted to the nodes in C_ℓ.*

Proof We relabel the nodes of G so that the natural ordering on $\{1, \ldots, p\}$ is a topological order. Clearly, this induces a topological order over the nodes of G_ℓ, as well. Recall that we have Equation (3); i.e., for each j,

$$X_j = b_j^T X_{1:j-1} + \epsilon_j, \qquad \text{where } \epsilon_j \perp\!\!\!\perp (X_1, \ldots, X_{j-1}). \tag{54}$$

For each $j \in C_\ell$, we define

$$\epsilon'_j := \epsilon_j + \sum_{k<j,\, k \notin \mathrm{Pa}_{G_\ell}(j)} b_{kj} X_k,$$

and note that

$$X_j = b_j^T X_{\mathrm{Pa}_{G_\ell}(j)} + \epsilon'_j,$$

where we have abused notation slightly and used b_j to denote the same vector restricted to $\mathrm{Pa}_{G_\ell}(j)$. We claim that

$$\epsilon'_j \perp\!\!\!\perp X_{\mathrm{Pa}_{G_\ell}(j)}, \tag{55}$$

for each j, implying that the marginal distribution of X over C_ℓ follows a linear SEM with the desired properties.

First consider the case when j is not contained in a separator set of the junction tree. Then all neighbors of j must be contained in C_ℓ, implying that $\mathrm{Pa}_{G_\ell}(j) = \mathrm{Pa}_G(j)$. Since $b_{kj} \neq 0$ only when $k < j$ and $k \in \mathrm{Pa}_G(j)$, this means $\epsilon'_j = \epsilon_j$. The desired independence (55) follows from Equation (54) and the simple fact that $\mathrm{Pa}_{G_\ell}(j) \subseteq \{1, \ldots, j-1\}$. If instead j is a separator node, then either $\mathrm{Pa}_G(j) \subseteq C_\ell$ or $\mathrm{Pa}_G(j) \cap C_\ell = \emptyset$. In the first case, we again have $\mathrm{Pa}_{G_\ell}(j) = \mathrm{Pa}_G(j)$, so the argument proceeds as before. In the second case, we have $\mathrm{Pa}_{G_\ell}(j) = \emptyset$, so the independence relation (55) is vacuous; indeed, we have $\epsilon'_j = \epsilon_j + b_j^T X_{\mathrm{Pa}_G(j)} = X_j$. Hence, condition (55) holds in every case. ■

Now consider any $G \in \mathcal{D}_\Theta$ such that $G \not\supseteq G_0$. Let $\{G^\ell\}_{\ell=1}^k$ denote the restrictions of G to the cliques. By Lemma 28, X follows a linear SEM when restricted to the nodes of C_ℓ; hence, by Lemma 6 and Theorem 7, we have

$$\mathrm{score}_\Omega(G_0^\ell) \leq \mathrm{score}_\Omega(G^\ell), \tag{56}$$

with equality if and only if $G_0^\ell \subseteq G^\ell$. Consider the graph G_1 constructed such that $G_1^\ell = G^\ell$ on cliques C_ℓ such that Inequality (56) holds with equality, and $G_1^\ell = G_0^\ell$ otherwise. In particular, we have $G_0^\ell \subseteq G_1^\ell$, for each ℓ, and

$$\mathrm{score}_\Omega(G_1^\ell) = \mathrm{score}_\Omega(G_0^\ell), \qquad \forall \ell, \tag{57}$$

by construction. Note that G_1 is always a DAG, but possibly $\mathcal{M}(G_1) \neq \mathcal{M}(G_0)$. However, since $G_0 \subseteq G_1$, we have

$$\text{score}_\Omega(G_0) = \text{score}_\Omega(G_1). \tag{58}$$

We also have

$$\text{score}_\Omega(G) = \sum_{\ell=1}^{k} \text{score}_\Omega(G^\ell) - \sum_{r=1}^{k'} (m_r - 1) f_{\sigma_{s_r}}(\emptyset), \tag{59}$$

$$\text{score}_\Omega(G_0) = \sum_{\ell=1}^{k} \text{score}_\Omega(G_0^\ell) - \sum_{r=1}^{k'} (m_r - 1) f_{\sigma_{s_r}}(\emptyset), \tag{60}$$

where $\{s_r\}_{r=1}^{k'}$ denote the indices of the $k' < k$ separator nodes, and $m_r := |\{\ell : s_r \in C_\ell\}|$. This is because both G and G_0 have the property that separator nodes only have parents contained in a single clique, so we include an extra term $f_{\sigma_{s_r}}(\emptyset)$ from each adjacent clique not containing $\text{Pa}(s_r)$ in computing the sum. Combining Equation (60) with Equations (57) and (58), we must also have

$$\text{score}_\Omega(G_1) = \sum_{\ell=1}^{k} \text{score}_\Omega(G_1^\ell) - \sum_{r=1}^{k'} (m_r - 1) f_{\sigma_{s_r}}(\emptyset). \tag{61}$$

Together with Equation (59), this implies

$$\max_{G_1 \supseteq G_0} \{\gamma_\Omega(G, G_1)\} = \frac{\sum_{\ell=1}^{k} \left(\text{score}_\Omega(G^\ell) - \text{score}_\Omega(G_1^\ell)\right)}{|H(G, G_1)|}. \tag{62}$$

Also note that by Lemma 28 and Theorem 7, we have

$$\text{score}_\Omega(G_1^\ell) \leq \text{score}_\Omega(G^\ell), \qquad \forall \ell,$$

and by assumption,

$$\frac{\text{score}_\Omega(G^\ell) - \text{score}_\Omega(G_1^\ell)}{|H(G^\ell, G_1^\ell)|} \geq \gamma_\Omega(G_0^\ell), \qquad \forall \ell. \tag{63}$$

Finally, reindexing the cliques so that $\{C_1, \ldots, C_{k''}\}$ are the cliques such that $G^\ell \neq G_1^\ell$, we have

$$H(G, G_1) \subseteq \bigcup_{\ell=1}^{k''} H(G^\ell, G_1^\ell),$$

implying that

$$|H(G, G_1)| \leq \sum_{\ell=1}^{k''} |H(G^\ell, G_1^\ell)|. \tag{64}$$

Using the simple fact that $\frac{a_\ell}{b_\ell} \geq \xi$ for all ℓ, with $a_\ell, b_\ell > 0$, implies $\frac{\sum_\ell a_\ell}{\sum_\ell b_\ell} > \xi$, we conclude from Equation (62) and Inequalities (63) and (64) that

$$\max_{G_1 \supseteq G_0} \{\gamma_\Omega(G, G_1)\} \geq \frac{\sum_{\ell=1}^{k''} \left(\text{score}_\Omega(G^\ell) - \text{score}_\Omega(G_1^\ell)\right)}{\sum_{\ell=1}^{k''} |H(G^\ell, G_1^\ell)|} \geq \min_{1 \leq \ell \leq k} \gamma_\Omega(G_0^\ell).$$

Since this result holds uniformly over all G, we have $\gamma_\Omega(G_0) \geq \min_{1 \leq \ell \leq k} \gamma_\Omega(G_0^\ell)$, as well.

D.6 Proof of Lemma 24

This proof is quite similar to the proof for the fully-observed case, so we only mention the high-level details here.

We write

$$\begin{aligned}\sigma_j^2|\widetilde{f}_{\sigma_j}(S) - f_{\sigma_j}(S)| &= \left|\left(\widehat{\Gamma}_{jj} - \widehat{\Gamma}_{j,S}\widehat{\Gamma}_{SS}^{-1}\widehat{\Gamma}_{S,j}^{-1}\right) - \left(\Sigma_{jj} - \Sigma_{j,S}\Sigma_{SS}^{-1}\Sigma_{S,j}\right)\right| \\ &\leq \left|\widehat{\Gamma}_{jj} - \Sigma_{jj}\right| + \underbrace{\left|\widehat{\Gamma}_{j,S}\widehat{\Gamma}_{SS}^{-1}\widehat{\Gamma}_{S,j} - \Sigma_{j,S}\Sigma_{SS}^{-1}\Sigma_{S,j}\right|}_{A}. \end{aligned} \tag{65}$$

The first term may be bounded directly using Inequality (33) and a union bound over j:

$$\mathbb{P}\left(\max_j \left|\widehat{\Gamma}_{jj} - \Sigma_{jj}\right| \geq c\sigma^2\sqrt{\frac{\log p}{n}}\right) \leq c_1' \exp(-c_2' \log p). \tag{66}$$

To bound the second term, we use the following expansion:

$$\begin{aligned} A &\leq \left|\widehat{\Gamma}_{j,S}\left(\widehat{\Gamma}_{SS}^{-1} - \Sigma_{SS}^{-1}\right)\widehat{\Gamma}_{S,j}\right| + \left|\widehat{\Gamma}_{j,S}\Sigma_{SS}^{-1}\left(\widehat{\Gamma}_{S,j} - \Sigma_{S,j}\right)\right| + \left|\left(\widehat{\Gamma}_{j,S} - \Sigma_{j,S}\right)\Sigma_{SS}^{-1}\Sigma_{S,j}\right| \\ &\leq \left|\left|\left|\widehat{\Gamma}_{SS}^{-1} - \Sigma_{SS}^{-1}\right|\right|\right|_2 \|\widehat{\Gamma}_{S,j}\|_2^2 + \left|\left|\left|\Sigma_{SS}^{-1}\right|\right|\right|_2 \left(\|\widehat{\Gamma}_{S,j}\|_2\|\widehat{\Gamma}_{S,j} - \Sigma_{S,j}\|_2 + \|\widehat{\Gamma}_{S,j} - \Sigma_{S,j}\|_2\|\Sigma_{S,j}\|_2\right). \end{aligned}$$

As in the proof of Lemma 29 in Appendix E, we may obtain a bound of the form

$$\mathbb{P}\left(\left|\left|\left|\widehat{\Gamma}_{SS}^{-1} - \Sigma_{SS}^{-1}\right|\right|\right|_2 \leq c\sigma^2\left(\sqrt{\frac{d}{n}} + t\right)\right) \leq c_1 \exp(-c_2 n t^2),$$

by inverting the deviation condition (33). Furthermore,

$$\|\widehat{\Gamma}_{S,j} - \Sigma_{S,j}\|_2 \leq \left|\left|\left|\widehat{\Gamma}_{S'S'} - \Sigma_{S'S'}\right|\right|\right|_2,$$

where $S' := S \cup \{j\}$, which may in turn be bounded using the deviation condition (33). We also have

$$\|\widehat{\Gamma}_{S,j}\|_2 \leq \|\Sigma_{S,j}\|_2 + \left|\left|\left|\widehat{\Gamma}_{S'S'} - \Sigma_{S'S'}\right|\right|\right|_2.$$

Combining these results and taking a union bound over the 2^d choices for S and p choices for j, we arrive at a uniform bound of the form

$$\mathbb{P}\left(A \leq c'\sigma^2\sqrt{\frac{\log p}{n}}\right) \geq 1 - c_1' \exp(-c_2' \log p).$$

Together with Inequality (66) and the expansion (65), we then obtain the desired result.

Appendix E. Matrix Concentration Results

This Appendix contains matrix concentration results that are used to prove our technical lemmas. We use $|||\cdot|||_2$ to denote the spectral norm of a matrix.

Lemma 29 *Suppose $\{x_i\}_{i=1}^n \subseteq \mathbb{R}^p$ are i.i.d. sub-Gaussian vectors with parameter σ^2 and covariance Σ. Then for all $t \geq 0$, we have*

$$\mathbb{P}\left(\left|\left|\left|\frac{X^TX}{n} - \Sigma\right|\right|\right|_2 \leq \sigma^2 \cdot \max\{\delta, \delta^2\}\right) \geq 1 - 2\exp(-cnt^2), \tag{67}$$

where $\delta = c'\sqrt{\frac{p}{n}} + c''t$. Furthermore, if $\frac{X^TX}{n}$ is invertible and

$$\sigma^2 \left|\left|\left|\Sigma^{-1}\right|\right|\right|_2 \cdot \max\{\delta, \delta^2\} \leq \frac{1}{2},$$

we have

$$\mathbb{P}\left(\left|\left|\left|\left(\frac{X^TX}{n}\right)^{-1} - \Sigma^{-1}\right|\right|\right|_2 \leq 2\sigma^2 \left|\left|\left|\Sigma^{-1}\right|\right|\right|_2^2 \cdot \max\{\delta, \delta^2\}\right) \geq 1 - 2\exp(-cnt^2). \tag{68}$$

Proof For Inequality (67), see Remark 5.40 of Vershynin (2012). For Inequality (68), we use the matrix expansion

$$(A+\Delta)^{-1} = (A(I + A^{-1}\Delta))^{-1} = (I + A^{-1}\Delta)^{-1}A^{-1} = A^{-1} + \sum_{k=1}^{\infty}(-1)^k(A^{-1}\Delta)^kA^{-1},$$

valid for any matrices A and Δ such that A and $A+\Delta$ are both invertible and the series converges. By the triangle inequality and multiplicativity of the spectral norm, we then have

$$\begin{aligned}
\left|\left|\left|(A+\Delta)^{-1} - A^{-1}\right|\right|\right|_2 &\leq \sum_{k=1}^{\infty}\left|\left|\left|(A^{-1}\Delta)^kA^{-1}\right|\right|\right|_2 \\
&\leq \left|\left|\left|A^{-1}\right|\right|\right|_2 \cdot \sum_{k=1}^{\infty}\left|\left|\left|A^{-1}\Delta\right|\right|\right|_2^k \\
&= \frac{\left|\left|\left|A^{-1}\right|\right|\right|_2 \cdot \left|\left|\left|A^{-1}\Delta\right|\right|\right|_2}{1 - \left|\left|\left|A^{-1}\Delta\right|\right|\right|_2} \\
&\leq \frac{\left|\left|\left|A^{-1}\right|\right|\right|_2^2 \cdot \left|\left|\left|\Delta\right|\right|\right|_2}{1 - \left|\left|\left|A^{-1}\Delta\right|\right|\right|_2}.
\end{aligned}$$

We now take $A = \Sigma$ and $\Delta = \frac{X^TX}{n} - \Sigma$. By the assumption and Inequality (67), we have

$$\left|\left|\left|A^{-1}\Delta\right|\right|\right|_2 \leq \left|\left|\left|A^{-1}\right|\right|\right|_2 \cdot \left|\left|\left|\Delta\right|\right|\right|_2 \leq \frac{1}{2},$$

implying that

$$\left|\left|\left|(A+\Delta)^{-1} - A^{-1}\right|\right|\right|_2 \leq 2\left|\left|\left|A^{-1}\right|\right|\right|_2^2 \cdot \left|\left|\left|\Delta\right|\right|\right|_2.$$

This gives the result. ■

References

O. O. Aalen, K. Røysland, J. M. Gran, and B. Ledergerber. Causality, mediation and time: A dynamic viewpoint. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 175(4):831–861, 2012.

H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(12):1 – 45, 1998.

P. Bühlmann, J. Peters, and J. Ernest. CAM: Causal additive models, high-dimensional order search, and penalized regression. *Annals of Statistics*, 2014. To appear.

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

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

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, C. D. Armour, H. A. Bennett, E. Coffey, H. Dai, Y. D. He, et al. Functional discovery via a compendium of expression profiles. *Cell*, 102:109–126, 2000.

T. Kloks. *Treewidth: Computations and Approximations*, volume 842. Springer, 1994.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

J. H. Korhonen and P. Parviainen. Exact learning of bounded tree-width Bayesian networks. In *Artificial Intelligence and Statistics (AISTATS 2013)*, pages 370–378. JMLR, 2013.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

P. Loh and M. J. Wainwright. High-dimensional regression with noisy and missing data: Provable guarantees with non-convexity. *Annals of Statistics*, 40(3):1637–1664, 2012.

P. Loh and M. J. Wainwright. Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses. *Annals of Statistics*, 41(6):3022–3049, 12 2013.

N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, 34:1436–1462, 2006.

C. Nowzohour and P. Bühlmann. Score-based causal learning in additive noise models. *arXiv e-prints*, April 2014. Available at `http://arxiv.org/abs/1311.6359v2`.

S. Ordyniak and S. Szeider. Algorithms and complexity results for exact Bayesian structure learning. *CoRR*, abs/1203.3501, 2012.

E. Perrier, S. Imoto, and S. Miyano. Finding optimal Bayesian network given a super-structure. *Journal of Machine Learning Research*, 9(2):2251–2286, 2008.

J. Peters and P. Bühlmann. Identifiability of Gaussian structural equation models with equal error variances. *Biometrika*, pages 1–10, 2013.

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*, 4:935–980, 2011.

S. Shimizu, P. O. Hoyer, A. Hyvärinen, and A. Kerminen. A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7:2003–2030, 2006.

S. Shimizu, T. Inazumi, Y. Sogawa, A. Hyvärinen, Y. Kawahara, T. Washio, P. O. Hoyer, and K. Bollen. DirectLiNGAM: A direct method for learning a linear non-Gaussian structural equation model. *Journal of Machine Learning Research*, 12:1225–1248, 2011.

A. Shojaie and G. Michailidis. Penalized likelihood methods for estimation of sparse high-dimensional directed acyclic graphs. *Biometrika*, 97(3):519–538, 2010.

T. Silander and P. Myllymaki. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the Twenty-Second Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*, pages 445–452, Arlington, VA, 2006. AUAI Press.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*, volume 81. The MIT Press, 2000.

D. J. Stekhoven, I. Moraes, G. Sveinbjörnsson, L. Hennig, M. H. Maathuis, and P. Bühlmann. Causal stability ranking. *Bioinformatics*, 28(21):2819–2823, 2012.

S. van de Geer and P. Bühlmann. ℓ_0-penalized maximum likelihood for sparse directed acyclic graphs. *Annals of Statistics*, 41(2):536–567, 2013.

R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. In Y. C. Eldar and G. Kutyniok, editors, *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012.

M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

Journal of Machine Learning Research 15 (2014) 3107-3131 Submitted 1/13; Revised 5/14; Published 10/14

Recursive Teaching Dimension, VC-Dimension and Sample Compression

Thorsten Doliwa THORSTEN.DOLIWA@RUB.DE
Faculty of Mathematics
Ruhr-Universität Bochum, D-44780 Bochum, Germany

Gaojian Fan GAOJIAN@UALBERTA.CA
Department of Computing Science
University of Alberta, Edmonton, AB, T6G 2E8, Canada

Hans Ulrich Simon HANS.SIMON@RUB.DE
Horst-Görtz Institute for IT Security and Faculty of Mathematics
Ruhr-Universität Bochum, D-44780 Bochum, Germany

Sandra Zilles ZILLES@CS.UREGINA.CA
Department of Computer Science
University of Regina, Regina, SK, S4S 0A2, Canada

Editor: Manfred Warmuth

Abstract

This paper is concerned with various combinatorial parameters of classes that can be learned from a small set of examples. We show that the recursive teaching dimension, recently introduced by Zilles et al. (2008), is strongly connected to known complexity notions in machine learning, e.g., the self-directed learning complexity and the VC-dimension. To the best of our knowledge these are the first results unveiling such relations between teaching and query learning as well as between teaching and the VC-dimension. It will turn out that for many natural classes the RTD is upper-bounded by the VCD, e.g., classes of VC-dimension 1, intersection-closed classes and finite maximum classes. However, we will also show that there are certain (but rare) classes for which the recursive teaching dimension exceeds the VC-dimension. Moreover, for maximum classes, the combinatorial structure induced by the RTD, called teaching plan, is highly similar to the structure of sample compression schemes. Indeed one can transform any *repetition-free teaching plan* for a maximum class $\mathcal{C}$ into an *unlabeled sample compression scheme* for $\mathcal{C}$ and vice versa, while the latter is produced by (i) the corner-peeling algorithm of Rubinstein and Rubinstein (2012) and (ii) the tail matching algorithm of Kuzmin and Warmuth (2007).

Keywords: recursive teaching, combinatorial parameters, Vapnik-Chervonenkis dimension, upper bounds, compression schemes, tail matching algorithm

1. Introduction

In the design and analysis of machine learning algorithms, the amount of training data that needs to be provided for the learning algorithm to be successful is an aspect of central

importance. In many applications, training data is expensive or difficult to obtain, and thus input-efficient learning algorithms are desirable. In computational learning theory therefore, one way of measuring the complexity of a concept class is to determine the worst-case number of input examples required by the best valid learning algorithm. What is a valid learning algorithm depends on the underlying model of learning. We refer to this kind of complexity measure as *information complexity*. For example, in PAC-learning (Valiant, 1984), the information complexity of a concept class $\mathcal{C}$ is the worst-case sample complexity a best possible PAC learner for $\mathcal{C}$ can achieve on all concepts in $\mathcal{C}$. In query learning (Angluin, 1988), it is the best worst-case number of queries a learner would have to ask to identify an arbitrary concept in $\mathcal{C}$. In the classical model of teaching (Goldman and Kearns, 1995; Shinohara and Miyano, 1991), the information complexity of $\mathcal{C}$ is given by its teaching dimension, i.e., the largest number of labeled examples that would have to be provided for distinguishing any concept in $\mathcal{C}$ from all other concepts in $\mathcal{C}$.

Besides the practical need to limit the required amount of training data, there are a number of reasons for formally studying information complexity. Firstly, a theoretical study of information complexity yields formal guarantees concerning the amount of data that needs to be processed to solve a learning problem. Secondly, analyzing information complexity often helps to understand the structural properties of concept classes that are particularly hard to learn or particularly easy to learn. Thirdly, the theoretical study of information complexity helps to identify connections between various formal models of learning, for example if it turns out that, for a certain type of concept class, the information complexity under learning model A is in some relationship with the information complexity under model B. This third aspect is the main motivation of our study.

In the past two decades, several learning models were defined with the aim of understanding in which way a low information complexity can be achieved. One such model is learning from partial equivalence queries (Maass and Turán, 1992), which subsume all types of queries for which negative answers are witnessed by counterexamples, e.g., membership, equivalence, subset, superset, and disjointness queries (Angluin, 1988). As lower bounds on the information complexity in this query model (here called query complexity) hold for numerous other query learning models, they are particularly interesting objects of study. Even more powerful are self-directed learners (Goldman et al., 1993). Each query of a self-directed learner is a prediction of a label for an instance of the learner's choice, and the learner gets charged only for wrong predictions. The query complexity in this model lower-bounds the one obtained from partial equivalence queries (Goldman and Sloan, 1994).

Dual to the models of query learning, in which the learner actively chooses the instances it wants information on, the literature proposes models of teaching (Goldman and Kearns, 1995; Shinohara and Miyano, 1991), in which a helpful teacher selects a set of examples and presents it to the learner, again aiming at a low information complexity. A recent model of teaching with low information complexity is recursive teaching, where a teacher chooses a sample based on a sequence of nested subclasses of the underlying concept class $\mathcal{C}$ (Zilles et al., 2008). The nesting is defined as follows. The outermost "layer" consists of all concepts in $\mathcal{C}$ that are easiest to teach, i.e., that have the smallest sets of examples distinguishing them from all other concepts in $\mathcal{C}$. The next layers are formed by recursively repeating this process with the remaining concepts. The largest number of examples required for teaching at any layer is the recursive teaching dimension (RTD) of $\mathcal{C}$. The RTD substantially

reduces the information complexity bounds obtained in previous teaching models. It lower bounds not only the teaching dimension—the measure of information complexity in the "classical" teaching model (Goldman and Kearns, 1995; Shinohara and Miyano, 1991)—but also the information complexity of iterated optimal teaching (Balbach, 2008), which is often substantially smaller than the classical teaching dimension.

A combinatorial parameter of central importance in learning theory is the VC-dimension (Vapnik and Chervonenkis, 1971). Among many relevant properties, it provides bounds on the sample complexity of PAC-learning (Blumer et al., 1989). Since the VC-dimension is the best-studied quantity related to information complexity in learning, it is a natural first parameter to compare to when it comes to identifying connections between information complexity notions across various models of learning. For example, even though the self-directed learning complexity can exceed the VC-dimension, existing results show some connection between these two complexity measures (Goldman and Sloan, 1994). However, the teaching dimension, i.e., the information complexity of the classical teaching model, does not exhibit any general relationship to the VC-dimension—the two parameters can be arbitrarily far apart in either direction (Goldman and Kearns, 1995). Similarly, there is no known connection between teaching dimension and query complexity.

In this paper, we establish the first known relationships between the information complexity of teaching and query complexity, as well as between the information complexity of teaching and the VC-dimension. All these relationships are exhibited by the RTD. Two of the main contributions of this work are the following:

- We show that the RTD is never higher (and often considerably lower) than the complexity of self-directed learning. Hence all lower bounds on the RTD hold likewise for self-directed learning, for learning from partial equivalence queries, and for a variety of other query learning models.

- We reveal a strong connection between the RTD and the VC-dimension. Though there are classes for which the RTD exceeds the VC-dimension, we present a number of quite general and natural cases in which the RTD is upper-bounded by the VC-dimension. These include classes of VC-dimension 1, intersection-closed classes, a variety of naturally structured Boolean function classes, and finite maximum classes in general (i.e., classes of maximum possible cardinality for a given VC-dimension and domain size). Many natural concept classes are maximum, e.g., the class of unions of up to k intervals, for any $k \in \mathbb{N}$, or the class of simple arrangements of positive halfspaces. It remains open whether every class of VC-dimension d has an RTD linear in d.

In proving that the RTD of a finite maximum class equals its VC-dimension, we also make a third contribution:

- We reveal a relationship between the RTD and *sample compression schemes* (Littlestone and Warmuth, 1996).

Sample compression schemes are schemes for "encoding" a set of examples in a small subset of examples. For instance, from the set of examples they process, learning algorithms often extract a subset of particularly "significant" examples in order to represent their hypotheses.

This way sample bounds for PAC-learning of a class $\mathcal{C}$ can be obtained from the size of a smallest sample compression scheme for $\mathcal{C}$ (Littlestone and Warmuth, 1996; Floyd and Warmuth, 1995). Here the size of a scheme is the size of the largest subset resulting from compression of any sample consistent with some concept in $\mathcal{C}$.

The relationship between RTD and unlabeled sample compression schemes (in which the compression sets consist only of instances without labels) is established via a recent result by Rubinstein and Rubinstein (2012). They show that, for any maximum class of VC-dimension d, a technique called corner-peeling defines unlabeled compression schemes of size d. Like the RTD, corner-peeling is associated with a nesting of subclasses of the underlying concept class. A crucial observation we make in this paper is that every maximum class of VC-dimension d allows corner-peeling with an additional property, which ensures that the resulting unlabeled samples contain exactly those instances a teacher following the RTD model would use. Similarly, we show that the unlabeled compression schemes constructed by Kuzmin and Warmuth's Tail Matching algorithm (Kuzmin and Warmuth, 2007) exactly coincide with the teaching sets used in the RTD model, all of which have size at most d.

This remarkable relationship between the RTD and sample compression suggests that the open question of whether or not the RTD is linear in the VC-dimension might be related to the long-standing open question of whether or not the best possible size of sample compression schemes is linear in the VC-dimension, cf. (Littlestone and Warmuth, 1996; Floyd and Warmuth, 1995). To this end, we observe that a negative answer to the former question would have implications on potential approaches to settling the second. In particular, if the RTD is not linear in the VC-dimension, then there is no mapping that maps every concept class of VC-dimension d to a superclass that is maximum of VC-dimension $O(d)$. Constructing such a mapping would be one way of proving that the best possible size of sample compression schemes is linear in the VC-dimension.

Note that sample compression schemes are not bound to any constraints as to how the compression sets have to be formed, other than that they be subsets of the set to be compressed. In particular, any kind of agreement on, say, an order over the instance space or an order over the concept class, can be exploited for creating the smallest possible compression scheme. As opposed to that, the RTD is defined following a strict "recipe" in which teaching sets are independent of orderings of the instance space or the concept class. These differences between the models make the relationship revealed in this paper even more remarkable. Further connections between teaching and sample compression can in fact be obtained when considering a variant of the RTD introduced by Darnstädt et al. (2013). This new teaching complexity parameter upper-bounds not only the RTD and the VC-dimension, but also the smallest possible size of a sample compression scheme for the underlying concept class. Darnstädt et al. (2013) dubbed this parameter *order compression number*, as it corresponds to the smallest possible size of a special form of compression scheme called *order compression scheme* of the class.

This paper is an extension of an earlier publication (Doliwa et al., 2010).

2. Definitions, Notation and Facts

Throughout this paper, X denotes a finite set and $\mathcal{C}$ denotes a concept class over domain X. For $X' \subseteq X$, we define $\mathcal{C}_{|X'} := \{C \cap X' |\ C \in \mathcal{C}\}$. We treat concepts interchangeably

as subsets of X and as $0,1$-valued functions on X. A labeled example is a pair (x,l) with $x \in X$ and $l \in \{0,1\}$. If S is a set of labeled examples, we define $X(S) = \{x \in X \mid (x,0) \in S \text{ or } (x,1) \in S\}$. For brevity, $[n] := \{1,\ldots,n\}$. $\mathrm{VCD}(\mathcal{C})$ denotes the VC-dimension of a concept class $\mathcal{C}$.

Definition 1 *Let K be a function that assigns a "complexity" $K(\mathcal{C}) \in \mathbb{N}$ to each concept class $\mathcal{C}$. We say that K is* monotonic *if $\mathcal{C}' \subseteq \mathcal{C}$ implies that $K(\mathcal{C}') \leq K(\mathcal{C})$. We say that K is* twofold monotonic *if K is monotonic and, for every concept class $\mathcal{C}$ over X and every $X' \subseteq X$, it holds that $K(\mathcal{C}_{|X'}) \leq K(\mathcal{C})$.*

2.1 Learning Complexity

A *partial equivalence query* (Maass and Turán, 1992) of a learner is given by a function $h : X \to \{0,1,*\}$ that is passed to an oracle. The latter returns "YES" if the target concept C^* coincides with h on all $x \in X$ for which $h(x) \in \{0,1\}$; it returns a "witness of inequivalence" (i.e., an $x \in X$ such that $C^*(x) \neq h(x) \in \{0,1\}$) otherwise. $\text{LC-PARTIAL}(\mathcal{C})$ denotes the smallest number q such that there is some learning algorithm which can exactly identify any concept $C^* \in \mathcal{C}$ with up to q partial equivalence queries (regardless of the oracle's answering strategy).

A query in the model of *self-directed learning* (Goldman et al., 1993; Goldman and Sloan, 1994) consists of an instance $x \in X$ and a label $b \in \{0,1\}$, passed to an oracle. The latter returns the true label $C^*(x)$ assigned to x by the target concept C^*. We say the learner *made a mistake* if $C^*(x) \neq b$. The *self-directed learning complexity* of $\mathcal{C}$, denoted $\mathrm{SDC}(\mathcal{C})$, is defined as the smallest number q such that there is some self-directed learning algorithm which can exactly identify any concept $C^* \in \mathcal{C}$ without making more than q mistakes.

In the model of *online-learning*, the learner A makes a prediction $b_i \in \{0,1\}$ for a given instance x_i but, in contrast to self-directed learning, the sequence of instances $x_1, x_2, \ldots$ is chosen by an adversary of A that aims at maximizing A's number of mistakes. The *optimal mistake bound* for a concept class $\mathcal{C}$, denoted $M_{opt}(\mathcal{C})$, is the smallest number q such that there exists an online-learning algorithm which which can exactly identify any concept $C^* \in \mathcal{C}$ without making more than q mistakes (regardless of the ordering in which the instances are presented to A).

Clearly, LC-PARTIAL and SDC are monotonic, and M_{opt} is twofold monotonic. The following chain of inequalities is well-known (Goldman and Sloan, 1994; Maass and Turán, 1992; Littlestone, 1988):

$$\mathrm{SDC}(\mathcal{C}) \leq \text{LC-PARTIAL}(\mathcal{C}) \leq M_{opt}(\mathcal{C}) \leq \log|\mathcal{C}|\,. \tag{1}$$

2.2 Teaching Complexity

A *teaching set* for a concept $C \in \mathcal{C}$ is a set S of labeled examples such that C, but no other concept in $\mathcal{C}$, is consistent with S. Let $\mathcal{TS}(C,\mathcal{C})$ denote the family of teaching sets for

$C \in \mathcal{C}$, let $\mathrm{TS}(C;\mathcal{C})$ denote the size of the smallest teaching set for $C \in \mathcal{C}$, and let

$$\mathrm{TS}_{min}(\mathcal{C}) := \min_{C\in\mathcal{C}} \mathrm{TS}(C;\mathcal{C})\,,$$
$$\mathrm{TS}_{max}(\mathcal{C}) := \max_{C\in\mathcal{C}} \mathrm{TS}(C;\mathcal{C})\,,$$
$$\mathrm{TS}_{avg}(\mathcal{C}) := \frac{1}{|\mathcal{C}|}\sum_{C\in\mathcal{C}} \mathrm{TS}(C;\mathcal{C})\,.$$

The quantity $\mathrm{TD}(\mathcal{C}) := \mathrm{TS}_{max}(\mathcal{C})$ is called the *teaching dimension* of $\mathcal{C}$ (Goldman and Kearns, 1995). It refers to the concept in $\mathcal{C}$ that is hardest to teach. In the sequel, $\mathrm{TS}_{min}(\mathcal{C})$ is called the *best-case teaching dimension* of $\mathcal{C}$, and $\mathrm{TS}_{avg}(\mathcal{C})$ is called the *average-case teaching dimension* of $\mathcal{C}$. Obviously, $\mathrm{TS}_{min}(\mathcal{C}) \le \mathrm{TS}_{avg}(\mathcal{C}) \le \mathrm{TS}_{max}(\mathcal{C}) = \mathrm{TD}(\mathcal{C})$.

We briefly note that TD is monotonic, and that a concept class $\mathcal{C}$ consisting of exactly one concept C has teaching dimension 0 because $\emptyset \in \mathcal{TS}(C, \{C\})$.

Definition 2 (Zilles et al. 2011) *A* teaching plan for $\mathcal{C}$ *is a sequence*

$$P = ((C_1, S_1), \ldots, (C_N, S_N)) \tag{2}$$

with the following properties:

- $N = |\mathcal{C}|$ *and* $\mathcal{C} = \{C_1, \ldots, C_N\}$.
- *For all* $t = 1, \ldots, N$, $S_t \in \mathcal{TS}(C_t, \{C_t, \ldots, C_N\})$.

The quantity $\mathrm{ord}(P) := \max_{t=1,\ldots,N} |S_t|$ *is called the* order of the teaching plan P. *Finally, we define*

$$\begin{aligned} \mathrm{RTD}(\mathcal{C}) &:= \min\{\mathrm{ord}(P) \mid P \text{ is a teaching plan for } \mathcal{C}\}\,, \\ \mathrm{RTD}^*(\mathcal{C}) &:= \max_{X'\subseteq X} \mathrm{RTD}(\mathcal{C}_{|X'})\,. \end{aligned}$$

The quantity $\mathrm{RTD}(\mathcal{C})$ *is called the* recursive teaching dimension of $\mathcal{C}$.

A teaching plan (2) is said to be *repetition-free* if the sets $X(S_1), \ldots, X(S_N)$ are pairwise distinct. (Clearly, the corresponding labeled sets, $S_1, \ldots, S_N$, are always pairwise distinct.) Similar to the recursive teaching dimension we define

$$\mathrm{rfRTD}(\mathcal{C}) := \min\{\mathrm{ord}(P) \mid P \text{ is a repetition-free teaching plan for } \mathcal{C}\}\,.$$

One can show that every concept class possesses a repetition-free teaching plan. First, by induction on $|X| = m$, the full cube 2^X has a repetition-free teaching plan of order m: It results from a repetition-free plan for the $(m-1)$-dimensional subcube of concepts for which a fixed instance x is labeled 1, where each teaching set is supplemented by the example $(x, 1)$, followed by a repetition-free teaching plan for the $(m-1)$-dimensional subcube of concepts with $x = 0$. Second, "projecting" a (repetition-free) teaching plan for a concept class $\mathcal{C}$ onto the concepts in a subclass $\mathcal{C}' \subseteq \mathcal{C}$ yields a (repetition-free) teaching plan for $\mathcal{C}'$. Putting these two observations together, it follows that every class over instance set X has a repetition-free teaching plan of order $|X|$.

	x_1	x_2	x_3	x_4	x_5	TS_{min}	$\mathrm{TS}_{min}(C_i,\mathcal{C}\setminus\{C_1\})$	$\mathrm{TS}_{min}(C_i,\mathcal{C}\setminus\{C_2\})$	$\mathrm{TS}_{min}(C_i,\mathcal{C}\setminus\{C_{1/2}\})$
C_1	0	0	0	0	0	2	-	2	-
C_2	1	1	0	0	0	2	2	-	-
C_3	0	1	0	0	0	4	3	3	2
C_4	0	1	0	1	0	4	4	4	4
C_5	0	1	0	1	1	3	3	3	3
C_6	0	1	1	1	0	3	3	3	3
C_7	0	1	1	0	1	3	3	3	3
C_8	0	1	1	1	1	3	3	3	3
C_9	1	0	1	0	0	3	3	3	3
C_{10}	1	0	0	1	0	4	3	3	3
C_{11}	1	0	0	1	1	3	3	3	3
C_{12}	1	0	1	1	0	3	3	3	3
C_{13}	1	0	1	0	1	3	3	3	3

Table 1: A class with $\mathrm{RTD}(\mathcal{C}) = 2$ but $\mathrm{rfRTD}(\mathcal{C}) = 3$.

It should be noted though that $\mathrm{rfRTD}(\mathcal{C})$ may exceed $\mathrm{RTD}(\mathcal{C})$. For example, consider the class in Table 1, which is of RTD 2. In any teaching plan of order 2, both C_1 and C_2 have to be taught first with the same teaching set $\{x_1, x_2\}$ augmented by the appropriate labels. The best repetition free teaching plan for this class is of order 3.

As observed by Zilles et al. (2011), the following holds:

- RTD is monotonic.
- The recursive teaching dimension coincides with the order of any teaching plan that is *in canonical form*, i.e., a teaching plan $((C_1, S_1), \ldots, (C_N, S_N))$ such that for all $t = 1, \ldots, N$ it holds that $|S_t| = \mathrm{TS}_{min}(\{C_t, \ldots, C_N\})$.

Intuitively, a canonical teaching plan is a sequence that is recursively built by always picking an easiest-to-teach concept C_t in the class $\mathcal{C} \setminus \{C_1, \ldots, C_{t-1}\}$ together with an appropriate teaching set S_t.

The definition of teaching plans immediately yields the following result:

Lemma 3 *1. If K is monotonic and $\mathrm{TS}_{min}(\mathcal{C}) \leq K(\mathcal{C})$ for every concept class $\mathcal{C}$, then $\mathrm{RTD}(\mathcal{C}) \leq K(\mathcal{C})$ for every concept class $\mathcal{C}$.*

2. If K is twofold monotonic and $\mathrm{TS}_{min}(\mathcal{C}) \leq K(\mathcal{C})$ for every concept class $\mathcal{C}$, then $\mathrm{RTD}^*(\mathcal{C}) \leq K(\mathcal{C})$ for every concept class $\mathcal{C}$.

RTD and TS_{min} are related as follows:

Lemma 4 $\mathrm{RTD}(\mathcal{C}) = \max_{\mathcal{C}' \subseteq \mathcal{C}} \mathrm{TS}_{min}(\mathcal{C}')$.

Proof Let C_1 be the first concept in a canonical teaching plan P for $\mathcal{C}$ so that $\mathrm{TS}(C_1; \mathcal{C}) = \mathrm{TS}_{min}(\mathcal{C})$ and the order of P equals $\mathrm{RTD}(\mathcal{C})$. It follows that

$$\mathrm{RTD}(\mathcal{C}) = \max\{\mathrm{TS}(C_1; \mathcal{C}), \mathrm{RTD}(\mathcal{C} \setminus \{C_1\})\} = \max\{\mathrm{TS}_{min}(\mathcal{C}), \mathrm{RTD}(\mathcal{C} \setminus \{C_1\})\},$$

and $\mathrm{RTD}(\mathcal{C}) \leq \max_{\mathcal{C}' \subseteq \mathcal{C}} \mathrm{TS}_{min}(\mathcal{C}')$ follows inductively. As for the reverse direction, let $\mathcal{C}'_0 \subseteq \mathcal{C}$ be a maximizer of TS_{min}. Since RTD is monotonic, we get $\mathrm{RTD}(\mathcal{C}) \geq \mathrm{RTD}(\mathcal{C}'_0) \geq \mathrm{TS}_{min}(\mathcal{C}'_0) = \max_{\mathcal{C}' \subseteq \mathcal{C}} \mathrm{TS}_{min}(\mathcal{C}')$. ■

2.3 Intersection-closed Classes and Nested Differences

A concept class $\mathcal{C}$ is called *intersection-closed* if $C \cap C' \in \mathcal{C}$ for all $C, C' \in \mathcal{C}$. Among the standard examples of intersection-closed classes are the d-dimensional boxes over domain $[n]^d$:

$$\mathrm{BOX}_n^d := \{[a_1 : b_1] \times \cdots \times [a_d : b_d] \mid \ \forall i = 1, \ldots, d : \ 1 \leq a_i, b_i \leq n\}$$

Here, $[a : b]$ is an abbreviation for $\{a, a+1, \ldots, b\}$, where $[a : b]$ is the empty set if $a > b$. For the remainder of this section, $\mathcal{C}$ is assumed to be intersection-closed.

For $T \subseteq X$, we define $\langle T \rangle_{\mathcal{C}}$ as the smallest concept in $\mathcal{C}$ containing T, i.e.,

$$\langle T \rangle_{\mathcal{C}} := \bigcap_{T \subseteq C \in \mathcal{C}} C \,.$$

A *spanning set* for $T \subseteq X$ w.r.t. $\mathcal{C}$ is a set $S \subseteq T$ such that $\langle S \rangle_{\mathcal{C}} = \langle T \rangle_{\mathcal{C}}$. S is called a *minimal spanning set* w.r.t. $\mathcal{C}$ if, for every proper subset S' of S, $\langle S' \rangle_{\mathcal{C}} \neq \langle S \rangle_{\mathcal{C}}$. $I(\mathcal{C})$ denotes the size of the largest minimal spanning set w.r.t. $\mathcal{C}$. It is well-known (Natarajan, 1987; Helmbold et al., 1990) that every minimal spanning set w.r.t. $\mathcal{C}$ is shattered by $\mathcal{C}$. Thus, $I(\mathcal{C}) \leq \mathrm{VCD}(\mathcal{C})$. Note that, for every $C^\circ \in \mathcal{C}$, $I(\mathcal{C}_{|C^\circ}) \leq I(\mathcal{C})$, because every spanning set for a set $T \subseteq C^\circ$ w.r.t. $\mathcal{C}$ is also a spanning set for T w.r.t. $\mathcal{C}_{|C^\circ}$.

The class of *nested differences* of depth d (at most d) with concepts from $\mathcal{C}$, denoted $\mathrm{DIFF}^d(\mathcal{C})$ ($\mathrm{DIFF}^{\leq d}(\mathcal{C})$, resp.), is defined inductively as follows:

$$\begin{aligned} \mathrm{DIFF}^1(\mathcal{C}) &:= \mathcal{C} \,, \\ \mathrm{DIFF}^d(\mathcal{C}) &:= \{C \setminus D \mid C \in \mathcal{C}, D \in \mathrm{DIFF}^{d-1}(\mathcal{C})\} \,, \\ \mathrm{DIFF}^{\leq d}(\mathcal{C}) &:= \bigcup_{i=1}^{d} \mathrm{DIFF}^i(\mathcal{C}) \,. \end{aligned}$$

Expanding the recursive definition of $\mathrm{DIFF}^d(\mathcal{C})$ shows that, e.g., a set in $\mathrm{DIFF}^4(\mathcal{C})$ has the form $C_1 \setminus (C_2 \setminus (C_3 \setminus C_4))$ where $C_1, C_2, C_3, C_4 \in \mathcal{C}$. We may assume without loss of generality that $C_1 \supseteq C_2 \supseteq \cdots$ because $\mathcal{C}$ is intersection-closed.

Nested differences of intersection-closed classes were studied in depth at the early stages of research in computational learning theory (Helmbold et al., 1990).

2.4 Maximum Classes and Unlabeled Compression Schemes

Let $\Phi_d(n) := \sum_{i=0}^{d} \binom{n}{i}$. For $d = \mathrm{VCD}(\mathcal{C})$ and for any subset X' of X, we have $\left|\mathcal{C}_{|X'}\right| \leq \Phi_d(|X'|)$, according to Sauer's Lemma (Vapnik and Chervonenkis, 1971; Sauer, 1972). The concept class $\mathcal{C}$ is called a *maximum class* if Sauer's inequality holds with equality for every subset X' of X. It is well-known (Welzl, 1987; Floyd and Warmuth, 1995) that a class over a finite domain X is maximum iff Sauer's inequality holds with equality for $X' = X$.

The following definition was introduced by Kuzmin and Warmuth (2007):

Definition 5 *An* unlabeled compression scheme for a maximum class of VC-dimension d *is given by an injective mapping r that assigns to every concept C a set $r(C) \subseteq X$ of size at most d such that the following condition is satisfied:*

$$\forall C, C' \in \mathcal{C}\ (C \neq C'), \exists x \in r(C) \cup r(C') :\ C(x) \neq C'(x)\,. \tag{3}$$

(3) is referred to as the *non-clashing property*. In order to ease notation, we add the following technical definitions. A *representation mapping of order k for a (not necessarily maximum) class $\mathcal{C}$* is any injective mapping r that assigns to every concept C a set $r(C) \subseteq X$ of size at most k such that (3) holds. A representation-mapping r is said to have the *acyclic non-clashing property* if there is an ordering $C_1, \ldots, C_N$ of the concepts in $\mathcal{C}$ such that

$$\forall 1 \leq i < j \leq N, \exists x \in r(C_i) :\ C_i(x) \neq C_j(x)\,. \tag{4}$$

Considering maximum classes, it was shown by Kuzmin and Warmuth (2007) that a representation mapping with the non-clashing property guarantees that, for every sample S labeled according to a concept in $\mathcal{C}$, there is exactly one concept $C \in \mathcal{C}$ that is consistent with S and satisfies $r(C) \subseteq X(S)$. This allows to encode (compress) a labeled sample S by $r(C)$ and, since r is injective, to decode (decompress) $r(C)$ by C (so that the labels in S can be reconstructed). This coined the term "unlabeled compression scheme".

A concept class $\mathcal{C}$ over a domain X of size n is identified with a subset of $\{0,1\}^n$. The *one-inclusion-graph* $\mathcal{G}(\mathcal{C})$ associated with $\mathcal{C}$ is defined as follows:

- The nodes are the concepts from $\mathcal{C}$.
- Two concepts are connected by an edge if and only if they differ in exactly one coordinate (when viewed as nodes in the Boolean cube).

A *cube* $\mathcal{C}'$ in $\mathcal{C}$ is a subcube of $\{0,1\}^n$ such that every node in $\mathcal{C}'$ represents a concept from $\mathcal{C}$. In the context of the one-inclusion graph, the instances (corresponding to the dimensions in the Boolean cube) are usually called "colors" (and an edge along dimension i is said to have color i). For a concept $C \in \mathcal{C}$, $I(C; \mathcal{G}(\mathcal{C}))$ denotes the union of the instances associated with the colors of the incident edges of C in $\mathcal{G}(\mathcal{C})$, called *incident instances* of C. Recall that the *density* of a graph with m edges and n nodes is defined as m/n. As shown by Haussler et al. (1994, Lemma 2.4), the density of the 1-inclusion graph lower-bounds the VC-dimension, i.e., $\mathrm{dens}(\mathcal{G}(\mathcal{C})) < \mathrm{VCD}(\mathcal{C})$.

The following definitions were introduced by Rubinstein and Rubinstein (2012); we reformulate the notation in order to stress the similarities to the definition of teaching plans.

Definition 6 *A* corner-peeling plan for $\mathcal{C}$ *is a sequence*

$$P = ((C_1, \mathcal{C}'_1), \ldots, (C_N, \mathcal{C}'_N)) \tag{5}$$

with the following properties:

1. *$N = |\mathcal{C}|$ and $\mathcal{C} = \{C_1, \ldots, C_N\}$.*
2. *For all $t = 1, \ldots, N$, $\mathcal{C}'_t$ is a cube in $\{C_t, \ldots, C_N\}$ which contains C_t and all its neighbors in $\mathcal{G}(\{C_t, \ldots, C_N\})$. (Note that this uniquely specifies $\mathcal{C}'_t$.)*

The nodes C_t are called the corners of the cubes $\mathcal{C}'_t$, *respectively. The dimension of the largest cube among $\mathcal{C}'_1, \ldots, \mathcal{C}'_N$ is called the* order of the corner-peeling plan P. $\mathcal{C}$ can be d-corner-peeled *if there exists a corner-peeling plan of order d.*

A concept class $\mathcal{C}$ is called *shortest-path closed* if, for every pair of distinct concepts $C, C' \in \mathcal{C}$, $\mathcal{G}(\mathcal{C})$ contains a path of length $|C \triangle C'|$ (known as the Hamming distance) that connects C and C', where $\triangle$ denotes the symmetric difference. Note that every maximum class is shortest-path closed, but not vice versa. Rubinstein and Rubinstein (2012) showed the following:

1. If a maximum class $\mathcal{C}$ has a corner-peeling plan (5) of order $\mathrm{VCD}(\mathcal{C})$, then an unlabeled compression scheme for $\mathcal{C}$ is obtained by defining $r(C_t)$ to be the set of colors in cube $\mathcal{C}'_t$ for $t = 1, \ldots, N$.

2. Every maximum class $\mathcal{C}$ can be $\mathrm{VCD}(\mathcal{C})$-corner-peeled.

Although it had previously been proved (Kuzmin and Warmuth, 2007) that any maximum class of VC-dimension d has an unlabeled compression scheme of size d, the corner-peeling technique still provides very useful insights. We will see an application in Section 4.3, where we show that $\mathrm{RTD}(\mathcal{C}) = \mathrm{VCD}(\mathcal{C})$ for every maximum class $\mathcal{C}$.

3. Recursive Teaching Dimension and Query Learning

Kuhlmann proved the following result:

Lemma 7 (Kuhlmann 1999) *For every concept class $\mathcal{C}$:* $\mathrm{TS}_{min}(\mathcal{C}) \leq \mathrm{SDC}(\mathcal{C})$.

In view of (1), the monotonicity of LC-PARTIAL and SDC, the twofold monotonicity of M_{opt}, and in view of Lemma 3, we obtain:

Corollary 8 *For every concept class $\mathcal{C}$, the following holds:*

1. $\mathrm{RTD}(\mathcal{C}) \leq \mathrm{SDC}(\mathcal{C}) \leq \mathit{LC\text{-}PARTIAL}(\mathcal{C}) \leq M_{opt}(\mathcal{C})$.

2. $\mathrm{RTD}^*(\mathcal{C}) \leq M_{opt}(\mathcal{C})$.

As demonstrated by Goldman and Sloan (1994), the model of self-directed learning is extremely powerful. According to Corollary 8, recursive teaching is an even more powerful model so that upper bounds on SDC apply to RTD as well, and lower bounds on RTD apply to SDC and LC-PARTIAL as well. The following result, which is partially known from the work by Goldman and Sloan (1994) and Zilles et al. (2011), illustrates this:

Corollary 9

1. *If* $\mathrm{VCD}(\mathcal{C}) = 1$, *then* $\mathrm{RTD}(\mathcal{C}) = \mathrm{SDC}(\mathcal{C}) = 1$.

2. $\mathrm{RTD}(\mathit{Monotone\ Monomials}) = \mathrm{SDC}(\mathit{Monotone\ Monomials}) = 1$.

3. $\mathrm{RTD}(\mathit{Monomials}) = \mathrm{SDC}(\mathit{Monomials}) = 2$.

4. $\mathrm{RTD}(\mathrm{BOX}_n^d) = \mathrm{SDC}(\mathrm{BOX}_n^d) = 2$.

5. $\mathrm{RTD}(m\text{-}\mathit{Term\ Monotone\ DNF}) \leq \mathrm{SDC}(m\text{-}\mathit{Term\ Monotone\ DNF}) \leq m$.

6. SDC(m-*Term Monotone DNF*) $\geq$ RTD(m-*Term Monotone DNF*) $\geq m$ *provided that the number of Boolean variables is at least* $m^2 + 1$.

Proof All upper bounds on SDC were proved by Goldman and Sloan (1994) and, as mentioned above, they apply to RTD as well. The lower bound 1 on RTD (for concept classes with at most two distinct concepts) is trivial. RTD(Monomials) = 2 was shown by Zilles et al. (2011). As a lower bound, this carries over to BOX_n^d which contains Monomials as a subclass. Thus the first five assertions are obvious from known results in combination with Corollary 8.

As for the last assertion, we have to show that RTD(m-Term Monotone DNF) $\geq m$. To this end assume that there are $n \geq m^2 + 1$ Boolean variables. According to Lemma 4, it suffices to find a subclass $\mathcal{C}'$ of m-Term Monotone DNF such that $\mathrm{TS}_{min}(\mathcal{C}') \geq m$. Let $\mathcal{C}'$ be the class of all DNF formulas that contain precisely m pairwise variable-disjoint terms of length m each. Let F be an arbitrary but fixed formula in $\mathcal{C}'$. Without loss of generality, the teacher always picks either a minimal positive example (such that flipping any 1-bit to 0 turns it negative) or a maximal negative example (such that flipping any 0-bit to 1 turns it positive). By construction of $\mathcal{C}'$, the former example has precisely m ones (and reveals one of the m terms in F) and the latter example has precisely m zeroes (and reveals one variable in each term). We may assume that the teacher consistently uses a numbering of the m terms from 1 to m and augments any 0-component (component i say) of a negative example by the number of the term that contains the corresponding Boolean variable (the term containing variable x_i). Since adding information is to the advantage of the learner, this will not corrupt the lower-bound argument. We can measure the knowledge that is still missing after having seen a collection of labeled instances by the following parameters:

- m', the number of still unknown terms
- $l_1, \ldots, l_m$, where l_k is the number of still unknown variables in term k

The effect of a teaching set on these parameters is as follows: a positive example decrements m', and a negative example decrements some of $l_1, \ldots, l_m$. Note that n was chosen sufficiently large[1] so that the formula F is not uniquely specified as long as none of the parameters has reached level 0. Since all parameters are initially of value m, the size of any teaching set for F must be at least m. ■

In powerful learning models, techniques for proving lower bounds become an issue. One technique for proving a lower bound on RTD was applied already in the proof of Corollary 9: select a subclass $\mathcal{C}' \subseteq \mathcal{C}$ and derive a lower bound on $\mathrm{TS}_{min}(\mathcal{C}')$. We now turn to the question whether known lower bounds for LC-PARTIAL or SDC remain valid for RTD. Maass and Turán (1992) showed that LC-PARTIAL is lower-bounded by the logarithm of the length of a longest inclusion chain in $\mathcal{C}$. This bound does not even apply to SDC, which follows from an inspection of the class of half-intervals over domain $[n]$. The longest inclusion chain in this class, $\emptyset \subset \{1\} \subset \{1,2\} \subset \cdots \subset \{1,2,\ldots,n\}$, has length $n+1$, but its self-directed learning complexity is 1. Theorem 8 in the paper by Ben-David and Eiron (1998) implies

1. A slightly refined argument shows that requiring $n \geq (m-1)^2 + 1$ would be sufficient. But we made no serious attempt to make this assumption as weak as possible.

that SDC is lower-bounded by $\log|\mathcal{C}|/\log|X|$ if $\mathrm{SDC}(\mathcal{C}) \geq 2$. We next show that the same bound applies to RTD:

Lemma 10 *Suppose* $\mathrm{RTD}(\mathcal{C}) \geq 2$. *Then,* $\mathrm{RTD}(\mathcal{C}) \geq \frac{\log|\mathcal{C}|}{\log|X|}$.

Proof Samei et al. (2012) have shown that Sauer's bound holds with $\mathrm{RTD}(\mathcal{C})$ in the role of $\mathrm{VCD}(\mathcal{C})$, i.e., for $k = \mathrm{RTD}(\mathcal{C})$, the following holds:

$$|\mathcal{C}| \leq \sum_{i=1}^{k} \binom{|X|}{i} = \Phi_k(|X|) \leq |X|^k$$

Solving for k yields the desired lower bound on $\mathrm{RTD}(\mathcal{C})$. ■

A subset $X' \subseteq X$ is called $\mathcal{C}$-*distinguishing* if, for each pair of distinct concepts $C, C' \in \mathcal{C}$, there is some $x \in X'$ such that $C(x) \neq C'(x)$. The matrix associated with a concept class $\mathcal{C}$ over domain X is given by $M(x, C) = C(x) \in \{0, 1\}$. We call two concept classes $\mathcal{C}, \mathcal{C}'$ equivalent if their matrices are equal up to permutation of rows or columns, and up to flipping all bits of a subset of the rows.[2] The following result characterizes the classes of recursive teaching dimension 1:

Theorem 11 *The following statements are equivalent:*

1. $\mathrm{SDC}(\mathcal{C}) = 1$.
2. $\mathrm{RTD}(\mathcal{C}) = 1$.
3. *There exists a* $\mathcal{C}$*-distinguishing set* $X' \subseteq X$ *such that* $\mathcal{C}_{|X'}$ *is equivalent to a concept class whose matrix* M *is of the form* $M = [M'|\vec{0}]$ *where* M' *is a lower-triangular square-matrix with ones on the main-diagonal and* $\vec{0}$ *denotes the all-zeroes vector.*

Proof *1 implies 2.* If $\mathrm{SDC}(\mathcal{C}) = 1$, $\mathcal{C}$ contains at least two distinct concepts. Thus, $\mathrm{RTD}(\mathcal{C}) \geq 1$. According to Corollary 8, $\mathrm{RTD}(\mathcal{C}) \leq \mathrm{SDC}(\mathcal{C}) = 1$.

2 implies 3. Let P be a teaching plan of order 1 for $\mathcal{C}$, and let X' be the set of instances occurring in P (which clearly is $\mathcal{C}$-distinguishing). Let $(C_1, \{(x_1, b_1)\})$ be the first item of P. Let M be the matrix associated with $\mathcal{C}$ (up to equivalence). We make C_1 the first column and x_1 the first row of M. We may assume that $b_1 = 1$. (Otherwise flip all bits in row 1.) Since $\{(x_1, 1)\}$ is a teaching set for C_1, the first row of M is of the form $(1, 0, \ldots, 0)$. We may repeat this argument for every item in P so that the resulting matrix M is of the desired form. (The last zero-column represents the final concept in P with the empty teaching set.)

3 implies 1. Since X' is $\mathcal{C}$-distinguishing, exact identification of a concept $C \in \mathcal{C}$ is the same as exact identification of C restricted to X'. Let $x_1, \ldots, x_{N-1}$ denote the instances corresponding to the rows of M. Let $C_1, \ldots, C_N$ denote the concepts corresponding to the columns of M. A self-directed learner passes $(x_1, 0), (x_2, 0), \ldots$ to the oracle until it makes the first mistake (if any). If the first mistake (if any) happens for $(x_k, 0)$, the target concept must be C_k (because of the form of M). If no mistake has occurred on items

2. Reasonable complexity measures (including RTD, SDC, VCD) are invariant under these operations.

$(x_1, 0), \ldots, (x_{N-1}, 0)$, there is only one possible target concept left, namely C_N. Thus the self-directed learner exactly identifies the target concept at the expense of at most one mistake. ■

As we have seen in this section, the gap between $\mathrm{SDC}(\mathcal{C})$ and $\text{LC-PARTIAL}(\mathcal{C})$ can be arbitrarily large (e.g., the class of half-intervals over domain $[n]$). We will see below, that a similar statement applies to $\mathrm{RTD}(\mathcal{C})$ and $\mathrm{SDC}(\mathcal{C})$ (despite the fact that both measures assign value 1 to the same family of concept classes).

4. Recursive Teaching Dimension and VC-Dimension

The main open question that we pursue in this section is whether there is a universal constant k such that, for all concept classes $\mathcal{C}$, $\mathrm{RTD}(\mathcal{C}) \leq k \cdot \mathrm{VCD}(\mathcal{C})$. Clearly, $\mathrm{TS}_{min}(\mathcal{C}) \leq \mathrm{RTD}(\mathcal{C}) \leq \mathrm{RTD}^*(\mathcal{C})$, so that the implications from left to right in

$$\begin{aligned} \forall \mathcal{C} : \mathrm{RTD}^*(\mathcal{C}) \leq k \cdot \mathrm{VCD}(\mathcal{C}) \quad &\Leftrightarrow \quad \forall \mathcal{C} : \mathrm{RTD}(\mathcal{C}) \leq k \cdot \mathrm{VCD}(\mathcal{C}) \\ &\Leftrightarrow \quad \forall \mathcal{C} : \mathrm{TS}_{min}(\mathcal{C}) \leq k \cdot \mathrm{VCD}(\mathcal{C}) \end{aligned} \tag{6}$$

are obvious. But the implications from right to left hold as well as can be seen from the following calculations based on the assumption that $\mathrm{TS}_{min}(\cdot) \leq k \cdot \mathrm{VCD}(\cdot)$:

$$\mathrm{RTD}^*(\mathcal{C}) = \max_{X' \subseteq X} \max_{\mathcal{C}' \subseteq \mathcal{C}} \mathrm{TS}_{min}(\mathcal{C}'_{|X'}) \leq k \cdot \max_{X' \subseteq X} \max_{\mathcal{C}' \subseteq \mathcal{C}} \mathrm{VCD}(\mathcal{C}'_{|X'}) \leq k \cdot \mathrm{VCD}(\mathcal{C})$$

Here, the first equation expands the definition of RTD^* and applies Lemma 4. The final inequality makes use of the fact that VCD is twofold monotonic. As a consequence, the question of whether $\mathrm{RTD}(\cdot) \leq k \cdot \mathrm{VCD}(\cdot)$ for a universal constant k remains equivalent if RTD is replaced by TS_{min} or RTD^*.

4.1 Classes with RTD Exceeding VCD

In general the recursive teaching dimension can exceed the VC-dimension. Kuhlmann (1999) presents a family $(\mathcal{C}_m)_{m \geq 1}$ of concept classes for which $\mathrm{VCD}(\mathcal{C}_m) = 2m$ but $\mathrm{RTD}(\mathcal{C}_m) \geq \mathrm{TS}_{min}(\mathcal{C}_m) = 3m$. The smallest class in Kuhlmann's family, $\mathcal{C}_1$, consists of 24 concepts over a domain of size 16.

A smaller class $\mathcal{C}_W$ with $\mathrm{RTD}(\mathcal{C}_W) = \mathrm{TS}_{min}(\mathcal{C}_W) = 3$ and $\mathrm{VCD}(\mathcal{C}_W) = 2$ was communicated to us by Manfred Warmuth. It is shown in Figure 1.

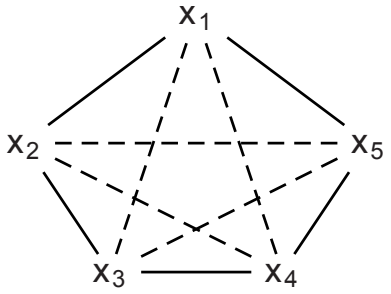
Brute-force enumeration shows that $\mathrm{RTD}(\mathcal{C}_W) = \mathrm{TS}_{min}(\mathcal{C}_W) = 3$ and $\mathrm{VCD}(\mathcal{C}_W) = 2$. Warmuth's class $\mathcal{C}_W$ is remarkable in the sense that it is the smallest concept class for which RTD exceeds VCD. In order to prove this, the following lemmas will be helpful.

Lemma 12 $\mathrm{RTD}(\mathcal{C}) \leq |X| - 1$ *unless* $\mathcal{C} = 2^X$.

Proof If $\mathcal{C} \neq 2^X$, then $\mathcal{C}$ must contain a concept C such that $C \triangle \{x\} \notin \mathcal{C}$ for some instance $x \in X$. Then, C can be uniquely identified within $\mathcal{C}$ using the instances from $X \setminus \{x\}$ and the corresponding labels. Iterative application of this argument leads to a teaching plan for $\mathcal{C}$ of order at most $|X| - 1$. ■

x_1	x_2	x_3	x_4	x_5
1	1	0	0	0
0	1	1	0	0
0	0	1	1	0
0	0	0	1	1
1	0	0	0	1
0	1	0	1	1
0	1	1	0	1
1	0	1	1	0
1	0	1	0	1
1	1	0	1	0

(a)



(b)

Figure 1: The smallest concept class $\mathcal{C}_W$ with $\mathrm{RTD}(\mathcal{C}_W) > \mathrm{VCD}(\mathcal{C}_W)$. The function table to the left can be extracted from the graph to the right by picking concept $\{x_i, x_j\}$ for every solid line and $\mathcal{X} \setminus \{x_i, x_j\}$ for every dashed line.

Note that Lemma 12 transfers to rfRTD, using an argument very similar to the one that implies the existence of a repetition-free teaching plan for every class (see the discussion just below Definition 2.)

For $x \in X$ and $\ell \in \{0,1\}$, $\mathcal{C}[x,\ell]$ is defined as the following subclass of $\mathcal{C}$:

$$\mathcal{C}[x,\ell] = \{C \in \mathcal{C} : \ C(x) = \ell\}$$

An instance x is called *redundant* if $\mathcal{C}[x,\ell] = \emptyset$ for some $\ell \in \{0,1\}$. Note that the label of a redundant instance does not contain any information about the underlying target concept from $\mathcal{C}$. With this notation, the following holds:

Lemma 13 *Let $\mathcal{C}$ be a concept class over domain X such that $\mathrm{TS}_{min}(\mathcal{C}) \geq 3$, and X does not contain redundant instances. Then, $\mathrm{VCD}(\mathcal{C}[x,\ell]) \geq 2$ for all $x \in X$ and $\ell \in \{0,1\}$.*

Proof By way of contradiction. Assume that $\mathrm{VCD}(\mathcal{C}[x,\ell]) \leq 1$ for some choice of x and ℓ. We will show that $\mathrm{TS}_{min}(\mathcal{C}) \leq 2$. According to Corollary 9, $\mathrm{VCD}(\mathcal{C}[x,\ell]) \leq 1$ implies that $\mathrm{TS}_{min}(\mathcal{C}[x,\ell]) \leq \mathrm{RTD}(\mathcal{C}[x,\ell]) \leq 1$. Now it can be seen that $\mathrm{TS}_{min}(\mathcal{C}) \leq 2$: choose (x,ℓ) as the first element in a teaching set and proceed with a teaching set of size $\mathrm{VCD}(\mathcal{C}[x,\ell]) \leq 1$ for the (non-empty) subclass $\mathcal{C}[x,\ell]$. ■

Lemma 14 *Let $\mathcal{C}$ be a concept class over domain X such that $\mathrm{VCD}(\mathcal{C}) = 2$, $\mathrm{TS}_{min}(\mathcal{C}) = 3$, and X does not contain redundant instances. Then $|X| \geq 5$ and, for all $x \in X$ and $\ell \in \{0,1\}$, $|C[x,\ell]| \geq 5$.*

Proof Let $x \in X$ and $\ell \in \{0,1\}$ be arbitrary but fixed. We first show that $|C[x,\ell]| \geq 5$. According to Lemma 13, $\mathrm{VCD}(\mathcal{C}[x,\ell]) \geq 2$. Since $\mathrm{VCD}(\mathcal{C}) = 2$, this implies that

$\mathrm{VCD}(\mathcal{C}[x,\ell]) = 2$. Let $C_1, C_2, C_3, C_4 \in \mathcal{C}[x,\ell]$ be concepts that shatter two points x', x'' in $X \setminus \{x\}$. For at least one of these four concepts, say for C_1, the neighboring concept $C_1 \triangle \{x\}$ does not belong to $\mathcal{C}$ (because otherwise the VC-dimension of $\mathcal{C}$ would be at least 3). If $C_1, \ldots, C_4$ were the only concepts in $\mathcal{C}[x,\ell]$, then $(x', C_1(x'))$ and $(x'', C_1(x''))$ would form a teaching set for C_1 in contradiction to $\mathrm{TS}_{min}(\mathcal{C}) = 3$. We conclude that C_1, C_2, C_3, C_4 are not the only concepts in $C[x,\ell]$ so that $|C[x,\ell]| \geq 5$.

We still have to show that $|X| \geq 5$. Clearly, $|X| \geq \mathrm{TS}_{min}(\mathcal{C}) = 3$. Let us assume by way of contradiction that $|X| = 4$, say $X = \{x, y, z, u\}$. We write concepts over X as 4-tuples $(C(x), C(y), C(z), C(u))$. The following considerations are illustrated in Figure 2. From Lemma 13 and from the assumption $\mathrm{VCD}(\mathcal{C}) = 2$, we may conclude that $\mathrm{VCD}(\mathcal{C}[u,0]) = 2 = \mathrm{VCD}(\mathcal{C}[u,1])$. The set of size 2 shattered by $\mathcal{C}[u,0]$ cannot coincide with the set of size 2 shattered by $\mathcal{C}[u,1]$ because, otherwise, the VC-dimension of $\mathcal{C}$ would be at least 3. Let's say, $\mathcal{C}[u,0]$ shatters $\{x,y\}$ but not $\{x,z\}$ and $\mathcal{C}[u,1]$ shatters $\{x,z\}$ but not $\{x,y\}$. By symmetry, we may assume that $\mathcal{C}[u,1]$ does not contain a concept that assigns label 1 to both x and y, i.e., the concepts $(1,1,0,1)$ and $(1,1,1,1)$ are missing in $\mathcal{C}[u,1]$. Since $\{x,z\}$ is shattered, $\mathcal{C}[u,1]$ must contain the concepts $(1,0,0,1)$ and $(1,0,1,1)$ so as to realize the label assignments $(1,0),(1,1)$ for (x,z). Recall from the first part of the proof that $|\mathcal{C}[u,\ell]| \geq 5$ for $\ell = 0,1$. Note that $|\mathcal{C}[u,\ell]| = 6$ would imply that $\{y,z\}$ is also shattered by $\mathcal{C}[u,\ell]$. Since $\mathrm{VCD}(\mathcal{C}) = 2$, this cannot occur for both subclasses $\mathcal{C}[u,1]$ and $\mathcal{C}[u,0]$ simultaneously. By symmetry, we may assume that $|\mathcal{C}[u,1]| = 5$. Thus, besides $(1,1,0,1)$ and $(1,1,1,1)$, exactly one more concept is missing in $\mathcal{C}[u,1]$. We proceed by case analysis:

Case 1: The additional missing concept in $\mathcal{C}[u,1]$, say C', has Hamming-distance 1 from one of $(1,1,0,1)$ and $(1,1,1,1)$. For reasons of symmetry, we may assume that $C' = (0,1,1,1)$. It follows that the concept $(0,1,0,1)$ belongs to $\mathcal{C}[u,1]$ and has the teaching set $\{(u,1),(y,1)\}$. This is a contradiction to $\mathrm{TS}_{min}(\mathcal{C}) = 3$.

Case 2: The additional missing concept in $\mathcal{C}[u,1]$ has Hamming-distance 2 from both of $(1,1,0,1)$ and $(1,1,1,1)$. Then $\mathcal{C}[u,1]$ contains $(0,1,1,1)$, $(0,1,0,1)$, $(1,0,1,1)$, and $(1,0,0,1)$. In particular, $\mathcal{C}[u,1]$ shatters $\{y,z\}$. In this case, it cannot happen that $\{y,z\}$ is shattered by $\mathcal{C}[u,0]$ too. Thus, $|\mathcal{C}[u,0]| = 5$. We may now expose $\mathcal{C}[u,0]$ to the same case analysis that we already applied to $\mathcal{C}[u,1]$. Since $\mathcal{C}[u,0]$ does not shatter $\{y,z\}$, Case 2 is excluded. As described above, Case 1 leads to a contradiction.

■

We are now ready to prove the minimality of Warmuth's class:

Theorem 15 *Let $\mathcal{C}$ be a concept class over domain X such that $\mathrm{RTD}(\mathcal{C}) > \mathrm{VCD}(\mathcal{C})$. Then $|\mathcal{C}| \geq 10$ and $|X| \geq 5$.*

Proof Obviously $\mathrm{VCD}(\mathcal{C}) = 0$ implies that $\mathrm{RTD}(\mathcal{C}) = 0$. According to Corollary 9, $\mathrm{VCD}(\mathcal{C}) = 1$ implies that $\mathrm{RTD}(\mathcal{C}) = 1$. So we may safely assume that $\mathrm{VCD}(\mathcal{C}) \geq 2$ and $\mathrm{RTD}(\mathcal{C}) \geq 3$. According to Lemma 4, we may assume that $\mathrm{RTD}(\mathcal{C}) = \mathrm{TS}_{min}(\mathcal{C})$ because, otherwise, our proof could proceed with the class $\mathcal{C}' \subseteq \mathcal{C}$ such that $\mathrm{RTD}(\mathcal{C}') = \mathrm{TS}_{min}(\mathcal{C}')$. We may furthermore assume that $\mathcal{C}[x,\ell] \neq \emptyset$ for all $x \in X$ and $\ell \in \{0,1\}$ because, otherwise, x is a redundant instance and the proof could proceed with the subdomain $X \setminus \{x\}$. We may

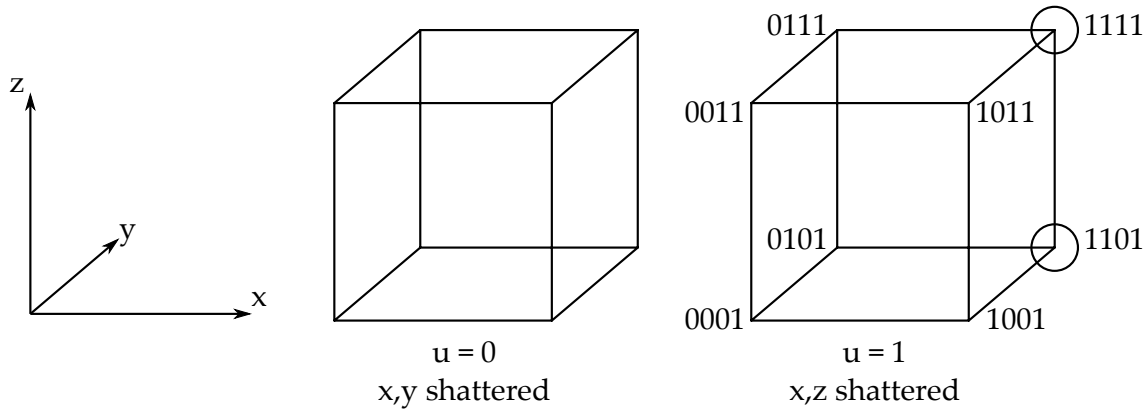


Figure 2: As indicated by circles, the concepts 1101 and 1111 are missing in $\mathcal{C}[u,1]$. There is exactly one additional concept C' which is missing. If $C' \in \{0101, 0111, 1001, 1011\}$, then C' has a teaching set of size 2. Otherwise, $\mathcal{C}[u,1]$ shatters y, z.

therefore apply Lemma 13 and conclude that $\mathrm{VCD}(\mathcal{C}[x,\ell]) \geq 2$ for all $x \in X$ and $\ell \in \{0,1\}$. Clearly $|X| \geq \mathrm{RTD}(\mathcal{C}) \geq 3$. We claim that $|X| \geq 5$, which can be seen as follows. First, note that $\mathcal{C} \neq 2^X$, because $\mathrm{RTD}(\mathcal{C}) > \mathrm{VCD}(\mathcal{C})$. Thus $\mathrm{RTD}(\mathcal{C}) \leq |X| - 1$ by Lemma 12 so that $|X| \geq \mathrm{RTD}(\mathcal{C}) + 1 \geq 4$. Assume $|X| = 4$ by way of contradiction. It follows that $\mathrm{RTD}(\mathcal{C}) \leq 3$ and $\mathrm{VCD}(\mathcal{C}) \leq 2$. Thus, $\mathrm{RTD}(\mathcal{C}) = 3$ and $\mathrm{VCD}(\mathcal{C}) = 2$. But then $|X| \geq 5$ by Lemma 14. Having established $|X| \geq 5$, it remains to prove that $|\mathcal{C}| \geq 10$. According to (1), $\mathrm{RTD}(\mathcal{C}) \leq \log|\mathcal{C}|$. $\mathrm{RTD}(\mathcal{C}) \geq 4$ would imply that $|\mathcal{C}| \geq 16 > 10$. We may therefore focus on the case $\mathrm{RTD}(\mathcal{C}) = 3$, which implies that $\mathrm{VCD}(\mathcal{C}) = 2$. But now it is immediate from Lemma 14 that $|\mathcal{C}| \geq 10$, as desired. ■

We close this section by showing that $\mathrm{RTD}(\mathcal{C}) - \mathrm{VCD}(\mathcal{C})$ can become arbitrarily large. This can be shown by a class whose concepts are disjoint unions of concepts taken from Warmuth's class $\mathcal{C}_W$. Details follow. Suppose that $\mathcal{C}_1$ and $\mathcal{C}_2$ are concept classes over domains X_1 and X_2, respectively, such that $X_1 \cap X_2 = \emptyset$. Then

$$\mathcal{C}_1 \uplus \mathcal{C}_2 := \{A \cup B |\ A \in \mathcal{C}_1, B \in \mathcal{C}_2\}\ .$$

We apply the same operation to arbitrary pairs of concept classes with the understanding that, after renaming instances if necessary, the underlying domains are disjoint. We claim that VCD, TS_{min} and RTD behave additively with respect to "$\uplus$", i.e., the following holds:

Lemma 16 *For all* $K \in \{\mathrm{VCD}, \mathrm{TS}_{min}, \mathrm{RTD}\}$*:* $K(\mathcal{C}_1 \uplus \mathcal{C}_2) = K(\mathcal{C}_1) + K(\mathcal{C}_2)$.

Proof The lemma is fairly obvious for $K = \mathrm{VCD}$ and $K = \mathrm{TS}_{min}$. Suppose that we have an optimal teaching plan that teaches the concepts from $\mathcal{C}_1$ in the order $A_1, \ldots, A_M$ (resp. the concepts from $\mathcal{C}_2$ in the order $B_1, \ldots, B_N$). Then, the teaching plan that proceeds in rounds and teaches $A_i \cup B_1, \ldots, A_i \cup B_N$ in round $i \in [M]$ witnesses that $\mathrm{RTD}(\mathcal{C}_1 \uplus \mathcal{C}_2) \leq$

$\mathrm{RTD}(\mathcal{C}_1) + \mathrm{RTD}(\mathcal{C}_2)$. The reverse direction is an easy application of Lemma 4. Choose $\mathcal{C}_1' \subseteq \mathcal{C}_1$ and $\mathcal{C}_2' \subseteq \mathcal{C}_2$ so that $\mathrm{RTD}(\mathcal{C}_1) = \mathrm{TS}_{min}(\mathcal{C}_1')$ and $\mathrm{RTD}(\mathcal{C}_2) = \mathrm{TS}_{min}(\mathcal{C}_2')$. Now it follows that

$$\mathrm{RTD}(\mathcal{C}_1 \uplus \mathcal{C}_2) \geq \mathrm{TS}_{min}(\mathcal{C}_1' \uplus \mathcal{C}_2') = \mathrm{TS}_{min}(\mathcal{C}_1') + \mathrm{TS}_{min}(\mathcal{C}_2') = \mathrm{RTD}(\mathcal{C}_1) + \mathrm{RTD}(\mathcal{C}_2) \ .$$

■

Setting $\mathcal{C}_W^n = \mathcal{C}_W \uplus \ldots \uplus \mathcal{C}_W$ with n duplicates of $\mathcal{C}_W$ on the right-hand side, we now obtain the following result as an immediate application of Lemma 16:

Theorem 17 $\mathrm{VCD}(\mathcal{C}_W^n) = 2n$ *and* $\mathrm{RTD}(\mathcal{C}_W^n) = 3n$.

We remark here that the same kind of reasoning cannot be applied to blow up rfRTD, because $\mathrm{rfRTD}(\mathcal{C} \uplus \mathcal{C})$ can in general be smaller than $2 \cdot \mathrm{rfRTD}(\mathcal{C})$: considering again the class $\mathcal{C}$ with $\mathrm{rfRTD}(\mathcal{C}) = 3$ from Table 1, simple brute-force computations show that $\mathrm{rfRTD}(\mathcal{C} \times \mathcal{C}) = 5$.

4.2 Intersection-closed Classes

As shown by Kuhlmann (1999), $\mathrm{TS}_{min}(\mathcal{C}) \leq I(\mathcal{C})$ holds for every intersection-closed concept class $\mathcal{C}$. Kuhlmann's central argument (which occurred first in a proof of a related result by Goldman and Sloan (1994)) can be applied recursively so that the following is obtained:

Lemma 18 *For every intersection-closed class* $\mathcal{C}$, $\mathrm{RTD}(\mathcal{C}) \leq I(\mathcal{C})$.

Proof Let $k := I(\mathcal{C})$. We present a teaching plan for $\mathcal{C}$ of order at most k. Let $C_1, \ldots, C_N$ be the concepts in $\mathcal{C}$ in topological order such that $C_i \supset C_j$ implies $i < j$. It follows that, for every $i \in [N]$, C_i is an inclusion-maximal concept in $\mathcal{C}_i := \{C_i, \ldots, C_N\}$. Let S_i denote a minimal spanning set for C_i w.r.t. $\mathcal{C}$. Then:

- $|S_i| \leq k$ and C_i is the unique minimal concept in $\mathcal{C}$ that contains S_i.
- As C_i is inclusion-maximal in $\mathcal{C}_i$, C_i is the only concept in $\mathcal{C}_i$ that contains S_i.

Thus $\{(x, 1) \mid x \in S_i\}$ is a teaching set of size at most k for C_i in $\mathcal{C}_i$. ■

Since $I(\mathcal{C}) \leq \mathrm{VCD}(\mathcal{C})$, we get

Corollary 19 *For every intersection-closed class* $\mathcal{C}$, $\mathrm{RTD}(\mathcal{C}) \leq \mathrm{VCD}(\mathcal{C})$.

This implies $\mathrm{RTD}^*(\mathcal{C}) \leq \mathrm{VCD}(\mathcal{C})$ for every intersection-closed class $\mathcal{C}$, since the property "intersection-closed" is preserved when reducing a class $\mathcal{C}$ to $\mathcal{C}_{|X'}$ for $X' \subseteq X$.

For every fixed constant d (e.g., $d = 2$), Kuhlmann (1999) presents a family $(\mathcal{C}_m)_{m \geq 1}$ of intersection-closed concept classes such that the following holds:[3]

$$\forall m \geq 1 : \ \mathrm{VCD}(\mathcal{C}_m) = d \text{ and } \mathrm{SDC}(\mathcal{C}_m) \geq m \, . \tag{7}$$

3. A family satisfying (7) but *not* being intersection-closed was presented previously by Ben-David and Eiron (1998).

This shows that $\mathrm{SDC}(\mathcal{C})$ can in general not be upper-bounded by $I(\mathcal{C})$ or $\mathrm{VCD}(\mathcal{C})$. It shows furthermore that the gap between $\mathrm{RTD}(\mathcal{C})$ and $\mathrm{SDC}(\mathcal{C})$ can be arbitrarily large (even for intersection-closed classes).

Lemma 18 generalizes to nested differences:

Theorem 20 *If $\mathcal{C}$ is intersection-closed then* $\mathrm{RTD}(\mathrm{DIFF}^{\leq d}(\mathcal{C})) \leq d \cdot I(\mathcal{C})$.

Proof Any concept $C \in \mathrm{DIFF}^{\leq d}(\mathcal{C})$ can be written in the form

$$C = C_1 \setminus \overbrace{(C_2 \setminus (\cdots (C_{d-1} \setminus C_d) \cdots))}^{=:D_1} \tag{8}$$

such that, for every j, $C_j \in \mathcal{C} \cup \{\emptyset\}$, $C_j \supseteq C_{j+1}$, and this inclusion is proper unless $C_j = \emptyset$. Let $D_j = C_{j+1} \setminus (C_{j+2} \setminus (\cdots (C_{d-1} \setminus C_d) \cdots))$. We may obviously assume that the representation (8) of C is *minimal* in the following sense:

$$\forall j = 1, \ldots, d : C_j = \langle C_j \setminus D_j \rangle_{\mathcal{C}} \tag{9}$$

We define a *lexicographic ordering*, $\sqsupset$, on concepts from $\mathrm{DIFF}^{\leq d}(\mathcal{C})$ as follows. Let C be a concept with a minimal representation of the form (8), and let the minimal representation of C' be given similarly in terms of C'_j, D'_j. Then, by definition, $C \sqsupset C'$ if $C_1 \supset C'_1$ or $C_1 = C'_1 \wedge D_1 \sqsupset D'_1$.

Let $k := I(\mathcal{C})$. We present a teaching plan of order at most dk for $\mathrm{DIFF}^{\leq d}(\mathcal{C})$. Therein, the concepts are in lexicographic order so that, when teaching concept C with minimal representation (8), the concepts preceding C w.r.t. $\sqsupset$ have been discarded already. A teaching set T for C is then obtained as follows:

- For every $j = 1, \ldots, d$, include in T a minimal spanning set for $C_j \setminus D_j$ w.r.t. $\mathcal{C}$. Augment its instances by label 1 if j is odd, and by label 0 otherwise.

By construction, C as given by (8) and (9) is the lexicographically smallest concept in $\mathrm{DIFF}^{\leq d}(\mathcal{C})$ that is consistent with T. Since concepts being lexicographically larger than C have been discarded already, T is a teaching set for C. ■

Corollary 21 *Let $\mathcal{C}_1, \ldots, \mathcal{C}_r$ be intersection-closed classes over the domain X. Assume that the "universal concept" X belongs to each of these classes.*[4] *Then,*

$$\mathrm{RTD}\left(\mathrm{DIFF}^{\leq d}(\mathcal{C}_1 \cup \cdots \cup \mathcal{C}_r)\right) \leq d \cdot \sum_{i=1}^{r} I(\mathcal{C}_i)\,.$$

Proof Consider the concept class $\mathcal{C} := \mathcal{C}_1 \wedge \cdots \wedge \mathcal{C}_r := \{C_1 \cap \cdots \cap C_r \mid C_i \in \mathcal{C}_i \text{ for } i = 1, \ldots, r\}$. According to Helmbold et al. (1990), we have:

1. $\mathcal{C}_1 \cup \cdots \cup \mathcal{C}_r$ is a subclass of $\mathcal{C}$.

4. This assumption is not restrictive: adding the universal concept to an intersection-closed class does not destroy the property of being intersection-closed.

2. $\mathcal{C}$ is intersection-closed.

3. Let $C = C_1 \cap \cdots \cap C_r \in \mathcal{C}$. For all i, let S_i be a spanning set for C w.r.t. $\mathcal{C}_i$, i.e., $S_i \subseteq C$ and $\langle S_i \rangle_{\mathcal{C}_i} = \langle C \rangle_{\mathcal{C}_i}$. Then $S_1 \cup \cdots \cup S_r$ is a spanning set for C w.r.t. $\mathcal{C}$.

Thus $I(\mathcal{C}) \leq I(\mathcal{C}_1) + \cdots + I(\mathcal{C}_r)$. The corollary follows from Theorem 20. ■

4.3 Maximum Classes

In this section, we show that the recursive teaching dimension coincides with the VC-dimension on the family of maximum classes. In a maximum class $\mathcal{C}$, every set of $k \leq \mathrm{VCD}(\mathcal{C})$ instances is shattered, which implies $\mathrm{RTD}(\mathcal{C}) \geq \mathrm{TS}_{min}(\mathcal{C}) \geq \mathrm{VCD}(\mathcal{C})$. Thus, we can focus on the reverse direction and pursue the question whether $\mathrm{RTD}(\mathcal{C}) \leq \mathrm{VCD}(\mathcal{C})$. We shall answer this question in the affirmative by establishing a connection between "teaching plans" and "corner-peeling plans".

We say that a corner-peeling plan (5) is *strong* if Condition 2 in Definition 6 is replaced as follows:

2'. For all $t = 1, \ldots, N$, $\mathcal{C}'_t$ is a cube in $\{C_t, \ldots, C_N\}$ which contains C_t and whose colors (augmented by their labels according to C_t) form a teaching set for $C_t \in \{C_t, \ldots, C_N\}$.

We denote the set of colors of $\mathcal{C}'_t$ as X_t and its augmentation by labels according to C_t as S_t in what follows. The following result is obvious:

Lemma 22 *A strong corner-peeling plan of the form (5) induces a teaching plan of the form (2) of the same order.*

The following result justifies the attribute "strong" of corner-peeling plans:

Lemma 23 *Every strong corner-peeling plan is a corner-peeling plan.*

Proof Assume that Condition 2 is violated. Then there is a color $x \in X \setminus X_t$ and a concept $C \in \{C_{t+1}, \ldots, C_N\}$ such that C coincides with C_t on all instances except x. But then C is consistent with set S_t so that S_t is *not* a teaching set for $C_t \in \{C_t, \ldots, C_N\}$, and Condition 2' is violated as well. ■

Lemma 24 *Let $\mathcal{C}$ be a shortest-path closed concept class. Then, every corner-peeling plan for $\mathcal{C}$ is strong.*

Proof Assume that Condition 2' is violated. Then some $C \in \{C_{t+1}, \ldots, C_N\}$ is consistent with S_t. Thus, the shortest path between C and C_t in $\mathcal{G}(\{C_t, \ldots, C_N\})$ does not enter the cube $\mathcal{C}'_t$. Hence there is a concept $C' \in \{C_{t+1}, \ldots, C_N\} \setminus \mathcal{C}'_t$ that is a neighbor of C_t in $\mathcal{G}(\{C_t, \ldots, C_N\})$, and Condition 2 is violated. ■

As maximum classes are shortest-path closed (Kuzmin and Warmuth, 2007), we obtain:

Corollary 25 *Every corner-peeling plan for a maximum class is strong, and therefore induces a teaching plan of the same order.*

Since Rubinstein and Rubinstein (2012) showed that every maximum class $\mathcal{C}$ can be $\mathrm{VCD}(\mathcal{C})$-corner-peeled, we may conclude that $\mathrm{RTD}(\mathcal{C}) \leq \mathrm{VCD}(\mathcal{C})$. As mentioned above, $\mathrm{RTD}(\mathcal{C}) \geq \mathrm{TS}_{min}(\mathcal{C}) \geq \mathrm{VCD}(\mathcal{C})$ for every maximum class $\mathcal{C}$. Thus the following holds:

Theorem 26 *For every maximum class $\mathcal{C}$, $\mathrm{RTD}(\mathcal{C}) = \mathrm{TS}_{min}(\mathcal{C}) = \mathrm{VCD}(\mathcal{C})$.*

The fact that, for every maximum class $\mathcal{C}$ and every $X' \subseteq X$, the class $\mathcal{C}_{|X'}$ is still maximum implies that $\mathrm{RTD}^*(\mathcal{C}) = \mathrm{VCD}(\mathcal{C})$ for every maximum class $\mathcal{C}$.

We establish a connection between repetition-free teaching plans and representations having the acyclic non-clashing property:

Lemma 27 *Let $\mathcal{C}$ be an arbitrary concept class. Then the following holds:*

1. *Every repetition-free teaching plan (2) of order d for $\mathcal{C}$ induces a representation mapping r of order d for $\mathcal{C}$ given by $r(C_t) = X(S_t)$ for $t = 1, \ldots, N$. Moreover, r has the acyclic non-clashing property.*

2. *Every representation mapping r of order d for $\mathcal{C}$ that has the acyclic non-clashing property (4) induces a teaching plan (2) given by $S_t = \{(x, C_t(x)) \mid x \in r(C_t)\}$ for $t = 1, \ldots, N$. Moreover, this plan is repetition-free.*

Proof

1. A clash between C_t and $C_{t'}$, $t < t'$, on $X(S_t)$ would contradict the fact that S_t is a teaching set for $C_t \in \{C_t, \ldots, C_N\}$.

2. Conversely, if $S_t = \{(x, C_t(x)) \mid x \in r(C_t)\}$ is not a teaching set for $C_t \in \{C_t, \ldots, C_N\}$, then there must be a clash on $X(S_t)$ between C_t and a concept from $\{C_{t+1}, \ldots, C_N\}$. The teaching plan induced by r is obviously repetition-free since r is injective.

■

Corollary 28 *Let $\mathcal{C}$ be maximum of VC-dimension d. Then, there is a one-one mapping between repetition-free teaching plans of order d for $\mathcal{C}$ and unlabeled compression schemes with the acyclic non-clashing property.*

A closer look at the work by Rubinstein and Rubinstein (2012) reveals that corner-peeling leads to an unlabeled compression scheme with the acyclic non-clashing property (again implying that $\mathrm{RTD}(\mathcal{C}) \leq \mathrm{VCD}(\mathcal{C})$ for maximum classes $\mathcal{C}$). Similarly, an inspection of the work by Kuzmin and Warmuth (2007) reveals that the unlabeled compression scheme obtained by the Tail Matching Algorithm has the acyclic non-clashing property, too. Thus, this algorithm too can be used to generate a recursive teaching plan of order $\mathrm{VCD}(\mathcal{C})$ for any maximum class $\mathcal{C}$.

It is not known to date whether every concept class $\mathcal{C}$ of VC-dimension d can be embedded into a maximum concept class $\mathcal{C}' \supseteq \mathcal{C}$ of VC-dimension $O(d)$. Indeed, finding such an

embedding is considered as a promising method for settling the sample compression conjecture. It is easy to see that a negative answer to our question "Is $\mathrm{RTD}(\mathcal{C}) \in O(\mathrm{VCD}(\mathcal{C}))$?" would deem this approach fruitless:

Theorem 29 *If* $\mathrm{RTD}(\mathcal{C})$ *is not linearly bounded in* $\mathrm{VCD}(\mathcal{C})$*, then there is no mapping* $\mathcal{C} \mapsto \mathcal{C}' \supseteq \mathcal{C}$ *such that* $\mathcal{C}'$ *is maximum and* $\mathrm{VCD}(\mathcal{C}')$ *is linearly bounded in* $\mathrm{VCD}(\mathcal{C})$*.*

Proof Suppose there is a universal constant k and a mapping MAXIMIZE that maps every concept class $\mathcal{C}$ to a concept class $\mathcal{C}' \supseteq \mathcal{C}$ such that $\mathcal{C}'$ is maximum and $\mathrm{VCD}(\mathcal{C}') \leq k \cdot \mathrm{VCD}(\mathcal{C})$. It follows that, for any concept class $\mathcal{C}$, the following holds:

$$\mathrm{RTD}(\mathcal{C}) \leq \mathrm{RTD}(\mathrm{MAXIMIZE}(\mathcal{C})) = \mathrm{VCD}(\mathrm{MAXIMIZE}(\mathcal{C})) \leq k \cdot \mathrm{VCD}(\mathcal{C}))$$

where the equation $\mathrm{RTD}(\mathrm{MAXIMIZE}(\mathcal{C})) = \mathrm{VCD}(\mathrm{MAXIMIZE}(\mathcal{C}))$ follows from Theorem 26. ■

According to (6), this theorem still holds if RTD is replaced by RTD^*.

4.4 Shortest-Path Closed Classes

In this section, we study the best-case teaching dimension, $\mathrm{TS}_{min}(\mathcal{C})$, and the average-case teaching-dimension, $\mathrm{TS}_{avg}(\mathcal{C})$, of a shortest-path closed concept class $\mathcal{C}$.

It is known that the instances of $I(C; \mathcal{G}(\mathcal{C}))$, augmented by their C-labels, form a unique minimal teaching set for C in $\mathcal{C}$ provided that $\mathcal{C}$ is a maximum class (Kuzmin and Warmuth, 2007). Lemma 30 slightly generalizes this observation.

Lemma 30 *Let* $\mathcal{C}$ *be any concept class. Then the following two statements are equivalent:*

1. $\mathcal{C}$ *is shortest-path closed.*
2. *Every* $C \in \mathcal{C}$ *has a unique minimum teaching set* S*, namely the set* S *such that* $X(S) = I(C; \mathcal{G}(\mathcal{C}))$*.*

Proof $1 \Rightarrow 2$ is easy to see. Let $\mathcal{C}$ be shortest-path closed, and let C be any concept in $\mathcal{C}$. Clearly, any teaching set S for C must satisfy $I(C; \mathcal{G}(\mathcal{C})) \subseteq X(S)$ because C must be distinguished from all its neighbors in $\mathcal{G}(\mathcal{C})$. Let $C' \neq C$ be any other concept in $\mathcal{C}$. Since C and C' are connected by a path P of length $|C \triangle C'|$, C and C' are distinguished by the color of the first edge in P, say by the color $x \in I(C; \mathcal{G}(\mathcal{C}))$. Thus, no other instances (=colors) besides $I(C; \mathcal{G}(\mathcal{C}))$ are needed to distinguish C from any other concept in $\mathcal{C}$.

To show $2 \Rightarrow 1$, we suppose 2 and prove by induction on k that any two concepts $C, C' \in \mathcal{C}$ with $k = |C \triangle C'|$ are connected by a path of length k in $\mathcal{G}(\mathcal{C})$. The case $k = 1$ is trivial. For a fixed k, assume all pairs of concepts of Hamming distance k are connected by a path of length k in $\mathcal{G}(\mathcal{C})$. Let $C, C' \in \mathcal{C}$ with $|C \triangle C'| = k+1 \geq 2$. Since $I(C; \mathcal{G}(\mathcal{C})) = X(S)$, there is an $x \in I(C; \mathcal{G}(\mathcal{C}))$ such that $C(x) \neq C'(x)$. Let C'' be the x-neighbor of C in $\mathcal{G}(\mathcal{C})$. Note that $C''(x) = C'(x)$ so that C'' and C' have Hamming-distance k. According to the inductive hypothesis, there is a path of length k from C'' to C' in $\mathcal{G}(\mathcal{C})$. It follows that C and C' are connected by a path of length $k + 1$. ■

Theorem 31 *Let $\mathcal{C}$ be a shortest-path closed concept class. Then,* $\mathrm{TS}_{avg}(\mathcal{C}) < 2\mathrm{VCD}(\mathcal{C})$.

Proof According to Lemma 30, the average-case teaching dimension of $\mathcal{C}$ coincides with the average vertex-degree in $\mathcal{G}(\mathcal{C})$, which is twice the density of $\mathcal{G}(\mathcal{C})$. As mentioned in Section 2.4 already, $\mathrm{dens}(\mathcal{G}(\mathcal{C})) < \mathrm{VCD}(\mathcal{C})$. ■

Theorem 31 generalizes a result by Kuhlmann (1999) who showed that the average-case teaching dimension of "d-balls" (sets of concepts of Hamming distance at most d from a center concept) is smaller than $2d$. It also simplifies Kuhlmann's proof substantially. In Theorem 4 of the same paper, Kuhlmann (1999) stated furthermore that $\mathrm{TS}_{avg}(\mathcal{C}) < 2$ if $\mathrm{VCD}(\mathcal{C}) = 1$, but his proof is flawed.[5] Despite the flawed proof, the claim itself is correct as we show now:

Theorem 32 *Let $\mathcal{C}$ be any concept class. If* $\mathrm{VCD}(\mathcal{C}) = 1$ *then* $\mathrm{TS}_{avg}(\mathcal{C}) < 2$.

Proof By Theorem 31, the average-case teaching dimension of a maximum class of VC-dimension 1 is less than 2. It thus suffices to show that any class $\mathcal{C}$ of VC-dimension 1 can be transformed into a maximum class $\mathcal{C}'$ of VC-dimension 1 without decreasing the average-case teaching dimension. Let $X' \subseteq X$ be a minimal set that is $\mathcal{C}$-distinguishing, i.e., for every pair of distinct concepts $C, C' \in \mathcal{C}$, there is some $x \in X'$ such that $C(x) \neq C(x')$. Let $m = |X|$ and $\mathcal{C}' = \mathcal{C}_{|X'}$. Obviously, $|\mathcal{C}'| = |\mathcal{C}|$ and $\mathrm{VCD}(\mathcal{C}') = 1$ so that $|\mathcal{C}'| \leq \binom{m}{0} + \binom{m}{1} = m + 1$. Now we prove that $\mathcal{C}'$ is maximum. Note that every $x \in X'$ occurs as a color in $\mathcal{G}(\mathcal{C}')$ because, otherwise, $X' \setminus \{x\}$ would still be $\mathcal{C}$-distinguishing (which would contradict the minimality of X'). As $\mathrm{VCD}(\mathcal{C}') = 1$, no color can occur twice. Thus $|E(\mathcal{G}(\mathcal{C}'))| = m$. Moreover, there is no cycle in $\mathcal{G}(\mathcal{C}')$ since a cycle would require at least one repeated color. As $\mathcal{G}(\mathcal{C}')$ is an acyclic graph of m edges, it has at least $m + 1$ vertices, i.e. $|\mathcal{C}'| \geq m + 1$. Thus, $|\mathcal{C}'| = m + 1$ and $\mathcal{C}'$ is maximum. This implies that $\mathrm{TS}_{avg}(\mathcal{C}') < 2\mathrm{VCD}(\mathcal{C}')$. Since $X' \subseteq X$ but X' is still $\mathcal{C}$-distinguishing, we obtain $\mathrm{TS}(C;\mathcal{C}) \leq \mathrm{TS}(C_{|X'}, \mathcal{C}')$ for all $C \in \mathcal{C}$. Thus, $\mathrm{TS}_{avg}(\mathcal{C}) \leq \mathrm{TS}_{avg}(\mathcal{C}') < 2\mathrm{VCD}(\mathcal{C}') = 2$, which concludes the proof. ■

We briefly note that $\mathrm{TS}_{avg}(\mathcal{C})$ cannot in general be bounded by $O(\mathrm{VCD}(\mathcal{C}))$. Kushilevitz et al. (1996) present a family $(\mathcal{C}_n)$ of concept classes such that $\mathrm{TS}_{avg}(\mathcal{C}_n) = \Omega(\sqrt{|\mathcal{C}_n|})$ but $\mathrm{VCD}(\mathcal{C}_n) \leq \log |\mathcal{C}_n|$.

We conclude this section by showing that there are shortest-path closed classes for which RTD exceeds VCD.

Lemma 33 *If* $\deg_{\mathcal{G}(\mathcal{C})}(C) \geq |X| - 1$ *for all* $C \in \mathcal{C}$, *then* $\mathcal{C}$ *is shortest-path closed.*

Proof Assume by way of contradiction that $\mathcal{C}$ is not shortest-path closed. Pick two concepts $C, C' \in \mathcal{C}$ of minimal Hamming-distance, say d, subject to the constraint of not being connected by a path of length d in $\mathcal{G}(\mathcal{C})$. It follows that $d \geq 2$. By the minimality of d, any

5. His Claim 2 states the following. *If* $\mathrm{VCD}(\mathcal{C}) = 1$, $C_1, C_2 \in \mathcal{C}$, $x \in X$, $x \notin C_1$, $C_2 = C_1 \cup \{x\}$, *then, for either* $(i,j) = (1,2)$ *or* $(i,j) = (2,1)$, *one obtains* $\mathrm{TS}(C_i;\mathcal{C}) = \mathrm{TS}(C_i - x; \mathcal{C} - x) + 1$ *and* $\mathrm{TS}(C_j;\mathcal{C}) = 1$. This is not correct, as can be shown by the class $\mathcal{C} = \{\{x_z : 1 \leq z \leq k\} : 0 \leq k \leq 5\}$ over $X = \{x_k : 1 \leq k \leq 5\}$, which has VC-dimension 1. For $C_1 = \{x_1, x_2\}$, $C_2 = \{x_1, x_2, x_3\}$, and $x = x_3$, we get $\mathrm{TS}(C_1;\mathcal{C}) = \mathrm{TS}(C_2;\mathcal{C}) = \mathrm{TS}(C_1 - x; \mathcal{C} - x) = 2$.

neighbor of C with Hamming-distance $d-1$ to C' does not belong to $\mathcal{C}$. Since there are d such missing neighbors, the degree of C in $\mathcal{G}(\mathcal{C})$ is bounded by $|X|-d \leq |X|-2$. This yields a contradiction. ■

Rubinstein et al. (2009) present a concept class $\mathcal{C}$ with $\mathrm{TS}_{min}(\mathcal{C}) > \mathrm{VCD}(\mathcal{C})$. An inspection of this class shows that the minimum vertex degree in its 1-inclusion graph is $|X|-1$. According to Lemma 33, this class must be shortest-path closed. Thus, the inequality $\mathrm{TS}_{min}(\mathcal{C}) \leq \mathrm{VCD}(\mathcal{C})$ does not generalize from maximum classes to shortest-path closed classes.

5. Conclusions

This paper relates the RTD, a recently introduced teaching complexity notion, to information complexity parameters of various classical learning models.

One of these parameters is SDC, the information complexity of self-directed learning, which constitutes the most information-efficient query learning model known to date. Our main result in this context, namely lower-bounding the SDC by the RTD, has implications for the analysis of information complexity in teaching and learning. In particular, every upper bound on SDC holds for RTD; every lower bound on RTD holds for SDC.

The central parameter in our comparison is the VC-dimension. Although the VC-dimension can be arbitrarily large for classes of recursive teaching dimension 1 (which is well-known and also evident from Theorem 11) and arbitrarily smaller than SDC (Ben-David and Eiron, 1998; Kuhlmann, 1999), it does not generally lie in between the two. However, while the SDC cannot be upper-bounded by any linear function of the VC-dimension, it is still open whether such a bound exists for the RTD. The existence of the latter would mean that the combinatorial properties that determine the information complexity of PAC-learning (i.e., of learning from randomly drawn examples) are essentially the same as those that determine the information complexity of teaching (i.e., of learning from helpfully selected examples), at least when using the recursive teaching model.

As a partial solution to this open question, we showed that the VC-dimension coincides with the RTD in the special case of maximum classes. Our results, and in particular the remarkable correspondence to unlabeled compression schemes, suggest that the RTD is based on a combinatorial structure that is of high relevance for the complexity of information-efficient learning and sample compression. Analyzing the circumstances under which teaching plans defining the RTD can be used to construct compression schemes (and to bound their size) seems to be a promising step towards new insights into the theory of sample compression.

Acknowledgments

We would like to thank Manfred Warmuth for the permission to include the concept class from Figure 1 in this paper. Moreover, we would like to thank Malte Darnstädt and Michael Kallweit for helpful and inspiring discussions, and the anonymous referees of both this article

and its earlier conference version for many helpful suggestions and for pointing out mistakes in the proofs of Theorems 20 and 32.

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

Frank Balbach. Measuring teachability using variants of the teaching dimension. *Theoretical Computer Science*, 397(1–3):94–113, 2008.

Shai Ben-David and Nadav Eiron. Self-directed learning and its relation to the VC-dimension and to teacher-directed learning. *Machine Learning*, 33(1):87–104, 1998.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association on Computing Machinery*, 36(4):929–965, 1989.

Malte Darnstädt, Thorsten Doliwa, Hans U. Simon, and Sandra Zilles. Order compression schemes. In *Proceedings of the 24th International Conference on Algorithmic Learning Theory*, pages 173–187, 2013.

Thorsten Doliwa, Hans U. Simon, and Sandra Zilles. Recursive teaching dimension, learning complexity, and maximum classes. In *Proceedings of the 21st International Conference on Algorithmic Learning Theory*, pages 209–223, 2010.

Sally Floyd and Manfred Warmuth. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, 21(3):1–36, 1995.

Sally A. Goldman and Michael J. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.

Sally A. Goldman and Robert H. Sloan. The power of self-directed learning. *Machine Learning*, 14(1):271–294, 1994.

Sally A. Goldman, Ronald L. Rivest, and Robert E. Schapire. Learning binary relations and total orders. *SIAM Journal on Computing*, 22(5):1006–1034, 1993.

David Haussler, Nick Littlestone, and Manfred K. Warmuth. Predicting $\{0,1\}$ functions on randomly drawn points. *Information and Computation*, 115(2):284–293, 1994.

David Helmbold, Robert Sloan, and Manfred K. Warmuth. Learning nested differences of intersection-closed concept classes. *Machine Learning*, 5:165–196, 1990.

Christian Kuhlmann. On teaching and learning intersection-closed concept classes. In *Proceedings of the 4th European Conference on Computational Learning Theory*, pages 168–182, 1999.

Eyal Kushilevitz, Nathan Linial, Yuri Rabinovich, and Michael E. Saks. Witness sets for families of binary vectors. *J. Comb. Theory, Ser. A*, 73(2):376–380, 1996.

Dima Kuzmin and Manfred K. Warmuth. Unlabeled compression schemes for maximum classes. *Journal of Machine Learning Research*, 8:2047–2081, 2007.

Nick Littlestone. Learning quickly when irrelevant attributes abound: a new linear threshold algorithm. *Machine Learning*, 2(4):245–318, 1988.

Nick Littlestone and Manfred K. Warmuth. Relating data compression and learnability. Technical Report, UC Santa Cruz, 1996.

Wolfgang Maass and György Turán. Lower bound methods and separation results for on-line learning models. *Machine Learning*, 9:107–145, 1992.

Balas K. Natarajan. On learning boolean functions. In *Proceedings of the 19th Annual Symposium on Theory of Computing*, pages 296–304, 1987.

Benjamin I. P. Rubinstein and J. Hyam Rubinstein. A geometric approach to sample compression. *Journal of Machine Learning Research*, 13:1221–1261, 2012.

Benjamin I. P. Rubinstein, Peter L. Bartlett, and J. Hyam Rubinstein. Shifting: One-inclusion mistake bounds and sample compression. *Journal of Computer and System Sciences*, 75(1):37–59, 2009.

Rahim Samei, Pavel Semukhin, Boting Yang, and Sandra Zilles. Sauer's bound for a notion of teaching complexity. In *Proceedings of the 23rd International Conference on Algorithmic Learning Theory*, pages 96–110, 2012.

Norbert Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, 1972.

Ayumi Shinohara and Satoru Miyano. Teachability in computational learning. *New Generation Computing*, 8(4):337–348, 1991.

Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Vladimir N. Vapnik and Alexey Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theor. Probability and Appl.*, 16(2):264–280, 1971.

Emo Welzl. Complete range spaces, 1987. Unpublished Notes.

Sandra Zilles, Steffen Lange, Robert Holte, and Martin Zinkevich. Teaching dimensions based on cooperative learning. In *Proceedings of the 21st Annual Conference on Learning Theory*, pages 135–146, 2008.

Sandra Zilles, Steffen Lange, Robert Holte, and Martin Zinkevich. Models of cooperative teaching and learning. *Journal of Machine Learning Research*, 12:349–384, 2011.

Journal of Machine Learning Research 15 (2014) 3133-3181 Submitted 11/13; Revised 4/14; Published 10/14

Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?

Manuel Fernández-Delgado MANUEL.FERNANDEZ.DELGADO@USC.ES
Eva Cernadas EVA.CERNADAS@USC.ES
Senén Barro SENEN.BARRO@USC.ES
CITIUS: Centro de Investigación en Tecnoloxías da Información da USC
University of Santiago de Compostela
Campus Vida, 15872, Santiago de Compostela, Spain

Dinani Amorim DINANIAMORIM@GMAIL.COM
Departamento de Tecnologia e Ciências Sociais- DTCS
Universidade do Estado da Bahia
Av. Edgard Chastinet S/N - São Geraldo - Juazeiro-BA, CEP: 48.305-680, Brasil

Editor: Russ Greiner

Abstract

We evaluate **179 classifiers** arising from **17 families** (discriminant analysis, Bayesian, neural networks, support vector machines, decision trees, rule-based classifiers, boosting, bagging, stacking, random forests and other ensembles, generalized linear models, nearest-neighbors, partial least squares and principal component regression, logistic and multinomial regression, multiple adaptive regression splines and other methods), implemented in Weka, R (with and without the caret package), C and Matlab, including all the relevant classifiers available today. We use **121 data sets**, which represent **the whole UCI** data base (excluding the large-scale problems) and other own real problems, in order to achieve significant conclusions about the classifier behavior, not dependent on the data set collection. **The classifiers most likely to be the bests are the random forest** (RF) versions, the best of which (implemented in R and accessed via caret) achieves 94.1% of the maximum accuracy overcoming 90% in the 84.3% of the data sets. However, the difference is not statistically significant with the second best, the SVM with Gaussian kernel implemented in C using LibSVM, which achieves 92.3% of the maximum accuracy. A few models are clearly better than the remaining ones: random forest, SVM with Gaussian and polynomial kernels, extreme learning machine with Gaussian kernel, C5.0 and avNNet (a committee of multi-layer perceptrons implemented in R with the caret package). The random forest is clearly the best family of classifiers (3 out of 5 bests classifiers are RF), followed by SVM (4 classifiers in the top-10), neural networks and boosting ensembles (5 and 3 members in the top-20, respectively).

Keywords: classification, UCI data base, random forest, support vector machine, neural networks, decision trees, ensembles, rule-based classifiers, discriminant analysis, Bayesian classifiers, generalized linear models, partial least squares and principal component regression, multiple adaptive regression splines, nearest-neighbors, logistic and multinomial regression

1. Introduction

When a researcher or data analyzer faces to the classification of a data set, he/she usually applies the classifier which he/she expects to be "the best one". This expectation is conditioned by the (often partial) researcher knowledge about the available classifiers. One reason is that they arise from different fields within computer science and mathematics, i.e., they belong to different "classifier families". For example, some classifiers (linear discriminant analysis or generalized linear models) come from statistics, while others come from symbolic artificial intelligence and data mining (rule-based classifiers or decision-trees), some others are connectionist approaches (neural networks), and others are ensembles, use regression or clustering approaches, etc. *A researcher may not be able to use classifiers arising from areas in which he/she is not an expert (for example, to develop parameter tuning), being often limited to use the methods within his/her domain of expertise.* However, there is no certainty that they work better, for a given data set, than other classifiers, which seem more "exotic" to him/her. The lack of available implementation for many classifiers is a major drawback, although it has been partially reduced due to the large amount of classifiers implemented in R[1] (mainly from Statistics), Weka[2] (from the data mining field) and, in a lesser extend, in Matlab using the Neural Network Toolbox[3]. Besides, the R package caret (Kuhn, 2008) provides a very easy interface for the execution of many classifiers, allowing automatic parameter tuning and reducing the requirements on the researcher's knowledge (about the tunable parameter values, among other issues). Of course, the researcher can review the literature to know about classifiers in families outside his/her domain of expertise and, if they work better, to use them instead of his/her preferred classifier. However, *usually the papers which propose a new classifier compare it only to classifiers within the same family, excluding families outside the author's area of expertise.* Thus, the researcher does not know whether these classifiers work better or not than the ones that he/she already knows. On the other hand, these comparisons are usually developed over a few, although expectedly relevant, data sets. Given that all the classifiers (even the "good" ones) show strong variations in their results among data sets, the average accuracy (over all the data sets) might be of limited significance if a reduced collection of data sets is used (Macià and Bernadó-Mansilla, 2014). Specifically, some classifiers with a good average performance over a reduced data set collection could achieve significantly worse results when the collection is extended, and conversely classifiers with sub-optimal performance on the reduced data collection could be not so bad when more data sets are included. There are useful guidelines (Hothorn et al., 2005; Eugster et al., 2014) to analyze and design benchmark exploratory and inferential experiments, giving also a very useful framework to inspect the relationship between data sets and classifiers.

Each time we find a new classifier or family of classifiers from areas outside our domain of expertise, we ask ourselves whether that classifier will work better than the ones that we use routinely. In order to have a clear idea of the capabilities of each classifier and family, it would be useful to develop a comparison of *a high number of classifiers* arising from *many different families and areas of knowledge* over a *large collection of data sets.* The objective

1. See `http://www.r-project.org`.
2. See `http://www.cs.waikato.ac.nz/ml/weka`.
3. See `http://www.mathworks.es/products/neural-network`.

is to select the classifier which more probably achieves the best performance for any data set. In the current paper we use a large collection of classifiers with publicly available implementations (in order to allow future comparisons), arising from a wide variety of classifier families, in order to achieve significant conclusions not conditioned by the number and variety of the classifiers considered. Using a *high number of classifiers* it is probable that some of them will achieve the "highest" possible performance for each data set, which can be used as reference (maximum accuracy) to evaluate the remaining classifiers. However, according to the No-Free-Lunch theorem (Wolpert, 1996), the best classifier will not be the same for all the data sets. Using *classifiers from many families*, we are not restricting the significance of our comparison to one specific family among many available methods. Using a *high number of data sets*, it is probable that each classifier will work well in some data sets and not so well in others, increasing the evaluation significance. Finally, considering the availability of several alternative implementations for the most popular classifiers, their comparison may also be interesting. The current work pursues: 1) to select the globally best classifier for the selected data set collection; 2) to rank each classifier and family according to its accuracy; 3) to determine, for each classifier, its probability of achieving the best accuracy, and the difference between its accuracy and the best one; 4) to evaluate the classifier behavior varying the data set properties (complexity, #patterns, #classes and #inputs).

Some recent papers have analyzed the comparison of classifiers over large collection of data sets. **OpenML** (Vanschoren et al., 2012), is a complete web interface[4] to anonymously access an experiment data base including 86 data sets from the UCI machine learning data base (Bache and Lichman, 2013) and 93 classifiers implemented in Weka. Although plug-ins for R, Knime and RapidMiner are under development, currently it only allows to use Weka classifiers. This environment allows to send queries about the classifier behavior with respect to tunable parameters, considering several common performance measures, feature selection techniques and bias-variance analysis. There is also an interesting analysis (Macià and Bernadó-Mansilla, 2014) about the use of the UCI repository launching several interesting criticisms about the usual practice in experimental comparisons. In the following, we synthesize these criticisms (the italicized sentences are literal cites) and describe how we tried to avoid them in our paper:

1. The criterion used to select the data set collection (which is usually reduced) may bias the comparison results. The same authors stated (Macià et al., 2013) that the superiority of a classifier may be restricted to a given domain characterized by some complexity measures, studying why and how the data set selection may change the results of classifier comparisons. Following these suggestions, we use all the data sets in the UCI classification repository, in order to avoid that a small data collection invalidate the conclusions of the comparison. This paper also emphasizes that the UCI repository was *not* designed to be a *complete, reliable framework composed of standardized real samples.*

2. The issue about *(1) whether the selection of learners is representative enough and (2) whether the selected learners are properly configured to work at their best performance*

4. See `http://expdb.cs.kuleuven.be/expdb`.

suggests that proposals of new classifiers usually design and tune them carefully, while the reference classifiers are run using a baseline configuration. This issue is also related to the lack of deep knowledge and experience about the details of all the classifiers with available implementations, so that the researchers *usually do not pay much attention about the selected reference algorithms, which may consequently bias the results in favour of the proposed algorithm*. With respect to this criticism, in the current paper we do not propose any new classifier nor changes on existing approaches, so we are not interested in favour any specific classifier, although we are more experienced with some classifier than others (for example, with respect to the tunable parameter values). We develop in this work a parameter tuning in the majority of the classifiers used (see below), selecting the best available configuration over a training set. Specifically, the classifiers implemented in R using caret automatically tune these parameters and, even more important, using pre-defined (and supposedly meaningful) values. This fact should compensate our lack of experience about some classifiers, and reduce its relevance on the results.

3. *It is still impossible to determine the maximum attainable accuracy for a data set, so that it is difficult to evaluate the true quality of each classifier*. In our paper, we use a large amount of classifiers (179) from many different families, so we hypothesize that the maximum accuracy achieved by some classifier is the maximum attainable accuracy for that data set: i.e., we suppose that if no classifier in our collection is able to reach higher accuracy, no one will reach. We can not test the validity of this hypothesis, but it seems reasonable that, when the number of classifiers increases, some of them will achieve the largest possible accuracy.

4. Since the data set complexity (measured somehow by the maximum attainable accuracy) is unknown, we do not know if the *classification error is caused by unfitted classifier design (learner's limitation) or by intrinsic difficulties of the problem (data limitation)*. In our work, since we consider that the attainable accuracy is the maximum accuracy achieved by some classifier in our collection, we can consider that low accuracies (with respect to this maximum accuracy) achieved by other classifiers are always caused by classifier limitations.

5. *The lack of standard data partitioning, defining training and testing data for cross-validation trials. Simply the use of different data partitionings will eventually bias the results, and make the comparison between experiments impossible*, something which is also emphasized by other researchers (Vanschoren et al., 2012). In the current paper, each data set uses the same partitioning for all the classifiers, so that this issue can not bias the results favouring any classifier. Besides, the partitions are publicly available (see Section 2.1), in order to make possible the experiment replication.

The paper is organized as follows: the Section 2 describes the collection of data sets and classifiers considered in this work; the Section 3 discusses the results of the experiments, and the Section 4 compiles the conclusions of the research developed.

2. Materials and Methods

In the following paragraphs we describe the materials (data sets) and methods (classifiers) used to develop this comparison.

Data set	#pat.	#inp.	#cl.	%Maj.	Data set	#pat.	#inp.	#cl.	%Maj.
abalone	4177	8	3	34.6	energy-y1	768	8	3	46.9
ac-inflam	120	6	2	50.8	energy-y2	768	8	3	49.9
acute-nephritis	120	6	2	58.3	fertility	100	9	2	88.0
adult	48842	14	2	75.9	flags	194	28	8	30.9
annealing	798	38	6	76.2	glass	214	9	6	35.5
arrhythmia	452	262	13	54.2	haberman-survival	306	3	2	73.5
audiology-std	226	59	18	26.3	hayes-roth	132	3	3	38.6
balance-scale	625	4	3	46.1	heart-cleveland	303	13	5	54.1
balloons	16	4	2	56.2	heart-hungarian	294	12	2	63.9
bank	45211	17	2	88.5	heart-switzerland	123	12	2	39.0
blood	748	4	2	76.2	heart-va	200	12	5	28.0
breast-cancer	286	9	2	70.3	hepatitis	155	19	2	79.3
bc-wisc	699	9	2	65.5	hill-valley	606	100	2	50.7
bc-wisc-diag	569	30	2	62.7	horse-colic	300	25	2	63.7
bc-wisc-prog	198	33	2	76.3	ilpd-indian-liver	583	9	2	71.4
breast-tissue	106	9	6	20.7	image-segmentation	210	19	7	14.3
car	1728	6	4	70.0	ionosphere	351	33	2	64.1
ctg-10classes	2126	21	10	27.2	iris	150	4	3	33.3
ctg-3classes	2126	21	3	77.8	led-display	1000	7	10	11.1
chess-krvk	28056	6	18	16.2	lenses	24	4	3	62.5
chess-krvkp	3196	36	2	52.2	letter	20000	16	26	4.1
congress-voting	435	16	2	61.4	libras	360	90	15	6.7
conn-bench-sonar	208	60	2	53.4	low-res-spect	531	100	9	51.9
conn-bench-vowel	528	11	11	9.1	lung-cancer	32	56	3	40.6
connect-4	67557	42	2	75.4	lymphography	148	18	4	54.7
contrac	1473	9	3	42.7	magic	19020	10	2	64.8
credit-approval	690	15	2	55.5	mammographic	961	5	2	53.7
cylinder-bands	512	35	2	60.9	miniboone	130064	50	2	71.9
dermatology	366	34	6	30.6	molec-biol-promoter	106	57	2	50.0
echocardiogram	131	10	2	67.2	molec-biol-splice	3190	60	3	51.9
ecoli	336	7	8	42.6	monks-1	124	6	2	50.0

Table 1: Collection of 121 data sets from the UCI data base and our real problems. It shows the number of patterns (#pat.), inputs (#inp.), classes (#cl.) and percentage of majority class (%Maj.) for each data set. Continued in Table 2. Some keys are: ac-inflam=acute-inflammation, bc=breast-cancer, congress-vot= congressional-voting, ctg=cardiotocography, conn-bench-sonar/vowel= connectionist-benchmark-sonar-mines-rocks/vowel-deterding, pb= pittsburg-bridges, st=statlog, vc=vertebral-column.

2.1 Data Sets

We use the whole UCI machine learning repository, the most widely used data base in the classification literature, to develop the classifier comparison. The UCI website[5] specifies a list of **165 data sets** which can be used for classification tasks (March, 2013). We **discarded 57 data sets** due to several reasons: 25 large-scale data sets (with very high #patterns and/or #inputs, for which our classifier implementations are not designed), 27 data sets which are not in the "common UCI format", and 5 data sets due to diverse reasons (just one input, classes without patterns, classes with only one pattern and sets not available). We also used **4 real-world data sets** (González-Rufino et al., 2013) not included in the UCI repository, about fecundity estimation for fisheries: they are denoted as oocMerl4D (2-class classification according to the presence/absence of oocyte nucleus), oocMerl2F (3-class classification according to the stage of development of the oocyte) for fish species *Merluccius*; and oocTris2F (nucleus) and oocTris5B (stages) for fish species *Trisopterus*. The inputs are texture features extracted from oocytes (cells) in histological images of fish gonads, and its calculation is described in the page 2400 (Table 4) of the cited paper.

Overall, we have 165 - 57 + 4 = **112 data sets**. However, some UCI data sets provide several "class" columns, so that actually they can be considered several classification problems. This is the case of data set *cardiotocography*, where the inputs can be classified into 3 or 10 classes, giving two classification problems (one additional data set); *energy*, where the classes can be given by columns y1 or y2 (one additional data set); *pittsburg-bridges*, where the classes can be material, rel-l, span, t-or-d and type (4 additional data sets); plant (whose complete UCI name is *One-hundred plant species*), with inputs margin, shape or texture (2 extra data sets); and *vertebral-column*, with 2 or 3 classes (1 extra data set). Therefore, we achieve a total of 112 + 1 + 1 + 4 + 2 + 1 = **121 data sets**[6], listed in the Tables 1 and 2 by alphabetic order (some data set names are reduced but significant versions of the UCI official names, which are often too long). OpenML (Vanschoren et al., 2012) includes only 86 data sets, of which seven do not belong to the UCI database: baseball, braziltourism, CoEPrA-2006_Classification_001/2/3, eucalyptus, labor, sick and solar-flare. In our work, the #patterns range from 10 (data set *trains*) to 130,064 (*miniboone*), with #inputs ranging from 3 (data set *hayes-roth*) to 262 (data set *arrhythmia*), and #classes between 2 and 100. We used even tiny data sets (such as *trains* or *balloons*), in order to assess that each classifier is able to learn these (expected to be "easy") data sets. In some data sets the classes with only two patterns were removed because they are not enough for training/test sets. The same data files were used for all the classifiers, excepting the ones provided by Weka, which require the ARFF format. We converted the nominal (or discrete) inputs to numeric values using a simple quantization: if an input x may take discrete values $\{v_1, \ldots, v_n\}$, when it takes the discrete value v_i it is converted to the numeric value $i \in \{1, \ldots, n\}$. We are conscious that *this change in the representation may have a high impact in the results of distance-based classifiers* (Macià and Bernadó-Mansilla, 2014), because contiguous discrete values (v_i and v_{i+1}) might not be nearer than non-contiguous values (v_1 and v_n). Each input

5. See `http://archive.ics.uci.edu/ml/datasets.html?task=cla`.
6. The whole data set and partitions are available from: `http://persoal.citius.usc.es/manuel.fernandez.delgado/papers/jmlr/data.tar.gz`.

Data set	#pat.	#inp.	#cl.	%Maj.	Data set	#pat.	#inp.	#cl.	%Maj.
monks-2	169	6	2	62.1	soybean	307	35	18	13.0
monks-3	3190	6	2	50.8	spambase	4601	57	2	60.6
mushroom	8124	21	2	51.8	spect	80	22	2	67.1
musk-1	476	166	2	56.5	spectf	80	44	2	50.0
musk-2	6598	166	2	84.6	st-australian-credit	690	14	2	67.8
nursery	12960	8	5	33.3	st-german-credit	1000	24	2	70.0
oocMerl2F	1022	25	3	67.0	st-heart	270	13	2	55.6
oocMerl4D	1022	41	2	68.7	st-image	2310	18	7	14.3
oocTris2F	912	25	2	57.8	st-landsat	4435	36	6	24.2
oocTris5B	912	32	3	57.6	st-shuttle	43500	9	7	78.4
optical	3823	62	10	10.2	st-vehicle	846	18	4	25.8
ozone	2536	72	2	97.1	steel-plates	1941	27	7	34.7
page-blocks	5473	10	5	89.8	synthetic-control	600	60	6	16.7
parkinsons	195	22	2	75.4	teaching	151	5	3	34.4
pendigits	7494	16	10	10.4	thyroid	3772	21	3	92.5
pima	768	8	2	65.1	tic-tac-toe	958	9	2	65.3
pb-MATERIAL	106	4	3	74.5	titanic	2201	3	2	67.7
pb-REL-L	103	4	3	51.5	trains	10	28	2	50.0
pb-SPAN	92	4	3	52.2	twonorm	7400	20	2	50.0
pb-T-OR-D	102	4	2	86.3	vc-2classes	310	6	2	67.7
pb-TYPE	105	4	6	41.9	vc-3classes	310	6	3	48.4
planning	182	12	2	71.4	wall-following	5456	24	4	40.4
plant-margin	1600	64	100	1.0	waveform	5000	21	3	33.9
plant-shape	1600	64	100	1.0	waveform-noise	5000	40	3	33.8
plant-texture	1600	64	100	1.0	wine	179	13	3	39.9
post-operative	90	8	3	71.1	wine-quality-red	1599	11	6	42.6
primary-tumor	330	17	15	25.4	wine-quality-white	4898	11	7	44.9
ringnorm	7400	20	2	50.5	yeast	1484	8	10	31.2
seeds	210	7	3	33.3	zoo	101	16	7	40.6
semeion	1593	256	10	10.2					

Table 2: Continuation of Table 1 (data set collection).

is pre-processed to have zero mean and standard deviation one, as is usual in the classifier literature. We do not use further pre-processing, data transformation or feature selection. The reasons are: 1) the impact of these transforms can be expected to be similar for all the classifiers; however, our objective is not to achieve the best possible performance for each data set (which eventually might require further pre-processing), but to compare classifiers on each set; 2) if pre-processing favours some classifier(s) with respect to others, this impact should be random, and therefore not statistically significant for the comparison; 3) in order to avoid comparison bias due to pre-processing, it seems advisable to use the original data; 4) in order to enhance the classification results, further pre-processing eventually should be specific to each data set, which would increase largely the present work; and 5) additional transformations would require a knowledge which is outside the scope of this paper, and should be explored in a different study. In those data sets with different training and test sets (annealing or audiology-std, among others), both files were not merged to follow the practice recommended by the data set creators, and to achieve "significant" accuracies on the right test data, using the right training data. In those data sets where the class attribute

must be defined grouping several values (in data set abalone) we follow the instructions in the data set description (file `data.names`). Given that our classifiers are not oriented to data with missing features, the missing inputs are treated as zero, which should not bias the comparison results. For each data set (abalone) two data files are created: `abalone_R.dat`, designed to be read by the R, C and Matlab classifiers, and `abalone.arff`, designed to be read by the Weka classifiers.

2.2 Classifiers

We use **179 classifiers** implemented in C/C++, Matlab, R and Weka. Excepting the Matlab classifiers, all of them are free software. We only developed own versions in C for the classifiers proposed by us (see below). Some of the R programs use directly the package that provides the classifier, but others use the classifier through the interface `train` provided by the caret[7] package. This function develops the parameter tuning, selecting the values which maximize the accuracy according to the validation selected (leave-one-out, k-fold, etc.). The caret package also allows to define the number of values used for each tunable parameter, although the specific values can not be selected. We used all the classifiers provided by Weka, running the command-line version of the java class for each classifier.

OpenML uses 93 Weka classifiers, from which we included 84. We could not include in our collection the remaining 9 classifiers: ADTree, alternating decision tree (Freund and Mason, 1999); AODE, aggregating one-dependence estimators (Webb et al., 2005); Id3 (Quinlan, 1986); LBR, lazy Bayesian rules (Zheng and Webb, 2000); M5Rules (Holmes et al., 1999); Prism (Cendrowska, 1987); ThresholdSelector; VotedPerceptron (Freund and Schapire, 1998) and Winnow (Littlestone, 1988). The reason is that they only accept nominal (not numerical) inputs, while we converted all the inputs to numeric values. Besides, we did not use classifiers ThresholdSelector, VotedPerceptron and Winnow, included in openML, because they accept only two-class problems. Note that classifiers LocallyWeightedLearning and RippleDownRuleLearner (Vanschoren et al., 2012) are included in our collection as LWL and Ridor respectively. Furthermore, we also included other 36 classifiers implemented in R, 48 classifiers in R using the caret package, as well as 6 classifiers implemented in C and other 5 in Matlab, summing up to 179 classifiers.

In the following, we briefly describe the 179 classifiers of the different families identified by acronyms (DA, BY, etc., see below), their names and implementations, coded as `name_implementation`, where `implementation` can be **C**, **m** (Matlab), **R**, **t** (in R using caret) and **w** (Weka), and their tunable parameter values (the notation A:B:C means from A to C step B). We found errors using several classifiers accessed via caret, but we used the corresponding R packages directly. This is the case of lvq, bdk, gaussprLinear, glmnet, kernelpls, widekernelpls, simpls, obliqueTree, spls, gpls, mars, multinom, lssvmRadial, partDSA, PenalizedLDA, qda, QdaCov, mda, rda, rpart, rrlda, sddaLDA, sddaQDA and sparseLDA. Some other classifiers as Linda, smda and xyf (not listed below) gave errors (both with and without caret) and could not be included in this work. In the R and caret implementations, we specify the function and, in typewriter font, the package which provide that classifier (the function name is absent when it is is equal to the classifier).

7. See `http://caret.r-forge.r-project.org`.

Discriminant analysis (DA): 20 classifiers.

1. **lda_R**, linear discriminant analysis, with the function lda in the `MASS` package.
2. **lda2_t**, from the `MASS` package, which develops LDA tuning the number of components to retain up to $\#classes - 1$.
3. **rrlda_R**, robust regularized LDA, from the `rrlda` package, tunes the parameters lambda (which controls the sparseness of the covariance matrix estimation) and alpha (robustness, it controls the number of outliers) with values {0.1, 0.01, 0.001} and {0.5, 0.75, 1.0} respectively.
4. **sda_t**, shrinkage discriminant analysis and CAT score variable selection (Ahdesmäki and Strimmer, 2010) from the `sda` package. It performs LDA or diagonal discriminant analysis (DDA) with variable selection using CAT (Correlation-Adjusted T) scores. The best classifier (LDA or DDA) is selected. The James-Stein method is used for shrinkage estimation.
5. **slda_t** with function slda from the `ipred` package, which develops LDA based on left-spherically distributed linear scores (Glimm et al., 1998).
6. **stepLDA_t** uses the function train in the `caret` package as interface to the function stepclass in the `klaR` package with method=lda. It develops classification by means of forward/backward feature selection, without upper bounds in the number of features.
7. **sddaLDA_R**, stepwise diagonal discriminant analysis, with function sdda in the `SDDA` package with method=lda. It creates a diagonal discriminant rule adding one input at a time using a forward stepwise strategy and LDA.
8. **PenalizedLDA_t** from the `penalizedLDA` package: it solves the high-dimensional discriminant problem using a diagonal covariance matrix and penalizing the discriminant vectors with lasso or fussed coefficients (Witten and Tibshirani, 2011). The lasso penalty parameter (lambda) is tuned with values $\{0.1, 0.0031, 10^{-4}\}$.
9. **sparseLDA_R**, with function sda in the `sparseLDA` package, minimizing the SDA criterion using an alternating method (Clemensen et al., 2011). The parameter lambda is tuned with values $0, \{10^i\}_{-1}^{4}$. The number of components is tuned from 2 to $\#classes - 1$.
10. **qda_t**, quadratic discriminant analysis (Venables and Ripley, 2002), with function qda in the `MASS` package.
11. **QdaCov_t** in the `rrcov` package, which develops Robust QDA (Todorov and Filzmoser, 2009).
12. **sddaQDA_R** uses the function sdda in the `SDDA` package with method=qda.
13. **stepQDA_t** uses function stepclass in the `klaR` package with method=qda, forward / backward variable selection (parameter direction=both) and without limit in the number of selected variables (maxvar=Inf).

14. **fda_R**, flexible discriminant analysis (Hastie et al., 1993), with function fda in the `mda` package and the default linear regression method.

15. **fda_t** is the same FDA, also with linear regression but tuning the parameter nprune with values 2:3:15 (5 values).

16. **mda_R**, mixture discriminant analysis (Hastie and Tibshirani, 1996), with function mda in the `mda` package.

17. **mda_t** uses the `caret` package as interface to function mda, tuning the parameter subclasses between 2 and 11.

18. **pda_t**, penalized discriminant analysis, uses the function gen.rigde in the `mda` package, which develops PDA tuning the shrinkage penalty coefficient lambda with values from 1 to 10.

19. **rda_R**, regularized discriminant analysis (Friedman, 1989), uses the function rda in the `klaR` package. This method uses regularized group covariance matrix to avoid the problems in LDA derived from collinearity in the data. The parameters lambda and gamma (used in the calculation of the robust covariance matrices) are tuned with values 0:0.25:1.

20. **hdda_R**, high-dimensional discriminant analysis (Bergé et al., 2012), assumes that each class lives in a different Gaussian subspace much smaller than the input space, calculating the subspace parameters in order to classify the test patterns. It uses the hdda function in the `HDclassif` package, selecting the best of the 14 available models.

Bayesian (BY) approaches: 6 classifiers.

21. **naiveBayes_R** uses the function NaiveBayes in R the `klaR` package, with Gaussian kernel, bandwidth 1 and Laplace correction 2.

22. **vbmpRadial_t**, variational Bayesian multinomial probit regression with Gaussian process priors (Girolami and Rogers, 2006), uses the function vbmp from the `vbmp` package, which fits a multinomial probit regression model with radial basis function kernel and covariance parameters estimated from the training patterns.

23. **NaiveBayes_w** (John and Langley, 1995) uses estimator precision values chosen from the analysis of the training data.

24. **NaiveBayesUpdateable_w** uses estimator precision values updated iteratively using the training patterns and starting from the scratch.

25. **BayesNet_w** is an ensemble of Bayes classifiers. It uses the K2 search method, which develops hill climbing restricted by the input order, using one parent and scores of type Bayes. It also uses the simpleEstimator method, which uses the training patterns to estimate the conditional probability tables in a Bayesian network once it has been learnt, which $\alpha = 0.5$ (initial count).

26. **NaiveBayesSimple_w** is a simple naive Bayes classifier (Duda et al., 2001) which uses a normal distribution to model numeric features.

Neural networks (NNET): 21 classifiers.

27. **rbf_m**, radial basis functions (RBF) neural network, uses the function newrb in the Matlab Neural Network Toolbox, tuning the spread of the Gaussian basis function with 19 values between 0.1 and 70. The network is created empty and new hidden neurons are added incrementally.

28. **rbf_t** uses `caret` as interface to the `RSNNS` package, tuning the size of the RBF network (number of hidden neurons) with values in the range 11:2:29.

29. **RBFNetwork_w** uses K-means to select the RBF centers and linear regression to learn the classification function, with symmetric multivariate Gaussians and normalized inputs. We use a number of clusters (or hidden neurons) equal to half the training patterns, ridge=10^{-8} for the linear regression and Gaussian minimum spread 0.1.

30. **rbfDDA_t** (Berthold and Diamond, 1995) creates incrementally from the scratch a RBF network with dynamic decay adjustment (DDA), using the `RSNNS` package and tuning the negativeThreshold parameter with values $\{10^{-i}\}_1^{10}$. The network grows incrementally adding new hidden neurons, avoiding the tuning of the network size.

31. **mlp_m**: multi-layer perceptron (MLP) implemented in Matlab (function newpr) tuning the number of hidden neurons with 11 values from 3 to 30.

32. **mlp_C**: MLP implemented in C using the fast artificial neural network (FANN) library[8], tuning the training algorithm (resilient, batch and incremental backpropagation, and quickprop), and the number of hidden neurons with 11 values between 3 and 30.

33. **mlp_t** uses the function mlp in the `RSNNS` package, tuning the network size with values 1:2:19.

34. **avNNet_t**, from the `caret` package, creates a committee of 5 MLPs (the number of MLPs is given by parameter repeat) trained with different random weight initializations and bag=false. The tunable parameters are the #hidden neurons (size) in {1, 3, 5} and the weight decay (values {0, 0.1, 10^{-4}}). This low number of hidden neurons is to reduce the computational cost of the ensemble.

35. **mlpWeightDecay_t** uses `caret` to access the `RSNNS` package tuning the parameters size and weight decay of the MLP network with values 1:2:9 and {0, 0.1, 0.01, 0.001, 0.0001} respectively.

36. **nnet_t** uses `caret` as interface to function nnet in the `nnet` package, training a MLP network with the same parameter tuning as in mlpWeightDecay_t.

37. **pcaNNet_t** trains the MLP using `caret` and the `nnet` package, but running principal component analysis (PCA) previously on the data set.

8. See `http://leenissen.dk/fann/wp`.

38. **MultilayerPerceptron_w** is a MLP network with sigmoid hidden neurons, unthresholded linear output neurons, learning rate 0.3, momentum 0.2, 500 training epochs, and #hidden neurons equal (#inputs and #classes)/2.

39. **pnn_m**: probabilistic neural network (Specht, 1990) in Matlab (function newpnn), tuning the Gaussian spread with 19 values in the range 0.01-10.

40. **elm_m**, extreme learning machine (Huang et al., 2012) implemented in Matlab using the code freely available[9]. We try 6 activation functions (sine, sign, sigmoid, hardlimit, triangular basis and radial basis) and 20 values for #hidden neurons between 3 and 200. As recommended, the inputs are scaled between [-1,1].

41. **elm_kernel_m** is the ELM with Gaussian kernel, which uses the code available from the previous site, tuning the regularization parameter and the kernel spread with values $2^{-5}..2^{14}$ and $2^{-16}..2^{8}$ respectively.

42. **cascor_C**, cascade correlation neural network (Fahlman, 1988) implemented in C using the FANN library (see classifier #32).

43. **lvq_R** is the learning vector quantization (Ripley, 1996) implemented using the function lvq in the `class` package, with codebook of size 50, and k=5 nearest neighbors. We selected the best results achieved using the functions lvq1, olvq2, lvq2 and lvq3.

44. **lvq_t** uses `caret` as interface to function lvq1 in the `class` package tuning the parameters size and k (the values are specific for each data set).

45. **bdk_R**, bi-directional Kohonen map (Melssen et al., 2006), with function bdk in the `kohonen` package, a kind of supervised Self Organized Map for classification, which maps high-dimensional patterns to 2D.

46. **dkp_C** (direct kernel perceptron) is a very simple and fast kernel-based classifier proposed by us (Fernández-Delgado et al., 2014) which achieves competitive results compared to SVM. The DKP requires the tuning of the kernel spread in the same range $2^{-16}..2^{8}$ as the SVM.

47. **dpp_C** (direct parallel perceptron) is a small and efficient Parallel Perceptron network proposed by us (Fernández-Delgado et al., 2011), based in the parallel-delta rule (Auer et al., 2008) with $n = 3$ perceptrons. The codes for DKP and DPP are freely available[10].

Support vector machines (SVM): 10 classifiers.

48. **svm_C** is the support vector machine, implemented in C using LibSVM (Chang and Lin, 2008) with Gaussian kernel. The regularization parameter C and kernel spread gamma are tuned in the ranges $2^{-5}..2^{14}$ and $2^{-16}..2^{8}$ respectively. LibSVM uses the one-vs.-one approach for multi-class data sets.

9. See `http://www.extreme-learning-machines.org`.
10. See `http://persoal.citius.usc.es/manuel.fernandez.delgado/papers/jmlr`.

49. **svmlight_C** (Joachims, 1999) is a very popular implementation of the SVM in C. It can only be used from the command-line and not as a library, so we could not use it so efficiently as LibSVM, and this fact leads us to errors for some large data sets (which are not taken into account in the calculation of the average accuracy). The parameters C and gamma (spread of the Gaussian kernel) are tuned with the same values as svm_C.

50. **LibSVM_w** uses the library LibSVM (Chang and Lin, 2008), calls from Weka for classification with Gaussian kernel, using the values of C and gamma selected for svm_C and tolerance=0.001.

51. **LibLINEAR_w** uses the library LibLinear (Fan et al., 2008) for large-scale linear high-dimensional classification, with L2-loss (dual) solver and parameters C=1, tolerance=0.01 and bias=1.

52. **svmRadial_t** is the SVM with Gaussian kernel (in the `kernlab` package), tuning C and kernel spread with values $2^{-2}..2^{2}$ and $10^{-2}..10^{2}$ respectively.

53. **svmRadialCost_t** (`kernlab` package) only tunes the cost C, while the spread of the Gaussian kernel is calculated automatically.

54. **svmLinear_t** uses the function ksvm (`kernlab` package) with linear kernel tuning C in the range $2^{-2}..2^{7}$.

55. **svmPoly_t** uses the `kernlab` package with linear, quadratic and cubic kernels $(s\mathbf{x}^T\mathbf{y}+o)^d$, using scale $s = \{0.001, 0.01, 0.1\}$, offset $o = 1$, degree $d = \{1, 2, 3\}$ and $C = \{0.25, 0.5, 1\}$.

56. **lssvmRadial_t** implements the least squares SVM (Suykens and Vandewalle, 1999), using the function lssvm in the `kernlab` package, with Gaussian kernel tuning the kernel spread with values $10^{-2}..10^{7}$.

57. **SMO_w** is a SVM trained using sequential minimal optimization (Platt, 1998) with one-against-one approach for multi-class classification, C=1, tolerance L=0.001, round-off error 10^{-12}, data normalization and quadratic kernel.

Decision trees (DT): 14 classifiers.

58. **rpart_R** uses the function rpart in the `rpart` package, which develops recursive partitioning (Breiman et al., 1984).

59. **rpart_t** uses the same function tuning the complexity parameter (threshold on the accuracy increasing achieved by a tentative split in order to be accepted) with 10 values from 0.18 to 0.01.

60. **rpart2_t** uses the function rpart tuning the tree depth with values up to 10.

61. **obliqueTree_R** uses the function obliqueTree in the `oblique.tree` package (Truong, 2009), with binary recursive partitioning, only oblique splits and linear combinations of the inputs.

62. **C5.0Tree_t** creates a single C5.0 decision tree (Quinlan, 1993) using the function C5.0 in the homonymous package without parameter tuning.

63. **ctree_t** uses the function ctree in the `party` package, which creates conditional inference trees by recursively making binary splittings on the variables with the highest association to the class (measured by a statistical test). The threshold in the association measure is given by the parameter mincriterion, tuned with the values 0.1:0.11:0.99 (10 values).

64. **ctree2_t** uses the function ctree tuning the maximum tree depth with values up to 10.

65. **J48_w** is a pruned C4.5 decision tree (Quinlan, 1993) with pruning confidence threshold C=0.25 and at least 2 training patterns per leaf.

66. **J48_t** uses the function J48 in the `RWeka` package, which learns pruned or unpruned C5.0 trees with C=0.25.

67. **RandomSubSpace_w** (Ho, 1998) trains multiple REPTrees classifiers selecting randomly subsets of inputs (random subspaces). Each REPTree is learnt using information gain/variance and error-based pruning with backfitting. Each subspace includes the 50% of the inputs. The minimum variance for splitting is 10^{-3}, with at least 2 pattern per leaf.

68. **NBTree_w** (Kohavi, 1996) is a decision tree with naive Bayes classifiers at the leafs.

69. **RandomTree_w** is a non-pruned tree where each leaf tests $\lfloor log_2(\#inputs + 1) \rfloor$ randomly chosen inputs, with at least 2 instances per leaf, unlimited tree depth, without backfitting and allowing unclassified patterns.

70. **REPTree_w** learns a pruned decision tree using information gain and reduced error pruning (REP). It uses at least 2 training patterns per leaf, 3 folds for reduced error pruning and unbounded tree depth. A split is executed when the class variance is more than 0.001 times the train variance.

71. **DecisionStump_w** is a one-node decision tree which develops classification or regression based on just one input using entropy.

Rule-based methods (RL): 12 classifiers.

72. **PART_w** builds a pruned partial C4.5 decision tree (Frank and Witten, 1999) in each iteration, converting the best leaf into a rule. It uses at least 2 objects per leaf, 3-fold REP (see classifier #70) and C=0.5.

73. **PART_t** uses the function PART in the `RWeka` package, which learns a pruned PART with C=0.25.

74. **C5.0Rules_t** uses the same function C5.0 (in the `C50` package) as classifiers C5.0Tree_t, but creating a collection of rules instead of a classification tree.

75. **JRip_t** uses the function JRip in the `RWeka` package, which learns a "repeated incremental pruning to produce error reduction" (RIPPER) classifier (Cohen, 1995), tuning the number of optimization runs (numOpt) from 1 to 5.

76. **JRip_w** learns a RIPPER classifier with 2 optimization runs and minimal weights of instances equal to 2.

77. **OneR_t** (Holte, 1993) uses function OneR in the `RWeka` package, which classifies using 1-rules applied on the input with the lowest error.

78. **OneR_w** creates a OneR classifier in Weka with at least 6 objects in a bucket.

79. **DTNB_w** learns a decision table/naive-Bayes hybrid classifier (Hall and Frank, 2008), using simultaneously both decision table and naive Bayes classifiers.

80. **Ridor_w** implements the ripple-down rule learner (Gaines and Compton, 1995) with at least 2 instance weights.

81. **ZeroR_w** predicts the mean class (i.e., the most populated class in the training data) for all the test patterns. Obviously, this classifier gives low accuracies, but it serves to give a lower limit on the accuracy.

82. **DecisionTable_w** (Kohavi, 1995) is a simple decision table majority classifier which uses BestFirst as search method.

83. **ConjunctiveRule_w** uses a single rule whose antecendent is the AND of several antecedents, and whose consequent is the distribution of available classes. It uses the antecedent information gain to classify each test pattern, and 3-fold REP (see classifier #70) to remove unnecessary rule antecedents.

Boosting (BST): 20 classifiers.

84. **adaboost_R** uses the function boosting in the `adabag` package (Alfaro et al., 2007), which implements the adaboost.M1 method (Freund and Schapire, 1996) to create an adaboost ensemble of classification trees.

85. **logitboost_R** is an ensemble of DecisionStump base classifiers (see classifier #71), using the function LogitBoost (Friedman et al., 1998) in the `caTools` package with 200 iterations.

86. **LogitBoost_w** uses additive logistic regressors (DecisionStump) base learners, the 100% of weight mass to base training on, without cross-validation, one run for internal cross-validation, threshold 1.79 on likelihood improvement, shrinkage parameter 1, and 10 iterations.

87. **RacedIncrementalLogitBoost_w** is a raced Logitboost committee (Frank et al., 2002) with incremental learning and DecisionStump base classifiers, chunks of size between 500 and 2000, validation set of size 1000 and log-likelihood pruning.

88. **AdaBoostM1_DecisionStump_w** implements the same Adaboost.M1 method with DecisionStump base classifiers.

89. **AdaBoostM1_J48_w** is an Adaboost.M1 ensemble which combines J48 base classifiers.

90. **C5.0_t** creates a Boosting ensemble of C5.0 decision trees and rule models (function C5.0 in the hononymous package), with and without winnow (feature selection), tuning the number of boosting trials in {1, 10, 20}.

91. **MultiBoostAB_DecisionStump_w** (Webb, 2000) is a MultiBoost ensemble, which combines Adaboost and Wagging using DecisionStump base classifiers, 3 sub-committees, 10 training iterations and 100% of the weight mass to base training on. The same options are used in the following MultiBoostAB ensembles.

92. **MultiBoostAB_DecisionTable_w** combines MultiBoost and DecisionTable, both with the same options as above.

93. **MultiBoostAB_IBk_w** uses MultiBoostAB with IBk base classifiers (see classifier #157).

94. **MultiBoostAB_J48_w** trains an ensemble of J48 decision trees, using pruning confidence C=0.25 and 2 training patterns per leaf.

95. **MultiBoostAB_LibSVM_w** uses LibSVM base classifiers with the optimal C and Gaussian kernel spread selected by the svm_C classifier (see classifier #48). We included it for comparison with previous papers (Vanschoren et al., 2012), although a strong classifier as LibSVM is in principle not recommended to use as base classifier.

96. **MultiBoostAB_Logistic_w** combines Logistic base classifiers (see classifier #86).

97. **MultiBoostAB_MultilayerPerceptron_w** uses MLP base classifiers with the same options as MultilayerPerceptron_w (which is another strong classifier).

98. **MultiBoostAB_NaiveBayes_w** uses NaiveBayes base classifiers.

99. **MultiBoostAB_OneR_w** uses OneR base classifiers.

100. **MultiBoostAB_PART_w** combines PART base classifiers.

101. **MultiBoostAB_RandomForest_w** combines RandomForest base classifiers. We tried this classifier for comparison with previous papers (Vanschoren et al., 2012), despite of RandomForest is itself an ensemble, so it seems not very useful to learn a MultiBoostAB ensemble of RandomForest ensembles.

102. **MultiBoostAB_RandomTree_w** uses RandomTrees with the same options as above.

103. **MultiBoostAB_REPTree_w** uses REPTree base classifiers.

Bagging (BAG): 24 classifiers.

104. **bagging_R** is a bagging (Breiman, 1996) ensemble of decision trees using the function bagging (in the `ipred` package).

105. **treebag_t** trains a bagging ensemble of classification trees using the `caret` interface to function bagging in the `ipred` package.

106. **ldaBag_R** creates a bagging ensemble of LDAs, using the function bag of the `caret` package (instead of the function train) with option bagControl=ldaBag.

107. **plsBag_R** is the previous one with bagControl=plsBag.

108. **nbBag_R** creates a bagging of naive Bayes classifiers using the previous bag function with bagControl=nbBag.

109. **ctreeBag_R** uses the same function bag with bagControl=ctreeBag (conditional inference tree base classifiers).

110. **svmBag_R** trains a bagging of SVMs, with bagControl=svmBag.

111. **nnetBag_R** learns a bagging of MLPs with bagControl=nnetBag.

112. **MetaCost_w** (Domingos, 1999) is based on bagging but using cost-sensitive ZeroR base classifiers and bags of the same size as the training set (the following bagging ensembles use the same configuration). The diagonal of the cost matrix is null and the remaining elements are one, so that each type of error is equally weighted.

113. **Bagging_DecisionStump_w** uses DecisionStump base classifiers with 10 bagging iterations.

114. **Bagging_DecisionTable_w** uses DecisionTable with BestFirst and forward search, leave-one-out validation and accuracy maximization for the input selection.

115. **Bagging_HyperPipes_w** with HyperPipes base classifiers.

116. **Bagging_IBk_w** uses IBk base classifiers, which develop KNN classification tuning K using cross-validation with linear neighbor search and Euclidean distance.

117. **Bagging_J48_w** with J48 base classifiers.

118. **Bagging_LibSVM_w**, with Gaussian kernel for LibSVM and the same options as the single LibSVM_w classifier.

119. **Bagging_Logistic_w**, with unlimited iterations and log-likelihood ridge 10^{-8} in the Logistic base classifier.

120. **Bagging_LWL_w** uses LocallyWeightedLearning base classifiers (see classifier #148) with linear weighted kernel shape and DecisionStump base classifiers.

121. **Bagging_MultilayerPerceptron_w** with the same configuration as the single MultilayerPerceptron_w.

122. **Bagging_NaiveBayes_w** with NaiveBayes classifiers.

123. **Bagging_OneR_w** uses OneR base classifiers with at least 6 objects per bucket.

124. **Bagging_PART_w** with at least 2 training patterns per leaf and pruning confidence C=0.25.

125. **Bagging_RandomForest_w** with forests of 500 trees, unlimited tree depth and $\lfloor log(\#inputs + 1) \rfloor$ inputs.

126. **Bagging_RandomTree_w** with RandomTree base classifiers without backfitting, investigating $\lfloor log_2(\#inputs) + 1 \rfloor$ random inputs, with unlimited tree depth and 2 training patterns per leaf.

127. **Bagging_REPTree_w** use REPTree with 2 patterns per leaf, minimum class variance 0.001, 3-fold for reduced error pruning and unlimited tree depth.

Stacking (STC): 2 classifiers.

128. **Stacking_w** is a stacking ensemble (Wolpert, 1992) using ZeroR as meta and base classifiers.

129. **StackingC_w** implements a more efficient stacking ensemble following (Seewald, 2002), with linear regression as meta-classifier.

Random Forests (RF): 8 classifiers.

130. **rforest_R** creates a random forest (Breiman, 2001) ensemble, using the R function randomForest in the `randomForest` package, with parameters ntree = 500 (number of trees in the forest) and mtry=$\sqrt{\#inputs}$.

131. **rf_t** creates a random forest using the `caret` interface to the function randomForest in the `randomForest` package, with ntree = 500 and tuning the parameter mtry with values 2:3:29.

132. **RRF_t** learns a regularized random forest (Deng and Runger, 2012) using `caret` as interface to the function RRF in the `RRF` package, with mtry=2 and tuning parameters coefReg={0.01, 0.5, 1} and coefImp={0, 0.5, 1}.

133. **cforest_t** is a random forest and bagging ensemble of conditional inference trees (ctrees) aggregated by averaging observation weights extracted from each ctree. The parameter mtry takes the values 2:2:8. It uses the `caret` package to access the `party` package.

134. **parRF_t** uses a parallel implementation of random forest using the `randomForest` package with mtry=2:2:8.

135. **RRFglobal_t** creates a RRF using the hononymous package with parameters mtry=2 and coefReg=0.01:0.12:1.

136. **RandomForest_w** implements a forest of RandomTree base classifiers with 500 trees, using $\lfloor log(\#inputs + 1) \rfloor$ inputs and unlimited depth trees.

137. **RotationForest_w** (Rodríguez et al., 2006) uses J48 as base classifier, principal component analysis filter, groups of 3 inputs, pruning confidence C=0.25 and 2 patterns per leaf.

Other ensembles (OEN): 11 classifiers.

138. **RandomCommittee_w** is an ensemble of RandomTrees (each one built using a different seed) whose output is the average of the base classifier outputs.

139. **OrdinalClassClassifier_w** is an ensemble method designed for ordinal classification problems (Frank and Hall, 2001) with J48 base classifiers, confidence threshold C=0.25 and 2 training patterns per leaf.

140. **MultiScheme_w** selects a classifier among several ZeroR classifiers using cross validation on the training set.

141. **MultiClassClassifier_w** solves multi-class problems with two-class Logistic_w base classifiers, combined with the One-Against-All approach, using multinomial logistic regression.

142. **CostSensitiveClassifier_w** combines ZeroR base classifiers on a training set where each pattern is weighted depending on the cost assigned to each error type. Similarly to MetaCost_w (see classifier #112), all the error types are equally weighted.

143. **Grading_w** is Grading ensemble (Seewald and Fuernkranz, 2001) with "graded" ZeroR base classifiers.

144. **END_w** is an Ensemble of Nested Dichotomies (Frank and Kramer, 2004) which classifies multi-class data sets with two-class J48 tree classifiers.

145. **Decorate_w** learns an ensemble of fifteen J48 tree classifiers with high diversity trained with specially constructed artificial training patterns (Melville and Mooney, 2004).

146. **Vote_w** (Kittler et al., 1998) trains an ensemble of ZeroR base classifiers combined using the average rule.

147. **Dagging_w** (Ting and Witten, 1997) is an ensemble of SMO_w (see classifier #57), with the same configuration as the single SMO classifier, trained on 4 different folds of the training data. The output is decided using the previous Vote_w meta-classifier.

148. **LWL_w**, Local Weighted Learning (Frank et al., 2003), is an ensemble of DecisionStump base classifiers. Each training pattern is weighted with a linear weighting kernel, using the Euclidean distance for a linear search of the nearest neighbor.

Generalized Linear Models (GLM): 5 classifiers.

149. **glm_R** (Dobson, 1990) uses the function glm in the `stats` package, with binomial and Poisson families for two-class and multi-class problems respectively.

150. **glmnet_R** trains a GLM via penalized maximum likelihood, with Lasso or elasticnet regularization parameter (Friedman et al., 2010) (function glmnet in the `glmnet` package). We use the binomial and multinomial distribution for two-class and multi-class problems respectively.

151. **mlm_R** (Multi-Log Linear Model) uses the function multinom in the `nnet` package, fitting the multi-log model with MLP neural networks.

152. **bayesglm_t**, Bayesian GLM (Gelman et al., 2009), with function bayesglm in the `arm` package. It creates a GLM using Bayesian functions, an approximated expectation-maximization method, and augmented regression to represent the prior probabilities.

153. **glmStepAIC_t** performs model selection by Akaike information criterion (Venables and Ripley, 2002) using the function stepAIC in the `MASS` package.

Nearest neighbor methods (NN): 5 classifiers.

154. **knn_R** uses the function knn in the `class` package, tuning the number of neighbors with values 1:2:37 (13 values).

155. **knn_t** uses function knn in the `caret` package with 10 number of neighbors in the range 5:2:23.

156. **NNge_w** is a NN classifier with non-nested generalized exemplars (Martin, 1995), using one folder for mutual information computation and 5 attempts for generalization.

157. **IBk_w** (Aha et al., 1991) is a KNN classifier which tunes K using cross-validation with linear neighbor search and Euclidean distance.

158. **IB1_w** is a simple 1-NN classifier.

Partial least squares and principal component regression (PLSR): 6 classifiers.

159. **pls_t** uses the function mvr in the `pls` package to fit a PLSR (Martens, 1989) model tuning the number of components from 1 to 10.

160. **gpls_R** trains a generalized PLS (Ding and Gentleman, 2005) model using the function gpls in the `gpls` package.

161. **spls_R** uses the function spls in the `spls` package to fit a sparse partial least squares (Chun and Keles, 2010) regression model tuning the parameters K and eta with values {1, 2, 3} and {0.1, 0.5, 0.9} respectively.

162. **simpls_R** fits a PLSR model using the SIMPLS (Jong, 1993) method, with the function plsr (in the `pls` package) and method=simpls.

163. **kernelpls_R** (Dayal and MacGregor, 1997) uses the same function plsr with method = kernelpls, with up to 8 principal components (always lower than $\#inputs - 1$). This method is faster when #patterns is much larger than #inputs.

164. **widekernelpls_R** fits a PLSR model with the function plsr and method = widekernelpls, faster when #inputs is larger than #patterns.

Logistic and multinomial regression (LMR): 3 classifiers.

165. **SimpleLogistic_w** learns linear logistic regression models (Landwehr et al., 2005) for classification. The logistic models are fitted using LogitBoost with simple regression functions as base classifiers.

166. **Logistic_w** learns a multinomial logistic regression model (Cessie and Houwelingen, 1992) with a ridge estimator, using ridge in the log-likelihood R=10^{-8}.

167. **multinom_t** uses the function multinom in the `nnet` package, which trains a MLP to learn a multinomial log-linear model. The parameter decay of the MLP is tuned with 10 values between 0 and 0.1.

Multivariate adaptive regression splines (MARS): 2 classifiers.

168. **mars_R** fits a MARS (Friedman, 1991) model using the function mars in the `mda` package.

169. **gcvEarth_t** uses the function earth in the `earth` package. It builds an additive MARS model without interaction terms using the fast MARS (Hastie et al., 2009) method.

Other Methods (OM): 10 classifiers.

170. **pam_t** (nearest shrunken centroids) uses the function pamr in the `pamr` package (Tibshirani et al., 2002).

171. **VFI_w** develops classification by voting feature intervals (Demiroz and Guvenir, 1997), with B=0.6 (exponential bias towards confident intervals).

172. **HyperPipes_w** classifies each test pattern to the class which most contains the pattern. Each class is defined by the bounds of each input in the patterns which belong to that class.

173. **FilteredClassifier_w** trains a J48 tree classifier on data filtered using the Discretize filter, which discretizes numerical into nominal attributes.

174. **CVParameterSelection_w** (Kohavi, 1995) selects the best parameters of classifier ZeroR using 10-fold cross-validation.

175. **ClassificationViaClustering_w** uses SimpleKmeans and EuclideanDistance to cluster the data. Following the Weka documentation, the number of clusters is set to #classes.

176. **AttributeSelectedClassifier_w** uses J48 trees to classify patterns reduced by attribute selection. The CfsSubsetEval method (Hall, 1998) selects the best group of attributes weighting their individual predictive ability and their degree of redundancy, preferring groups with high correlation within classes and low inter-class correlation. The BestFirst forward search method is used, stopping the search when five non-improving nodes are found.

177. **ClassificationViaRegression_w** (Frank et al., 1998) binarizes each class and learns its corresponding M5P tree/rule regression model (Quinlan, 1992), with at least 4 training patterns per leaf.

178. **KStar_w** (Cleary and Trigg, 1995) is an instance-based classifier which uses entropy-based similarity to assign a test pattern to the class of its nearest training patterns.

179. **gausspr Radial_t** uses the function gausspr in the `kernlab` package, which trains a Gaussian process-based classifier, with kernel= rbfdot and kernel spread (parameter sigma) tuned with values $\{10^i\}_{-2}^{7}$.

3. Results and Discussion

In the experimental work we evaluate 179 classifiers over 121 data sets, giving 21,659 combinations classifier-data set. We use Weka v. 3.6.8, R v. 2.15.3 with caret v. 5.16-04, Matlab v. 7.9.0 (R2009b) with Neural Network Toolbox v. 6.0.3, the C/C++ compiler v. gcc/g++ 4.7.2 and fast artificial neural networks (FANN) library v. 2.2.0 on a computer with Debian GNU/Linux v. 3.2.46-1 (64 bits). We found errors with some classifiers and data sets caused by a variety of reasons. Some classifiers (lda_R, qda_t, QdaCov_t, among others) give errors in some data sets due to collinearity of data, singular covariance matrices, and equal inputs for all the training patterns in some classes; rrlda_R requires that all the inputs must have different values in more than 50% of the training patterns; other errors are caused by discrete inputs, classes with low populations (specially in data sets with many classes), or too few classes (vbmpRadial requires 3 classes). Large data sets (miniboone and connect-4) give some lack of memory errors, and few small data sets (trains and balloons) give errors for some Weka classifiers requiring a minimum #patterns per class. Overall, we found 449 errors, which represent 2.1% of the 21,659 cases. These error cases are excluded from the average accuracy calculation for each classifier.

The validation methodology is the following. One training and one test set are generated randomly (each with 50% of the available patterns), but imposing that each class has the same number of training and test patterns (in order to have enough training and test patterns of every class). This couple of sets is used only for **parameter tuning** (in those classifiers which have tunable parameters), selecting the parameter values which provide the best accuracy on the test set. The indexes of the training and test patterns (i.e., the data partitioning) are given by the file `conxuntos.dat` for each data set, and are the same for all the classifiers. Then, using the selected values for the tunable parameters, a **4-fold cross validation** is developed using the whole available data. The indexes of the training and test patterns for each fold are the same for all the classifiers, and they are listed in the file `conxuntos_kfold.dat` for each data set. The test results is the average over the 4 test sets. However, for some data sets, which provide **separate data for training and testing** (data sets annealing and audiology-std, among others), the classifier (with the tuned parameter values) is trained and tested on the respective data sets. In this case, the test result is calculated on the test set. We used this methodology in order to keep low the computational cost of the experimental work. However, we are aware of that this methodology may lead to poor bias and variance, and that the classifier results for each data

Rank	Acc.	κ	Classifier	Rank	Acc.	κ	Classifier
32.9	82.0	63.5	parRF_t (RF)	67.3	77.7	55.6	pda_t (DA)
33.1	**82.3**	**63.6**	rf_t (RF)	67.6	78.7	55.2	elm_m (NNET)
36.8	81.8	62.2	svm_C (SVM)	67.6	77.8	54.2	SimpleLogistic_w (LMR)
38.0	81.2	60.1	svmPoly_t (SVM)	69.2	78.3	57.4	MAB_J48_w (BST)
39.4	81.9	62.5	rforest_R (RF)	69.8	78.8	56.7	BG_REPTree_w (BAG)
39.6	82.0	62.0	elm_kernel_m (NNET)	69.8	78.1	55.4	SMO_w (SVM)
40.3	81.4	61.1	svmRadialCost_t (SVM)	70.6	78.3	58.0	MLP_w (NNET)
42.5	81.0	60.0	svmRadial_t (SVM)	71.0	78.8	58.23	BG_RandomTree_w (BAG)
42.9	80.6	61.0	C5.0_t (BST)	71.0	77.1	55.1	mlm_R (GLM)
44.1	79.4	60.5	avNNet_t (NNET)	71.0	77.8	56.2	BG_J48_w (BAG)
45.5	79.5	61.0	nnet_t (NNET)	72.0	75.7	52.6	rbf_t (NNET)
47.0	78.7	59.4	pcaNNet_t (NNET)	72.1	77.1	54.8	fda_R (DA)
47.1	80.8	53.0	BG_LibSVM_w (BAG)	72.4	77.0	54.7	lda_R (DA)
47.3	80.3	62.0	mlp_t (NNET)	72.4	79.1	55.6	svmlight_C (NNET)
47.6	80.6	60.0	RotationForest_w (RF)	72.6	78.4	57.9	AdaBoostM1_J48_w (BST)
50.1	80.9	61.6	RRF_t (RF)	72.7	78.4	56.2	BG_IBk_w (BAG)
51.6	80.7	61.4	RRFglobal_t (RF)	72.9	77.1	54.6	ldaBag_R (BAG)
52.5	80.6	58.0	MAB_LibSVM_w (BST)	73.2	78.3	56.2	BG_LWL_w (BAG)
52.6	79.9	56.9	LibSVM_w (SVM)	73.7	77.9	56.0	MAB_REPTree_w (BST)
57.6	79.1	59.3	adaboost_R (BST)	74.0	77.4	52.6	RandomSubSpace_w (DT)
58.5	79.7	57.2	pnn_m (NNET)	74.4	76.9	54.2	lda2_t (DA)
58.9	78.5	54.7	cforest_t (RF)	74.6	74.1	51.8	svmBag_R (BAG)
59.9	79.7	42.6	dkp_C (NNET)	74.6	77.5	55.2	LibLINEAR_w (SVM)
60.4	80.1	55.8	gausssprRadial_R (OM)	75.9	77.2	55.6	rbfDDA_t (NNET)
60.5	80.0	57.4	RandomForest_w (RF)	76.5	76.9	53.8	sda_t (DA)
62.1	78.7	56.0	svmLinear_t (SVM)	76.6	78.1	56.5	END_w (OEN)
62.5	78.4	57.5	fda_t (DA)	76.6	77.3	54.8	LogitBoost_w (BST)
62.6	78.6	56.0	knn_t (NN)	76.6	78.2	57.3	MAB_RandomTree_w (BST)
62.8	78.5	58.1	mlp_C (NNET)	77.1	78.4	54.0	BG_RandomForest_w (BAG)
63.0	79.9	59.4	RandomCommittee_w (OEN)	78.5	76.5	53.7	Logistic_w (LMR)
63.4	78.7	58.4	Decorate_w (OEN)	78.7	76.6	50.5	ctreeBag_R (BAG)
63.6	76.9	56.0	mlpWeightDecay_t (NNET)	79.0	76.8	53.5	BG_Logistic_w (BAG)
63.8	78.7	56.7	rda_R (DA)	79.1	77.4	53.0	lvq_t (NNET)
64.0	79.0	58.6	MAB_MLP_w (BST)	79.1	74.4	50.7	pls_t (PLSR)
64.1	79.9	56.9	MAB_RandomForest_w (BST)	79.8	76.9	54.7	hdda_R (DA)
65.0	79.0	56.8	knn_R (NN)	80.6	75.9	53.3	MCC_w (OEN)
65.2	77.9	56.2	multinom_t (LMR)	80.9	76.9	54.5	mda_R (DA)
65.5	77.4	56.6	gcvEarth_t (MARS)	81.4	76.7	55.2	C5.0Rules_t (RL)
65.5	77.8	55.7	glmnet_R (GLM)	81.6	78.3	55.8	lssvmRadial_t (SVM)
65.6	78.6	58.4	MAB_PART_w (BST)	81.7	75.6	50.9	JRip_t (RL)
66.0	78.5	56.5	CVR_w (OM)	82.0	76.1	53.3	MAB_Logistic_w (BST)
66.4	79.2	58.9	treebag_t (BAG)	84.2	75.8	53.9	C5.0Tree_t (DT)
66.6	78.2	56.8	BG_PART_w (BAG)	84.6	75.7	50.8	BG_DecisionTable_w (BAG)
66.7	75.5	55.2	mda_t (DA)	84.9	76.5	53.4	NBTree_w (DT)

Table 3: Friedman ranking, average accuracy and Cohen κ (both in %) for each classifier, ordered by increasing Friedman ranking. Continued in the Table 4. BG = Bagging, MAB=MultiBoostAB.

Rank	Acc.	κ	Classifier	Rank	Acc.	κ	Classifier
86.4	76.3	52.6	ASC_w (OM)	110.4	71.6	46.5	BG_NaiveBayes_w (BAG)
87.2	77.1	54.2	KStar_w (OM)	111.3	62.5	38.4	widekernelpls_R (PLSR)
87.2	74.6	50.3	MAB_DecisionTable_w (BST)	111.9	63.3	43.7	mars_R (MARS)
87.6	76.4	51.3	J48_t (DT)	111.9	62.2	39.6	simpls_R (PLSR)
87.9	76.2	55.0	J48_w (DT)	112.6	70.1	38.0	sddaLDA_R (DA)
88.0	76.0	51.7	PART_t (DT)	113.1	61.0	38.2	kernelpls_R (PLSR)
89.0	76.1	52.4	DTNB_w (RL)	113.3	68.2	39.5	sparseLDA_R (DA)
89.5	75.8	54.8	PART_w (DT)	113.5	70.1	46.5	NBUpdateable_w (BY)
90.2	76.6	48.5	RBFNetwork_w (NNET)	113.5	70.7	39.9	stepLDA_t (DA)
90.5	67.5	45.8	bagging_R (BAG)	114.8	58.1	32.4	bayesglm_t (GLM)
91.2	74.0	50.9	rpart_t (DT)	115.8	70.6	46.4	QdaCov_t (DA)
91.5	74.0	48.9	ctree_t (DT)	116.0	69.5	39.6	stepQDA_t (DA)
91.7	76.6	54.1	NNge_w (NN)	118.3	67.5	34.3	sddaQDA_R (DA)
92.4	72.8	48.5	ctree2_t (DT)	118.9	72.0	45.9	NaiveBayesSimple_w (BY)
93.0	74.7	50.1	FilteredClassifier_w (OM)	120.1	55.3	33.3	gpls_R (PLSR)
93.1	74.8	51.4	JRip_w (RL)	120.8	57.6	32.5	glmStepAIC_t (GLM)
93.6	75.3	51.1	REPTree_w (DT)	122.2	63.5	35.1	AdaBoostM1_w (BST)
93.6	74.7	52.3	rpart2_t (DT)	122.7	68.3	39.4	LWL_w (OEN)
94.3	75.1	50.7	BayesNet_w (BY)	126.1	50.8	30.5	glm_R (GLM)
94.4	73.5	49.5	rpart_R (DT)	126.2	65.7	44.7	dpp_C (NNET)
94.5	76.4	54.5	IB1_w (NN)	129.6	62.3	31.8	MAB_w (BST)
94.6	76.5	51.6	Ridor_w (RL)	130.9	64.2	33.2	BG_OneR_w (BAG)
95.1	71.8	48.7	lvq_R (NNET)	130.9	62.1	29.6	MAB_IBk_w (BST)
95.3	76.0	53.9	IBk_w (NN)	132.1	63.3	36.2	OneR_t (RL)
95.3	73.9	45.8	Dagging_w (OEN)	133.2	64.2	34.3	MAB_OneR_w (BST)
96.0	74.4	50.7	qda_t (DA)	133.4	63.3	33.3	OneR_w (RL)
96.5	71.9	48.1	obliqueTree_R (DT)	133.7	61.8	28.3	BG_DecisionStump_w (BAG)
97.0	68.9	42.0	plsBag_R (BAG)	135.5	64.9	42.4	VFI_w (OM)
97.2	73.9	52.1	OCC_w (OEN)	136.6	60.4	27.7	ConjunctiveRule_w (RL)
99.5	71.3	44.9	mlp_m (NNET)	137.5	60.3	26.5	DecisionStump_w (DT)
99.6	74.4	51.6	cascor_C (NNET)	138.0	56.6	15.1	RILB_w (BST)
99.8	75.3	52.7	bdk_R (NNET)	138.6	60.3	26.1	BG_HyperPipes_w (BAG)
100.8	73.8	48.9	nbBag_R (BAG)	143.3	53.2	17.9	spls_R (PLSR)
101.6	73.6	49.3	naiveBayes_R (BY)	143.8	57.8	24.3	HyperPipes_w (OM)
103.2	72.2	44.5	slda_t (DA)	145.8	53.9	15.3	BG_MLP_w (BAG)
103.6	72.8	41.3	pam_t (OM)	154.0	49.3	3.2	Stacking_w (STC)
104.5	62.6	33.1	nnetBag_R (BAG)	154.0	49.3	3.2	Grading_w (OEN)
105.5	72.1	46.7	DecisionTable_w (RL)	154.0	49.3	3.2	CVPS_w (OM)
106.2	72.7	48.0	MAB_NaiveBayes_w (BST)	154.1	49.3	3.2	StackingC_w (STC)
106.6	59.3	71.7	logitboost_R (BST)	154.5	49.2	7.6	MetaCost_w (BAG)
106.8	68.1	41.5	PenalizedLDA_R (DA)	154.6	49.2	2.7	ZeroR_w (RL)
107.5	72.5	48.3	NaiveBayes_w (BY)	154.6	49.2	2.7	MultiScheme_w (OEN)
108.1	69.4	44.6	rbf_m (NNET)	154.6	49.2	5.6	CSC_w (OEN)
108.2	71.5	49.8	rrlda_R (DA)	154.6	49.2	2.7	Vote_w (OEN)
109.4	65.2	46.5	vbmpRadial_t (BY)	157.4	52.1	25.13	CVC_w (OM)
110.0	73.9	51.0	RandomTree_w (DT)				

Table 4: Continuation of Table 3. ASC = AttributeSelectedClassifier, BG = Bagging, CSC = CostSensitiveClassifier, CVPS = CVParameterSelection, CVC = ClassificationViaClustering, CVR = ClassificationViaRegression, MAB = MultiBoostAB, MCC = MultiClassClassifier, MLP = MultilayerPerceptron, NBUpdeatable = NaiveBayesUpdateable, OCC = OrdinalClassClassifier, RILB = RacedIncrementalLogitBoost.

set may vary with respect to previous papers in the literature due to resampling differences. Although a leave-one-out validation might be more adequate (because it does not depend

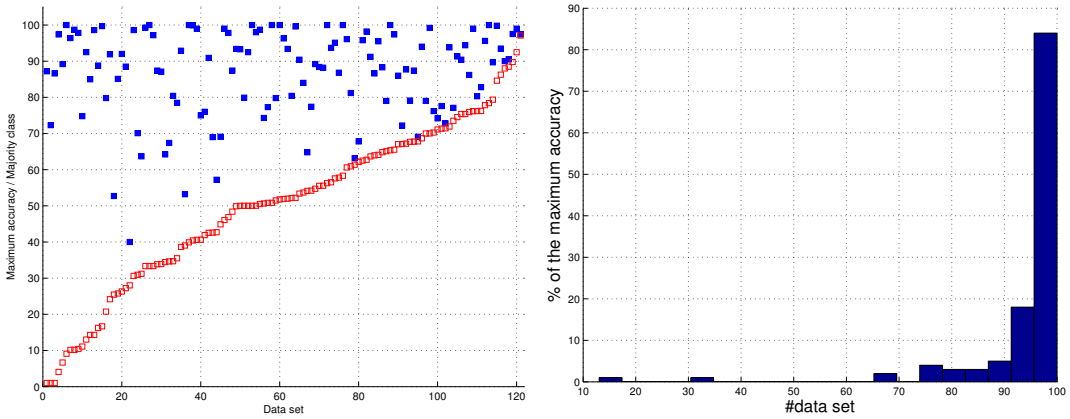


Figure 1: **Left**: Maximum accuracy (blue) and majority class (red), both in % ordered by increasing %Maj. for each data set. **Right**: Histogram of the accuracy achieved by parRF_t (measured as percentage of the best accuracy for each data set).

on the data partitioning), specially for the small data sets, it would not be feasible for some other larger data sets included in this study.

3.1 Average Accuracy and Friedman Ranking

Given its huge size (21,659 entries), the table with the complete results[11] is not included in the paper. Taking into account all the trials developed for parameter tuning in many classifiers (number of tunable parameters and number of values used for tuning), the total number of experiments is 241,637. The average accuracy for each classifier is calculated excluding the data sets in which that classifier found errors (denoted as -- in the complete table). The Figure 1 (left panel) plots, for each data set, the percentage of majority class (see columns %Maj. in Tables 1 and 2) and the maximum accuracy achieved by some classifier, ordered by increasing %Maj. Except for very few unbalanced data sets (with very populated majority classes), the best accuracy is much higher than the %Maj. (which is the accuracy achieved by classifier ZeroR_w). The Friedman ranking (Sheskin, 2006) was also computed to statistically sort the classifiers (this rank is increasing with the classifier error) taking into account the whole data set collection. Given that this test requires the same number of accuracy values for all the classifiers, in the error cases we use (only for this test) the average accuracy for that data set over all the classifiers.

The Tables 3 and 4 report the Friedman ranking, the average accuracy and the Cohen κ (Carletta, 1996), which excludes the probability of classifier success by chance, for the 179 classifiers, ordered following the Friedman ranking. The **best classifier is parRF_t (parallel random forest implemented in R using the randomForest and caret packages)**, with rank 32.9, average accuracy 82.0%($\pm$16.3) and κ=63.5%($\pm$30.6), **followed by rf_t (random forest using the randomForest package and tuned with caret)**, with rank 33.1 and the highest accuracy 82.3%($\pm$15.3) and κ=63.6($\pm$30.0). This result is

11. See `http://persoal.citius.usc.es/manuel.fernandez.delgado/papers/jmlr/results.txt`.

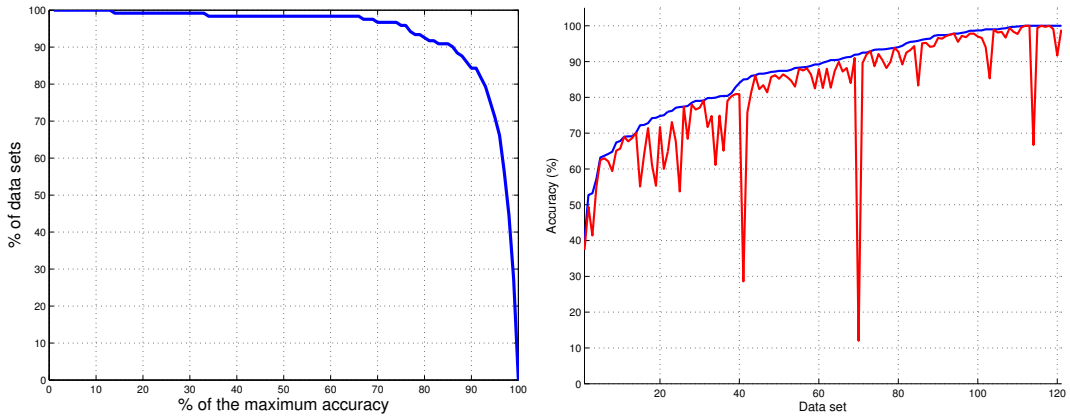


Figure 2: **Left**: for each % of the maximum accuracy in the horizontal axis, the vertical axis shows the percentage of data sets for which parRF_t overcomes that % of the maximum accuracy. **Right**: Accuracy (in %) achieved by parRF_t (in red) and maximum accuracy (in blue) for each data set (ordered by increasing maximum accuracies).

somehow surprising, because Random Forest is an old method, but it works better than other newer classifiers. The high deviations in accuracies and κ are expected, due to the large amount and variability of data sets. Since parRF_t is a parallel version of rf_t, using different random seeds, the difference between both can be considered not significant: parRF_t achieves better Friedman ranking, while rf_t achieves better accuracy and κ. Similar situations arise with other couples of classifiers within the same family, which are slightly different versions of the same classifier or versions with/without parameter tuning (svmRadial_t and svmRadialCost_t, lda_R and lda2_t, among others), with similar results, being the difference between them caused by noise, random initializations, etc. The parRF_t is the best classifier in 12 out of 121 data sets, and its average accuracy is 4.9% below the maximum average accuracy (i.e., the maximum accuracy over all the classifiers for each data set, averaged over all the data sets), which is 86.9%. It is very significant (and it can not be casual) that, among so many classifiers (179), the two bests ones (parRF_t and rf_t, according both to average accuracy and Friedman rank) are random forests implemented with the randomForest package and tuned with caret: this fact shows a clear superiority with respect to the remaining classifiers. It is also interesting that an "old" classifier as RF works better than many other, more recent, approaches. The Figure 1 (right panel) shows that for the majority of the data sets (specifically for 102 out of 121, which represents the 84.3%), the parRF_t achieves more than 90% of the maximum accuracy, being very near to the best accuracy for almost all the data sets. The Figure 2 (left panel) plots, for each % of the maximum accuracy in the horizontal axis, the % of data sets for which parRF overcomes that percentage: for the 93% (resp. for 84.3%) of the data sets parRF achieves more than 80% (resp. 90%) of the maximum accuracy. In this figure, the area under curve (AUC) of the three bests classifiers (parRF_t, rf_t and svm_C) are 0.9349, 0.9382 and 0.9312 respectively, being rf_t slightly better than parRF_t (as the accuracy in Table 3) and

svm_C slightly worse. As we commented in the introduction, given the large number of classifiers used in this work, it is reasonable to estimate the maximum attainable accuracy for a data set as the maximum accuracy achieved by some classifier. Therefore, although the No-Free-Lunch theorem states that no classifier can be always the best, in the practice, parRF_t is very near to the best attainable accuracy for almost all the data sets. Specifically, the Figure 2 (right panel) shows that parRF is very near to the maximum accuracy for almost all the data sets, excepting three data sets: #41 (image-segmentation, 33.6%), #70 (audiology-std, 13.0%) and #114 (balloons, 66.7%).

The third best classifier is svm_C (LibSVM with Gaussian kernel), with rank (36.8, two points above parRT_t) and average accuracy 81.8%($\pm$16.2). The following classifiers are: svmPoly_t (SVM with polynomial kernel, rank 38.0), rforest_R (random forest without mtry tuning, rank 39.4), elm_kernel_m (extreme learning machine, rank 39.6), svmRadialCost_t (40.3), svmRadial_t (42.5), C5.0_t (42.9) and avNNet_t (44.1). It may not be a casuality the presence of three RF and two SVM among the five best classifiers, identifying both classifier families as the best ones. Besides, there are also two neural networks and one boosting ensemble (C5.0_t) among the top-10. The Figure 3 shows the 25 classifiers with the lowest Friedman ranks (upper panel) and the classifiers with the highest average accuracies (lower panel): parRF_t and rf_t have ranks clearly lower than svm_C and the following classifiers. In fact, the highest increment (3.7) between two classifier ranks is between rf_t and svm_C, which shows that parRF_t and rf_t are clearly better than the remaining classifiers in the plot. Besides, rf_t, parRF, svm_C, rforest_R and elm_kernel_m have higher accuracies than the others (the largest accuracy reduction, 0.37, is between svm_C and svmRadialCost_t). Our proposal dkp_C is in the 23th (resp. 21th) position according to the Friedman ranking (resp. to the accuracy, 79.7%), but this apparently good result is somehow obscured by the low value of κ (42.6%). It is caused by some data sets where dkp_C assigns all the patterns to the most populated class: for these data sets, $\kappa = 0$, which reduces the average κ over all the data sets.

We developed **paired T-tests** comparing the accuracies of parRF_t and the following 9 classifiers in Table 3 (the null hypothesis is that the two accuracies compared are not significantly different, so that, within a tolerance $\alpha = 0.05$, when $p < 0.05$ parRF_t is significantly better than the other classifier. The Figure 4 (left panel) plots the T-statistic, 95%-limits and p-values, showing that parRF_t is only significantly better (high T-statistic, $p < 0.05$) than with C5.0_t and avNNet_t. Although parRF_t is better than svm_C in 56 of 121 data sets, worse than svm_C in 55 sets, and equal in 10 sets, the Figure 4 (right panel) compares their percentages of the maximum accuracy for each data set (ordered by increasing percentages): for the majority of the data sets they are almost 100% (i.e., parRF_t and svm_C are near to the maximum accuracy). Besides, svm_C is never much better than parRF_t: when svm_C outperforms parRF_t, the difference is small, but when parRF_t outperforms svm_C, the difference is higher (data sets 1-20). In fact, calculating for each data set the difference between the accuracies of parRF_t and svm_C, the sum of positive differences (parRF is better) is 193.8, while the negative ones (svm_C better) sum 139.8.

All the classifiers of the random forest and SVM families are included among the 25 best classifiers, with accuracies above 79% (while the best is 82.3%), which identify both families as the best ones. Other classifiers included among the top-20, not belonging to RF

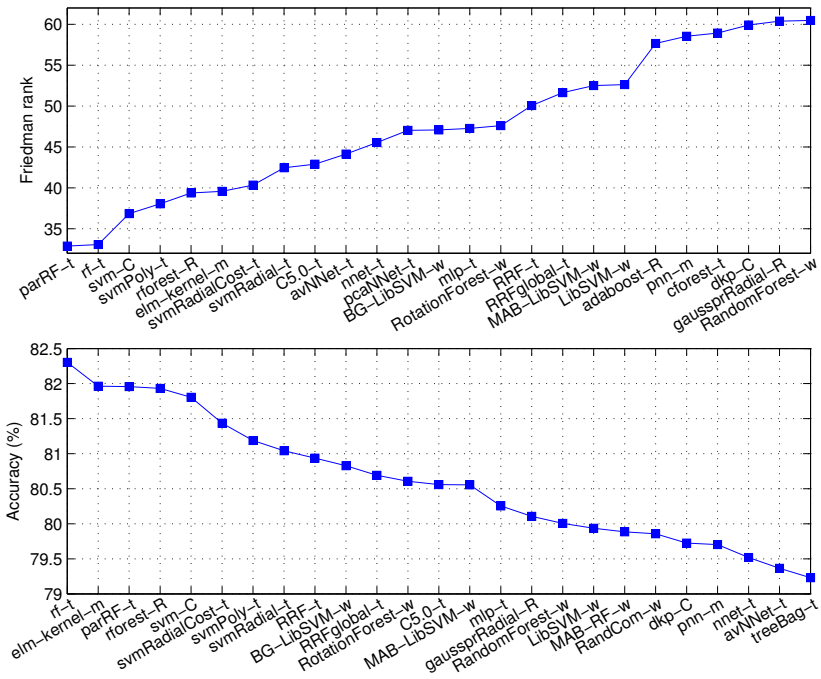


Figure 3: Friedman rank (upper panel, increasing order) and average accuracies (lower panel, decreasing order) for the 25 best classifiers.

and SVM families, are nnet_t (MLP network, rank 45.5), pcaNNet_t (MLP + PCA network, rank 47.0), Bagging_LibSVM_w (ensemble of Gaussian LibSVMs, rank 47.1), mlp_t (RSNNS MLP with tunable network size, rank 38.0), MultiBoostAB_LibSVM_w (MultiBoostAB ensemble of Gaussian LibSVMs, rank 52.5) and adaboost_R (Adaboost.M1 ensemble of decision trees, rank 57.6). Beyond the 20th position are pnn_m (Probabilistic Neural Network with tunable Gaussian spread, rank 58.5), and our proposal dkp_C (rank 59.9). Besides, note that 12 classifiers in the top-20 use caret, which might be due to the automatic parameter tuning (only rforest_R and adaboost_R have no tunable parameter). We must emphasize that, since parameter tuning and testing use different data sets, the final result can not be biased by parameter optimization, because the set of parameter values selected in the tuning stage is not necessarily the best on the test set. In some cases, the tuning is not relevant: for C5.0_t the differences among the performances using different parameter values are low, so it would work similarly without parameter tuning.

OpenML Vanschoren et al. (2012) uses only 86 data sets and 93 classifiers, while our work is much wider (121 and 179, respectively), including Weka classifiers in a later version (3.6.9), for example openML uses Bagging 1.31.2.2, while we use Bagging version 6502. Besides, as we commented above, we do not use 9 of 93 classifiers included in the previous reference. The results in Figure 17 of that paper rank Bagging-NBayesTree as the best classifier, followed by Bagging-PART, SVM-Polynomial, MultilayerPerceptron, Boosting-NBayesTree, RandomForest, Boosting-PART, Bagging-C45, Boosting-C45 and SVM-RBF. However, in our results the best Weka classifiers (in the top-20) are Baggging_LibSVM_w, RotationFor-

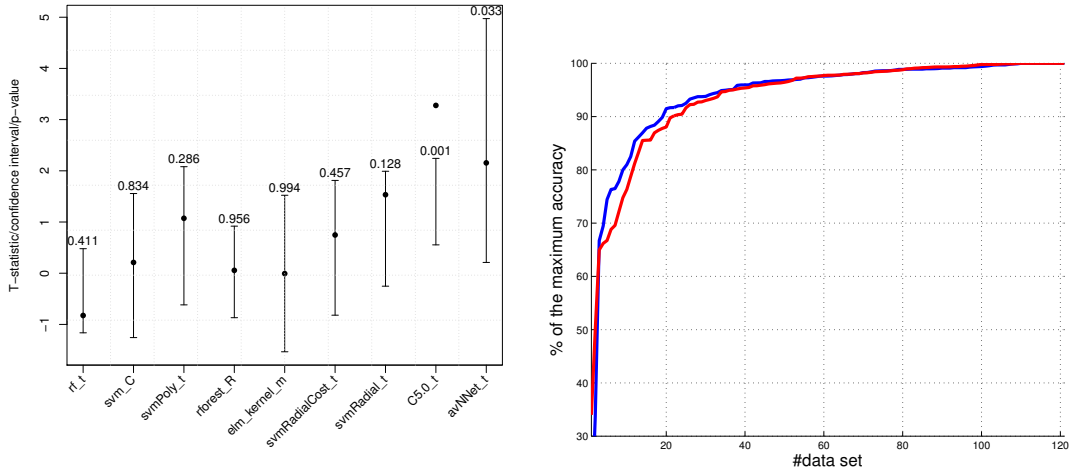


Figure 4: **Left panel**: T-statistics (point), confidence intervals and p-values (above upper interval limits) of the T-tests comparing parRF_T and the remaining 9 best classifiers. **Right panel**: Percentage of the maximum accuracy achieved by parRF_t (blue) and svm_C (red) for the 121 data sets (ordered by increasing percentage)

est_w, MultiBoostAB_LibSVM_w and LibSVM_w, i.e., a Random Forest, a SVM and two ensembles of SVMs. This is expected, because it is known that ensembles of strong classifiers do not work better than the single classifier. Therefore, Bagging and MultiBoostAB of LibSVM do not work better than LibSVM_w, although the three are worse than svm_C (the same Gaussian LibSVM in C) and the caret SVM versions (svmPoly_t, svmRadialCost_t and svmRadial_t). Besides, similarly to openML, in our work svmPoly_t (polynomial kernel) is near to svm_C (Gaussian kernel). However, in our results Bagging_NaiveBayes_w works very bad (rank 110.4, Table 4), while other Baggging ensembles are better: Bagging_PART_w (66.6), MultilayerPerceptron_w (70.6), MultiBoostAB_NaiveBayes_w (equivalent to Boosting-NaiveBayes in openML, rank 106.2) and MultiBoostAB_PART_w (65.6). Therefore, in our experiments the bagging and multiBoostAB ensembles (except of LibSVM_w) do not work well. We use the same configurations for bagging (10 bagging iterations, 100% of the training set for bag size, changing only the base learner) and MultiBoostAB (3 sub-committees, 10 boost iterations and 100% of build mass used to build classifiers) as openML, so these bad results can not be caused by improper configuration (or parameter tuning) of the ensemble or base classifier. Therefore, they might be caused by the larger number of data sets, or by the inclusion in our collection of other classifiers and implementations (in R, caret, C and Matlab), with better accuracies, not considered by OpenML.

No.	Classifier	PAMA	No.	Classifier	PAMA
1	elm_kernel_m	13.2	11	mlp_t	5.0
2	svm_C	10.7	12	pnn_m	5.0
3	parRF_t	9.9	13	dkp_C	5.0
4	C5.0_t	9.1	14	LibSVM_w	5.0
5	adaboost_R	9.1	15	svmPoly_t	5.0
6	rforest_R	8.3	16	treebag_t	5.0
7	nnet_t	6.6	17	RRFglobal_t	5.0
8	svmRadialCost_t	6.6	18	svmlight_C	5.0
9	rf_t	5.8	19	Bagging_RandomForest_w	4.1
10	RRF_t	5.8	20	mda_t	4.1

No.	Classifier	P95	No.	Classifier	P95
1	parRF_t	71.1	11	elm_kernel_m	60.3
2	svm_C	70.2	12	MAB-LibSVM_w	60.3
3	rf_t	68.6	13	RandomForest_w	57.0
4	rforest_R	65.3	14	RRF_t	56.2
5	Bagging-LibSVM_w	63.6	15	pcaNNet_t	55.4
6	svmRadialCost_t	63.6	16	RotationForest_w	54.5
7	svmRadial_t	62.8	17	avNNet_t	53.7
8	svmPoly_t	62.8	18	nnet_t	53.7
9	LibSVM_w	62.0	19	RRFglobal_t	53.7
10	C5.0_t	61.2	20	mlp_t	52.1

No.	Classifier	PMA	No.	Classifier	PMA
1	parRF_t	94.1	11	RandomCommittee_w	91.4
2	rf_t	93.6	12	nnet_t	91.3
3	rforest_R	93.3	13	avNNet_t	91.1
4	C5.0_t	92.5	14	RRFglobal_t	91.0
5	RotationForest_w	92.5	15	knn_R	90.5
6	svm_C	92.3	16	Bagging-LibSVM_w	90.5
7	mlp_t	92.1	17	Bagging_REPTree_w	90.4
8	LibSVM_w	91.7	18	MAB_MLP_w	90.4
9	RRF_t	91.4	19	elm_m	90.3
10	dkp_C	91.4	20	rda_R	90.3

Table 5: **Up**: list of the 20 classifiers with the highest Probabilities of Achieving the Maximum Accuracies (PAMA, in %). **Middle**: List of the 20 classifiers with the highest probabilities of achieving 95% (P95) of the maximum accuracy over all the data sets. **Down**: Classifiers sorted by its Percentage of the Maximum Accuracy (PMA) for each data set, averaged over all the data sets. MAB means MultiBoostAB.

3.2 Probability of Achieving the Best Accuracy

One of the objectives of this paper (Section 1) is to estimate, for each classifier, the **Probability of Achieving the Maximum Accuracy** (**PAMA**) for a given data set, as the number of data sets for which it achieves the highest accuracy, divided by the number of data sets. The Table 5 (upper part) shows the 20 classifiers with the highest values for these probabilities (in %), being elm_kernel_m the best (for 13.2% of the data sets) followed by svm_C (10.7%) and parRF (9.9%). These values are very far from 100%, which confirms

that no classifier is the best for most data sets (following the No-Free-Lunch theorem). The C5.0_t and adaboost_R have about 9%. The remaining classifiers are about 4-8%, so that many classifiers are the best for only few data sets. There are 5 classifiers of family RF, 5 SVM, 5 NNET and 4 ensembles among the 20 classifiers with the highest probabilities of being the best. Our proposal dkp_C achieves the 13th position.

The PAMA does not take into account that a classifier may be very near from the best accuracy without being the best one. Therefore, an alternative, more significant, measure is the **probability of achieving more than 95% of the maximum accuracy** (**P95**) (middle part of Table 5 for the best 20 values). This probability (in %), for a given classifier, is estimated dividing the number of data sets in which it achieves 95% or more of the maximum accuracy (achieved by any other classifier on that data set), by the number of data sets. The ten classifiers with the highest P95 are almost the same as in the Friedman rank, with a different order. In this table, ParRF_t achieves more than 95% of the maximum accuracy for 71.1% of the data sets (again far from the 100%), followed by svm_C (70.2%) and rf_t (68.6). The other classifiers have P95 below 65%. The low P95 of elm_kernel_w (60.3%, 11th position), being the best for a highest number of data sets, shows a behavior less stable than rf_t, parRF_t and svm_C, because its accuracy in the other data sets is lower in average.

Another interesting measurement is the **Percentage of the Maximum Accuracy** (**PMA**) achieved by each classifier, averaged over the whole collection of data sets (the 20 first are shown in the lower part of the Table 5). Again, parRF_t is the best achieving 94.1%($\pm$11.3) of the maximum accuracy, followed by other two Random Forests: rf_t and rforest_R (93.6% and 93.3% respectively). The svm_C is in the 6th position, with PMA 92.3%($\pm$15.9). Note that six out of eight Random Forest classifiers are in the top-20. The PMA values are high, very near to, but below, the threshold of 95% used in the middle part of Table 5. This explains the low values of P95: the bests classifiers have PMA about 94%, so their probability of achieving 95% or more of the maximum accuracy is low (about 70%). Setting the threshold in 90% of the maximum accuracies, the corresponding probabilities would be much higher. The elm_kernel_m is not included in this table: this confirms its unstable behavior, because in average it does not achieve PAM above 90.3% (even elm_m, without kernels, has better PMA). The mlp_t (92.2%) has also a good value. The dkp_C is the 10th position, achieving in average the 91.4%, only 2.7 below the best. The 20 classifiers are in a narrow margin between 90%-94% of the maximum accuracy, so that there are many classifiers which a high percentage of the maximum accuracy. The Figure 5 shows that the three Random Forests (parRT_t, rf_t and rforest_R) achieve PMAs clearly higher than the remaining classifiers (including svm_C), being the greatest gap (0.8) between rforest_R and C5.0_t.

3.3 Discussion by Classifier Family

The Figure 6 compares the classifier families showing in the upper panel the error bars with the mean (blue square), minimum and maximum values of the Friedman ranks for each family. The lower panel shows the minimum rank (corresponding to the best classifier) for each family, by ascending order. The family RF has the lowest minimum rank (32.9) and mean (46.7), and also a narrow interval (up to 60.5), which means that all the RF classifiers

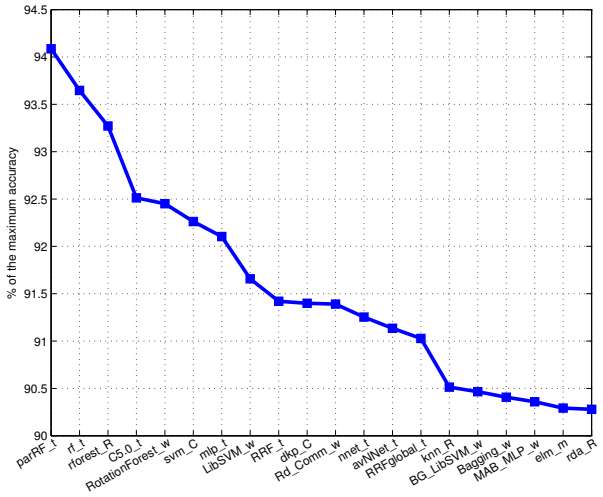


Figure 5: Twenty classifiers with the highest percentages of the maximum accuracy. MAB means MultiBoostAB, BG means Bagging.

work very well. The SVM has the following minimum (36.8), but the mean is much higher (55.4), and the interval is also much wider (up to 81.6). The third best type is NNET, whose minimum and mean rank are 39.6 (elm_kernel_m) and 73.8 respectively. The DTs have the following minimum (42.9), followed by BAG (47.1), BST (52.5), OM (other methods, specifically gaussprRadial, 60.4), DA (62.5), NN (62.6), OEN (other ensembles, specifically RandomCommittee_w, 63.0), LMR (65.2), MARS and GLM (65.5), PLSR (79.1), RL (81.4), BY (94.3) and STC (154.0). We can make three family groups in the lower panel of Figure 6: a) the best ones (RF, SVM, NNET, DT, BAG and BST), with the lowest ranks (about 30-50); b) the intermediate families (OM, DA, NN, OEN, LMR, MARS and GLM), about 60-70; and c) the worst families (PLSR, RL, BY and STC), with ranks above 80.

Now, we discuss the results for each classifier family (see Tables 6 and 7). The **discriminant analysis** (**DA**) classifiers work relatively well, being fda_t the best one, followed by rda_R, mda_t and pda_t. The lda_R works better than the caret version lda2_t (74.4), which however tunes of the number of retained components. In other DA classifiers (fda and mda) the parameter tuning developed in the caret versions allows to achieve better accuracies than their R counterparts (without tuning). It is surprising that sophisticated versions of LDA are worse: slda_t, PenalizedLDA_t, rrlda_R, sddaLDA_R, sparseLDA_R and stepLDA_t. Finally, the QDA classifiers are very bad, achieving again the classical qda_t the best results compared to more advanced versions (QdaCov_t, stepQDA_t and sddaQDA_R). The **Bayesian methods** (**BY**) are clearly worse than DA, and they are not competitive at all to the globally best classifiers, achieving the best (BayesNet_w) a high rank (94.3).

Among the **neural networks** (**NNET**), the elm_kernel_m is the best one, followed by several caret MLP implementations (avNNet_t, nnet_t, pcaNNet_t and mlp_t), included in the top-20, better than other MLP implementations: mlp_C (LibFANN), MultilayerPerceptron_w (Weka) and mlp_m (Matlab). The good result of avNNet (an ensemble of 4 small MLPs with up to 9 hidden neurons whose weights are randomly initialized), compared to

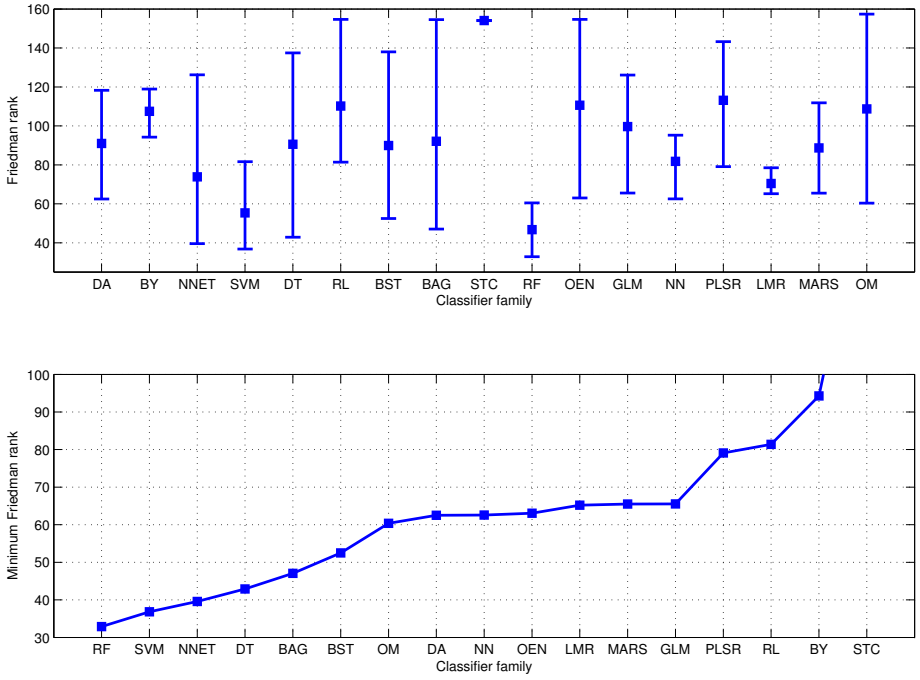


Figure 6: Friedman rank interval for the classifiers of each family (upper panel) and minimum rank (by ascending order) for each family (lower panel).

greater MLPs, as mlp_C and mlp_m (up to 30 hidden neurons), is due to its ensemble nature, because mlpWeightDecay_t also has up to 9 hidden neurons, with worse results. The rule used for size selection by MultilayerPerceptron_w (#inputs + #classes)/2 does not achieve good results. The pnn_m (probabilistic neural network) and our proposal dkp_C (direct kernel perceptron) are very near to the top-20. The bad results of elm_m (67.6) are surprising taking into account the good behavior of the Gaussian elm_kernel_w. Similarly, the LVQ versions are not good: lvq_t, which tunes the size and k, works much better than lvq_R, being bdk_R (99.8) the worst one. The cascor_C (cascade correlation), which uses LibFANN, is also worse (99.6) than the best MLP version (avNNet_t). Finally, the RBF networks are also bad, although the caret versions outperform the Weka and Matlab versions. The dpp_C is not competitive at all with the other networks.

The svm_C, with Gaussian kernel using LibSVM is the best **Support vector machine** (**SVM**), followed by the caret versions svmPoly_t (polynomial kernel), svmRadialCost_t and svmRadial_t (Gaussian kernel), better than the Weka versions LibSVM_w and SMO_w and that svmlight_C. The linear kernel versions (svmLinear_t and LibLINEAR_w) are clearly worse, and lssvmRadial_t is the worst one. Overall, the ten SVM classifiers achieve very good results, with ranks in the (relatively narrow) interval 36.8—72.4 (excluding linear kernels).

RandomSubSpace_w is the best **decision tree** (**DT**), with a bad rank (74.0); both J48_t and J48_w achieve similar results (the former runs the latter in the RWeka package tuned with caret). The best **Rule-based** (**RL**) classifiers are C5.0Rules_t and JRip_t,

Discriminant analysis (DA)					
1	fda_t	62.5	11	qda_t	96.0
2	rda_R	63.8	12	slda_t	103.2
3	mda_t	66.7	13	PenalizedLDA_t	106.8
4	pda_t	67.3	14	rrlda_R	108.2
5	fda_R	72.1	15	sddaLDA_R	112.6
6	lda_R	72.4	16	sparseLDA_R	113.3
7	lda2_t	74.4	17	stepLDA_t	113.5
8	sda_t	76.5	18	QdaCov_t	115.8
9	hdda_R	79.8	19	stepQDA_t	116.0
10	mda_R	80.9	20	sddaQDA_R	118.3
Bayesian methods (BY)					
1	BayesNet_w	94.3	4	vbmpRadial_t	109.4
2	naiveBayes_R	101.6	5	NBUpdateable_w	113.5
3	NaiveBayes_w	107.5			
Neural networks (NNET)					
1	elm_kernel_m	39.6	12	rbf_t	72.0
2	avNNet_t	44.1	13	rbfDDA_t	75.9
3	nnet_t	45.5	14	lvq_t	79.1
4	pcaNNet_t	47.0	15	RBFNetwork_w	90.2
5	mlp_t	47.3	16	lvq_R	95.1
6	pnn_m	58.5	17	mlp_m	99.5
7	dkp_C	59.9	18	cascor_C	99.6
8	mlp_C	62.8	19	bdk_R	99.8
9	mlpWeightDecay	63.6	20	rbf_m	108.1
10	elm_m	67.6	21	dpp_C	126.2
11	MultilayerPerceptron_w	70.6			
Support vector machines (SVM)					
1	svm_C	36.8	6	svmLinear_t	62.1
2	svmPoly_t	38.0	7	SMO_w	62.8
3	svmRadialCost_t	40.3	8	svmlight_C	72.4
4	svmRadial_t	42.5	9	LibLINEAR_w	74.6
5	LibSVM_w	52.6	19	lssvmRadial_t	81.6

Table 6: Friedman ranks of the classifiers in each family (continued in Table 7).

slightly worse than the best DT. The difference between JRip_t and JRip_w suggests that the tuning of the number of optimization runs developed in the caret version, but not with Weka, is important. ZeroR_w is among the worst ones, because it only predicts the same mean class for every test pattern: we included it to define the "zero-level" for the accuracy (49.2%, there is no classifier with lower accuracy).

Among the **boosting** (**BST**) ensembles, C5.0_t is the best (position 9), followed by MultiBoostAB_LibSVM (position 18), adaboost_R (position 19) and other MultiBoostAB ensembles with strong base classifiers (MultilayerPerceptron, RandomForest, PART and J48), while the ones with weak classifiers (OneR, IBk, DecisionStump, NaiveBayes, among others) are worse. The Figure 7 (upper panel, only Weka ensembles and base classifiers are plotted) shows that MultiBoostAB ensembles achieves much lower ranks than their corresponding base classifiers, excepting LibSVM, RandomForest and Logistic, where both are similar, and IBk, where the base classifier works much better. The same happens with

Decision trees (DT)					
1	RandomSubSpace_w	74.0	9	ctree2_t	92.4
2	C5.0Tree_t	84.2	10	REPTree_w	93.6
3			11	rpart2_t	93.6
4	NBTree_w	84.6	12	rpart_R	94.4
5	J48_t	87.6	13	obliqueTree_R	96.5
6	J48_w	87.9	14	RandomTree_w	110.0
7	rpart_t	91.2	15	DecisionStump_w	137.5
8	ctree_t	91.5			
Rule-based classifiers (RL)					
1	C5.0Rules_t	81.4	7	Ridor_w	94.6
2	JRip_t	81.7	8	DecisionTable_w	105.5
3	PART_t	88.0	9	OneR_t	132.1
4	DTNB_w	89.0	10	OneR_w	133.4
5	PART_w	89.5	11	ConjunctiveRule_w	136.6
6	JRip_w	93.1	12	ZeroR_w	154.6
Boosting (BST)					
1	C5.0_t	42.9	11	MAB_RandomTree_w	76.6
2	MAB_LibSVM_w	52.5	12	MAB_Logistic_w	82.0
3	adaboost_R	57.9	13	MAB_DecisionTable_w	87.2
4	MAB_MultilayerPerceptron_w	64.0	14	MAB_NaiveBayes_w	106.2
5	MAB_RandomForest_w	64.1	15	logitboost_R	106.6
6	MAB_PART_w	65.6	16	AdaBoostM1_DecisionStump_w	122.2
7	MAB_J48_w	69.2	17	MAB_DecisionStump_w	129.6
8	AdaBoostM1_J48_w	72.6	18	MAB_IBk_w	130.9
9	MAB_REPTree_w	73.7	19	MAB_OneR_w	133.2
10	LogitBoost_w	76.6	20	RILB_w	138.0
Bagging (BAG)					
1	BG_LibSVM_w	47.1	13	BG_Logistic_w	79.0
2	treebag_t	66.4	14	BG_DecisionTable_w	84.6
3	BG_PART_w	66.6	15	bagging_R	90.5
4	Bagging_REPTree_w	69.8	16	plsBag_R	97.0
5	BG_RandomTree_w	71.0	17	nbBag_R	100.8
6	BG_J48_w	71.0	18	nnetBag_R	104.5
7	BG_IBk_w	72.7	19	BG_NaiveBayes_w	110.4
8	ldaBag_R	72.9	20	BG_OneR_w	130.9
9	BG_LWL_w	73.2	21	BG_DecisionStump_w	133.7
10	svmBag_R	74.6	22	BG_HyperPipes_w	143.8
11	BG_RandomForest_w	77.1	23	BG_MLP_w	145.8
12	ctreeBag_R	78.7	24	MetaCost_w	154.5

Table 7: Continuation of Table 6. MAB means MultiBoostAB. RILB means RacedIncrementalLogitBoost. BG means Bagging. Continued in Table 8.

AdaBoostM1 (J48 much better than DecisionStump). The adaboost_R (AdaboostM1 with classification trees) works very well (included in the top-20), while AdaBoostM1_J48_w and AdaBoostM1_DecisionStump_w work much worse: this big difference might be in the AdaboostM1 implementation or in the base classifiers. There is also difference between LogitBoost_w and logitboost_R, despite of using the same base classifier (DecisionStump):

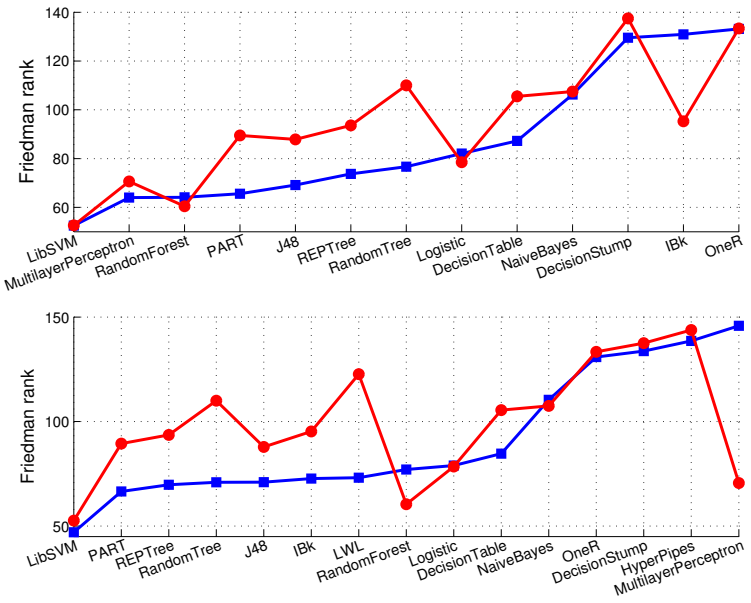


Figure 7: Upper panel: Friedman rank (ordered increasingly) of each Weka MultiBoostAB ensemble (blue squares) and its corresponding Weka base classifier (red circles). Lower panel: the same for Weka bagging ensembles (blue squares) and base classifiers (red circles).

the Weka implementation is clearly better. The RacedIncrementalLogitBoost_w is the worst one, despite of being a committee of LogitBoost.

The best **bagging (BG)** ensemble is also the Baggging_LibSVM_w (included in the top-20), although the svmBag_R is not so good, revealing big differences between implementations. The Figure 7 (lower panel) compares 15 Bagging ensembles to their respective base classifiers (both implemented in Weka), being the ensembles better except for RandomForest, NaiveBayes and MultilayerPerceptron. This means that RandomForest works better than in MultiBoostAB and Baggging ensembles. The remaining Bagging classifiers are not good: The ldaBag_R, ctreeBag_R, nbBag_R and nnetBag_R work also bad, similarly to their Weka correspondents (Bagging_NaiveBayes_w and Baggging_MultilayerPerceptron_w). Both **stacking** classifiers Stacking_w and StackingC_w work equally bad. The eight **random forest** classifiers are included among the 25 best classifiers having all of them low ranks, so this is clearly the best family of classifiers. Although one could think that there is a redundancy in RF models that might over-emphasize some results (parRF_t and rf_t are very similar classifiers), we must note that RRF_t (Regularized RF), RRFglobal_t (for which the caret documentation does not give differences with RRF_t, except in the tunable parameters) and cforest_t are different classifiers. Besides, the Weka RF implementations (RandomForest_w and RotationForest_w) are also among the 25 best classifiers, confirming that good positions of RF classifiers are not due to redundancy. Finally, none of the **other ensembles** (**OEN**) achieves good results, being RandomCommittee_w and Decorate_w the bests, but many of them are at the end of the list (rank 154.0).

Stacking (STC)					
1	Stacking_w	154.0	2	StackingC_w	154.1
Random forests (RF)					
1	parRF_t	32.9	5	RRF_t	50.1
2	rf_t	33.1	6	RRFglobal_t	51.6
3	rforest_R	39.4	7	cforest_t	58.9
4	RotationForest_w	47.6	8	RandomForest_w	60.5
Other ensembles (OEN)					
1	RandomCommittee_w	63.0	7	LWL_w	122.7
2	Decorate_w	63.4	8	Grading_w	154.0
3	END_w	76.6	9	MultiScheme_w	154.6
4	MultiClassClassifier_w	80.6	10	CostSensitiveClassifier_w	154.6
5	Dagging_w	95.3	11	Vote_w	154.6
6	OrdinalClassClassifier_w	97.2			
Generalized linear models (GLM)					
1	gmlnet_R	65.5	4	glmStepAIC_t	120.8
2	mlm_R	71.0	5	glm_R	126.1
3	bayesglm_t	114.8			
Nearest neighbors (NN)					
1	knn_t	62.6	4	IBk_w	94.5
2	knn_R	65.0	5	IB1_w	95.3
3	NNge_w	91.7			
Partial least squares and principal component regression (PLSR)					
1	pls_t	79.1	4	kernelpls_R	113.1
2	widekernelpls_R	111.3	5	gpls_R	120.1
3	simpls_R	111.9	6	spls_R	143.3
Logistic and multinomial regression(LMR)					
1	multinom_t	65.2	3	Logistic_w	78.5
2	SimpleLogistic_w	67.6			
Multivariate adaptive regression splines (MARS)					
1	gcvEarth_t	65.5	2	mars_R	111.9
Other methods (OM)					
1	gaussprRadial	60.4	6	pam_t	103.6
2	ClassificationViaRegression_w	66.0	7	VFI_w	135.5
3	AttributeSelectedClassifier_w	86.4	8	HyperPipes_w	143.8
4	KStar_w	87.2	9	CVParameterSelection	154.0
5	FilteredClassifier_w	93.0	10	ClassificationViaClustering_w	157.4

Table 8: Continuation of Tables 6 and 7.

The **GLM** classifiers are divided in two groups: gmlnet_R and mlm_R, with relatively good ranks (60-70), and the others, with much worse results. Something similar happens with **NN**, where the R and caret versions (knn_t and knn_R) are about 70, while the Weka variants NNge_w, IBk_w and IB1_w are much worse (about 90). With respect to the **PLSR** classifiers, the simplest one (pls_t) is the best, while the remaining, more sophisticated, versions are much worse. The three **LMR** classifiers achieve ranks about 65-75, being multinom_t the best one. The original **MARS** classifier (mars_R) is very bad, while the fast MARS version (gcvEarth_t) works much better. Finally, only the gaussprRadial_R and

ClassificationViaRegression_w achieve good results among the **Other methods**, while the remaining ones have ranks about 90 (AttributeSelectedClassifier_w, KStar_w and FilteredClassifier_w), and more, being some of the worse classifiers in the collection (ClassificationViaClustering_w).

Rank	Classifier	Acc. (%)	Rank	Classifier	Acc (%)
36.2	avNNet_t	83.0	50.0	mlp_t	82.2
39.9	svmPoly_t	79.9	51.4	elm_kernel_m	77.5
41.0	pcaNNet_t	82.9	54.1	RotationForest_w	82.0
42.2	svmRadialCost_t	80.0	54.9	rforest_R	80.9
44.2	parRF_t	82.6	57.6	mlpWeightDecay_t	79.7
44.7	rf_t	81.2	57.7	svmBag_R	78.8
47.1	C5.0_t	82.0	59.7	fda_t	81.0
47.2	svm_C	79.0	60.8	cforest_t	74.7
47.5	nnet_t	82.1	61.5	Bagging_LibSVM_w	77.9
48.0	svmRadial_t	79.4	62.9	knn_t	80.4

No.	Classifier	P95	No.	Classifier	P95
1	svmRadialCost_t	78.2	11	MultiBoostAB_LibSVM_w	65.5
2	svm_C	74.5	12	pcaNNet_t	63.6
3	svmPoly_t	74.5	13	svmBag_R	63.6
4	svmRadial_t	72.7	14	elm_kernel_m	61.8
5	Bagging_LibSVM_w	70.9	15	nnet_t	61.8
6	avNNet_t	69.1	16	RotationForest_w	61.8
7	parRF_t	69.1	17	fda_t	60.0
8	LibSVM_w	67.3	18	mlp_t	60.0
9	C5.0_t	67.3	19	MultiBoostAB_REPTree_w	58.2
10	rf_t	67.3	20	RandomForest_w	58.2

No.	Classifier	PMA	No.	Classifier	PMA
1	avNNet_t	95.0	11	pda_t	92.8
2	pcaNNet_t	94.9	12	mlm_R	92.7
3	parRF_t	94.3	13	fda_t	92.7
4	nnet_t	94.1	14	MAB_MLP_w	92.7
5	mlp_t	94.1	15	bayesglm_t	92.6
6	C5.0_t	93.8	16	simpls_R	92.5
7	RotationForest_w	93.7	17	rforest_R	92.5
8	glmnet_R	93.5	18	MultiBoostAB_PART_w	92.5
9	rda_R	93.2	19	fda_R	92.3
10	rf_t	92.8	20	nnetBag_R	92.2

Table 9: Results for **two class** data sets. **Up**: Friedman rank and average accuracies for the 20 best classifiers. RF_w = RotationForest_w. MWD_t = mlpWeightDecay_t. **Middle**: Probability (in %) of achieving 95% or more of the maximum accuracy. **Down**: 20 classifiers with the highest average Percentage of the Maximum Accuracy (PMA) over the two-class data sets. MAB_MLP_w means MultiBoostAB_MultilayerPerceptron_w.

3.4 Two-Class Data Sets

Since 45.4% of the data sets (55 out of 121) have only two classes, it is interesting to see what happens when only 2-class data sets are considered. We repeated our analysis of the Subsections 3.1 and 3.2, calculating the Friedman rank and the average accuracy, alongside with the P95 and PMA, for all the classifiers and two-class data sets. Although it should be recommendable, we did not use the area under ROC curve as quality measure, nor develop cutoff tuning (Kuhn and Johnson, 2013), because some classifiers do not give probabilistic output. The Table 9 reports the results:

- The **upper part** shows the 20 classifiers with the best Friedman rank (calculated using only 2-class data sets), alongside with their average accuracies. The classifiers in this new list are approximately the same as in the top-20 of Table 3, but the order is different: avNNet_t (rank 36.2) is now the best, while the parRF_t, rf_t and svm_C (the three bests ones in Table 3) are now the 5th, 6th and 8th respectively. Besides, the best average accuracy (83.0%) is almost the same as in Table 3 (82.3%), so the classification results are not globally better for two class problems. Except the C5.0_t, all the classifiers in the top-10 are NMP neural networks, SVMs and Random Forests. As well, these families occupy 6 places in positions 11-20. Besides, 14 of 20 classifiers use caret. The elm_kernel_m is worse than in Table 3.

- The **middle part** reports the probabilities (in %) of achieving 95% or more of the maximum accuracy (P95). The best one is 78.2% (svmRadialCost), higher than in Table 3 (71.1%, parRF_t). The first four classifiers are SVMs, while parRF_t and rf_t are in 7th and 10th positions. The avNNet_t, Baggging_LibSVM_w, LibSVM_w and C5.0_t also are in the top-10. In positions 11-20 there are two MultiBoostAB ensembles (LibSVM and REPTree), svmBag_R and fda_t, alongside with several neural networks (pcaNNet_t, elm_kernel_m, nnet_t and mlp_t) and Random Forests (RotationForest_w and RandomForest_w).

- The **lower part** shows the 20 classifiers with the highest average Percentage of the Maximum Accuracy (PMA). The maximum value (95.0%, avNNet_t) is similar to the multi-class value (94.1%, lower part of Table 5), being parRF_t in the 3th position (94.3%). Other NNET classifiers also achieve good PMAs: pcaNNet_t, nnet_t and mlp_t. The C5.0_t keeps its good results, while rf_t falls to the 10th position. The table also includes some classifiers with bad multi-class results: glmnet_R, rda_R, pda_t, fda_t, bayesglm_t and simpls_R (both with bad multi-class rank), belonging to families GLM, DA and PLSR, which behave well for two-class problems. The best ensembles, apart from Random Forests, are MultiBoostAB_MultilayerPerceptron_w, MultiBoostAB_PART_w and nnetBag_R. Overall, the 20 classifiers are in a narrow range between 92.2%-95% of accuracy.

3.5 Discussion by Data Set Properties

In this section we study the classifier behavior in function of five data set properties: its "complexity", increasing and decreasing #patterns, #inputs and #classes. This study will be developed by calculating a modified average accuracy μ_j (in %) for each classifier j, in

which each data set is "weighted" according to each property as $\mu_j = \frac{1}{N_d}\sum_{i=1}^{N_d} w_i A_{ij}, j = 1,\ldots,N_c$, being w_i is the weight measuring the property for data set i ($0 \leq w_i \leq N_d$), defined in the following subsections; $N_d = 121$ is the number of data sets; A_{ij} is the accuracy (in %) achieved by classifier j in data set i; and $N_c = 179$ is the number of classifiers. The **classifier behavior with the data complexity** is difficult to evaluate, because the own data set complexity is hard to define (Ho and Basu, 2002), and it may be relative to the classifier used. In our case, since we are trying a large number of classifiers, we can suppose that some of them achieves the highest possible accuracy for each data set. Since this maximum accuracy is higher for some data sets than for others, we can believe that some data sets are harder, independently of the classifier used. Therefore, we can calculate the weighted average accuracy μ_j^C (the C superscript denotes "complexity") of classifier j using the weights w_i^C (which evaluate the complexity of data set i) defined as $w_i^C = \frac{N_d(1-M_i)}{N_d-\sum_{k=1}^{N_d} M_k}, i = 1,\ldots,N_d$, being $M_i = \max_{j=1,\ldots,N_c}\{A_{ij}/100\}$, the maximum accuracy for data set i divided by 100. Note that $\sum_{i=1}^{N_d} w_i^C = N_d$. The weighted accuracy μ_j^C (see below) with w_i^C defined above weights more the data sets i with maximum accuracy M_i low, which are expected to be more complex. The Table 10 (upper panel) shows the 20 classifiers with the highest μ^C, which exhibit the best behavior when the hardest data sets have stronger weight (data sets with maximum accuracy M_i low). The parRF_t is the best one, and the three best classifiers (5 in the top-10) belong to the family RF. Other two classifiers are neural networks (mlp_t and avNNet_t), C5.0_t is the 4th, and two SVMs (svm_C and LibSVM_w) are 6th and 9th respectively. Our proposal dkp_C exhibits a good behavior (12th position), while other classifiers in the top-20 of Table 3 as nnet_t, Bagging_LibSVM_w and RRFglobal_t are also included. The 20 classifiers are in a narrow range between 70.0% and 66.9% (3.1 points), so the differences among them are not too high. In order to study the **classifier behavior increasing #patterns**, the weighted accuracy μ^P uses the following weights $w_i^P = \frac{N_d N_i}{\sum_{k=1}^{N_d} N_k}, i = 1,\ldots,N_d$, where N_i is the #patterns (population) of data set i. The middle part of the Table 10 shows the weighted accuracy μ^P (the two largest data sets, connect-4 and miniboone, give errors for some classifiers which disturb this measure, so that they are excluded). Although the range is narrow (89.4%-91.1%), again the rf_t and parRF_t are the bests, and svm_C is the 3rd. There are six random forests in the top-10. The and treebag_t are also in the top-10. The positions 11-20 are completely filled by ensembles: Bagging, MultiBoostAB and AdaboostM1.

The **classifier behavior decreasing #patterns** in the data set can be analyzed calculating the weighted accuracy using weights w_i^D decreasing with the #patterns (N_m is the maximum #patterns for all the data sets) $w_i^D = \frac{N_d(N_m-N_i)}{N_m-\sum_{k=1}^{N_d} N_k}, i = 1,\ldots,N_d; N_m = \max_{j=1,\ldots,N_d}\{N_j\}, i = 1,\ldots,N_d$. The lower part of Table 10 shows the accuracies μ^D weighting each data set decreasingly with the #patterns. The rf_t is the best, followed by rforest_R, svm_C and parRF_t, which are only slightly worse than rf_t. Again, there are 6 random forests in the top-10. The positions 11-20 include dkp_C, elm_kernel_m, and MultiBoostAB ensembles of LibSVM and MultilayerPerceptron. The **dependence of the results with the #classes** N_i^c of the data set i can be analyzed calculating the weighted accuracy μ^L with data set weights w_i^L given by $w_i^L = \frac{N_d N_i^c}{\sum_{k=1}^{N_d} N_k^c}, i = 1,\ldots,N_d$. The Table 11

No.	Classifier	μ^C	No.	Classifier	μ^C
1	parRF_t	69.9	11	nnet_t	67.7
2	rf_t	69.6	12	dkp_C	67.6
3	rforest_R	69.3	13	RRFglobal_t	67.4
4	C5.0_t	69.0	14	Bagging_LibSVM_w	67.3
5	RotationForest_w	68.6	15	Decorate_w	67.1
6	svm_C	68.4	16	knn_t	67.1
7	mlp_t	68.4	17	Bagging_REPTree_w	67.0
8	RRF_t	68.1	18	elm_m	67.0
9	LibSVM_w	67.8	19	pda_t	67.0
10	avNNet_t	67.8	20	RandomCommittee_w	66.9
No.	Classifier	μ^P	No.	Classifier	μ^P
1	rf_t	91.1	11	Bagging_LibSVM_w	89.9
2	parRF_t	91.1	12	RandomCommittee_w	89.9
3	svm_C	90.7	13	Bagging_RandomTree_w	89.8
4	RRF_t	90.6	14	MultiBoostAB_RandomTree_w	89.8
5	RRFglobal_t	90.6	15	MultiBoostAB_LibSVM_w	89.8
6	LibSVM_w	90.6	16	MultiBoostAB_PART_w	89.7
7	RotationForest_w	90.5	17	Bagging_PART_w	89.7
8	C5.0_t	90.5	18	AdaBoostM1_J48_w	89.5
9	rforest_R	90.3	19	Bagging_REPTree_w	89.5
10	treebag_t	90.2	20	MultiBoostAB_J48_w	89.4
No.	Classifier	μ^D	No.	Classifier	μ^D
1	rf_t	82.1	11	MultiBoostAB_LibSVM_w	79.7
2	rforest_R	81.8	12	LibSVM_w	79.6
3	svm_C	81.6	13	RandomCommittee_w	79.5
4	parRF_t	81.6	14	dkp_C	79.5
5	RRF_t	80.8	15	nnet_t	79.3
6	RotationForest_w	80.3	16	elm_kernel_m	79.2
7	C5.0_t	80.2	17	avNNet_t	79.2
8	mlp_t	80.0	18	treebag_t	79.0
9	Bagging_LibSVM_w	80.0	19	MAB_MLP_w	78.8
10	RRFglobal_t	79.8	20	knn_R	78.7

Table 10: Twenty best classifiers depending on the data set complexity and population. **Up**: average accuracy μ^C (in %) weighting each data set decreasingly with its complexity. **Middle**: accuracy μ^P weighting the data sets increasingly with #patterns. **Down**: average accuracy μ^D weighted decreasingly with #patterns.

(upper part) shows the accuracy μ^L for the 20 best classifiers. The best classifiers are svm_C and rf_t (with the same accuracy), followed by rforest_t, Bagging_LibSVM_w, parRF_t and others, only 1% below the bests. There are 4 Random Forests and 2 SVMs in the top-10. The Bagging_LibSVM_w, MultiBoostAB_LibSVM_w and MultiBoostAB_Multilayer Perceptron_w ensembles are also included in the top-10. The best neural networks are dkp_C (9th position), MultilayerPerceptron_w and elm_m. Two DA classifiers (rda_R and hdda_R) and two NN classifiers (knn_R and IBk_w) are included. With respect to the **number of inputs**, the weighted average accuracy μ^I according to the #inputs N_i^I can be calculated

No.	Classifier	μ^L	No.	Classifier	μ^L
1	svm_C	80.5	11	RotationForest_w	76.6
2	rf_t	80.5	12	RRFglobal_t	76.1
3	rforest_R	79.8	13	MultilayerPerceptron_w	76.1
4	Bagging_LibSVM_w	79.7	14	rda_R	76.0
5	parRF_t	79.5	15	knn_R	75.9
6	MultiBoostAB_LibSVM_w	79.5	16	SMO_w	75.6
7	LibSVM_w	79.5	17	hdda_R	75.4
8	RRF_t	77.9	18	KStar_w	75.3
9	dkp_C	77.7	19	elm_m	75.1
10	MAB_MLP_w	76.9	20	RandomCommittee_w	75.1

No.	Classifier	μ^I	No.	Classifier	μ^I
1	parRF_t	84.0	11	mlp_t	81.5
2	rf_t	83.3	12	SMO_w	81.3
3	rforest_R	82.9	13	Bagging_RandomTree_w	81.3
4	RotationForest_w	82.8	14	elm_kernel_m	81.1
5	MAB_MLP_w	82.5	15	mlp_C	81.0
6	LibSVM_w	82.4	16	dkp_C	80.8
7	MultilayerPerceptron_w	82.0	17	fda_t	80.8
8	svm_C	82.0	18	rda_R	80.8
9	RandomCommittee_w	81.8	19	SimpleLogistic_w	80.7
10	C5.0_t	81.6	20	RRF_t	80.4

Table 11: **Up**: average accuracy μ^L weighted using the #classes w^L (only 20 first classifiers). **Down**: average accuracy μ^I weighted with the #inputs w^I.

defining the weights w^I as $w_i^I = \frac{N_d N_i^I}{\sum_{k=1}^{N_d} N_k^I}, i = 1, \ldots, N_d$. The lower part of the Table 11 shows μ^I for the 20 best classifiers: parRF_t and rf_t are the bests, with 4 random forests among the top-5 (the other is MultiBoostAB_Multilayer Perceptron_w), while the svm_C falls to the 8th position, below LibSVM_w (6th). The MultilayerPerceptron_w and mlp_t are also included in the top-10. The dkp_C is again in the top-20. **Considering jointly the four dependencies** (complexity, population, #classes and #inputs), parRF_t and rf_t are always in the first positions, while the svm_C is not so regular: good behavior with #classes and #patterns, but not so good with complexity and #inputs (6th and 8th positions). The svm_C and parRF_t are worse than rf_t with decreasing #patterns. Besides, the averages of $\mu^C, \mu^P, \mu^D, \mu^L, \mu^I$ are 81.3%, 81.2% and 80.8% for rf_t, parRF_t and svm_C respectively, which shows the similarity between rf_t and parRF_t, and their difference to svm_C. Most of the random forest versions (rforest_R, RotationForest_w, RRF_t and RRFglobal_t), and LibSVM_w, are in the five tables. Apart from the RF and SVM classifiers, which fill most of the 10 best positions in the five tables, it is remarkable the good behavior of C5.0_t (family DT), included in the four tables and three times in the top-10. Among the *neural networks*, the dkp_C appears more often (in four of five tables): in fact, the μ^P table does not include any neural network, showing a bad behavior for populated data sets. The Bagging_LibSVM_w is also the first **bagging** classifier in four tables, while MultiBoostAB of LibSVM or MLP is the best **boosting** classifier, appearing in four tables. The Random-

Committee_w (the best classifier of family **OEN**) is also included in five tables, and in the top-10 for μ^I. On the other hand, three of five tables include a classifier of family **NN** (knn_t or knn_R). The **DA** classifiers show bad behavior with population, being included only pda_t in μ^C; rda_R and hdda_R in μ^L; fda_t and rda_R in μ^I.

4. Conclusion

This paper presents an exhaustive evaluation of 179 classifiers belonging to a wide collection of 17 families over the whole UCI machine learning classification database, discarding the large-scale data sets due to technical reasons, plus 4 own real sets, summing up to 121 data sets from 10 to 130,064 patterns, from 3 to 262 inputs and from 2 to 100 classes. **The best results are achieved by the parallel random forest** (parRF_t), implemented in R with caret, tuning the parameter mtry. The parRF_t achieves in average 94.1% of the maximum accuracy over all the data sets (Table 5, lower part), and overcomes the 90% of the maximum accuracy in 102 out of 121 data sets. Its average accuracy over all the data sets is 82.0%, while the maximum average accuracy (achieved by the best classifier for each data set) is 86.9%. The random forest in R and tuned with caret (rf_t) is slightly worse (93.6% of the maximum accuracy), although it achieves slightly better average accuracy (82.3%) than parRF_t. The LibSVM implementation of SVM in C with Gaussian kernel (svm_C), tuning the regularization and kernel spread, achieves 92.3% of the maximum accuracy. Six RFs and five SVMs are included among the 20 best classifiers, which are the bests families. The parRF_t may be considered as a reference ("gold-standard") to compare with new classifier proposals in order to assess their performance for general classification in general (not requiring special features as large-scale, on-line learning, non-stationary data, etc.). Other classifiers with good results are the extreme learning machine with Gaussian kernel, the C5.0 decision tree and the multi-layer perceptron (avNNet_t, a committee of 5 multi-layer perceptrons randomly initialized tuning the size and decay rate). The best boosting and bagging ensembles use LibSVM as base classifiers (in Weka), being slightly better than the single LibSVM classifier, and adaboost_R (ensemble of decision trees trained using Adaboost.M1). For **two-class data sets**, avNNet_t is the best (95% of the maximum accuracy), being the parRF_t also very good (94.3%). It is also the best when the complexity, #patterns and #inputs of the data set increase, being also good when #patterns decrease (rf_t is the best) and #classes increase (svm_C is the best). The probabilistic neural network in Matlab, tuning the Gaussian kernel spread (pnn_m), and the direct kernel perceptron in C (dkp_C), a very simple and fast neural network proposed by us (Fernández-Delgado et al., 2014), are also very near to the top-20. The remaining families of classifiers, including other neural networks (radial basis functions, learning vector quantization and cascade correlation), discriminant analysis, decision trees other than C5.0, rule-based classifiers, other bagging and boosting ensembles, nearest neighbors, Bayesian, GLM, PLSR, MARS, etc., are not competitive at all. Most of the best classifiers are implemented in R and tuned using caret, which seems the best alternative to select a classifier implementation.

Acknowledgments

We would like to acknowledge support from the Spanish Ministry of Science and Innovation (MICINN), which supported this work under projects TIN2011-22935 and TIN2012-32262.

References

David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

Miika Ahdesmäki and Korbinian Strimmer. Feature selection in omics prediction problems using cat scores and false non-discovery rate control. *Annals of Applied Stat.*, 4:503–519, 2010.

Esteban Alfaro, Matías Gámez, and Noelia García. Multiclass corporate failure prediction by Adaboost.M1. *Int. Advances in Economic Research*, 13:301–312, 2007.

Peter Auer, Harald Burgsteiner, and Wolfang Maass. A learning rule for very simple universal approximators consisting of a single layer of perceptrons. *Neural Networks*, 1(21): 786–795, 2008.

Kevin Bache and Moshe Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Laurent Bergé, Charles Bouveyron, and Stéphane Girard. HDclassif: an R package for model-based clustering and discriminant analysis of high-dimensional data. *J. Stat. Softw.*, 46(6):1–29, 2012.

Michael R. Berthold and Jay Diamond. Boosting the performance of RBF networks with dynamic decay adjustment. In *Advances in Neural Information Processing Systems*, pages 521–528. MIT Press, 1995.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Leo Breiman, Jerome Friedman, R.A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, 1984.

Jean Carletta. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22(2):249–254, 1996.

Jadzia Cendrowska. PRISM: An algorithm for inducing modular rules. *Int. J. of Man-Machine Studies*, 27(4):349–370, 1987.

S. Le Cessie and J.C. Van Houwelingen. Ridge estimators in logistic regression. *Applied Stat.*, 41(1):191–201, 1992.

Chih-Chung Chang and Chih-Jen. Lin. Libsvm: a library for support vector machines, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Hyonho Chun and Sunduz Keles. Sparse partial least squares for simultaneous dimension reduction and variable selection. *J. of the Royal Stat. Soc. - Series B*, 72:3–25, 2010.

John G. Cleary and Leonard E. Trigg. K*: an instance-based learner using an entropic distance measure. In *Int. Conf. on Machine Learning*, pages 108–114, 1995.

Line H. Clemensen, Trevor Hastie, Daniela Witten, and Bjarne Ersboll. Sparse discriminant analysis. *Technometrics*, 53(4):406–413, 2011.

William W. Cohen. Fast effective rule induction. In *Int. Conf. on Machine Learning*, pages 115–123, 1995.

Bhupinder S. Dayal and John F. MacGregor. Improved PLS algorithms. *J. of Chemometrics*, 11:73–85, 1997.

Gülsen Demiroz and H. Altay Guvenir. Classification by voting feature intervals. In *European Conf. on Machine Learning*, pages 85–92. Springer, 1997.

Houtao Deng and George Runger. Feature selection via regularized trees. In *Int. Joint Conf. on Neural Networks*, pages 1–8, 2012.

Beijing Ding and Robert Gentleman. Classification using generalized partial least squares. *J. of Computational and Graphical Stat.*, 14(2):280–298, 2005.

Annette J. Dobson. *An Introduction to Generalized Linear Models*. Chapman and Hall, 1990.

Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Int. Conf. on Knowledge Discovery and Data Mining*, pages 155–164, 1999.

Richard Duda, Peter Hart, and David Stork. *Pattern Classification*. Wiley, 2001.

Manuel J.A. Eugster, Torsten Hothorn, and Friedrich Leisch. Domain-based benchmark experiments: exploratory and inferential analysis. *Austrian J. of Stat.*, 41:5–26, 2014.

Scott E. Fahlman. Faster-learning variations on back-propagation: an empirical study. In *1988 Connectionist Models Summer School*, pages 38–50. Morgan-Kaufmann, 1988.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: a library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, 2008.

Manuel Fernández-Delgado, Jorge Ribeiro, Eva Cernadas, and Senén Barro. Direct parallel perceptrons (DPPs): fast analytical calculation of the parallel perceptrons weights with margin control for classification tasks. *IEEE Trans. on Neural Networks*, 22:1837–1848, 2011.

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, Jorge Ribeiro, and José Neves. Direct kernel perceptron (DKP): ultra-fast kernel ELM-based classification with non-iterative closed-form weight calculation. *Neural Networks*, 50:60–71, 2014.

Eibe Frank and Mark Hall. A simple approach to ordinal classification. In *European Conf. on Machine Learning*, pages 145–156, 2001.

Eibe Frank and Stefan Kramer. Ensembles of nested dichotomies for multi-class problems. In *Int. Conf. on Machine Learning*, pages 305–312. ACM, 2004.

Eibe Frank and Ian H. Witten. Generating accurate rule sets without global optimization. In *Int. Conf. on Machine Learning*, pages 144–151, 1999.

Eibe Frank, Yong Wang, Stuart Inglis, Geoffrey Holmes, and Ian H. Witten. Using model trees for classification. *Machine Learning*, 32(1):63–76, 1998.

Eibe Frank, Geoffrey Holmes, Richard Kirkby, and Mark Hall. Racing committees for large datasets. In *Int. Conf. on Discovery Science*, pages 153–164, 2002.

Eibe Frank, Mark Hall, and Bernhard Pfahringer. Locally weighted naive Bayes. In *Conf. on Uncertainty in Artificial Intelligence*, pages 249–256, 2003.

Yoav Freund and Llew Mason. The alternating decision tree learning algorithm. In *Int. Conf. on Machine Learning*, pages 124–133, 1999.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Int. Conf. on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.

Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. In *Conf. on Computational Learning Theory*, pages 209–217, 1998.

Jerome Friedman. Regularized discriminant analysis. *J. of the American Stat. Assoc.*, 84: 165–175, 1989.

Jerome Friedman. Multivariate adaptive regression splines. *Annals of Stat.*, 19(1):1–141, 1991.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Stat.*, 28:2000, 1998.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *J. of Stat. Softw.*, 33(1):1–22, 2010.

Brian R. Gaines and Paul Compton. Induction of ripple-down rules applied to modeling large databases. *J. Intell. Inf. Syst.*, 5(3):211–228, 1995.

Andrew Gelman, Aleks Jakulin, Maria G. Pittau, and Yu-Sung Su. A weakly informative default prior distribution for logistic and other regression models. *The Annals of Applied Stat.*, 2(4):1360–1383, 2009.

Mark Girolami and Simon Rogers. Variational bayesian multinomial probit regression with Gaussian process priors. *Neural Computation*, 18:1790–1817, 2006.

Ekkehard Glimm, Siegfried Kropf, and Jürgen Läuter. Multivariate tests based on left-spherically distributed linear scores. *The Annals of Stat.*, 26(5):1972–1988, 1998.

Encarnación González-Rufino, Pilar Carrión, Eva Cernadas, Manuel Fernández-Delgado, and Rosario Domínguez-Petit. Exhaustive comparison of colour texture features and classification methods to discriminate cells categories in histological images of fish ovary. *Pattern Recognition*, 46:2391–2407, 2013.

Mark Hall. *Correlation-Based Feature Subset Selection for Machine Learning*. PhD thesis, University of Waikato, 1998.

Mark Hall and Eibe Frank. Combining naive Bayes and decision tables. In *Florida Artificial Intel. Soc. Conf.*, pages 318–319. AAAI press, 2008.

Trevor Hastie and Robert Tibshirani. Discriminant analysis by Gaussian mixtures. *J. of the Royal Stat. Soc. series B*, 58:158–176, 1996.

Trevor Hastie, Robert Tibshirani, and Andreas Buja. Flexible discriminant analysis by optimal scoring. *J. of the American Stat. Assoc.*, 89:1255–1270, 1993.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2009.

Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

Tin Kam Ho and Mitra Basu. Complexity measures of supervised classification problems. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.

Geoffrey Holmes, Mark Hall, and Eibe Frank. Generating rule sets from model trees. In *Australian Joint Conf. on Artificial Intelligence*, pages 1–12, 1999.

Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

Torsten Hothorn, Friedrich Leisch, Achim Zeileis, and Kurt Hornik. The design and analysis of benchmark experiments. *J. Computational and Graphical Stat.*, 14:675–699, 2005.

Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Trans. Syst. Man Cybern. - Part B: Cybernetics*, 42:513–529, 2012.

Torsten Joachims. Making Large-Scale Support Vector Machine Learning Practical. In Bernhard Scholköpf, Cristopher J.C. Burges, and Alexander Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 169–184. MIT-Press, 1999.

George H. John and Pat Langley. Estimating continuous distributions in Bayesian classifiers. In *Conf. on Uncertainty in Artificial Intelligence*, pages 338–345, 1995.

Sijmen De Jong. SIMPLS: an alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 18:251–263, 1993.

Josef Kittler, Mohammad Hatef, Robert P.W. Duin, and Jiri Matas. On combining classifiers. *IEEE Trans. on Pat. Anal. and Machine Intel.*, 20:226–239, 1998.

Ron Kohavi. The power of decision tables. In *European Conf. on Machine Learning*, pages 174–189. Springer, 1995.

Ron Kohavi. Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid. In *Int. Conf. on Knoledge Discovery and Data Mining*, pages 202–207, 1996.

Max Kuhn. Building predictive models in R using the caret package. *J. Stat. Softw.*, 28(5): 1–26, 2008.

Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*. Springer, New York, 2013.

Niels Landwehr, Mark Hall, and Eibe Frank. Logistic model trees. *Machine Learning*, 95 (1-2):161–205, 2005.

Nick Littlestone. Learning quickly when irrelevant attributes are abound: a new linear threshold algorithm. *Machine Learning*, 2:285–318, 1988.

Nuria Macià and Ester Bernadó-Mansilla. Towards UCI+: a mindful repository design. *Information Sciences*, 261(10):237–262, 2014.

Nuria Macià, Ester Bernadó-Mansilla, Albert Orriols-Puig, and Tin Kam Ho. Learner excellence biased by data set selection: a case for data characterisation and artificial data sets. *Pattern Recognition*, 46:1054–1066, 2013.

Harald Martens. *Multivariate Calibration*. Wiley, 1989.

Brent Martin. *Instance-Based Learning: Nearest Neighbor with Generalization*. PhD thesis, Univ. of Waikato, Hamilton, New Zealand, 1995.

Willem Melssen, Ron Wehrens, and Lutgarde Buydens. Supervised Kohonen networks for classification problems. *Chemom. Intell. Lab. Syst.*, 83:99–113, 2006.

Prem Melville and Raymond J. Mooney. Creating diversity in ensembles using artificial data. *Information Fusion: Special Issue on Diversity in Multiclassifier Systems*, 6(1): 99–111, 2004.

John C. Platt. Fast training of support vector machines using sequential minimal optimization. In Bernhard Scholköpf, Cristopher J.C. Burges, and Alexander Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, 1998.

Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

Ross Quinlan. Learning with continuous classes. In *Australian Joint Conf. on Artificial Intelligence*, pages 343–348, 1992.

Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge Univ. Press, 1996.

Juan J. Rodríguez, Ludmila I. Kuncheva, and Carlos J. Alonso. Rotation forest: a new classifier ensemble method. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.

Alexander K. Seewald. How to make stacking better and faster while also taking care of an unknown weakness. In *Int. Conf. on Machine Learning*, pages 554–561. Morgan Kaufmann Publishers, 2002.

Alexander K. Seewald and Johannes Fuernkranz. An evaluation of grading classifiers. In *Int. Conf. on Advances in Intelligent Data Analysis*, pages 115–124, 2001.

David J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. CRC Press, 2006.

Donald F. Specht. Probabilistic neural networks. *Neural Networks*, 3(1):109–118, 1990.

Johan A.K. Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proc. of the National Academy of Sciences*, 99(10):6567–6572, 2002.

Kai M. Ting and Ian H. Witten. Stacking bagged and dagged models. In *Int. Conf. on Machine Learning*, pages 367–375, 1997.

Valentin Todorov and Peter Filzmoser. An object oriented framework for robust multivariate analysis. *J. Stat. Softw.*, 32(3):1–47, 2009.

Alfred Truong. *Fast Growing and Interpretable Oblique Trees via Probabilistic Models*. PhD thesis, Univ. Oxford, 2009.

Joaquin Vanschoren, Hendrik Blockeel, Bernhard. Pfahringer, and Geoffrey Holmes. Experiment databases. A new way to share, organize and learn from experiments. *Machine Learning*, 87(2):127–158, 2012.

William N. Venables and Brian D. Ripley. *Modern Applied Statistics with S*. Springer, 2002.

Geoffrey Webb, Janice Boughton, and Zhihai Wang. Not so naive Bayes: aggregating one-dependence estimators. *Machine Learning*, 58(1):5–24, 2005.

Geoffrey I. Webb. Multiboosting: a technique for combining boosting and wagging. *Machine Learning*, 40(2):159–196, 2000.

Daniela M. Witten and Robert Tibshirani. Penalized classification using Fisher's linear discriminant. *J. of the Royal Stat. Soc. Series B*, 73(5):753–772, 2011.

David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

David H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 9:1341–1390, 1996.

Zijian Zheng and Goeffrey I. Webb. Lazy learning of Bayesian rules. *Machine Learning*, 4 (1):53–84, 2000.

Journal of Machine Learning Research 15 (2014) 3183-3186 Submitted 6/12; Revised 6/13; Published 10/14

ooDACE Toolbox: A Flexible Object-Oriented Kriging Implementation

Ivo Couckuyt* IVO.COUCKUYT@UGENT.BE
Tom Dhaene TOM.DHAENE@UGENT.BE
Piet Demeester PIET.DEMEESTER@UGENT.BE
Ghent University - iMinds
Department of Information Technology (INTEC)
Gaston Crommenlaan 8
9050 Gent, Belgium

Editor: Mikio Braun

Abstract

When analyzing data from computationally expensive simulation codes, surrogate modeling methods are firmly established as facilitators for design space exploration, sensitivity analysis, visualization and optimization. Kriging is a popular surrogate modeling technique used for the Design and Analysis of Computer Experiments (DACE). Hence, the past decade Kriging has been the subject of extensive research and many extensions have been proposed, e.g., co-Kriging, stochastic Kriging, blind Kriging, etc. However, few Kriging implementations are publicly available and tailored towards scientists and engineers. Furthermore, no Kriging toolbox exists that unifies several Kriging flavors. This paper addresses this need by presenting an efficient object-oriented Kriging implementation and several Kriging extensions, providing a flexible and easily extendable framework to test and implement new Kriging flavors while reusing as much code as possible.

Keywords: Kriging, Gaussian process, co-Kriging, blind Kriging, surrogate modeling, metamodeling, DACE

1. Introduction

This paper is concerned with efficiently solving complex, computational expensive problems using surrogate modeling techniques (Gorissen et al., 2010). Surrogate models, also known as metamodels, are cheap approximation models for computational expensive (black-box) simulations. Surrogate modeling techniques are well-suited to handle, for example, expensive finite element (FE) simulations and computational fluid dynamic (CFD) simulations.

Kriging is a popular surrogate model type to approximate deterministic noise-free data. First conceived by Danie Krige in geostatistics and later introduced for the Design and Analysis of Computer Experiments (DACE) by Sacks et al. (1989), these Gaussian process (Rasmussen and Williams, 2006) based surrogate models are compact and cheap to evaluate, and have proven to be very useful for tasks such as optimization, design space exploration, visualization, prototyping, and sensitivity analysis (Viana et al., 2014). Note

∗. Ivo Couckuyt is a post-doctoral research fellow of FWO-Vlaanderen.

that Kriging surrogate models are primarily known as Gaussian processes in the machine learning community. Except for the utilized terminology there is no difference between the terms and associated methodologies.

While Kriging is a popular surrogate model type, not many publicly available, easy-to-use Kriging implementations exist. Many Kriging implementations are outdated and often limited to one specific type of Kriging. Perhaps the most well-known Kriging toolbox is the DACE toolbox[1] of Lophaven et al. (2002), but, unfortunately, the toolbox has not been updated for some time and only the standard Kriging model is provided. Other freely available Kriging codes include: stochastic Kriging (Staum, 2009),[2] DiceKriging,[3] Gaussian processes for Machine Learning (Rasmussen and Nickisch, 2010) (GPML),[4] demo code provided with Forrester et al. (2008),[5] and the Matlab Krigeage toolbox.[6]

This paper addresses this need by presenting an object-oriented Kriging implementation and several Kriging extensions, providing a flexible and easily extendable framework to test and implement new Kriging flavors while reusing as much code as possible.

2. ooDACE Toolbox

The ooDACE toolbox is an object-oriented Matlab toolbox implementing a variety of Kriging flavors and extensions. The most important features and Kriging flavors include:

- Simple Kriging, ordinary Kriging, universal Kriging, stochastic Kriging (regression Kriging), blind- and co-Kriging.
- Derivatives of the prediction and prediction variance.
- Flexible hyperparameter optimization.
- Useful utilities include: cross-validation, integrated mean squared error, empirical variogram plot, debug plot of the likelihood surface, robustness-criterion value, etc.
- Proper object-oriented design (compatible interface with the DACE toolbox[1] is available).

Documentation of the ooDACE toolbox is provided in the form of a getting started guide (for users), a wiki[7] and doxygen documentation[8] (for developers and more advanced users). In addition, the code is well-documented, providing references to research papers where appropriate. A quick-start demo script is provided with five surrogate modeling use cases, as well as script to run a suite of regression tests.

A simplified UML class diagram, showing only the most important public operations, of the toolbox is shown in Figure 1. The toolbox is designed with efficiency and flexibility in mind. The process of constructing (and predicting) a Kriging model is decomposed in several smaller, logical steps, e.g., constructing the correlation matrix, constructing the

1. The DACE toolbox can be downloaded at `http://www2.imm.dtu.dk/~hbn/dace/`.
2. The stochastic Kriging toolbox can be downloaded at `http://stochasticKriging.net/`.
3. The DiceKriging toolbox can be downloaded at `http://cran.r-project.org/web/packages/DiceKriging/index.html`.
4. The GPML toolbox can be downloaded at `http://mloss.org/software/view/263/`.
5. Demo code of Kriging can be downloaded at `http://www.wiley.com//legacy/wileychi/forrester/`.
6. The Krigeage toolbox can be downloaded at `http://globec.whoi.edu/software/kriging/`.
7. The wiki documentation of the ooDACE toolbox is found at `http://sumowiki.intec.ugent.be/index.php/ooDACE:ooDACE_toolbox`.
8. The doxygen documentation of the ooDACE toolbox is found at `http://sumo.intec.ugent.be/buildbot/ooDACE/doc/`.

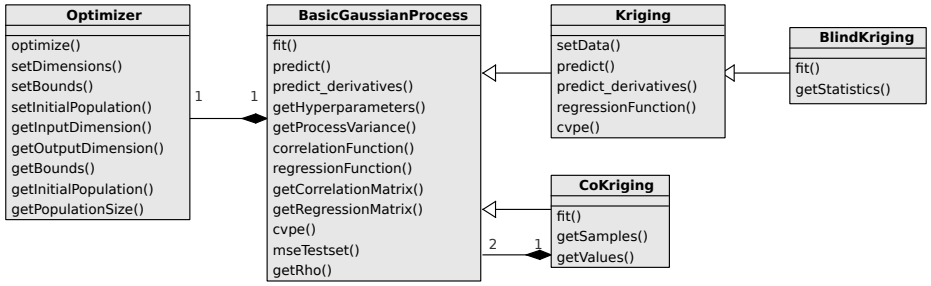


Figure 1: Class diagram of the ooDACE toolbox.

regression matrix, updating the model, optimizing the parameters, etc. These steps are linked together by higher-level steps, e.g., fitting the Kriging model and making predictions. The basic steps needed for Kriging are implemented as (protected) operations in the BasicGaussianProcess superclass. Implementing a new Kriging type, or extending an existing one, is now done by subclassing the Kriging class of your choice and inheriting the (protected) methods that need to be reimplemented. Similarly, to implement a new hyperparameter optimization strategy it suffices to create a new class inherited from the Optimizer class.

To assess the performance of the ooDACE toolbox a comparison between the ooDACE toolbox and the DACE toolbox[1] is performed using the 2D Branin function. To that end, 20 data sets of increasing size are constructed, each drawn from an uniform random distribution. The number of observations ranges from 10 to 200 samples with steps of 10 samples. For each data set, a DACE toolbox[1] model, a ooDACE ordinary Kriging and a ooDACE blind Kriging model have been constructed and the accuracy is measured on a dense test set using the Average Euclidean Error (AEE). Moreover, each test is repeated 1000 times to remove any random factor, hence the average accuracy of all repetitions is used. Results are shown in Figure 2a. Clearly, the ordinary Kriging model of the ooDACE toolbox consistently outperforms the DACE toolbox for any given sample size, mostly due to a better hyperparameter optimization, while the blind Kriging model is able improve the accuracy even more.

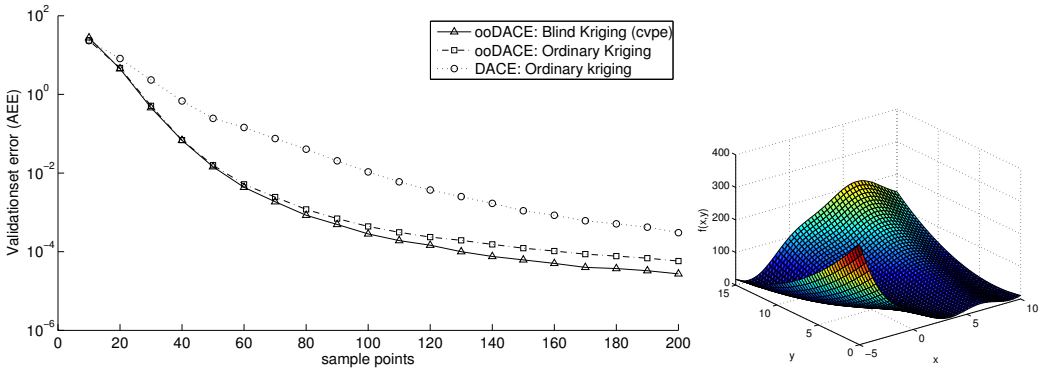
3. Applications

The ooDACE Toolbox has already been applied successfully to a wide range of problems, e.g., optimization of a textile antenna (Couckuyt et al., 2010), identification of the elasticity of the middle-ear drum (Aernouts et al., 2010), etc.

In sum, the ooDACE toolbox aims to provide a modern, up to date Kriging framework catered to scientists and engineers. Usage instructions, design documentation, and stable releases can be found at `http://sumo.intec.ugent.be/?q=ooDACE`.

References

J. Aernouts, I. Couckuyt, K. Crombecq, and J.J.J. Dirckx. Elastic characterization of membranes with a complex shape using point indentation measurements and inverse



(a) (b)

Figure 2: (a) Evolution of the average AEE versus the number of samples (Branin function). (b) Landscape plot of the Branin function.

modelling. *International Journal of Engineering Science*, 48:599–611, 2010.

I. Couckuyt, F. Declercq, T. Dhaene, and H. Rogier. Surrogate-based infill optimization applied to electromagnetic problems. *Journal of RF and Microwave Computer-Aided Engineering: Advances in Design Optimization of Microwave/RF Circuits and Systems*, 20(5):492–501, 2010.

A. Forrester, A. Sobester, and A. Keane. *Engineering Design Via Surrogate Modelling: A Practical Guide*. Wiley, Chichester, 2008.

D. Gorissen, K. Crombecq, I. Couckuyt, P. Demeester, and T. Dhaene. A surrogate modeling and adaptive sampling toolbox for computer based design. *Journal of Machine Learning Research*, 11:2051–2055, 2010. URL `http://sumo.intec.ugent.be/`.

S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. Aspects of the Matlab toolbox DACE. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2002.

C. E. Rasmussen and H. Nickisch. Gaussian processes for machine learning (GPML) toolbox. *Journal of Machine Learning Research*, 11:3011–3015, 2010.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

J. Sacks, W. J. Welch, T.J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, 1989.

J. Staum. Better simulation metamodeling: The why, what, and how of stochastic Kriging. In *Proceedings of the Winter Simulation Conference*, 2009.

F. A. C. Viana, T. W. Simpson, V. Balabanov, and V. Toropov. Metamodeling in multidisciplinary design optimization: How far have we really come? *AIAA Journal*, 52(4): 670–690, 2014.

Journal of Machine Learning Research 15 (2014) 3187-3220 Submitted 2/14; Revised 5/14; Published 10/14

Robust Online Gesture Recognition with Crowdsourced Annotations

Long-Van Nguyen-Dinh LONGVAN@IFE.EE.ETHZ.CH
Alberto Calatroni ALBERTO.CALATRONI@IFE.EE.ETHZ.CH
Gerhard Tröster TROESTER@IFE.EE.ETHZ.CH
Wearable Computing Lab
ETH Zürich
ETZ H 95, Gloriastrasse 35, Zürich 8092, Switzerland

Editor: Isabelle Guyon, Vassilis Athitsos and Sergio Escalera

Abstract

Crowdsourcing is a promising way to reduce the effort of collecting annotations for training gesture recognition systems. Crowdsourced annotations suffer from "noise" such as mislabeling, or inaccurate identification of start and end time of gesture instances. In this paper we present SegmentedLCSS and WarpingLCSS, two template-matching methods offering robustness when trained with noisy crowdsourced annotations to spot gestures from wearable motion sensors. The methods quantize signals into strings of characters and then apply variations of the longest common subsequence algorithm (LCSS) to spot gestures. We compare the noise robustness of our methods against baselines which use dynamic time warping (DTW) and support vector machines (SVM). The experiments are performed on data sets with various gesture classes (10-17 classes) recorded from accelerometers on arms, with both real and synthetic crowdsourced annotations. WarpingLCSS has similar or better performance than baselines in absence of noisy annotations. In presence of 60% mislabeled instances, WarpingLCSS outperformed SVM by 22% F1-score and outperformed DTW-based methods by 36% F1-score on average. SegmentedLCSS yields similar performance as WarpingLCSS, however it performs one order of magnitude slower. Additionally, we show to use our methods to filter out the noise in the crowdsourced annotation before training a traditional classifier. The filtering increases the performance of SVM by 20% F1-score and of DTW-based methods by 8% F1-score on average in the noisy real crowdsourced annotations.

Keywords: gesture spotting, crowdsourced annotation, longest common subsequence, template matching methods, accelerometer sensors

1. Introduction

Wearable computing is gaining momentum through the availability of an increasing choice of devices, like smart watches, glasses and sensor-equipped garments. A core component to allow these devices to understand our context is online gesture recognition (*spotting*) in which types of gestures and their temporal boundaries must be recognized in the incoming streaming sensor data. This is carried out using machine learning approaches on different sensing modalities, like acceleration (Bao and Intille, 2004) and video (Elmezain et al., 2009; Yoon et al., 2001).

Training a gesture recognition system requires an annotated training data set that is used to perform supervised learning (Bao and Intille, 2004; Ravi et al., 2005; Aggarwal and Ryoo, 2011; Chen et al., 2012). Specifically, the annotations comprise the start and end times (i.e., temporal boundaries) of gestures of interest and their corresponding labels. Reference data sets are usually annotated by a small number of experts to be as accurate as possible. However, the labeling process is extremely time-consuming: it may take up to 7-10 hours to annotate gestures in a 30-min video (Roggen et al., 2010). Moreover, it is also costly to hire experts to annotate data corpora.

Crowdsourcing has been emerged recently to address these issues (Howe, 2006; Doan et al., 2011). Crowdsourcing is defined as a model that outsources tasks which are traditionally performed by experts to a crowd of ordinary people. Thus, crowdsourcing is promising to reduce the cost and time of labeling. Recently, crowdsourcing has been exploited to get labeling for training data sets for gesture recognition (Nguyen-Dinh et al., 2013c). However, labels obtained from crowdsourcing are provided by low-commitment anonymous workers, thus they are commonly unreliable and noisy (Sheng et al., 2008). In gesture annotation from crowdsourcing, the challenge is to obtain labels matching ground truth, attaining both correct labels and correct temporal boundaries.

Using multiple annotators for the same annotation task by watching videos or audios is a popular strategy to get a good annotation from crowdsourcing (Yuen et al., 2011; Nguyen-Dinh et al., 2013c). However, multiple annotators may not be applicable in some cases, either due to the higher cost or because of some privacy concerns. This latter case occurs when the annotation involves some personal context information, including for example location or other sensitive data. Hence, the annotation is often provided and relied on the crowdsourced user for his recorded data. Moreover, it is very difficult to ask the anonymous low-commitment user to clean his annotation because it is time consuming and he may not remember exactly what he has done. In these cases, the large presence of noise in the training data annotation can degrade significantly the performance.

While other research is focusing on how to improve the quality of crowdsourced annotations, we here point out the need for algorithms that can cope with the kinds of annotation errors that will anyway remain. In this work, we show that our proposed template matching methods (TMMs) based on the longest common subsequence algorithm (known as LCSS or LCS in the literature) are suitable for online gesture recognition in a setting where training data are affected significantly by labeling noise. Additionally, the work targets the recognition of gestures based on acceleration data recorded from only one accelerometer mounted on the user's arm. The reason to just use one sensor is that this setting will be the most common one with smart watches in the close future. Recognizing gestures with just motion data from one sensor is challenging due to the ambiguities in the sensor data, especially with high percentage of *null* class (no gesture of interest).

1.1 Contributions

In this paper, we make the following contributions:

1. We discuss how gesture recognition systems can leverage crowdsourcing to collect annotated data. We address the challenges that arise and then propose a taxonomy

of annotation noise which occur in a crowdsourcing setting. We also give analysis on annotation noise in the real crowdsourced annotated data set.

2. We propose SegmentedLCSS and WarpingLCSS as TMMs for online gesture recognition. These methods were first presented in our previous work (Nguyen-Dinh et al., 2012) and have been shown to perform well in clean annotated gesture data sets both in terms of computational complexity and accuracy. In this work, we show their robustness to the labeling noise from crowdsourcing.

3. We compare the robustness of our gesture recognition methods against three baselines using two variations of dynamic time warping and support vector machines. The algorithms are tested with annotations collected in real crowdsourcing scenarios as well as the synthetic crowdsourced annotations in three data sets recorded from accelerometers on arms. We also investigate the impact of different kinds of noises in crowdsourced annotation on the performance of the gesture recognition methods.

4. We investigate the property of LCSS of being able to select clean templates, which makes it suitable also as a filtering component to select good training examples despite noisy annotations. This filter can be used in combination with other classifiers. We show how inserting this filtering step improves the performance of SVMs and TMMs based on dynamic time warping.

The rest of the paper is organized as follows. In Section 2, we first review existing work in online gesture recognition and crowdsourcing. In Section 3, we discuss crowdsourcing in gesture recognition and propose a taxonomy of annotation noise in gesture labeling by crowdsourcing. Then, in Section 4, we present our proposed SegmentedLCSS and WarpingLCSS methods. The experiments are described in Section 5. We present quantitative results evaluating the robustness of our proposed methods against the baselines in Section 6. Finally, Section 7 concludes our work and gives some potential research directions.

2. Related Work

In this section we discuss related work in the fields of gesture recognition and crowdsourcing, pointing out the lack of an analysis of how noise present in typical crowdsourced annotations impacts gesture recognition algorithms.

2.1 Annotation Techniques

Supervised learning techniques require a set of annotated training samples to build gesture models. Therefore, many annotation techniques have been proposed to collect annotated data. There are *offline annotation* techniques which rely on video and audio recordings (Roggen et al., 2010), subject self-report of activities at the end of the day (Van Laerhoven et al., 2008). *Online annotation* (i.e., real-time) techniques perform the annotation during execution of the activities, like experience sampling (Froehlich et al., 2007) which prompts periodically to a user to ask information about his current activities, or direct annotation in which users responsibly provide a label before an activity begins and indicate when the activity ends (Rossi et al., 2012). There is a trade-off between accuracy of an annotation

technique and the amount of time required for annotation (Stikic et al., 2011). For example, offline annotation on video recordings by experts can provide accurate annotations, however it is extremely time consuming (Roggen et al., 2010), and non-scalable to large number of users. In contrast, the self-report of the subject may require less time but the accuracy depends on the subject's ability to recall activities. Therefore, most of the existing works require video annotation by experts to obtain clean and correct annotated data sets (Roggen et al., 2010) or provide a course to teach subjects carefully how they should record and annotate their data correctly (Bao and Intille, 2004).

2.2 Crowdsourcing

Crowdsourcing services, like Amazon Mechanical Turk (AMT)[1] and Crowdflower[2], have emerged recently as a new cheap labor pool to distribute annotation tasks to a large number of workers (Yuen et al., 2011). Crowdsourcing tasks are performed by low-commitment anonymous workers, thus acquired data is commonly unreliable and noisy (Sheng et al., 2008). Therefore, the same task is often redundantly performed by multiple workers and majority voting is a popular decision making method used to identify the correct answers (Yuen et al., 2011). Moreover, in crowdsourcing, malicious workers often take advantage of the verification difficulty (the ground truth is unknown) and submit low-quality answers.

Due to the error-prone nature of crowdsourcing, several strategies were proposed to estimate the quality of workers, in order to reject low-performing and malicious workers. Verifiable questions or pilot tasks for which the requester knows the correct answers is a common empirical strategy to screen workers from crowdsourcing (Kittur et al., 2008; Yuen et al., 2011). Another way to ensure quality is to check the agreement in annotations among workers to detect non-serious workers (Nguyen-Dinh et al., 2013c). Dawid and Skene (1979) proposed a theoretical model that used the redundancy in acquiring answers (i.e., the same task is completed by multiple workers) to measure the labeling quality of the workers. Recently, Raykar et al. (2010) proposed Bayesian versions of worker quality inference. Ipeirotis et al. (2010) improved the method by separating spammers who provide low-quality answers intentionally from biased workers who are careful but biased.

Recently, crowdsourcing has been exploited also in the field of activity recognition to collect annotated training data sets (Rossi et al., 2012; Nguyen-Dinh et al., 2013a,b,c; Lasecki et al., 2013). These works showed that crowdsourced data is erroneous, therefore, filtering strategies such as multiple labelers and outlier removal should be used to reduce labeling noise.

Although many strategies are used to reduce noise in crowdsourced data annotation, there is no guarantee to have a perfect annotation, especially when using multiple labelers can not be applied. Until now, the impact of the noisy annotations in crowdsourcing on the training of gesture recognition methods was not investigated. Furthermore, the nature of the noise that affects the annotations in a crowdsourcing scenario for gesture recognition has not been analyzed yet. These two latter topics are subject of the present paper.

1. The home page for AMT is `http://www.mturk.com`.
2. The home page for Crowdflower is `http://crowdflower.com`.

2.3 Online Gesture Recognition Methods

Signals from body-worn sensors belong to the category of time series data. Suitable machine learning and pattern recognition techniques for online gesture recognition include Hidden Markov Models (HMM) (Lee and Kim, 1999; Deng and Tsui, 2000; Junker et al., 2008; Schlömer et al., 2008), template matching methods (TMM) using mostly dynamic time warping—in short DTW (Ko et al., 2005; Stiefmeier et al., 2008; Hartmann and Link, 2010) and support vector machines (Ravi et al., 2005; He et al., 2008; Wu et al., 2009).

HMMs are not appealing since a large amount of training data is required to get results comparable to other TMMs and SVM. In Vogler and Metaxas (1999) for example, about 1300 instances for 22 classes (i.e., about 60 instances per class) are used to train the HMM, whereas TMMs can work with as little as one training instance per class. The issue of the amount of training data is mentioned also in Cooper et al. (2012), where the authors state, referring to HMMs: "While they have been employed for sign recognition, they have issues due to the large training requirements". In Alon et al. (2009), a variation of HMMs is selected but the parameters could not be learnt because of the scarcity of training data: "We fix the transition probabilities to simplify the learning task, because we do not have sufficient training data to learn more parameters". HMMs remain nevertheless an interesting approach for cases where a large data corpus is available, which is often the case in the field of video-based gesture or sign language recognition, see for example Wilson and Bobick (1999); Lee and Kim (1999); Keskin et al. (2011).

Segmented DTW (Ko et al., 2005; Hartmann and Link, 2010) performs online gesture recognition by first buffering the streaming signals into an observation window. A *t*est segment is a sequence that is examined to classify whether it is an instance of a gesture class. The start and end boundaries of a test segment can vary inside the window. A DTW distance is computed between all templates which represents gesture classes and the test segment, and the class of the closest template is eventually selected as label for the test segment if the distance falls below a certain rejection threshold. As the sensor delivers a new reading, the window is shifted by one sample and the process is repeated. Segmented DTW is time consuming since DTW is recomputed to find the best boundaries for the test segment inside the window and it is also recomputed every time the window shifts by one sample. A *nonsegmented DTW* variation was proposed by Stiefmeier et al. (2008) to reuse the computation of previous readings, recognize gestures and determine their boundaries without segmenting the stream.

Along with DTW, the other commonly used similarity measure for matching two time series is *LCSS* (Fu, 2011). In previous work (Nguyen-Dinh et al., 2012), we introduced two variations of LCSS-based template matching for online gesture spotting and recognition. We applied the methods to accelerometer data. These LCSS-based classifiers (SegmentedLCSS and WarpingLCSS) proved to outperform DTW-based TMMs, both in terms of computational complexity and accuracy (especially for data sets containing high variability in gesture execution as shown in Nguyen-Dinh et al., 2012). Furthermore, our methods were designed with the goal of being robust in case of noisy annotations. The validation of this aspect is the main topic of the present article. The impact of the various kinds of noise occurring in crowdsourced annotations on TMMs has not been investigated in previous literature, to the best of our knowledge.

In sign language recognition literature, we find two other works proposing the use of LCSS as a classifier, applied to video data (Frolova et al., 2013; Stern et al., 2013). In both cases, the methods use a sliding window to set temporal boundaries of a gesture inside the window, similarly to our SegmentedLCSS. With our WarpingLCSS, this need of using a window is removed, reducing the computational complexity. It is interesting to note how Stern et al. (2013) states that "It can then be said that the MDSLCS algorithm can outperform the HMM classifier for both pre-cut and streaming gestures", which supports the idea of using TMMs instead of HMMs to make best use of the available training data. TMMs are competitive with HMMs also with respect to null-class rejection, meaning the ability to spot a gesture within a continuous stream.

Some algorithms present in the literature rely on k-means or spatio-temporal clustering to transform the raw signals into so-called "fenemes", or subunits (Bauer and Karl-Friedrich, 2002; Fang et al., 2004), which allows to reduce the amount of training data, due to the fact that more gestures can contain the same feneme, so that a critical mass can be achieved in terms of amount of training data. We use a similar approach based on k-means clustering to find a quantization of the signals which gives good results.

A large body of literature focuses on a recognition performed on video data, for example for the recognition of sign language (see for example Wilson and Bobick, 1999; Bowden et al., 2004; Alon et al., 2009; Keskin et al., 2011). However, gesture recognition from wearable sensors, e.g., one accelerometer at the wrist, would allow to scale up the recognition system to many users immediately because the system can be deployed easily wherever a user goes with the motion sensor mounted on hand. It does not need any other infrastructure like cameras, which do not follow us everywhere in practice. Of the video-based approaches, the one of Hao and Shibata (2009) captures the videos directly by a moving camera, which could be easily wearable. However, from the practical point of view, such an option has some limitations: first, such a device would be quite costly; second, processing signals from a camera is more computationally intensive than processing those from a motion sensor; third, capturing video data is much more intrusive due to privacy concerns.

2.4 Robustness against Annotation Noise

The impact of noise in annotations on the performance of classifiers has been investigated in the literature (Angluin and Laird, 1988; Amini and Gallinari, 2005; Gayar et al., 2006; Lawrence and Schölkopf, 2001; Stikic et al., 2011). The above cited studies do not concern template matching methods. Moreover, they conducted experiments on synthetic noisy data. Additionally, under "annotation noise", or "class noise", only the case of having wrong labels (i.e., labels are substituted as other classes) was considered. Noise in gestures annotation can nevertheless also mean having labelings with temporal boundaries differing from the ground truth, e.g., a gesture marked as starting earlier and ending later than the ground truth. These other kinds of noise were neglected until now, and they are investigated in this paper in both synthetic and real crowdsourced annotated data.

3. Crowdsourcing in Gesture Recognition

In this section we discuss how gesture recognition systems can leverage crowdsourcing. We outline the challenges that arise and provide a taxonomy of the annotation noises, i.e., the

mistakes that affect crowdsourced annotations. We then measure these annotation noises in a real crowdsourced data set.

Gesture recognition systems can take advantages of crowdsourcing in three ways:

1. Crowdsourcing can be used to acquire annotations for an existing gesture data set by asking crowdsourced workers to watch video footage synchronized with the sensor data (Nguyen-Dinh et al., 2013c; Lasecki et al., 2013).

2. Berchtold et al. (2010) proposed a system that asks users to both record and annotate activities. This system can be deployed in a crowdsourcing manner. Users can sporadically select gestures they want to perform and record them with a device (e.g, smart watch, smart phone, etc.). This way, multiple annotated gestures provided by a large user base could contribute to a central repository which grows in time. The data set would capture the variability in gesture execution due to the different people contributing.

3. A more obtrusive crowdsourcing task would ask users to record and annotate as many activities and gestures as possible over a long time span (e.g., weeks). This type of crowdsourced data collection would be useful to gather data for long-term health care monitoring systems.

In any of the previous scenarios, the outcome would be an annotated training data set, with which algorithms can be trained. The benefit of the crowdsourcing setting is that a large data set can be collected quickly, if the crowdsourced user base is large enough.

3.1 Taxonomy of Sources of Annotation Noises

The major challenge in any of the settings outlined above is the quality of the labels obtained, which are unreliable for many reasons. We define the following taxonomy of annotation noises along with examples:

- Some gestures or activities can be understood differently with respect to when they actually start and end. The temporal boundaries of the gesture *drink* can be set from the time when the user picks up a glass to when he or she puts it back to the table. Another variation is that the gesture is annotated only when the person is actually drinking. Both annotations are valid, but this uncertainty of temporal boundaries has an impact on the algorithms that will be trained with the collected annotated data. However, even when we assume the definition of gesture boundary is given, the errors in gesture boundary still happen due to the carelessness of crowdsourced labelers. We call this form of noise *boundary jitter*. We define *boundary jitter* as the presence of a shift in the annotation boundaries, while the label matches the actual gesture (ground truth).

- Some instances of gestures can be wrongly annotated or missed altogether. This can occur for example if the video footage does not have enough resolution to spot subtle manipulative gestures, or more simply if the labeler does not annotate all gestures that are occurring. We use the term *label noise* to denote instances where gestures are associated to wrong labels or to no label at all.

We further categorize *boundary jitter* into four error types, namely *extend, shrink, shift left* and *shift right* according to how the temporal boundary of a gesture is shifted compared to the ground truth. Figure 1a illustrates the subclasses of *boundary jitter*.

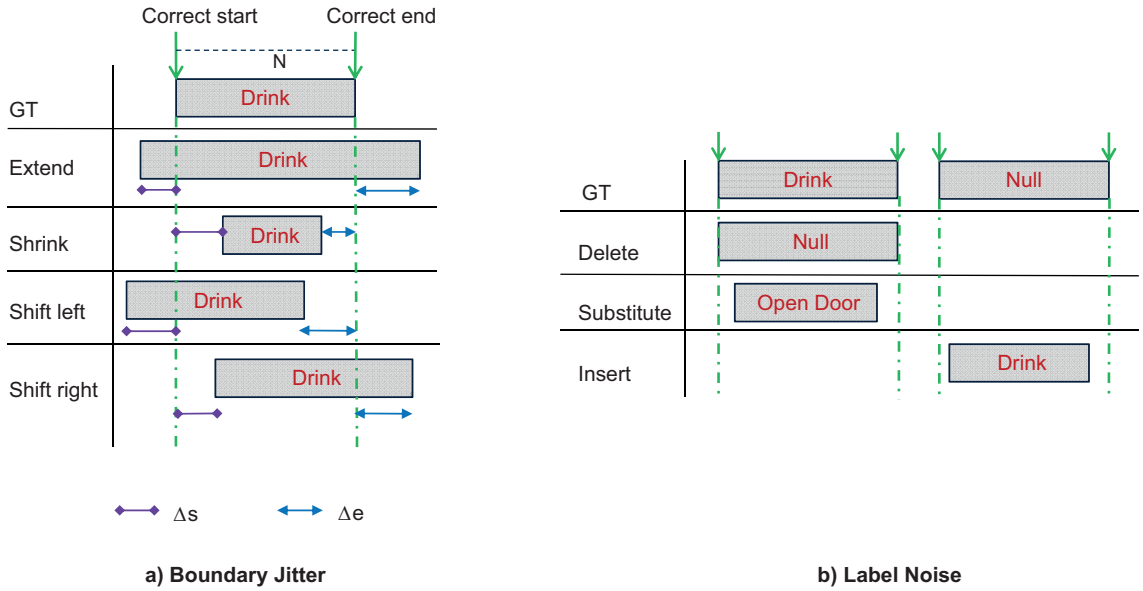


Figure 1: Illustrations of boundary jitter and label noise in crowdsourcing annotation. GT stands for ground truth. The blue dash-dotted lines indicate the correct boundary of a gesture.

- *Extend*: The starting boundary is set earlier and the ending boundary is set later. The information of the gesture instance is preserved, but noise is attached to the gesture instance in the form of samples which belong actually to another gesture class or to no class of interest at all (i.e., *null* class).
- *Shrink*: The starting boundary is set later and the ending boundary is set earlier. In this case, some information of the gesture instance is missed.
- *Shift left*: Both starting and ending boundaries are set earlier. In this case, some information of the gesture instance is missed and noise is added at the end of the gesture.
- *Shift right*: Both starting and ending boundaries are set later. In this case, some information of the gesture instance is missed and noise is added at the beginning of the gesture.

We also categorize *label noise* into three error types, namely *delete, substitute* and *insert*.

- *Delete*: A gesture instance is not annotated. It is automatically marked as *null* class.

- *Substitute*: A gesture instance is labeled as another gesture class.
- *Insert*: A gesture instance is labeled where no gesture of interest actually occurs.

Figure 1b illustrates the subclasses of *label noise*. The subclasses of *label noise* are similar to the definition of classification errors evaluated in performance metrics proposed by Ward et al. (2011). However, in this work, we consider those errors existing in annotations of training data set.

3.2 Annotation Noise Parameters

Along with the taxonomy provided in the previous section, we here list the parameters that quantify the amount of noise in the annotation. Given a gesture instance, let *start* and *end* be the start time and end time of the crowdsourced annotation. Let *GT_start* and *GT_end* be the corresponding ground truth boundaries. Let N denote the time length of the gesture ($N = |GT\_end - GT\_start|$). We define Δs as the time difference between the crowdsourced start time and the correct start time ($\Delta s = |start - GT\_start|$). Similarly, we define Δe as the time difference between the crowdsourced end time and the correct end time ($\Delta e = |end - GT\_end|$). Δs and Δe are illustrated in Figure 1a for the different boundary jitter noises.

For *boundary jitter* and for the corresponding subclasses, we define a *jitter level* to quantify the proportion of time that is wrongly annotated in a gesture due to the jitter. The jitter level also indicates how much the boundaries stray from the correct annotation. These jitter parameters are calculated as follows:

$$\begin{aligned}
\text{extend level} &= \text{proportion of time noisy samples added} \\
&= \frac{\Delta s + \Delta e}{N}.
\end{aligned}$$

$$\begin{aligned}
\text{shrink level} &= \text{proportion of time good samples missed} \\
&= \frac{\Delta s + \Delta e}{N}.
\end{aligned}$$

$$\begin{aligned}
\text{shift-left level} &= \text{proportion of time noisy samples added and good samples missed / 2} \\
&= \frac{\Delta s + \Delta e}{2 * N}.
\end{aligned}$$

$$\begin{aligned}
\text{shift-right level} &= \text{proportion of time noisy samples added and good samples missed / 2} \\
&= \frac{\Delta s + \Delta e}{2 * N}.
\end{aligned}$$

3.3 Annotation Noise Statistics from A Real Crowdsourcing Experiment

To give a flavor of typical values encountered for the annotation noise levels, we report these levels measured in a real crowdsourcing experiment that we conducted in a previous study (Nguyen-Dinh et al., 2013c). In the crowdsourcing experiment we used video footage belonging to the Opportunity data set (Roggen et al., 2010), which contains gestures of normal daily routines (e.g., drink, open or close doors). We showed each short video to ten workers in Amazon Mechanical Turk (AMT), described the task and collected their

annotations. The AMT labelers must annotate the start, end boundaries and the label of all occurrences of gestures of interest in the videos. We applied two strategies to detect and filter non-serious labelers and erroneous labeling (Nguyen-Dinh et al., 2013c). Individual filtering checks the correctness in the answers of each labeler for qualification questions whose answers are known in advance. Collaborative filtering checks the agreement in annotations among workers to detect non-serious labelers. Specifically, the labeler X who has a disagreement score $d(X) = \frac{\text{Annotation times of X disagree with majority}}{\text{Total annotation times of X}} > threshold$ is a spammer. We chose a threshold = 0.3, it means if the disagreement score $d \geq 0.3$ (i.e., less than 70% of annotation of a labeler agrees with the majority), the labeler is a spammer and his annotations are removed. The collaborative filtering is illustrated in Figure 2. After filtering, the majority voting among qualified annotations is performed to generate a final crowdsourced gesture annotation. A more detail on the crowdsourcing experiment is given in Nguyen-Dinh et al. (2013c).

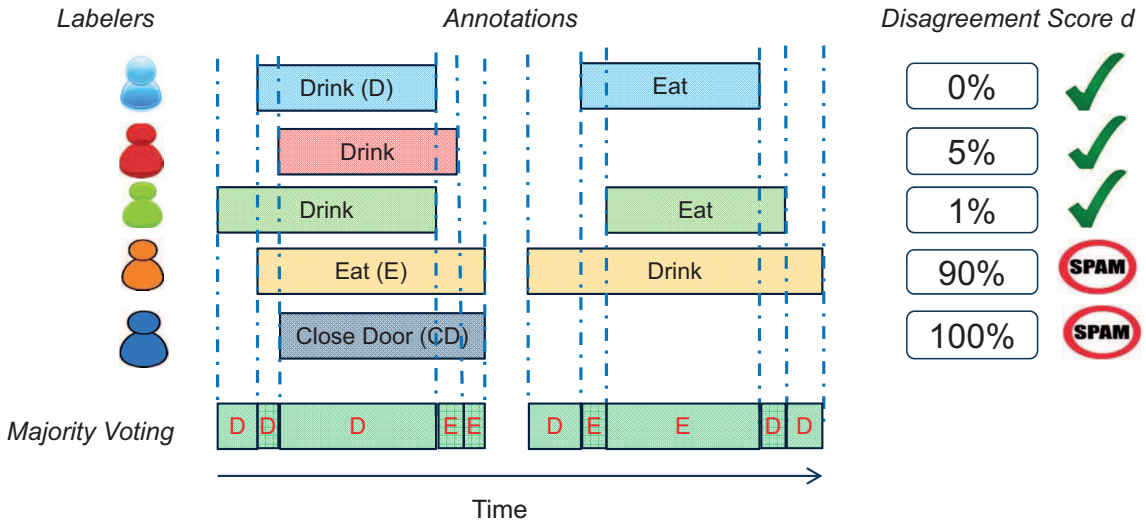


Figure 2: An illustration of the collaborative filtering technique to calculate the disagreement score of each labeler against the majority. The last two labelers are spammers and then their annotations will be removed.

Each video footage of the Opportunity data set was already examined and annotated carefully by one expert (Roggen et al., 2010) and the expert's annotations are used as a ground truth to evaluate our crowdsourced annotation. Here we report the sample-based accuracy (i.e., fraction of correctly labeled samples compared to expert's annotation) for a one-labeler annotation scenario where only one crowdsourced labeler is selected, and for a multiple-labeler scenario where the filterings and majority voting are applied for the ten workers. For a one-labeler annotation, the sample-based accuracy gets as low as 55%. In the multiple-labeler annotation, the accuracy reaches 80%. A breakdown of the types of annotation mistakes, according to the taxonomy introduced in Section 3.1, is shown in Figure 3a. The values for *label noise* and for the *boundary jitter* are shown for one and

for multiple labelers. In the scenario of only one labeler, about 52% of the instances are affected by *label noises*, comprising mostly *substitute* and *delete* errors. In the multiple-labeler scenario, *label noise* decreases to 18%. In Figure 3b, we give the average, the min and the max values of jitter level of boundary jitters for one and for multiple labelers. On average, jitter levels ranges from 27% to 60%. However, there are good annotated instances with very low jitter levels (only 2%).

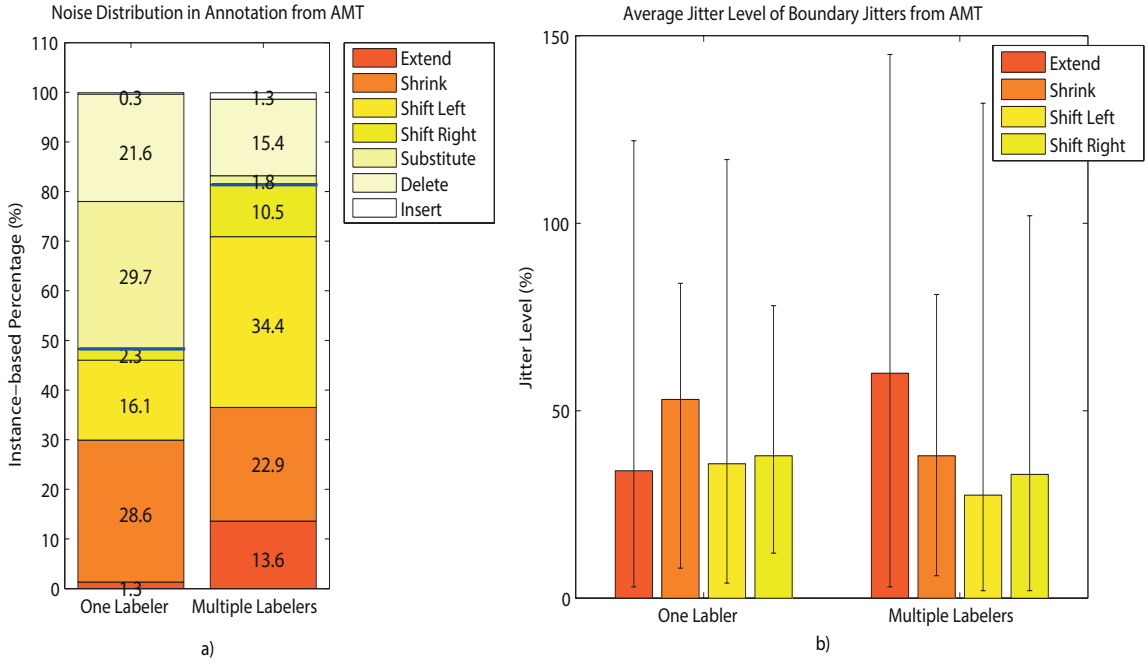


Figure 3: Analysis of crowdsourcing annotation from AMT. Blue lines in the figure a separate boundary jitter part and label noise part. Black lines in the figure b show the minimum and maximum level of jitter in each type of noise.

It can be seen that requesting multiple labelers for an annotation task can reduce labeling errors. However, the result from a one-labeler annotation represents for the scenarios where multiple labelers cannot be applied. Our experiment belongs to the first crowdsourcing category described at the beginning of the present section, i.e., crowdsourcing labeling of data which were previously recorded. The amount and distribution of annotation noises will change depending on the crowdsourcing scenario and on the kind of gesture data, but there is no reason to think that some scenarios will achieve much lower noise levels. On the contrary, in real-time annotation (i.e., providing labels while recording data) , it is more likely that the level of noise increases: more gestures could be forgotten and others would be labeled only after they really occurred, leading to imprecise time boundaries. We therefore argue that annotation noise is a fact that cannot be completely removed and that calls the attention of robust methods when designing gesture recognition systems which use noisy crowdsourced annotations.

In the next sections we present our SegmentedLCSS and WarpingLCSS TMMs which are designed with the aim of being robust to annotation noise for gesture recognition.

4. SegmentedLCSS and WarpingLCSS Gesture Recognition Methods

In this section, we describe in details our proposed methods, Segmented LCSS and WarpingLCSS for online gesture recognition using signals obtained from body-worn sensors.

The methods proposed to recognize gestures are based on template matching (TM). The training phase uses a set of labeled signals to train the gesture recognition algorithm. In the training phase, the sensor signals are quantized and converted into sequences of symbols (strings); furthermore, one template is created for each gesture of interest. When deploying the recognition algorithm, the quantization scheme is again applied to the streaming signals. The strings obtained are then compared with the learned templates by either using the longest common subsequence (LCSS) algorithm in segmented windows (SegmentedLCSS) or using our faster variant of LCSS (namely WarpingLCSS). Figure 4 shows the data flow through different processing components in the training phase and the recognition phase of our proposed system.

The rationale using LCSS is that it gives a measure of similarity between templates and signals to be recognized. Moreover, LCSS is robust to the high variability in gesture execution as shown in our previous work (Nguyen-Dinh et al., 2012) because LCSS can ignore the dissimilarity and accumulate the similarity between two gesture instances.

In the following, we first briefly review LCSS, then we describe the different processing components of the recognition system in Figure 4.

4.1 The Longest Common Subsequence Algorithm (LCSS)

Let s_A and s_B be two strings comprising l_A and l_B symbols respectively. Let $s(i)$ denote the i-th symbol within a string s. For each pair of positions $0 \leq i \leq l_A$ and $0 \leq j \leq l_B$ within the strings, we call $LCSS_{(A,B)}(i,j)$ the length of the longest symbol subsequence in common between the first i symbols of s_A and the first j symbols of s_B. The LCSS between the complete strings is then denoted as $L_{(A,B)}$ or, when the strings are clear from the context, just with L.

$$L_{(A,B)}(i,j) = \begin{cases} 0 \quad \text{, if } i = 0 \text{ or } j = 0 \\ L_{(A,B)}(i-1,j-1) + 1 \quad \text{, if } s_A(i) = s_B(j) \\ \max \begin{cases} L_{(A,B)}(i-1,j) \\ L_{(A,B)}(i,j-1) \end{cases} \quad \text{, otherwise.} \end{cases} \tag{1}$$

Let Ω_A and Ω_B be the sets of indices corresponding to the longest subsequences of s_A and s_B that are matching. The sets $\Omega_A = \omega_A^{(0)} \ldots \omega_A^{(L-1)}$ and $\Omega_B = \omega_B^{(0)} \ldots \omega_B^{(L-1)}$ contain then $L_{(A,B)}$ indices. $L_{(A,B)}$ and the corresponding matching subsequences, hence the sets Ω_A and Ω_B, can be found using dynamic programming (see Cormen et al., 2001).

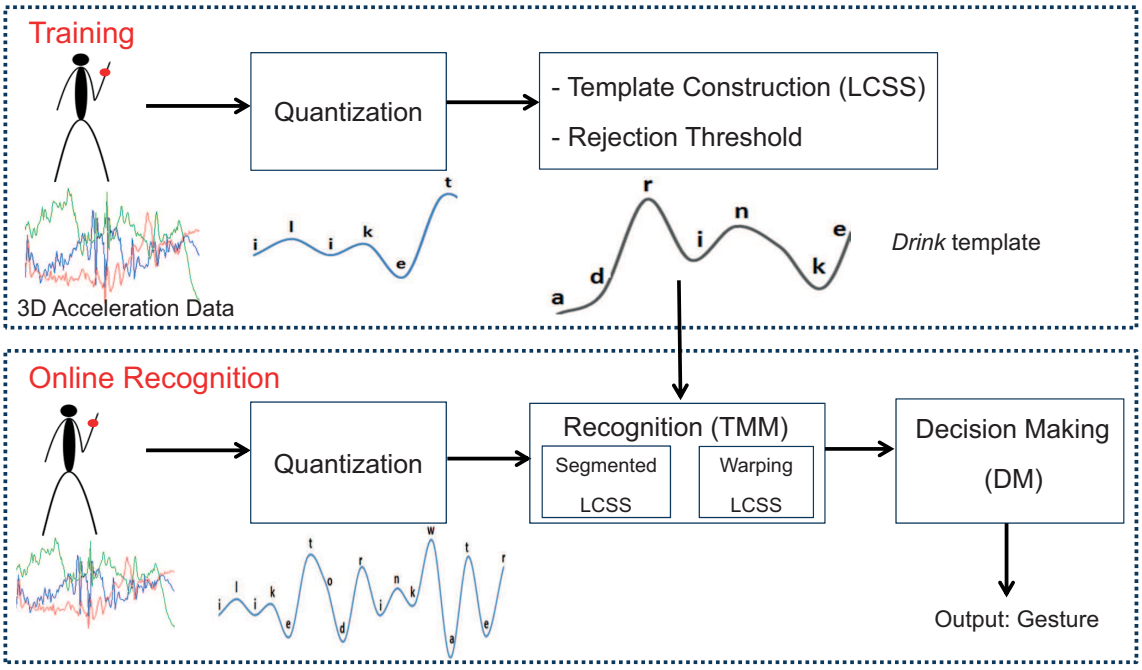


Figure 4: Data processing flow of the proposed LCSS-based template matching methods for gesture recognition.

4.2 Training Phase: Quantization Step

Let n denote the number of signal channels provided by the body-worn sensors (e.g., $n = 3$ for one triaxial accelerometer). Let N be the number of available samples. Let x_i be the time series corresponding to the i-th signal channel, with $1 \leq i \leq n$ and $x_i(t)$ be the value of the time series x_i at time t, with $1 \leq t \leq N$. Let the n-dimensional vector $\mathbf{x}(t) = [x_1(t) \dots x_n(t)]$ denote one sample from all channels at time t.

The quantization step converts the vectors $\mathbf{x}(t)$ into a sequence of symbols (string) $\mathbf{s}(t)$. This is done by performing k-means clustering on the set of n-dimensional vectors $\mathbf{x}(t)$, $\forall t, 1 \leq t \leq N$. The choice of k is performed through cross-validation or empirically. For the gesture data sets used in this paper, $k = 20$ provided a good tradeoff between complexity (k-means' complexity scales linearly with k) and performance. The output of k-means is a set of k n-dimensional cluster centers, $\boldsymbol{\zeta_0} \dots \boldsymbol{\zeta_{k-1}}$, to which k symbols $\alpha_0 \dots \alpha_{k-1}$ are assigned. The quantization procedure then operates on each sample $\mathbf{x}(t)$ to obtain the symbols $s(t)$ as follows:

$$s(t) = \alpha_i | i = \underset{i}{\operatorname{argmin}} ||\mathbf{x}(t) - \boldsymbol{\zeta_i}||_2 \quad .$$

Let us denote with $d(\alpha_l, \alpha_m)$ the distance between two symbols, given by the correspondent distance between their assigned cluster centers, normalized to fall in the interval $[0, 1]$.

$$d(\alpha_i, \alpha_j) = \frac{||\zeta_\mathbf{i} - \zeta_\mathbf{j}||_2}{max_{i,j}||\zeta_\mathbf{i} - \zeta_\mathbf{j}||_2} \quad . \tag{2}$$

4.3 Training Phase: Template Construction

For each labeled gesture in the training data set, a corresponding string is derived used the quantization described in Section 4.2. Denote with $s_i^{(c)}$ the i-th string belonging to class c. The template $\bar{s}^{(c)}$ that represents a gesture class c is then chosen as the string that has the highest average LCSS to all other strings of the same class.

$$\bar{s}^{(c)} = \underset{s_i^{(c)}}{\text{argmax}} \sum_{j \neq i} L_{(s_i^{(c)}, s_j^{(c)})} \quad .$$

4.4 Training Phase: Calculation of Rejection Thresholds

In order to be able to reject signals not belonging to a gesture class upon deployment, a threshold needs to be calculated in the training phase. We define one rejection threshold ϵ_c for each class c. Let $\mu^{(c)}$ and and $\sigma^{(c)}$ be the mean and the standard deviation, respectively, of LCSS values between the template of a class c and any string belonging to the same class. We calculate the rejection threshold to be below $\mu^{(c)}$ by some standard deviations.

$$\epsilon_c = \mu^{(c)} - h * \sigma(c),$$

with h = 0,1,2,...

The rationale is that the good instances belonging to a class should have the similarity with the template around the mean value. ϵ_c is also chosen to be robust with the existence of noisy training instances in gesture class. In our experiments, $h = 1$ provided a good performance.

4.5 Recognition Phase: Quantization Step

In the online recognition, streaming data from a body-worn sensor are quantized to the k-means centroids (i.e., symbols) identified during training, then come to template matching module (TM) which uses either Segmented LCSS or WarpingLCSS to recognize gestures.

4.6 Recognition Phase: SegmentedLCSS

In the SegmentedLCSS approach, the sensor readings $\mathbf{x}(t)$ are first quantized into a string s through the quantization step described in Section 4.5. For each gesture class c, the string s is then segmented into a sliding observation window OW_c. The length of OW_c is chosen as the length of the template $\bar{s}^{(c)}$. A substring of s in OW_c is denoted as s_{OW}^c. Each substring is compared to the template $\bar{s}^{(c)}$ for class c.

The LCSS algorithm is used to calculate $L_{(s_{OW}^c, \bar{s}^{(c)})}$ and the set Ω_s of reference indices of the symbols of s_{OW}^c in the string s matching with symbols in the template. Because the LCSS algorithm can find matching points, the boundaries of the detected gesture can be decided easily. Specifically, if $L_{(s_{OW}^c, \bar{s}^{(c)})} \geq \epsilon_c$, the symbols ranging from $s(\omega_s^{(0)})$ and $s^c(\omega_s^{(L-1)})$ are marked as belonging to class c.

In order to reduce the computational complexity, the next observation window is started at the index $\omega_s^{(0)}$ of the first matching symbol of the previous observation window. In case the set Ω_s is empty, the next observation window is shifted quickly by the window length. Figure 5 illustrates the SegmentedLCSS.

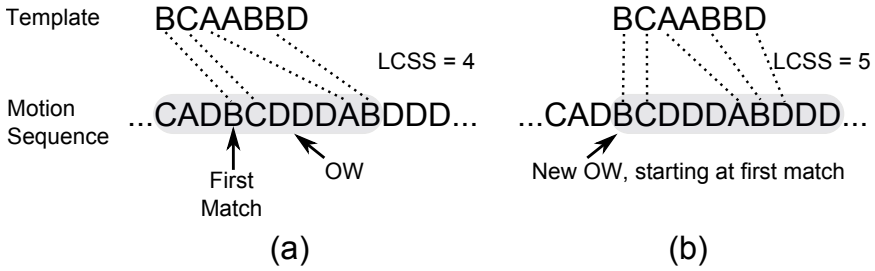


Figure 5: The SegmentedLCSS recognition process. The shaded part represents the observation window OW_c. For each class c, the LCSS is computed between the gesture template $\bar{s}^{(c)}$ and the quantized signal in the window. If the LCSS exceeds the rejection threshold, the samples between the first and the last matching symbols are assigned to class c. The next observation window will start at the first matched point of the previous calculation as illustrated in Figure b.

4.6.1 Computational Complexity of SegmentedLCSS

Let T_c denote the length of a gesture template of class c ($|OW_c| = T_c$). The worst case computational complexity of SegmentedLCSS occurs when new observation windows are shifted by just one sample compared to the preceding ones. In this case, for each class c, the time complexity of SegmentedLCSS is $\mathcal{O}(T_c^2)$. The overall time complexity is then $\mathcal{O}(C * \overline{T}^2)$, where C is the number of classes and $\overline{T}$ stands for the average template length across the classes. The memory usage in SegmentedLCSS is at most $\mathcal{O}(T^2)$, where T is the length of the longest template.

4.7 Recognition Phase: WarpingLCSS

In the SegmentedLCSS, the LCSS must be recomputed every time the observation window shifts, in order to find the beginning and end of each gesture. WarpingLCSS is our variant of LCSS that can find the gesture boundaries without the need of sliding windows, thereby reducing the computational complexity.

In WarpingLCSS, after each new sample of $\mathbf{x}(t)$ is available, the string s is updated by appending the symbol obtained through the quantization of the sample and the LCSS value is recomputed accordingly, relying on the previous values.

Given the gesture template for class c, $\bar{s}^{(c)}$, the WarpingLCSS score $W_{(\bar{s}^{(c)},s)}(i,j)$ between the first i symbols of the template $\bar{s}^{(c)}$ and the first j symbols of the string s is obtained through a modified version of Equation 1 as follows.

$$W_{(\bar{s}^{(c)},s)}(i,j) = \begin{cases} 0 & \text{, if } i=0 \text{ or } j=0 \\ W_{(\bar{s}^{(c)},s)}(i-1,j-1)+1 & \text{, if } \bar{s}^{(c)}(i)=s(j) \\ \max \begin{cases} W_{(\bar{s}^{(c)},s)}(i-1,j-1) - p*d(\bar{s}^{(c)}(i),s(j)) \\ W_{(\bar{s}^{(c)},s)}(i-1,j) - p*d(\bar{s}^{(c)}(i),\bar{s}^{(c)}(i-1)) \\ W_{(\bar{s}^{(c)},s)}(i,j-1) - p*d(s(j),s(j-1)) \end{cases} & \text{, otherwise,} \end{cases} \tag{3}$$

where p is a penalty parameter of the dissimilarity and $d(\cdot,\cdot)$ is the distance between two symbols as defined in Equation 2. The rationale of the WarpingLCSS is the following: if the WarpingLCSS algorithm encounters the same symbol in a template and in the current string, W is increased by a reward of 1. Otherwise, W is decreased by a penalty which depends on the parameter p and on the distance between the symbols. Furthermore, if the string s is "warped", that is, it contains contiguous repetitions of a symbol due to a slower execution of a gesture, the penalty is counted only once.

The algorithm starts with an empty string s and $W(0,0) = 0$. As new symbols are appended, W is updated according to Equation 3. If a gesture of a class is performed, it symbols matching the corresponding template are found and W grows, until reaching a local maximum and eventually decreasing again, as soon as the gesture is over. A gesture of class c is recognized for each local maximum of W that also exceeds the rejection threshold ϵ_c. The end point of the gesture is set to the local maximum itself. The start point is found by tracing back the matching path. The LCSS between the template and the matched gesture is accumulated during the trace-back process if necessary (i.e., when a gesture is spotted as belonging to multiple classes) to make a decision (discussed in next section).

When gestures differ from those encoded by the stored templates, W drops significantly due to the penalty terms. The value of the penalty parameter p depends on the application and can be chosen by cross-validation to maximize the recognition accuracy.

Figure 6 illustrates an example of behavior of W. Figure 7 shows a close-up of W where a gesture was matched to a template. It also shows how the WarpingLCSS detects the temporal boundaries of matched gestures.

4.7.1 Computational Complexity of WarpingLCSS

WarpingLCSS only needs to update the value of W for each new sample. Thus, the time complexity of WarpingLCSS is $\mathcal{O}(T)$. WarpingLCSS has a linear complexity in T compared to SegmentedLCSS, whose complexity grows quadratically in T. The WarpingLCSS maintains at most $\mathcal{O}(T^2)$ memory for the need to trace back the starting boundary of detected gestures.

4.8 Decision Making and Solving Conflicts

The incoming streaming string is concurrently "compared" with templates of all concerned gesture classes in TM module. If a gesture is spotted as belonging to multiple classes (i.e., boundaries of spotted instances are overlapping), the decision making module (DM) will

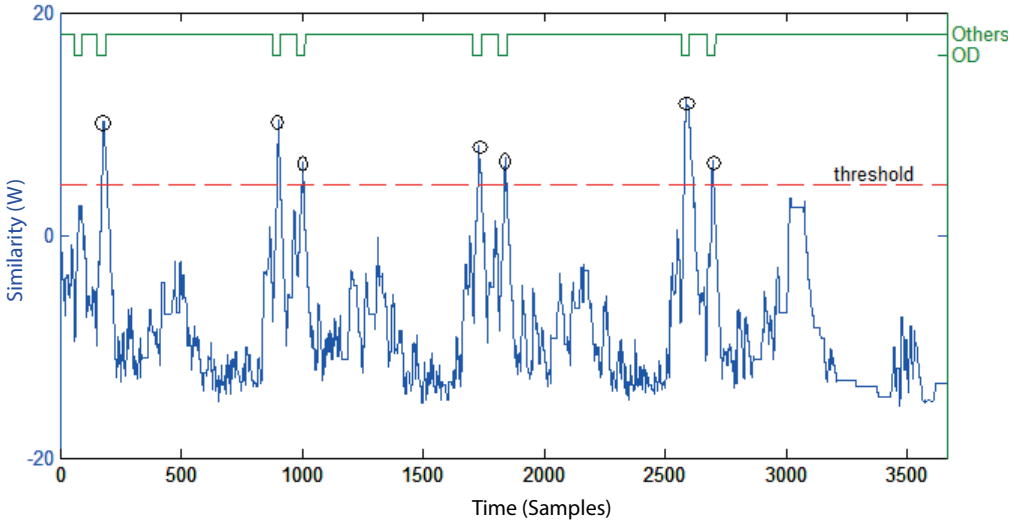


Figure 6: WarpingLCSS between a template of the gesture "open door" (OD) and a streaming string s, p = 3. The value W is updated for each new sample. The line on the top shows the ground truth. The small circles show gesture detection at spotting time.

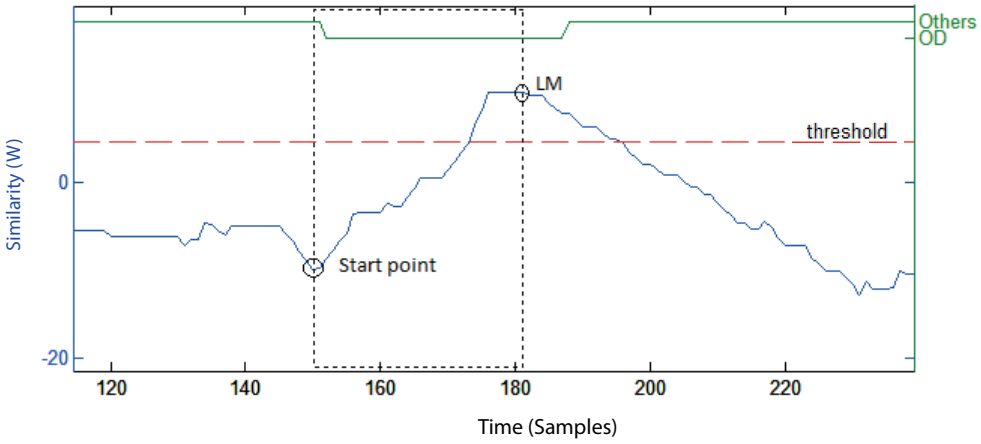


Figure 7: Close-up of the first detected "open door" gesture (OD) in the string s (see Figure 6). The local maximum (LM) marks the end of the gesture, while the start is traced back through the matching symbols.

resolve conflicts (as discussed below) by deciding which class is the best match. If a gesture is classified into only one gesture class, the DM will output the class. Otherwise, if no gesture class is spotted, the DM will output *null*.

Resolving spotting conflicts: We define the normalized similarity between two strings A and

B as NormSim(A,B) = $LCSS(A,B)/max(\|A\|,\|B\|)$, with $\|A\|$ and $\|B\|$ are the lengths of the strings A and B, respectively. The NormSim between the template and the matched gesture is output to the decision making module (DM). The class with highest NormSim is chosen as the best match. This process is the same for both SegmentedLCSS and WarpingLCSS.

5. Experiments

To analyze the effect of annotation noise in terms of performance of gesture recognition algorithms, we compare our SegmentedLCSS and WarpingLCSS TMMs against state-of-the-art recognition methods to assess their robustness. We first present three gesture data sets used to evaluate the recognition systems. We then describe how synthetic crowdsourced annotations are obtained. Finally, we discuss baseline methods and evaluation metrics.

5.1 Description of Data Sets

We used three data sets including various gestures which have been labeled manually by experts. The experts' annotation is the ground truth of the data sets. The data sets used also include *null* class, data which do not correspond to any of the gestures of interest. The list of gestures of these data sets are shown in Table 1. In each data set, we use a 3D accelerometer at a subject' dominant (right) lower arm for the evaluations (30Hz sampling rate). Following, we describe briefly each data set[3].

HCI Gestures				
Circle	Triangle	Square	Infinity	Slider
Their Speculars				Null

Opportunity Gestures		
Null	clean Table	open Drawer 1-2-3
close Drawer 1-2-3	open Door 1-2	close Door 1-2
open Fridge	close Fridge	open Dishwasher
close Dishwasher	drink Cup	toggle Switch

Skoda Gestures			
write on notepad	check gaps on the front door	open hood	close hood
open left front door	close left front door	close both left door	check trunk gaps
check steering wheel	open and close trunk	Null	

Table 1: Gestures in Opportunity, Skoda, and HCI data sets.

5.1.1 Skoda

The Skoda data set (Zappi et al., 2008) contains 10 manipulative gestures performed in a car maintenance scenario by one subject. The *null* class takes 23%. Each gesture class has about 70 instances. This data set is characterized as low variant in execution because the subject performed carefully each manipulative gesture in the same manner.

3. Skoda and Opportunity data sets can be downloaded from `http://www.wearable.ethz.ch/resources/Dataset`.

5.1.2 HCI

The HCI data set (Banos et al., 2012) contains 10 gestures executed by a single person. The gestures are geometric shapes executed with the arm in the vertical plane. This data set has a low variability in the execution of gestures and well-defined labeling. The *null* class takes 57% and each gesture class has about 50 instances.

5.1.3 Opportunity

The Opportunity data set (Roggen et al., 2010) is a rich multi-modal data set collected in a naturalistic environment akin to an apartment, where users execute 16 daily gestures. The data set is characterized by a predominance of *null* class (37%) and a large variability in the execution of the daily activities. Each gesture class has 20 instances excepts "Drink Cup" and "Toggle Switch" each having 40 instances. Note that in Opportunity data set, there are three drawers at different heights which makes the recognition more challenging.

5.2 Experiments on Synthesized Crowdsourced Annotation

To analyze how much noise in annotation the gesture recognition methods can tolerate, we conduct experiments with synthesized annotations. We modify clean annotations from the three data sets described above by emulating *label noise* and *boundary jitter* as discussed in the taxonomy in Section 3.1. In order to evaluate the effect of the different types of noise, we run simulations for each type of noise separately.

5.2.1 Label Noise Simulation

In the *label noise* simulation, we assume the label boundaries are perfect. Let α be the label noise percentage in each class. This means that α percent of the instances are selected and their labels are randomly flipped to other classes (including *null class*). Consequently, each gesture class will have $(1 - \alpha)$ percent of clean instances.

5.2.2 Boundary Jitter Simulation

We run different simulations for different error types in boundary jitter. We assume that all gesture instances get affected from boundary jitter. Let β be the *jitter level* defined in Section 3.2. In the *extend* simulation, each gesture instance will have an *extend level* of β, with boundaries extended at both ends equally ($\beta/2$ per side). Similarly, in the *shrink* simulation, each gesture instance will be shrunk at both ends equally by $\beta/2$. In the *shift left* and *shift right* simulations, each gesture instance is shifted to the left or to the right respectively by β compared to the correct starting point.

We assume that all gesture instances have the same jitter level β. This assumption is not realistic however it can show how much jitter level in the training data set the spotting methods can tolerate given the same style of annotation (for example, a labeler always extends all his annotation 20% level). For a more realistic scenario where jitter levels vary from one instance to another instance, the experiment on the real crowdsourced annotation is presented in Section 6.2.

5.3 Evaluation with Baseline Methods

To investigate the effect of noisy crowdsourced data sets on gesture recognition, we compare the performance of recognition methods trained with ground truth annotations against those trained with crowdsourced annotations. With crowdsourcing-based experiments, the recognition system is trained on crowdsourced annotations and tested on clean data (i.e., annotated by experts). For each data set, we perform a 5-fold cross-validation.

We compare our proposed LCSS-based TMMs with three baselines approaches: the Segmented DTW (Ko et al., 2005; Hartmann and Link, 2010), Nonsegmented DTW (Stiefmeier et al., 2008) and support vector machines (SVM). For all TMM methods, we use the same strategy to select templates, i.e., the maximum similarity average for our LCSS-based methods and the minimum distance average for DTW-based ones. They all have the same quantization preprocessing step as presented in Section 4.2. The rejection thresholds are selected as discussed in Section 4.4. For SegmentedLCSS and Segmented DTW, the window length is chosen as the template length.

For SVM, the signals are passed through a sliding window, with 50% overlap. For each window, mean and variance of the signals are calculated and the obtained feature vectors are fed into a SVM classifier. We use RBF kernels and the two RBF parameters are selected by using cross-validation. In this work, we use the LIBSVM library (Chang and Lin, 2011) for training SVM.

5.3.1 Complexity of Baseline Methods

Segmented DTW belongs, like Segmented LCSS, to the category of sliding window based template matching algorithms. Therefore, roughly, they have the same computational cost. However, unlike SegmentedLCSS, in SegmentedDTW the boundaries of the gestures must be swept exhaustively in the observation window and DTW must be recomputed for each choice to find the best match (Ko et al., 2005; Hartmann and Link, 2010). Therefore, when one new sample arrives, the complexity of the SegmentedDTW is $\mathcal{O}(T^3)$ in the worst case. Meanwhile, in SegmentedLCSS the boundary of gesture inside the window can be found easily via matching points and the observation window is shifted to the first matched point in the previous recognition process instead of being shifted forward by only one sample. Thus, SegmentedLCSS has one order of magnitude lower than SegmentedDTW.

Nonsegmented DTW and WarpingLCSS determine gesture occurrences without segmenting the stream. Therefore, they achieve the same computational cost and they are faster than SegmentedLCSS by one order of magnitude.

In the recognition phase, the running time of SVM grows linearly with the length of the window. Hence, SVM has roughly the same computation cost as WarpingLCSS in the recognition phase.

5.4 Evaluation Metrics

The distribution of the gesture classes may be highly unbalanced in real-life data sets. Especially, in our data sets, *null class* is predominant. Therefore, we assess the performance of gesture recognition with the weighted average F1 score. The weighted average F1 score is the sum of the F1 scores of all classes, each weighted according to the proportion of samples

of that particular class. Specifically,

$$F1score = \sum_c 2 * w_c \frac{precision_c * recall_c}{precision_c + recall_c},$$

where c is the class index and w_c is the proportion of samples of class c; $precision_c$ is the proportion of samples of class c predicted correctly over the total samples predicted as class c; $recall_c$ is the proportion of samples of class c predicted correctly over the total samples of class c.

We present two ways of computing the F1 score, either including (F1-Null) or excluding the *null class* (F1-NoNull). F1-NoNull does not consider the *null class*, but still takes into account false predictions of gesture samples or instances misclassified as *null class*. The recognition system that has high values of both F1-Null and F1-NoNull predicts well both gesture classes and *null* class.

6. Results and Discussion

In this section we present and discuss the results of the experiments conducted with synthesized and real crowdsourced annotations.

6.1 Results on Synthesized Crowdsourced Annotations

We first present the results with synthesized crowdsourced annotations, sweeping the noise levels as described in Section 5. The results show that F1-Null and F1-NoNull have a similar trend of performance as the noise levels increase, therefore we report F1-Null score only.

6.1.1 Label Noise Simulation

Figure 8 shows the results of label noise simulations on the three data sets. WarpingLCSS and SegmentedLCSS are more robust against label noise compared to SVM and DTW-based methods. The performance of LCSS-based methods is stable until a label noise percentage (α) in each class exceeding 70% in Opportunity and HCI data sets and 50% in the Skoda data set. On average, WarpingLCSS outperforms SVM by 22% F1-Null and outperforms DTW-based methods by 36% F1-Null in presence of 60% mislabeled instances. SegmentedLCSS yields similar performance as WarpingLCSS.

SVM performs worse than our LCSS-based methods when α increases. As more label substitutions are added to each class, SVM gets more confused and its performance decreases quickly. The degradation of SVM in performance is expected, since each instance contributes equally to the model building. Hence, wrongly labeled instances can induce the model to choose incorrect support vectors, which severely degrades the performance. Moreover, since the SVM method models *null* class explicitly, it is very sensitive to *delete* noise. Meanwhile, TMMs examine patterns of gesture classes and ignore *null* class in the training phase, thus, TMMs are not influenced with the *delete* noise at all.

The reason why LCSS-based TMMs outperform the ones based on DTW lies in the distance metrics used when selecting the template for each class. Each template is chosen as the one with the highest average similarity to the other instances belonging to the same class. This translates into choosing respectively highest average LCSS and lowest average

DTW distance. While LCSS values between a template and an instance of the same class are bounded between 0 and the length of the template, DTW can grow indefinitely. For this reason, when calculating average DTW distances, mislabeled instances bias the average towards high values, regardless whether correctly labeled instances have a low DTW distance. Consequently, DTW-based TMMs are more likely to pick wrong templates, leading to poor performance when α increases.

The difference between LCSS and DTW in choosing templates can be illustrated with a toy-example. Consider three instances A_1, A_2 and B which are all labeled as belonging to class c_A but let B be mislabeled, that is, B actually belongs to class c_B. To simplify matters, let us assume $LCSS(A_1, A_2) = 1$, $LCSS(A_1, B) = 0$ and $LCSS(A_2, B) = 0$. Similarly, let us assume $DTW(A_1, A_2) = 0$, $DTW(A_1, B) = \infty$ and $DTW(A_2, B) = \infty$. With LCSS, A_1 would have an average similarity of .5 to A_2 and B; A_2 would have an average similarity of .5 to A_1 and B; B would have an average similarity of 0 to A_1 and A_2. Thus, LCSS would pick either A_1 or A_2 as template for the class c_A: both choices would be reasonable. With DTW, A_1 would have an average distance of ∞ to A_2 and B; A_2 would have an average distance of ∞ to A_1 and B; B would have an average distance of ∞ to A_1 and A_2. In this case, the algorithm would not prefer A_1 or A_2 over B, which can lead to choosing as template the mislabeled instance B to represent class c_A. Of course in practice the values of the DTW distance are not infinity, in fact the degradation of DTW-based approaches is not occurring already for a small amount of label noise.

The illustration explains the capability of our LCSS-based methods to pick a good template among noisy instances for a gesture class as long as the number of good instances in a gesture class is still predominant.

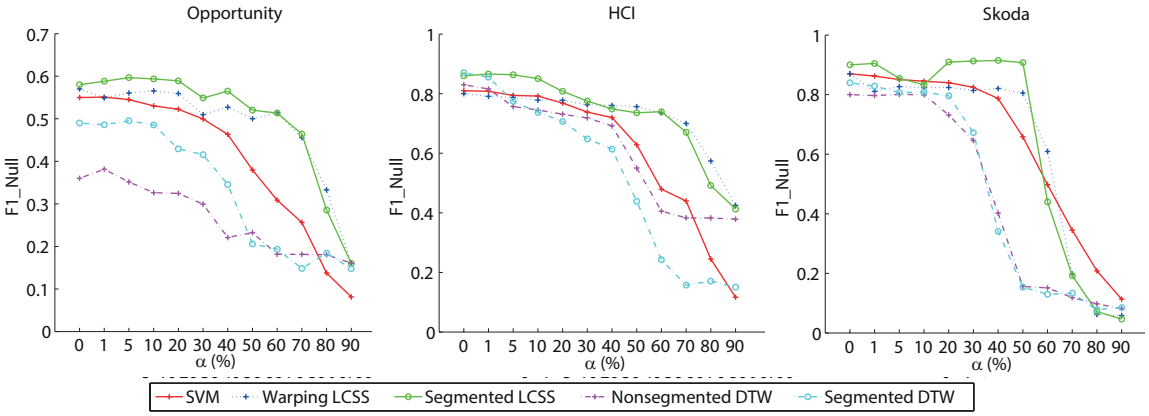


Figure 8: Performance of label noise simulation for the three data sets.

By analyzing the starting points of the curves of Figure 8, obtained with $\alpha = 0$ (no noise), we can conclude that our LCSS-based methods have a similar or better performance compared to the baselines also for the case of clean training data sets.

6.1.2 Extend Jitter Simulation

When temporal boundaries are extended, data belonging to the *null* class (before and after the gesture) are labeled as belonging to the gesture class. This impacts SVM and TMMs

differently. In the case of SVM, the *null class* is modeled explicitly. The noisy feature vectors extracted from extended parts are added into the feature space of each gesture class. Besides that, the data really belonging to the gesture are preserved, thus the models of gesture classes maintain good feature spaces correctly. Therefore, the performance of SVM depends on how much the noisy feature vectors added into the model of each gesture class. Accordingly, it relies on the levels of variability of the signals belonging to the *null* class. If the variability of the signals belong to the *null* class is low, even when the extend level is large, the noisy feature vectors in each gesture class does not grow, leading to the stable of SVM performance. In the converse case, the noisy feature vectors in each gesture class will explode as the extend level increases, causing the decrease in the performance of SVM.

For TMMs instead the *null class* is recognized in the test data by means of the rejection threshold ϵ_c and no template is built for it. Thus, if symbols belonging to the *null class* are present in a test sequence, these will be matched to the symbols present in the extended gesture instances, inducing the TMMs to recognize gestures instead of *null class*.

This is confirmed by an analysis of the results, as shown in Figure 9. TMMs can tolerate up to about 40% *extend level* in the Opportunity and HCI data sets and about 10% *extend level* in the Skoda data set. As the extend level is high, the performance of SVM is stable in HCI and Skoda data sets, but degrades quickly in Opportunity data set. As explained above, the reason of the differences among data sets lie in the different levels of variability of the signals belonging to the *null class* in the different data sets.

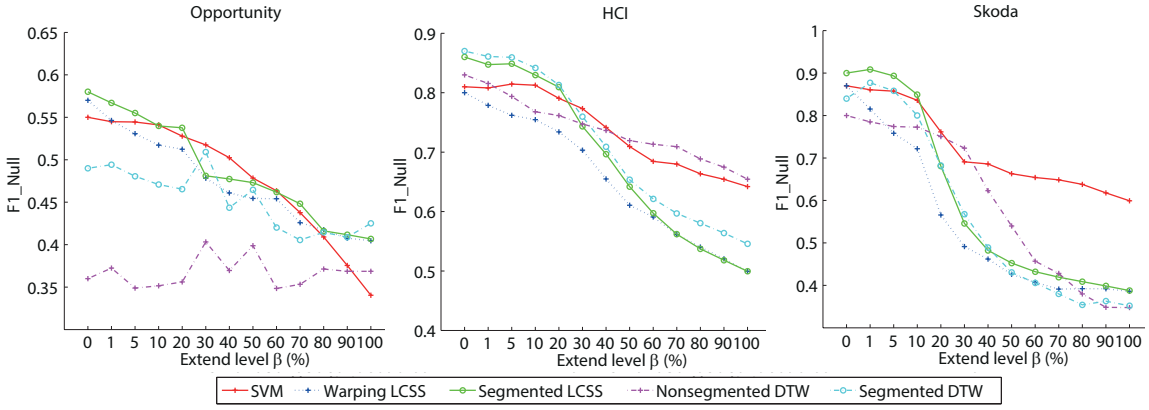


Figure 9: Performance of extend jitter simulation.

6.1.3 Shrink Jitter Simulation

When having a *shrink jitter* noise, the effect is that the methods lose information about the gesture data, since only parts of the gestures are labeled. This has a stronger effect in SVM, since the model is corrupted. For TMMs, subsequences are matched, with the effect that shrunk instances still contain information in form of shorter subsequences that can still be matched to the test data. This is confirmed by the results, shown in Figure 10.

Our proposed LCSS-based methods achieve the best performance in the three data sets. All methods can tolerate about 30% shrink level before a degradation compared to training

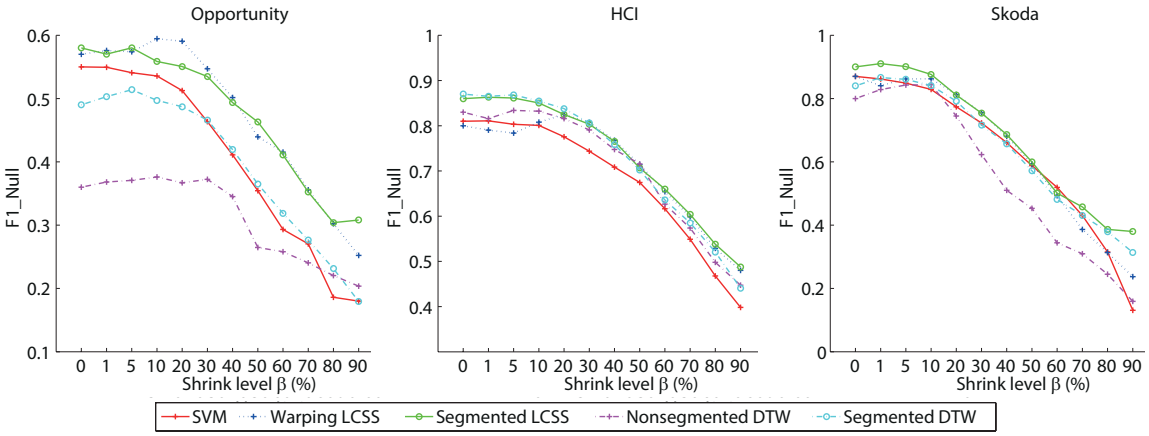


Figure 10: Performance of shrink jitter simulation.

with clean data occurs. The Segmented DTW has a similar results as LCSS-based methods in low-variability data sets (HCI and Skoda). However, Segmented DTW takes a higher computational cost. Moreover, in our experiments, all gesture instances have the same shrink level, i.e., after shrinking, instances of a gesture class are still aligned well and DTW can still achieve a reasonable performance. In a real crowdsourcing annotation setting, different instances may have different shrink levels (see Figure 3b). In that case, DTW will accumulate higher distances due to data misalignment at the beginning and the end of instances (see Nguyen-Dinh et al., 2012 for a more thorough discussion of the weakness of DTW with misalignment in temporal boundaries).

6.1.4 Shift-Left and Shift-Right Jitter Simulation

When annotations are shifted, a mixture of the effects described in Sections 6.1.2 and 6.1.3 are present. Some samples belonging to gestures are lost and some null class samples are labeled as belonging to a gesture. Figure 11 shows the results of *shift-right* jitter simulations (the *shift-left* simulations yield similar results). All methods can sustain about 20% *shift level* before the performance degrades compared to a clean training data set. LCSS-based methods perform often better, or as good as DTW-based methods on the data sets that we examined. TMMs outperform SVM with up to 30% *shift level*.

6.2 Results on Real Crowdsourced Annotation

To further validate the outcome of the previous experiments, we use the real crowdsourced annotations discussed in Section 3.3. The annotations were performed by AMT workers on the Opportunity data set. We use both the annotations obtained in the one-labeler and in the multiple-labeler scenarios. In these annotations, mixtures of all kinds of the errors listed in the taxonomy (Section 3.1) are present and jitter levels are varied from one instance to another instance (see Figure 3).

Figure 12 reports the performance of the different recognition methods on our real crowdsourced annotation. In the clean annotated Opportunity data set, the performance of SVM is slightly lower than that of LCSS-based TMMs (only lower by 3% for F1-Null

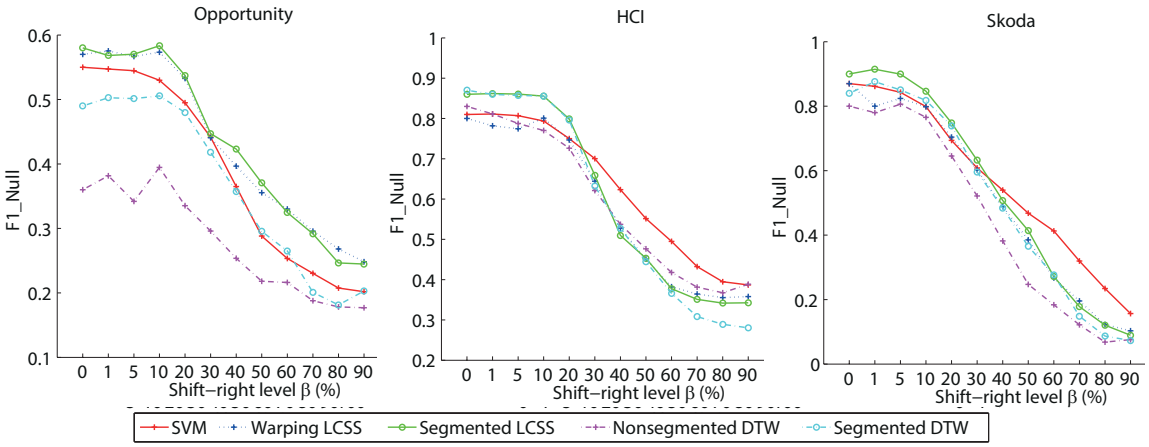


Figure 11: Performance of shift-right jitter simulation.

and by 7% for F1-NoNull). Two DTW approaches underperform the others. The reason is that DTW is very sensitive to high variation in gesture execution (Nguyen-Dinh et al., 2012) and the Opportunity data set contains large variability in the executions of the daily activities.

In the multiple-labeler annotation, labels of 80% of the data samples match the ground truth. Moreover, only 18% of gesture instances are labeled incorrectly and the remainder are correctly labeled with a *jitter level* of at least 2% (see Figure 3). The results show that the performances of all recognition methods are slightly decreased by up to 4% for F1-Null and 6% for F1-NoNull compared to the training with clean training sets. Our LCSS-based TMMs yield the best performance. As stated also in Section 6.1.1, the reason for the robustness of LCSS-based methods lies in their ability to select clean templates also in presence of annotation noise.

In the AMT one-labeler annotation, only 55% samples are annotated correctly. Additionally, about 50% of gesture instances are affected by *label noise*, with many deletions and substitutions. In each gesture class, instances which are labeled correctly are still the majority. The result shows that our LCSS-based TMMs still achieve the best performance. The F1-Null measure decreases by 10% and the F1-NoNull by 16% compared to training with clean annotations.

In the one-labeler annotation, there is a significant difference in performance between TMMs and SVM. The performance of SVM decreases dramatically, down to a F1-NoNull of 5%, which is less than random guessing (which would be around 6% in a 16-class data set like Opportunity). This result confirms what was already measured with the synthetic annotations and discussed in Section 6.1.1.

Additionally, we conduct a 2-sided hypothesis test at the 0.01 level of significance as in Guyon et al. (1998) among the performance of the methods in the three scenarios. The tests showed that the performance differences among the methods are statistically significant except the comparison of the F1-Null between SVM and WarpingLCSS and the comparison of the F1-NoNull between WarpingLCSS and SegmentedLCSS in the multiple-labeler annotation.

The results on the real crowdsourcing annotation confirm that our proposed WarpingLCSS and SegmentedLCSS are robust to noise and yield better performance on crowdsourcing data set. WarpingLCSS is preferable in online recognition, since it has a lower computational cost.

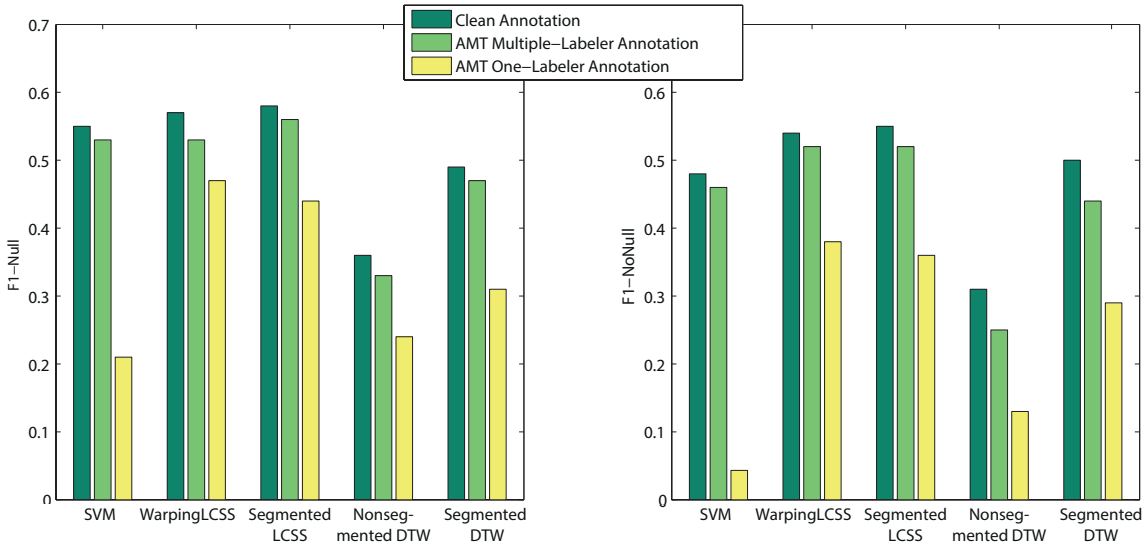


Figure 12: Performance of real crowdsourcing annotation on Opportunity data set.

6.3 A LCSS-based Filtering Component

The results have shown that SVM is very sensitive to the high *label noise* in the training data set. Therefore, a preprocessing component to clean the noisy annotation would be beneficial before using SVM. Given the robustness of our LCSS approaches in selecting templates among noisy instances, as well as in spotting, we further propose a LCSS-based filtering component to filter out noise in crowdsourced annotations before training a SVM. We call this approach FSVM. For each gesture class, the LCSS-based filtering component first computes a LCSS similarity matrix among all pairs of instances in the class. It then keeps only the instances that have an average similarity to other instances of the same class exceeding the average of all the average similarities of all instances in the class. To clean noise inside the *null* instances (e.g., *delete* noise), the filtering component runs the WarpingLCSS on the data annotated as *null* and discards any parts which get classified as any gestures of interest.

For DTW-based TMMs, the performance degrades quickly when the *label noise* percentage in the training data set increases (see Figure 8) because DTW cannot pick a good template among noisy instances. It is interesting to know how templates selected by LCSS perform in the DTW spotting methods. Therefore, we conduct experiments for Segmented DTW and Nonsegmented DTW with templates trained by LCSS. We call these approaches LCSS-SegDTW and LCSS-NonSegDTW respectively. Note that the algorithm running time when the system is deployed remains unchanged: only the training phase is affected.

The performances of FSVM, LCSS-SegDTW and LCSS-NonSegDTW are shown in Figure 13 for the real crowdsourced annotation and in Figure 14 for the synthetic label noise simulation. We present again the performances of the other methods that we discuss above for the sake of comparison.

In the real crowdsourced annotation, the filtering increases the performance of SVM by 20% F1-score and of DTW-based methods by 8% F1-score on average in the one-labeler annotation scenario where high *label noise* exists (see Figure 3). In the clean annotation and multiple-labeler annotation, FSVM performs just slightly worse than SVM (only 2%). This slight decrease can be explained with the fact that the FSVM method decreases the amount training data compared to pure SVM, because the LCSS-based filtering component in the FSVM removes some part of training data, considered noisy. Our proposed LCSS-based methods still outperform FSVM.

The LCSS-NonSegDTW outperforms Nonsegmented DTW in all three scenarios (expert's annotation, AMT multiple-labeler annotation and AMT one-labeler annotation). Similarly, LCSS-SegDTW outperforms SegmentedDTW. The result clarifies that LCSS is capable of picking a better template among noisy instances, compared to DTW. However, LCSS-NonSegDTW and LCSS-SegDTW still underperform compared to our LCSS-based TMMs. The rationale is the same as discussed before. LCSS is more robust to high variation in daily gesture execution, therefore LCSS-based spotting approaches have a better performance than DTW-based ones even with the same templates.

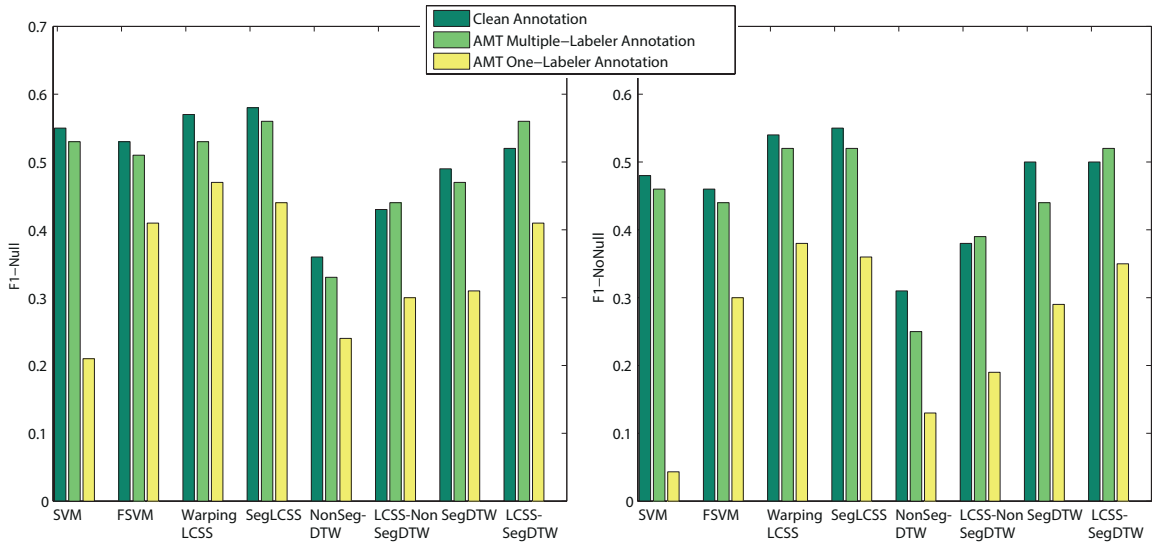


Figure 13: Performance of real crowdsourcing annotation on Opportunity data set for the methods with and without filtering. SegLCSS, NonSegDTW, and SegDTW stand for Segmented LCSS, Nonsegmented DTW and Segmented DTW respectively.

In the synthetic label noise simulation, the FSVM, LCSS-NonSegDTW and LCSS-SegDTW methods outperform SVM, Nonsegmented DTW and Segmented DTW respec-

tively and keep the performance stable much longer when α increases. Our proposed LCSS-based TMMs have similar or better performance than the other methods. Interestingly, with the same templates picked by LCSS, LCSS-SegDTW and LCSS-NonSegDTW have a performance which is similar to our LCSS-based methods in the HCI and Skoda data sets. In the Opportunity data set, the LCSS-NonSegDTW still performs worse than our SegmentedLCSS and WarpingLCSS methods because LCSS is more robust than DTW to high variability in daily gestures (Nguyen-Dinh et al., 2012).

The results show that our LCSS approaches can be used in a preprocessing step for cleaning noisy annotation in the training data for SVM or for selecting templates for DTW-based TMMs.

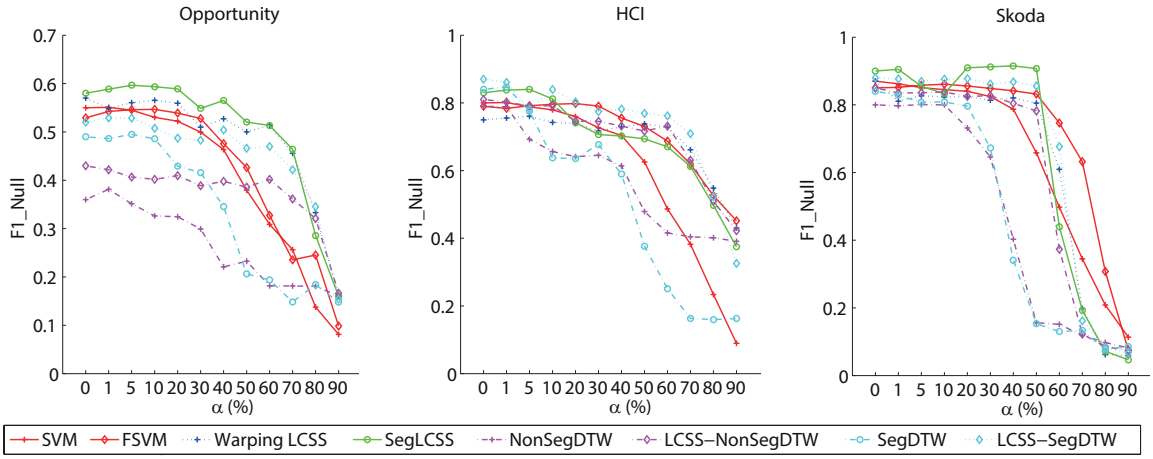


Figure 14: Performance of label noise simulation for the methods with and without filtering.

6.4 Wrapping up

Our LCSS-based TMMs are robust to labeling noise in crowdsourced gesture data sets. Moreover, the LCSS-based TMMs also offer other advantages. (1) They are easy to deploy in online gesture recognition system due to low time complexity. (2) In our systems, signals are converted into symbols, thus SegmentedLCSS lends itself even to embedded implementations. Specifically, string matching in the deployment phase does not involve floating-point operations, thus it can be deployed easily in cheap entry-level microcontroller units. (3) The deployed TMM-based systems are scalable to new gesture classes of interest. After collecting a training data set for a new class, the training phase only works with this class to find a template and the rejection threshold for the class. The template is then integrated directly into the deployed system. Thus, the whole process works smoothly with the new class without interfering with other existing gesture classes.

Our LCSS-based TMMs have been investigated in online gesture recognition with accelerometer data only. Their ability to work with other sensor modalities (e.g., gyroscopes, sound) has been investigated and it has shown promising preliminary results in Nguyen-Dinh et al. (2014).

7. Conclusion and Future Work

In this paper, we investigated the robustness of our proposed LCSS-based TMMs for online gesture recognition on crowdsourced annotated data sets. The results show that SegmentedLCSS and WarpingLCSS are robust to crowdsourced annotation noise and yield better performance than DTW-based methods and SVM. We also introduced a taxonomy of annotation noise in crowdsourcing settings and analyzed the distribution of that noise in real crowdsourced scenarios. Our LCSS-based methods are very robust to label noise because they are capable of selecting a good template among noisy instances for a class. In presence of 60% mislabeled instances, LCSS-based methods outperform SVM by 22% F1-score and outperform DTW-based methods by 36% F1-score on average.

With boundary jitter, the performance of the proposed approaches is comparable to that on clean data sets if annotations can keep most of the information indicating gestures (at most 30%-40% jitter level). In extreme cases when jitter levels go beyond that limit, our LCSS-based TMMS and the other machine learning techniques fail to recognize the complete segment of gestures. This can be the case for example in real-time labeling, where labelers tend to indicate quickly when a gesture occurs with only one time point, without providing the start and end time of the gesture (e.g., the boundary shrinks to a point). Other techniques (e.g., active learning) are necessary to acquire more labels and improve label quality in such cases.

We showed that our LCSS-based methods can be also used as a preprocessing filtering component to clean crowdsourced training data set with severe label noise before feeding the training sets into other learning techniques such as SVM or select templates for DTW. The filtering increases the performance of SVM by 20% F1-score and DTW-based methods by 8% F1-score on average in the noisy real crowdsourced annotations.

In future work, we plan to deploy the system that crowdsources annotated data to a large number of users who record and contribute gestures. Our methods will then be tested on such real large crowdsourced data sets, with the ultimate goal of having a collaborative database of gestures and associated models with direct applications with wearable sensors.

Acknowledgments

The authors would like to thank Dr. Daniel Roggen (University of Sussex) for his useful comments. This work has been supported by the Swiss Hasler Foundation project SmartDAYS.

References

J. Aggarwal and M. Ryoo. Human activity analysis: A review. *ACM Computing Surveys*, 43(3):16:1–16:43, April 2011.

J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff. A unified framework for gesture recognition and spatiotemporal gesture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(9):1685–1699, Sept 2009.

R. Amini and P. Gallinari. Semi-supervised learning with an imperfect supervisor. *Knowledge and Information Systems*, 8:385–413, November 2005.

D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2:343–370, April 1988.

O. Banos, A. Calatroni, M. Damas, H. Pomares, I. Rojas, H. Sagha, J. del R. Millán, G. Tröster, R. Chavarriaga, and D. Roggen. Kinect=imu? learning mimo signal mappings to automatically translate activity recognition systems across sensor modalities. In *Proceedings of the 2012 16th International Symposium on Wearable Computers (ISWC)*, pages 92–99, 2012.

L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. In *Proceedings of the 2nd International Conference on Pervasive Computing*, 2004.

B. Bauer and K. Karl-Friedrich. Towards an automatic sign language recognition system using subunits. In *International Gesture Workshop on Gesture and Sign Languages in Human-Computer Interaction*, pages 64–75. 2002.

M. Berchtold, M. Budde, D. Gordon, H. Schmidtke, and M. Beigl. Actiserv: Activity recognition service for mobile phones. In *Proceedings of the 2010 14th International Symposium on Wearable Computers (ISWC)*, pages 1–8, Oct 2010.

R. Bowden, D. Windridge, T. Kadir, A. Zisserman, and M. Brady. A linguistic feature vector for the visual interpretation of sign language. In *European Conference on Computer Vision*, ECCV '04. 2004.

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

L. Chen, J. Hoey, C. D. Nugent, D. J. Cook, and Z. Yu. Sensor-based activity recognition. In *IEEE Transactions on Systems, Man and Cybernetics*, 2012.

H. Cooper, E.-J. Ong, N. Pugeault, and R. Bowden. Sign language recognition using subunits. *Journal of Machine Learning Research*, 13(1):2205–2231, July 2012.

T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. 2nd edition, 2001. ISBN 0070131511.

A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, 28(1):20–28, 1979.

J. Deng and H. Tsui. An HMM-based approach for gesture segmentation and recognition. In *Proceedings of the International Conference on Pattern Recognition*, ICPR '00, 2000.

A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the world-wide web. *Communications of the ACM*, 54(4):86–96, April 2011.

M. Elmezain, A. Al-Hamadi, and B. Michaelis. Improving hand gesture recognition using 3D combined features. In *Proceedings of the 2nd International Conference on Machine Vision*, ICMV '09, pages 128–132, Dec 2009.

G. Fang, X. Gao, W. Gao, and Y. Chen. A novel approach to automatically extracting basic units from chinese sign language. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 4, pages 454–457, Aug 2004.

J. Froehlich, M. Y. Chen, S. Consolvo, B. Harrison, and J. A. Landay. Myexperience: A system for in situ tracing and capturing of user feedback on mobile phones. In *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services*, MobiSys '07, 2007.

D. Frolova, H. Stern, and S. Berman. Most probable longest common subsequence for recognition of gesture character input. *IEEE Transactions on Cybernetics*, 43(3):871–880, June 2013.

T.-C. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, February 2011.

N. Gayar, F. Schwenker, and G. Palm. A study of the robustness of KNN classifiers trained using soft labels. In *Artificial Neural Networks in Pattern Recognition*, volume 4087. 2006.

I. Guyon, J. Makhoul, R. Schwartz, and V. Vapnik. What size test set gives good error rate estimates? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20 (1):52–64, Jan 1998.

J. Hao and T. Shibata. Digit-writing hand gesture recognition by hand-held camera motion analysis. In *Proceedings of the 3rd International Conference on Signal Processing and Communication Systems*, ICSPCS '09, pages 1–5, Sept 2009.

B. Hartmann and N. Link. Gesture recognition with inertial sensors and optimized DTW prototypes. In *Proceedings of the 2010 IEEE International Conference on Systems Man and Cybernetics (SMC)*, 2010.

Z. He, L. Jin, L. Zhen, and J. Huang. Gesture recognition based on 3D accelerometer for cell phones interaction. In *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pages 217–220, Nov 2008.

J. Howe. The Rise of Crowdsourcing. (accessed July 20, 2010), jun 2006. URL `http://www.wired.com/wired/archive/14.06/crowds.html`.

P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on Amazon Mechanical Turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, pages 64–67, 2010.

H. Junker, O. Amft, P. Lukowicz, and G. Tröster. Gesture spotting with body-worn inertial sensors to detect user activities. *Pattern Recognition*, 41(6), 2008.

C. Keskin, A. Cemgil, and L. Akarun. DTW based clustering to improve hand gesture recognition. In *Proceedings of the 2nd International Conference on Human Behavior Unterstanding*, HBU'11, pages 72–81. 2011.

A. Kittur, E. H. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the Twenty-sixth SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 453–456, 2008.

M. H. Ko, G. West, S. Venkatesh, and M. Kumar. Online context recognition in multisensor systems using dynamic time warping. In *Proceedings of the Intelligent Sensors, Sensor Networks and Information Processing Conference*, 2005.

W. S. Lasecki, Y. C. Song, H. Kautz, and J. P. Bigham. Real-time crowd labeling for deployable activity recognition. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 1203–1212, 2013.

N. D. Lawrence and B. Schölkopf. Estimating a kernel fisher discriminant in the presence of label noise. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 306–313, 2001.

H.-K. Lee and J. H. Kim. An hmm-based threshold model approach for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):961–973, October 1999.

L.-V. Nguyen-Dinh, D. Roggen, A. Calatroni, and G. Tröster. Improving online gesture recognition with template matching methods in accelerometer data. In *Proceedings of the 12th International Conference on Intelligent Systems Design and Applications (ISDA)*, 2012.

L.-V. Nguyen-Dinh, U. Blanke, and G. Tröster. Towards scalable activity recognition: Adapting zero-effort crowdsourced acoustic models. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, MUM '13, 2013a.

L.-V. Nguyen-Dinh, M. Rossi, U. Blanke, and G. Tröster. Combining crowd-generated media and personal data: Semi-supervised learning for context recognition. In *Proceedings of the 1st ACM International Workshop on Personal Data Meets Distributed Multimedia*, PDM '13, 2013b.

L.-V. Nguyen-Dinh, C. Waldburger, D. Roggen, and G. Tröster. Tagging human activities in video by crowdsourcing. In *Proceedings of the ACM International Conference on Multimedia Retrieval*, ICMR '13, 2013c.

L.-V. Nguyen-Dinh, A. Calatroni, and G. Tröster. Towards a unified system for multimodal activity spotting: Challenges and a proposal. In *Proceedings of the ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, UbiComp '14 Adjunct, 2014.

N. Ravi, N. D, P. Mysore, and M. L. Littman. Activity recognition from accelerometer data. In *Proceedings of the Seventeenth Conference on Innovative Applications of Artificial Intelligence(IAAI)*, pages 1541–1546. AAAI Press, 2005.

V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *Journal of Machine Learning Research*, 11, 2010.

D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Forster, G. Troster, and et al. Collecting complex activity data sets in highly rich networked sensor environments. In *Proceedings of the 7th International Conference on Networked Sensing Systems*. IEEE Press, 2010.

M. Rossi, O. Amft, and G. Tröster. Recognizing daily life context using web-collected audio data. In *Proceedings of the 16th IEEE International Symposium on Wearable Computers (ISWC)*, June 2012.

T. Schlömer, B. Poppinga, N. Henze, and S. Boll. Gesture recognition with a Wii controller. In *Proceedings of the 2nd International Conference on Tangible and Embedded Interaction*, 2008.

V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, 2008.

H. Stern, M. Shmueli, and S. Berman. Most discriminating segment - longest common subsequence (MDSLCS) algorithm for dynamic hand gesture classification. *Pattern Recognition Letters*, 34(15):1980–1989, 2013.

T. Stiefmeier, D. Roggen, G. Ogris, P. Lukowicz, and G. Tröster. Wearable activity tracking in car manufacturing. *IEEE Pervasive Computing Magazine*, 7(2), 2008.

M. Stikic, D. Larlus, S. Ebert, and B. Schiele. Weakly supervised recognition of daily life activities with wearable sensors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2521–2537, December 2011.

K. Van Laerhoven, D. Kilian, and B. Schiele. Using rhythm awareness in long-term activity recognition. In *Proceedings of the IEEE International Symposium on Wearable Computers (ISWC)*, October 2008.

C. Vogler and D. N. Metaxas. Toward scalability in ASL recognition: Breaking down signs into phonemes. In *Gesture-Based Communication in Human-Computer Interaction*, Lecture Notes in Computer Science, pages 211–224, 1999.

J. A. Ward, P. Lukowicz, and H. W. Gellersen. Performance metrics for activity recognition. *ACM Transactions on Intelligent Systems and Technology*, 2(1), January 2011.

A. Wilson and A. Bobick. Parametric hidden markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21, 1999.

J. Wu, G. Pan, D. Zhang, G. Qi, and S. Li. Gesture recognition with a 3-D accelerometer. In *Proceedings of the 6th International Conference on Ubiquitous Intelligence and Computing*, UIC '09, pages 25–38, 2009.

H.-S. Yoon, J. Soh, Y. J. Bae, and H. S. Yang. Hand gesture recognition using combined features of location, angle and velocity. *Pattern Recognition*, 34, 2001.

M.-C. Yuen, I. King, and K.-S. Leung. A survey of crowdsourcing systems. In *SocialCom/PASSAT*, pages 766–773, 2011.

P. Zappi, C. Lombriser, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Tröster. Activity recognition from on-body sensors: Accuracy-power trade-off by dynamic sensor selection. In *Proceedings of the 5th European Conference on Wireless Sensor Networks*, EWSN'08, pages 17–33, 2008.

Journal of Machine Learning Research 15 (2014) 3221-3245 Submitted 6/13; Published 10/14

Accelerating t-SNE using Tree-Based Algorithms

Laurens van der Maaten LVDMAATEN@GMAIL.COM
Pattern Recognition and Bioinformatics Group
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Editors: Aaron Courville, Rob Fergus, and Christopher Manning

Abstract

The paper investigates the acceleration of t-SNE—an embedding technique that is commonly used for the visualization of high-dimensional data in scatter plots—using two tree-based algorithms. In particular, the paper develops variants of the Barnes-Hut algorithm and of the dual-tree algorithm that approximate the gradient used for learning t-SNE embeddings in $\mathcal{O}(N \log N)$. Our experiments show that the resulting algorithms substantially accelerate t-SNE, and that they make it possible to learn embeddings of data sets with millions of objects. Somewhat counterintuitively, the Barnes-Hut variant of t-SNE appears to outperform the dual-tree variant.

Keywords: embedding, multidimensional scaling, t-SNE, space-partitioning trees, Barnes-Hut algorithm, dual-tree algorithm.

1. Introduction

Visual exploration is an essential component of data analysis, as it allows for the development of intuitions and hypotheses for the processes that generated the data. Visual analytics provides and develops approaches to obtain such understanding from complex data: it aims to develop methods that allow analysts to examine the processes underlying the data (Keim et al., 2010). Unfortunately, modern visual-analytics approaches are still largely based on traditional visualization techniques such as histograms, scatter plots, and parallel coordinate plots; see, e.g., Heer et al. (2010) for an overview of visualization techniques. The drawback of these visualization techniques is that they only facilitate the visualization of one or a few data variables at a time, which prohibits their use on large, high-dimensional data sets. In order to develop hypotheses about processes that generate a large number of variables, it is therefore necessary to perform an automatic analysis of the data before making visualizations. A popular way to perform such an automatic analysis is by learning a low-dimensional embedding of the data. In a low-dimensional embedding, each (high-dimensional) object is represented by a low-dimensional point in such a way, that nearby points correspond to similar objects and that distant points correspond to dissimilar objects. The low-dimensional embedding can readily be visualized in, e.g., a scatter plot or a parallel coordinate plot, or it can be used as the basis for the construction of more advanced visualizations, such as class-conditional density maps (van Eck and Waltman, 2010) or hierarchical visualizations (Tiňo and Nabney, 2002).

A plethora of embedding techniques have been proposed over the last decade, e.g., by Carreira-Perpiñán (2010); Lawrence (2011); Roweis and Saul (2000); Tenenbaum et al. (2000); Saul et al. (2006); and van der Maaten and Hinton (2008). Reviews of these methods are given by, e.g., van der Maaten et al. (2009) and Burges (2010). Because in high-dimensional spaces, only small pairwise distances are reliable, most of these techniques only try to accurately model such small pairwise distances in the low-dimensional embedding. In particular, a family of techniques that preserves small pairwise distances via *stochastic neighbor embedding* (SNE; Hinton and Roweis, 2003) has recently gained popularity (Carreira-Perpiñán, 2010; van der Maaten and Hinton, 2008; Venna et al., 2010). Stochastic neighbor embedding techniques compute an $N \times N$ similarity matrix in both the original data space and in the low-dimensional embedding space in such a way, that the similarities form a probability distribution over pairs of objects. The distribution over pairs of objects is defined such that pairs of similar objects have a high probability under the distribution, whilst pairs of dissimilar points have a low probability. Specifically, the probabilities are generally given by a normalized Gaussian or Student-t kernel computed from the input data or from the embedding. The low-dimensional embedding is learned by minimizing the Kullback-Leibler divergence between the two probability distributions (computed in the original data space and the embedding space) with respect to the locations of the points in the embedding. Because of the asymmetry of the Kullback-Leibler divergence, stochastic neighbor embedding focuses on accurately modeling small pairwise distances, i.e., on preserving *local* data structure in the low-dimensional embedding.

Because the objective functions of most stochastic neighbor embedding techniques are non-convex,[1] the minimization of the objective is typically performed using first-order or second-order gradient-descent techniques (Carreira-Perpiñán, 2010; Hinton and Roweis, 2003; Vladymyrov and Carreira-Perpiñán, 2012). The gradient of the Kullback-Leibler divergence that is minimized has a very natural interpretation as an N-body system in which all of the N points in the low-dimensional embedding exert forces on each other, and the resultant force on each of the points needs to be computed.

Because the computation of stochastic neighbor embedding gradients involves the evaluation of forces between all $N \times N$ pairs of points, one of the main limitations of stochastic neighbor embedding is that its computational complexity scales quadratically in the number of input objects N. In practice, this limits the applicability of stochastic neighbor embedding to data sets with only a few thousand points. To visualize larger data sets, landmark implementations of stochastic neighbor embedding may be used (van der Maaten and Hinton, 2008), but this is hardly a satisfactory solution because it does not facilitate visualization of all available data. Alternatively, computational problems may be circumvented by learning a parametric function between the input space and the embedding using a type of stochastic gradient descent (van der Maaten, 2009), but such an approach substantially complicates learning and is only applicable when the input data takes the form of high-dimensional data vectors.

In this paper, we explore tree-based approaches for stochastic neighbor embedding that require only $\mathcal{O}(N \log N)$ computation and $\mathcal{O}(N)$ memory. Our approaches compute a sparse

1. We note that it is possible to define a convex variant of traditional stochastic neighbor embedding (Hinton and Roweis, 2003) by performing the minimization with respect to the Gram matrix of the low-dimensional embedding instead of with respect to the low-dimensional embedding itself.

approximation of the similarities between the input objects using vantage-point trees (Yianilos, 1993), and subsequently, they approximate the forces between the points in the embedding with the help of either a Barnes-Hut algorithm (Barnes and Hut, 1986) or a dual-tree algorithm (Gray and Moore, 2001, 2003). The Barnes-Hut and dual-tree algorithms reduce the number of pairwise forces that needs to be computed by exploiting the fact that the forces exerted between two small groups of points are all very similar whenever these two groups are relatively far away. We will study the performance of the tree-based algorithms in the context of the successful t-distributed stochastic neighbor embedding (t-SNE; van der Maaten and Hinton, 2008) algorithm, but similar computational approaches may be used to speed up, e.g., standard stochastic neighbor embedding (Hinton and Roweis, 2003), the neighborhood retrieval visualizer (NeRV; Venna et al., 2010), and elastic embedding (Carreira-Perpiñán, 2010; Vladymyrov and Carreira-Perpiñán, 2014). Source code of our tree-based t-SNE algorithms is publicly available on `http://homepage.tudelft.nl/19j49/tsne`; this software has recently been successfully used to create large-scale embeddings of, among others, mouse brain data (Ji, 2013), metagenomic data (Laczny et al., 2014), and word embeddings (Cho et al., 2014). This paper is an extended version of an earlier conference publication (van der Maaten, 2013) on a Barnes-Hut approximation to t-SNE, which was recently independently investigated by Yang et al. (2013). Compared to these prior papers, this paper: (1) investigates a second tree-based algorithm to speed up t-SNE, *viz.* the dual-tree algorithm, whereas van der Maaten (2013) and Yang et al. (2013) only considered the Barnes-Hut algorithm; (2) contains more detailed explanations of the techniques and experiments; and (3) contains additional experimental results on a number of large data sets.

The outline of the remainder of this paper is as follows. In Section 2, we discuss related work on accelerating algorithms that scale quadratically in the data set size. Section 3 reviews the t-SNE algorithm of van der Maaten and Hinton (2008). In Section 4, we present accelerated variants of t-SNE that are based on the Barnes-Hut and on the dual-tree algorithm. Our experimental results are presented in Section 5. Section 6 concludes the paper and presents directions for future work.

2. Related Work

This work fits in a larger body of prior work that has focused on decreasing the computational complexity of algorithms that scale quadratically in the amount of data when implemented naively, such as nearest neighbor search and Parzen density estimation.

In nearest-neighbor search problems, substantial speed-ups are generally obtained using space-partitioning (metric) trees such as kd-trees (Freidman et al., 1977; Silpa-Anan and Hartley, 2008), b-trees (Bayer and McCreight, 1972), cover trees (Beygelzimer et al., 2006), vantage-point trees (Yianilos, 1993), and trees constructed using hierarchical clustering (Fukunaga and Narendra, 1975; Brin, 1995; Nister and Stewenius, 2006). An approach that automatically selects the best-performing tree-based algorithm for a particular data set was presented by Muja and Lowe (2009). Alternative approaches to speed up nearest-neighbor search use approximate search algorithms based on locality sensitive hashing (Indyk and Motwani, 1998; Weiss et al., 2008; Salakhutdinov and Hinton, 2007). Motivated by their

strong performance reported in earlier work by Liu et al. (2004), we opt to use metric trees to approximate the similarities of the input objects in our algorithms.

Prior work on accelerating N-body computations, e.g., to perform fast Parzen density estimation, is generally based on $\mathcal{O}(N \log N)$ tree-based algorithms such as the Barnes-Hut algorithm (Barnes and Hut, 1986) and the dual-tree algorithm (Gray and Moore, 2001, 2003), or on $\mathcal{O}(N)$ fast multipole methods (Rokhlin, 1985). We explain the Barnes-Hut and dual-tree algorithms in more detail in Section 4. Fast multipole methods perform expansions of the forces that points exert on each other that are specific to the functional form of those forces, and use these expansions to speed up the computations (Rokhlin, 1985). For instance, if the strength of the interactions is governed by a Gaussian function, the interactions may be approximated by a weighted sum of Hermite polynomials: the interaction $I(\cdot, \cdot)$ between objects $\mathbf{x}$ and $\mathbf{y}$ then factorizes as $I(\mathbf{x}, \mathbf{y}) = f(\mathbf{x})g(\mathbf{y})$, which facilitates the computation of all resultant forces in $\mathcal{O}(N)$. For Gaussian forces, the fast multipole approach is generally referred to as the fast Gauss transform (Greengard and Rokhlin, 1987; Yang et al., 2003). The aforementioned algorithms for fast N-body computations are commonly used in astronomy, e.g., for simulating large galaxies (Springel et al., 2001; Croton et al., 2006), and in information visualization, e.g., for constructing force-directed layouts and for graph drawing (Fruchterman and Reingold, 1991; Chalmers, 1996; Quigley and Eades, 2000; Hu, 2005). Lang et al. (2005) presents an experimental comparison of many of the algorithms.

In machine learning, dual-tree algorithms have been used for, among others, density estimation (Gray and Moore, 2001, 2003) and Gaussian process regression (Gray, 2004). Raykar and Duraiswami (2006) have used fast multipole methods to speed up Parzen density estimators. de Freitas et al. (2006) also used fast multipole approaches to speed up the computation of Gaussian N-body interactions, in particular, in order to speed up generalized eigenvalue solvers based on Krylov subspace iteration as well as to speed up active learning (Mahdaviani et al., 2005) and stochastic neighbor embedding. Recently, Vladymyrov and Carreira-Perpiñán (2014) have also explored a fast multipole approach for constructing embeddings, focusing on the elastic-embedding algorithm. Unfortunately, the fast multipole approach cannot be readily applied to t-SNE because, to the best of our knowledge, there exists no appropriate expansion for forces governed by Student-t interactions: using fast multipole methods for t-SNE would thus require further approximations that, for instance, replace the Student-t interactions in the learning gradient by Gaussian interactions.

3. t-Distributed Stochastic Neighbor Embedding

t-Distributed stochastic neighbor embedding (t-SNE) minimizes the divergence between two distributions: a distribution that measures pairwise similarities of the input objects and a distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding. Assume we are given a data set of (high-dimensional) input objects $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ and a function $d(\mathbf{x}_i, \mathbf{x}_j)$ that computes a distance between a pair of objects, e.g., the Euclidean distance $d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|$. Our aim is to learn an s-dimensional embedding in which each object is represented by a point, $\mathcal{E} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$ with $\mathbf{y}_i \in \mathbb{R}^s$ (typical values for s are 2 or 3). To this end, t-SNE defines joint probabilities p_{ij} that measure the pairwise similarity between objects $\mathbf{x}_i$ and $\mathbf{x}_j$ by symmetrizing two

conditional probabilities

$$p_{j|i} = \frac{\exp(-d(\mathbf{x}_i, \mathbf{x}_j)^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-d(\mathbf{x}_i, \mathbf{x}_k)^2/2\sigma_i^2)}, \qquad p_{i|i} = 0,$$
$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}.$$

In the above equation, the bandwidth of the Gaussian kernels, σ_i, is set in such a way that the perplexity of the conditional distribution P_i equals a predefined perplexity u. As a result, the optimal value of σ_i varies per object: in regions of the data space with a higher data density, σ_i tends to be smaller than in regions of the data space with lower density. The optimal value of σ_i for each input object can be found using a simple binary search (Hinton and Roweis, 2003) or using a robust root-finding method (Vladymyrov and Carreira-Perpiñán, 2013).

In the s-dimensional embedding $\mathcal{E}$, the similarities between two points $\mathbf{y}_i$ and $\mathbf{y}_j$ (i.e., the low-dimensional models of $\mathbf{x}_i$ and $\mathbf{x}_j$) are measured using a normalized heavy-tailed kernel. Specifically, the embedding similarity q_{ij} between the two points $\mathbf{y}_i$ and $\mathbf{y}_j$ is computed as a normalized Student-t kernel with a single degree of freedom

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}, \qquad q_{ii} = 0.$$

The heavy tails of the normalized Student-t kernel allow dissimilar input objects $\mathbf{x}_i$ and $\mathbf{x}_j$ to be modeled by low-dimensional counterparts $\mathbf{y}_i$ and $\mathbf{y}_j$ that are too far apart. This is desirable because it creates more space to accurately model the small pairwise distances (i.e., the local data structure) in the low-dimensional embedding.

The locations of the embedding points $\mathbf{y}_i$ are determined by minimizing the Kullback-Leibler divergence between the joint distributions P and Q:

$$C(\mathcal{E}) = KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

Due to the asymmetry of the Kullback-Leibler divergence, the objective function focuses on modeling high values of p_{ij} (similar objects) by high values of q_{ij} (nearby points in the embedding space). The objective function is non-convex in the embedding $\mathcal{E}$. It is typically minimized by descending along the gradient

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) q_{ij} Z (\mathbf{y}_i - \mathbf{y}_j),$$

where we defined the normalization term $Z = \sum_{k \neq l}(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}$.

It is straightforward to see that the evaluation of the joint distributions P and Q is $\mathcal{O}(N^2)$, because both distributions involve a normalization term that sum over all $N(N-1)$ pairs of unique objects. Since t-SNE scales quadratically in the number of objects N, its applicability is limited to data sets with only a few thousand input objects; beyond that, learning becomes too slow to be practical (and the memory requirements become too large).

4. Tree-Based Algorithms for t-SNE

We explore two fast algorithms to approximate the t-SNE gradient $\frac{\partial C}{\partial \mathbf{y}_i}$: (1) an algorithm based on the Barnes-Hut approximation and (2) an algorithm based on the dual-tree approximation. Both variants use the same algorithm to approximate the similarities computed between the input data, *viz.* they use a metric tree to approximate P by a sparse distribution in which only $\mathcal{O}(uN)$ values are non-zero. This approximation of the input similarities is described in detail in Section 4.1. The Barnes-Hut and dual-tree approximations are presented in Section 4.2 and 4.3, respectively.

4.1 Approximating Input Similarities

The input similarities in t-SNE are defined as normalized Gaussian kernel values. As a result, probabilities p_{ij} that correspond to dissimilar input objects i and j are nearly infinitesimal. Therefore, we can develop a sparse approximation for the probabilities p_{ij} without negatively affecting the quality of the final embeddings. In particular, we compute the sparse approximation by finding the $\lfloor 3u \rfloor$ nearest neighbors of each of the N input objects (recall that u is the perplexity of the conditional distributions), and we redefine the pairwise similarities between the input objects, p_{ij}, as

$$p_{j|i} = \begin{cases} \frac{\exp(-d(\mathbf{x}_i, \mathbf{x}_j)^2 / 2\sigma_i^2)}{\sum_{k \in \mathcal{N}_i} \exp(-d(\mathbf{x}_i, \mathbf{x}_k)^2 / 2\sigma_i^2)}, & \text{if } j \in \mathcal{N}_i \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}. \tag{2}$$

Herein, $\mathcal{N}_i$ represents the set of the $\lfloor 3u \rfloor$ nearest neighbors of $\mathbf{x}_i$, and the bandwidth σ_i is set such that the perplexity of the conditional distribution equals a predefined perplexity u via a binary search over σ_i. The nearest neighbor sets $\mathcal{N}_i$ are found in $\mathcal{O}(uN \log N)$ time by building a vantage-point tree on the input data and performing an exact nearest-neighbor search with the help of the resulting tree.

In a vantage-point tree, each node stores an input object and the radius of a (hyper)ball that is centered on this object (Yianilos, 1993). All non-leaf nodes in the tree have two children: objects that are located *inside* the ball are stored under the left child of the node, whereas objects that are located *outside* the ball are stored under the right child. The tree is constructed by presenting the objects one-by-one, traversing the tree based on whether the current object lies inside or outside a ball, and creating a new leaf node in which the object is stored. The radius of the new leaf node is set to the median distance between its object and all other objects that lie inside the ball represented by its parent node. To construct a vantage-point tree, the objects need not necessarily be points in a high-dimensional feature space; the availability of a metric $d(\mathbf{x}_i, \mathbf{x}_j)$ suffices. Therefore, the use of vantage-point trees facilitates the application of our algorithms even on complex data types, provided a metric $d(\mathbf{x}_i, \mathbf{x}_j)$ is available. In all our experiments, the input data comprises high-dimensional vectors, $\mathbf{x}_i \in \mathbb{R}^D$, and we use the metric $d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|$.

A nearest-neighbor search to construct the set $\mathcal{N}_i$ is performed using a depth-first search on the vantage-point tree that computes the distance of the objects stored in the nodes to the target object, whilst maintaining: (1) a list of the current nearest neighbors and (2)

the distance τ to the furthest nearest neighbor in the current neighbor list. The value of τ determines whether or not a node should be explored: if there can still be objects inside the ball whose distance to the target object is smaller than τ, the left node is searched, and if there can still be objects outside the ball whose distance to the target object is smaller than τ, the right node is searched. The order in which children are explored depends on whether or not the target object lies inside or outside the current node ball: the left child is examined first if the object lies inside the ball, because the odds are that the nearest neighbors of the target object are also located inside the ball. Conversely, the right child is examined first whenever the target object lies outside of the ball.

The nearest-neighbor search is performed for all N input objects in $\mathcal{D}$ in order to obtain the nearest-neighbor sets $\mathcal{N}_i$. Afterwards, it is straightforward to compute the input similarities via Equation 1 and 2.

4.2 Barnes-Hut Approximation

To approximate the t-SNE gradient, we start by splitting the gradient into two parts

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4(F_{attr} + F_{rep}) = 4\left(\sum_{j \neq i} p_{ij} q_{ij} Z(\mathbf{y}_i - \mathbf{y}_j) - \sum_{j \neq i} q_{ij}^2 Z(\mathbf{y}_i - \mathbf{y}_j)\right), \tag{3}$$

where F_{attr} denotes the sum of all attractive forces (the left sum), whereas F_{rep} denotes the sum of all repulsive forces (the right sum). Computing the sum of all attractive forces, F_{attr}, is computationally efficient; it can be done by summing over all non-zero elements of the sparse distribution P that was constructed using the procedure described in the previous subsection in $\mathcal{O}(uN)$.[2] However, a naive computation of the sum of all repulsive forces, F_{rep}, is still $\mathcal{O}(N^2)$. We now develop a Barnes-Hut algorithm to approximate F_{rep} efficiently in $\mathcal{O}(N \log N)$.

Consider three points $\mathbf{y}_i$, $\mathbf{y}_j$, and $\mathbf{y}_k$ with $\|\mathbf{y}_i - \mathbf{y}_j\| \approx \|\mathbf{y}_i - \mathbf{y}_k\| \gg \|\mathbf{y}_j - \mathbf{y}_k\|$. In this situation, the contributions of $\mathbf{y}_j$ and $\mathbf{y}_k$ to F_{rep} will be roughly equal. The Barnes-Hut algorithm (Barnes and Hut, 1986) exploits this by (1) constructing a quadtree or octtree on the current embedding,[3] (2) traversing the quadtree using a depth-first search, and (3) at every node in the quadtree, deciding whether the corresponding cell can be used as a "summary" for the contributions to F_{rep} of all points in that cell.

A quadtree is a tree in which each node represents a rectangular *cell* with a particular center, width, and height. Non-leaf nodes have four children that split up the cell into four smaller cells (quadrants) that lie "northwest", "northeast", "southwest", and "southeast" of the center of the parent node; see Figure 1 for an illustration of a quadtree. Leaf nodes represent cells that contain at most one point of the embedding; the root node represents the cell that contains the complete embedding. In each node, we store the center-of-mass of the embedding points that are located inside the corresponding cell, $\mathbf{y}_{cell}$, and the total number of points that lie inside the cell, N_{cell}. A quadtree has $\mathcal{O}(N)$ nodes and can be constructed in $\mathcal{O}(N)$ time by inserting the points one-by-one, splitting a leaf node whenever

2. Note that the term $q_{ij}Z = (1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$ can be computed in $\mathcal{O}(1)$.
3. Throughout the paper, we assume the data is embedded in a two-dimensional space, prompting the use of a quadtree. When embedding the data in three dimensions, an octtree is used instead.

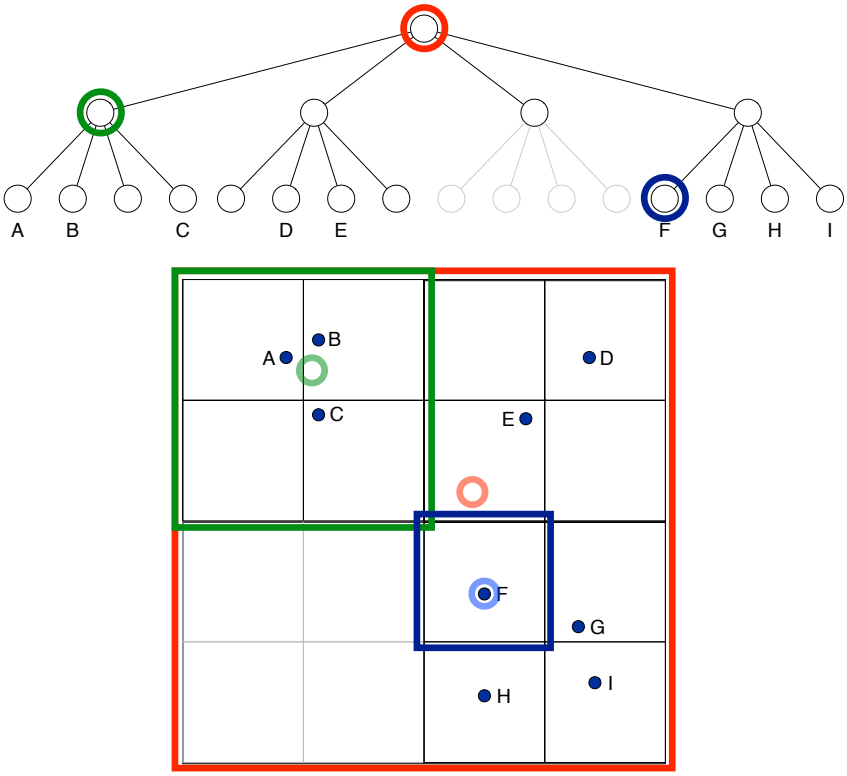


Figure 1: Illustration of a quadtree that was constructed on a data set of nine two-dimensional data points. The top half of the figure illustrates the structure of the tree that represents the partitioning of the two-dimensional space shown in the lower half of the figure. Corresponding colors are used to highlight corresponding elements of the graph and the space partitioning. Nodes in the graph correspond to square cells in the space (deeper nodes correspond to smaller cells). In each node, we store: (1) the number of points that are located in the corresponding cell and (2) the center-of-mass of those points (the centers-of-mass of the three highlighted cells are illustrated by the opaque circles in the space partitioning). The opaque parts of the tree are not actually created, because the corresponding parts of the space do not contain any data points. Leaf nodes represent cells that contain at most one data point. As a result, denser areas of the space correspond to parts of the tree that are deeper.

a second point is inserted in its cell, and updating $\mathbf{y}_{cell}$ and N_{cell} of all visited nodes. Note that the in denser regions of the embedding, the quadtree is deeper than in regions with sparse data.

To approximate the repulsive part of the gradient, F_{rep}, we note that if a cell is sufficiently small and sufficiently far away from point $\mathbf{y}_i$, the contributions $-q_{ij}^2 Z(\mathbf{y}_i - \mathbf{y}_j)$ to F_{rep} will be roughly similar for all points $\mathbf{y}_j$ inside that cell. We can, therefore, approximate these contributions by $-N_{cell} q_{i,cell}^2 Z(\mathbf{y}_i - \mathbf{y}_{cell})$, where N_{cell} represents the number of

points inside the cell, $\mathbf{y}_{cell}$ represents the center-of-mass of the cell, and where we define $q_{i,cell}Z = (1 + \|\mathbf{y}_i - \mathbf{y}_{cell}\|^2)^{-1}$. This approximation is illustrated in Figure 2. We first approximate $F_{rep}Z = -q_{ij}^2 Z^2(\mathbf{y}_i - \mathbf{y}_j)$ by performing a depth-first search on the quadtree, assessing at each node whether or not that node may be used as a "summary" for all the embedding points that are located in the corresponding cell. During this search, we also construct an estimate of $Z = \sum_{i \neq j}(1+\|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$ in the same way. The two approximations thus obtained are then used to compute F_{rep} via $F_{rep} = \frac{F_{rep}Z}{Z}$.

We use the condition proposed by Barnes and Hut (1986) to decide whether a cell may be used as a "summary" for all points in that cell. The condition compares the distance between the cell and the target point with the size of that cell by evaluating

$$\frac{r_{cell}}{\|\mathbf{y}_i - \mathbf{y}_{cell}\|^2} < \theta, \tag{4}$$

where r_{cell} represents the length of the diagonal of the cell under consideration and θ is a threshold that trades off speed and accuracy (higher values of θ lead to faster but coarser approximations). Note that when $\theta = 0$, all pairwise interactions are computed, and the Barnes-Hut approximation reduces to naive computation of the t-SNE gradient. In preliminary experiments, we also explored various other conditions that take into account the rapid decay of the Student-t tail, but we did not find these alternative conditions to lead to a better accuracy-speed trade-off. The problem of more complex conditions is that they require expensive computations at each cell. By contrast, the condition in Equation 4 can be evaluated very rapidly.

4.3 Dual-tree Approximation

Whilst the Barnes-Hut algorithm considers *point-cell* interactions, further speed-ups may be obtained by computing only *cell-cell* interactions. This can be done using a dual-tree algorithm of Gray and Moore (2001). The dual-tree algorithm simultaneously traverses the same quadtree twice in a depth-first manner. For every pair of nodes, the dual-tree algorithm decides whether or not the interaction between the cells of quadtree A and quadtree B can be used as "summary" for the interactions between all points inside these two cells (note that quadtree A and B are identical trees). If the summary condition is passed, the corresponding force is computed. Subsequently, we perform the following additions: (1) we add to all children of the node under consideration in tree A the product of the force and the number of children in the relevant node of tree B; and (2) we add to all children of the node under consideration in tree B the product of the force and the number of children in the node of tree A. Subsequently, all children of the cells in quadtree A and B are pruned. In the dual-tree approximation, we check whether the interaction between a pair of nodes may be used as a "summary" interaction using the condition

$$\frac{\max(r_{cell-A}, r_{cell-B})}{\|\mathbf{y}_{cell-A} - \mathbf{y}_{cell-B}\|^2} < \theta,$$

where $\mathbf{y}_{cell-A}$ and $\mathbf{y}_{cell-B}$ represent the center-of-mass of the two cells from quadtree A and B under consideration and where r_{cell-A} and r_{cell-B} represent the diameter of these two cells. As before, we compute the attractive part of the t-SNE gradient in Equation 3

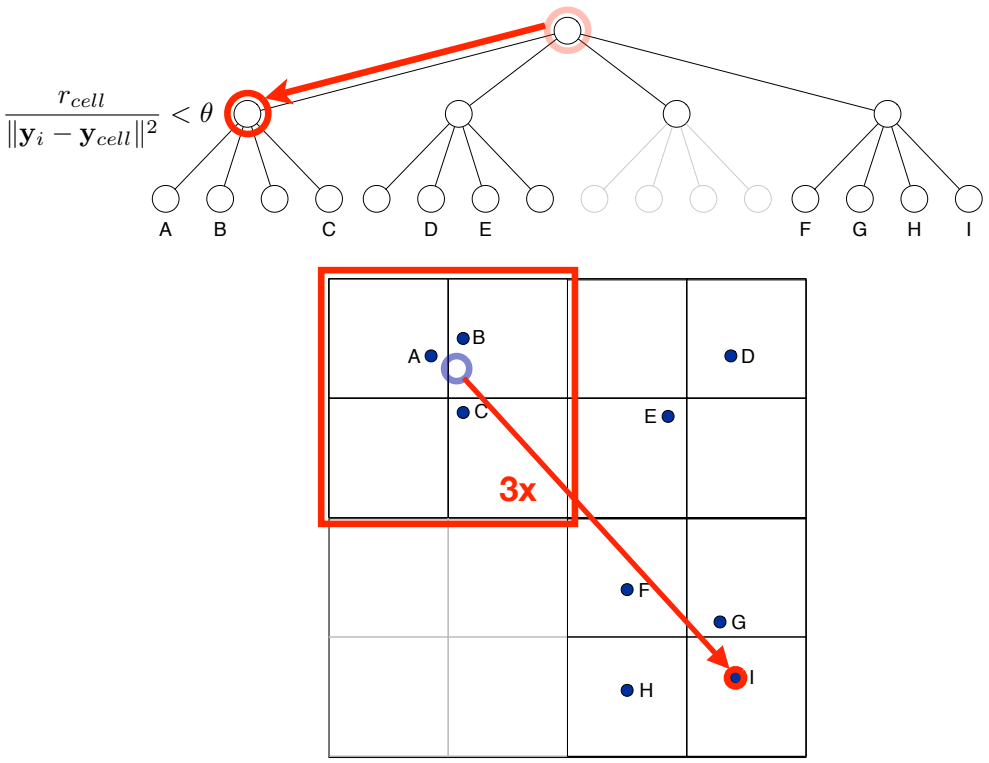


Figure 2: Illustration of the Barnes-Hut approximation. To evaluate the t-SNE gradient for point I, the Barnes-Hut algorithm performs a depth-first search on the embedding quadtree, checking at every node whether or not the node may be used as a "summary". In the illustration, the cell containing points A, B, and C satisfies the summary-condition: the force between the center-of-mass of the three points (which is stored in the quadtree node) and point I is computed, multiplied by the number of points in the cell (i.e., by three), and added to the gradient for point I. All children of the summary node are pruned from the depth-first search.

exactly in dual-tree t-SNE. However, in dual-tree t-SNE, the dual-tree algorithm is used to compute the repulsive part, F_{rep}, of the t-SNE gradient. Note that the optimal value for θ generally differs between Barnes-Hut and dual-tree algorithms, because both algorithms summarize interactions differently.

Whilst the dual-tree algorithm may lead to significant reductions in the number of pairwise forces that needs to be computed compared to the Barnes-Hut algorithm, the computational advantages of the dual-tree algorithm are smaller than one might initially expect when the dual-tree algorithm is used to approximate the t-SNE gradient. Specifically, the problem is that after computing an interaction between two cells, one still needs to determine to which set of points the interaction applies. That is, we need to perform an additional search to determine which points are located in the cell corresponding to the nodes under consideration (in both tree A and B), because the force needs to be added to all

those points (after multiplication with the appropriate number of children). Alternatively, we could construct and store a list of all children for each node during tree construction, but this is computationally equally costly and requires substantial additional memory.[4]

5. Experiments

We performed experiments on five large data sets to evaluate the performance of the Barnes-Hut and dual-tree variants of t-SNE. An implementation of the two algorithms (as well as an implementation of the original t-SNE algorithm) is available from `http://homepage.tudelft.nl/19j49/tsne`. We describe the data sets we used in our experiments in Section 5.1. The setup of our experiments is presented in Section 5.2, and the results of our experiments are presented in Section 5.3.

5.1 Data Sets

We performed experiments on five data sets: (1) the MNIST data set, (2) the CIFAR-10 data set, (3) the NORB data set, (4) the street view house numbers data set, and (5) the TIMIT data set. We briefly describe each of the five data sets as well as the preprocessing we applied on the data below.

MNIST. The MNIST data set contains $N = 70,000$ gray scale handwritten digit images of size $D = 28 \times 28 = 784$ pixels (real-valued between 0 and 1), each of which corresponds to one of ten classes. We directly use the pixel values as input into our embedding algorithms without any further preprocessing.

CIFAR-10. The CIFAR-10 data set (Krizhevsky, 2009) is an annotated subset of the 80 million tiny images data set of Torralba et al. (2008) that contains $N = 70,000$ RGB images of size 32×32 pixels. Each image corresponds to one of ten classes. To extract features from the images, we trained a convolutional network with three convolutional layers on the training images using Caffe (Jia, 2013). We used a network with the following structure: (1) two convolutional layers that contain 32 filters of size 5×5, compute rectified linear unit (ReLU) activations, perform max-pooling over 3×3 patches, and perform local response normalization over 3×3 patches; (2) one convolutional layer with 64 filters of size 5×5, ReLU activations, and average pooling over 3×3 patches; and (3) a final fully connected layer followed by a softmax activation function. The weights of the network were randomly initialized by sampling from a Gaussian distribution with a small variance; all biases were initialized to zero. The network was trained to minimize cross-entropy loss with hundred full sweeps through the data using mini-batches of size 100, a slowly decaying learning rate, and a momentum term of 0.9. The network was regularized using standard (L2) weight decay, using $\lambda = 0.004$. The resulting network obtained a training error of 0.1087 and a test error of 0.1870 on the CIFAR-10 data set, which is on par with the performance of convolutional networks (without data augmentation) on this data set reported in prior studies (Krizhevsky, 2009; Hinton et al., 2012). We used the activations in the last convolutional layer (after the average pooling) as $D = 1,024$-dimensional features for the images. Please note that supervised information was used to obtain these features.

4. Note that the problem sketched does not come into play when approximating the value of the t-SNE cost function, as in that computation, the interaction sums are summed over all N points anyway. Therefore, evaluation of the t-SNE cost function is indeed much faster via a dual-tree algorithm.

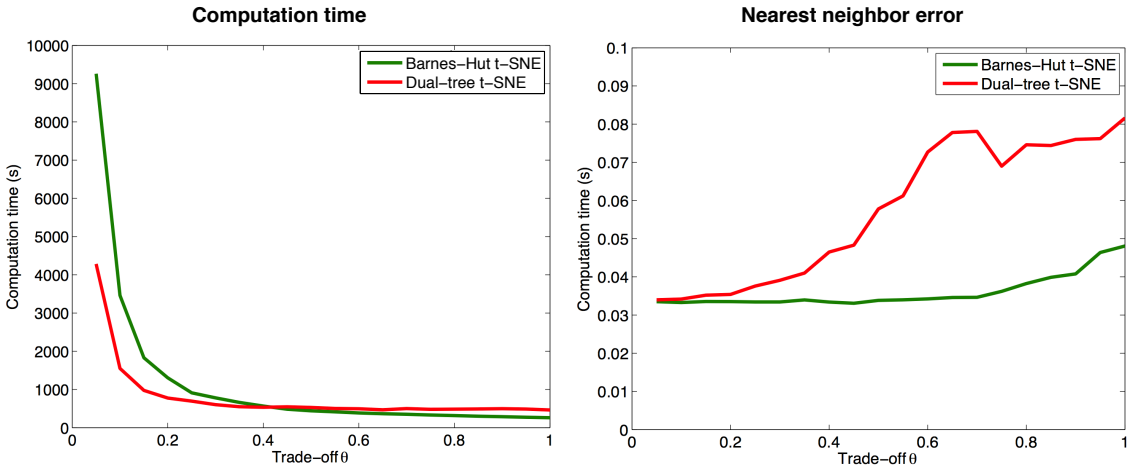


Figure 3: Computation time (in seconds) required to embed 70,000 MNIST digits using two accelerated variants of t-SNE (left) and the 1-nearest neighbor errors of the corresponding embeddings (right) as a function of the trade-off parameter θ. The green lines represent the performance of the Barnes-Hut approximation, whereas the red lines represent the performance of the dual-tree approximation. Note that the special case $\theta = 0$ corresponds to standard t-SNE.

NORB. The (small) NORB data set (LeCun et al., 2004) contains gray scale images of toys from five different classes, rendered on a uniform background under 6 lighting conditions, 9 elevations (30 to 70 degrees every 5 degrees), and 18 azimuths (0 to 340 every 20 degrees). All $N = 48,600$ images contain $96 \times 96 = 9,216$ pixels. We preprocess the images using a simple high-pass filter (specifically, a Laplacian-of-Gaussian filter with $\sigma^2 = 1$ pixels) in order to remove low-frequency information such as the intensity value of the image background. This leads to feature representations of dimensionality $D = 9,216$, which were used as input into the embedding algorithms.

Street View House Numbers. The street view house numbers (SVHN) data set contains $N = 630,420$ labeled color images of house numbers from Google Street View (Netzer et al., 2011). The images are cropped to a size of 32×32 pixels. To extract features from these images, we trained a convolutional network with the following architecture: (1) a convolutional layer with 32 filters of size 5×5, max-pooling over 3×3-pixel regions, and ReLU activations; (2) a convolutional layer with 32 filters of size 5×5, ReLU activations, and average-pooling over 3×3-pixel regions; (3) a convolutional layer with 64 filters of size 5×5, ReLU activations, and average-pooling over 3×3-pixel regions; and (4) a fully-connected layer with softmax units. We trained the network to minimize the cross-entropy loss using Caffe (Jia, 2013) with one full sweep through the training data using mini-batches of size 100, a fixed learning rate of 0.001, and a momentum term of 0.9. The network was regularized using weight decay with $\lambda = 0.004$. The resulting network has a training error of 5.06% and a test error of 10.28%, which is roughly on par with the performance of vanilla convolutional networks reported by Sermanet et al. (2012). We used the $D = 64$ activations

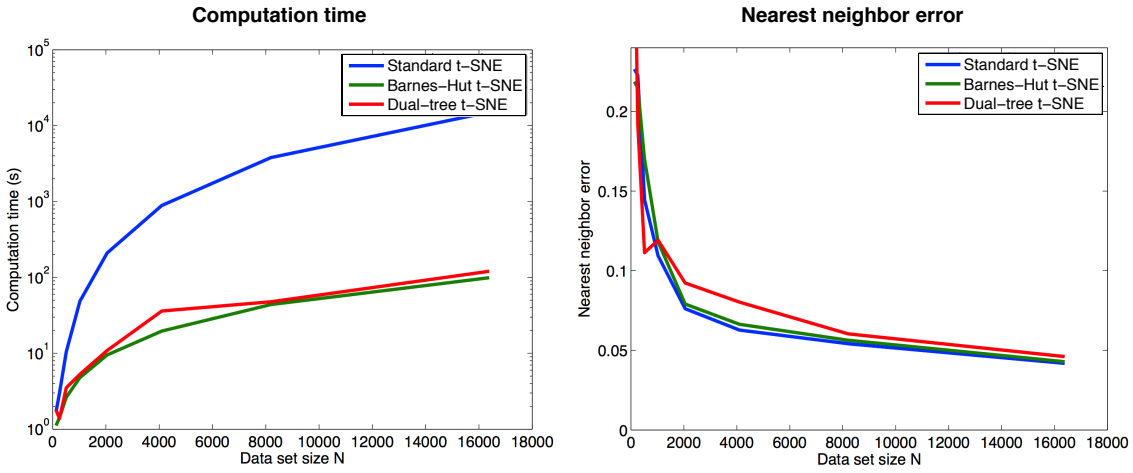


Figure 4: Computation time (in seconds) required to embed MNIST digits (left) and the 1-nearest neighbor errors of the corresponding embeddings (right) as a function of data set size N for standard t-SNE (in blue), Barnes-Hut t-SNE (in green), and dual-tree t-SNE (in red). Note that the required computation time, which is shown on the y-axis of the left figure, is plotted on a logarithmic scale.

in the last convolutional layer as features for the house number images. Please note that supervised information was used to obtain these features.

TIMIT. The TIMIT data set contains $3,696$ spoken utterances (with a total of $N = 1,105,455$ frames) by both male and female speakers.[5] Each frame of the utterances is labeled according to one of 39 phones. The features that we used in our experiments are 13 mel-frequency cepstral coefficients (MFCC features) computed on sliding windows of speech with 25 ms windows at a 10 ms frame rate. In addition, we employ the corresponding delta features and delta-delta features (Sha and Saul, 2006), which leads to a 39-dimensional feature representation. For each frame, all MFCC features within a window of width 7 are concatenated, leading to $D = 273$-dimensional feature vectors that are used as input data.

5.2 Experimental Setup

In all experiments, we follow the experimental setup of van der Maaten and Hinton (2008) as closely as possible. In particular, we initialize the embedding $\mathcal{E}$ by sampling the points $\mathbf{y}_i$ from a Gaussian with a variance of 10^{-4}, and we run a gradient-descent optimizer for $1,000$ iterations, setting the initial step size to 200. We update the step size during the optimization using the scheme of Jacobs (1988). We use an additional momentum term that has weight 0.5 during the first 250 iterations, and 0.8 afterwards. In all experiments, the perplexity u used to compute the input similarities is fixed to 50. All data sets were preprocessed using PCA to reduce their dimensionality to 50 before t-SNE was performed.

5. We only used the TIMIT training set in our experiments.

During the first 250 learning iterations, we multiplied all p_{ij}-values by a user-defined constant $\alpha > 1$. As explained by van der Maaten and Hinton (2008), this trick enables t-SNE to find a better global structure in the early stages of the optimization by creating very tight clusters of points that can easily move around in the embedding space. In preliminary experiments, we found that this trick becomes increasingly important to obtain good embeddings when the data set size increases, as it becomes harder for the optimization to find a good global structure when there are more points in the embedding because there is less space for clusters to move around. In our experiments, we fix $\alpha = 12$ (by contrast, van der Maaten and Hinton, 2008 used $\alpha = 4$).

5.3 Results

We present the results of three sets of experiments. In the first experiment, we investigate the effect of the trade-off parameter θ on the speed and the quality of embeddings produced by Barnes-Hut t-SNE and dual-tree t-SNE on the MNIST data set. In the second experiment, we investigate the computation time required by both approaches as a function of the number of input objects N (also on the MNIST data set). In the third experiment, we construct and visualize embeddings of all five data sets. All computation times were measured on a laptop computer with an Intel Core i5 4258U CPU running at 2.6GHz.

Experiment 1. Figure 3 presents the results of experiments with Barnes-Hut t-SNE and dual-tree t-SNE in which we varied the speed-accuracy trade-off parameter θ used to construct the embedding. The figure shows the computation time required to construct embeddings of all $70,000$ MNIST digit images, as well as the 1-nearest neighbor error (computed based on the digit labels) of the corresponding embeddings. The nearest-neighbor error of an embedding is a measure for the quality of an embedding. Note that the special case $\theta = 0$ corresponds to standard t-SNE of van der Maaten and Hinton (2008); we did not perform an experiment with $\theta = 0$ because standard t-SNE would take too long to complete on the full MNIST data set.

The results presented in the figure highlight the merits of using tree-based t-SNE algorithms. In particular, the results show that Barnes-Hut t-SNE with $\theta = 0.5$ and dual-tree t-SNE with $\theta = 0.2$ lead to embeddings that are of the same quality as those obtained with standard t-SNE (when quality is measured in terms of nearest-neighbor errors in the embedding). At the same time, increasing the value of θ to these values leads to very substantial improvements in terms of the amount of computation required to construct the embedding: for example, Barnes-Hut t-SNE requires only 751 seconds to embed all $70,000$ MNIST digits when $\theta = 0.5$, whereas the original t-SNE algorithm would have taken many days to complete. The results presented in the figure also suggest that dual-tree t-SNE has a slightly worse speed-accuracy trade-off than Barnes-Hut t-SNE: Barnes-Hut t-SNE with $\theta = 0.5$ leads to an embedding of slightly higher quality than dual-tree t-SNE with $\theta = 0.2$, whilst at the same time requiring fewer computational resources.

Experiment 2. In Figure 4, we compare standard t-SNE, Barnes-Hut t-SNE, and dual-tree t-SNE in terms of: (1) the computation time required for the embedding of MNIST digit images as a function of the data set size N and (2) the 1-nearest neighbor errors of the corresponding embeddings. Note that the y-axis of the left figure, which represents the required computation time in seconds, uses a logarithmic scale. Based on the results of the

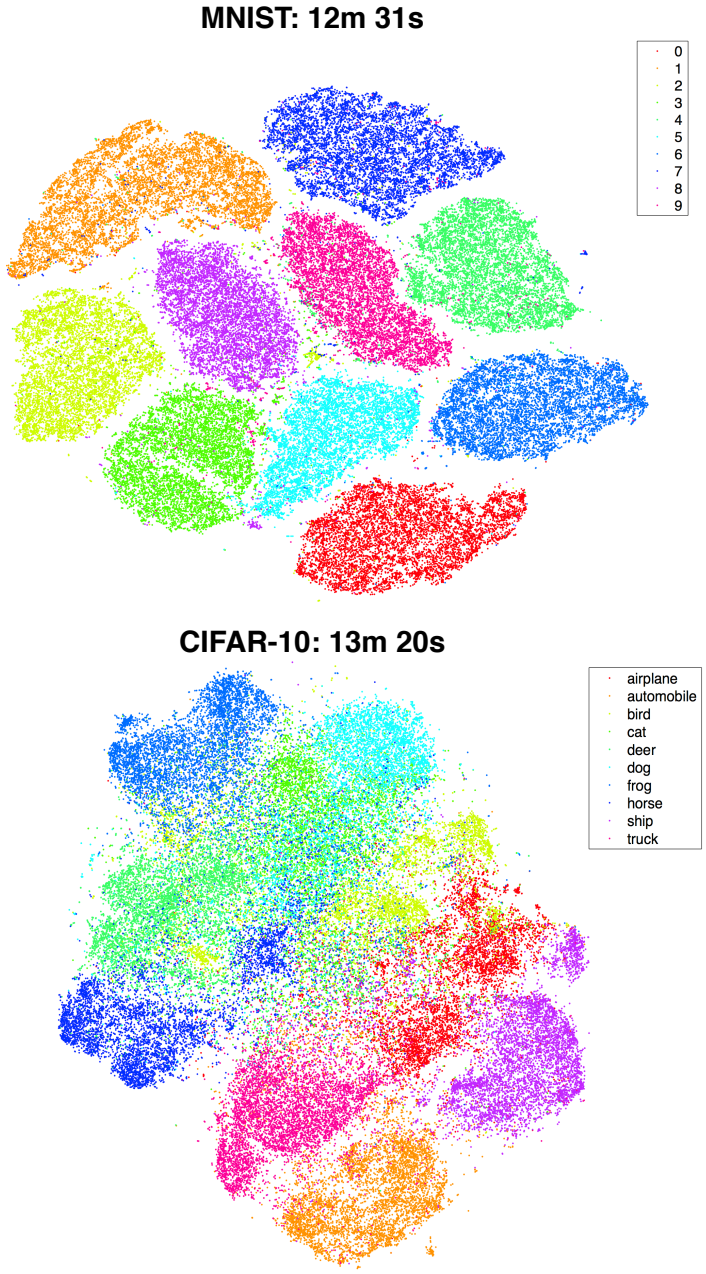


Figure 5: Barnes-Hut t-SNE visualizations obtained with $\theta = 0.5$ of two data sets: MNIST handwritten digits (top) and CIFAR-10 tiny images (bottom). The colors of the points indicate the classes of the corresponding objects. The titles of the figures indicate the computation time that was used to construct the corresponding embeddings. Figure best viewed in color.

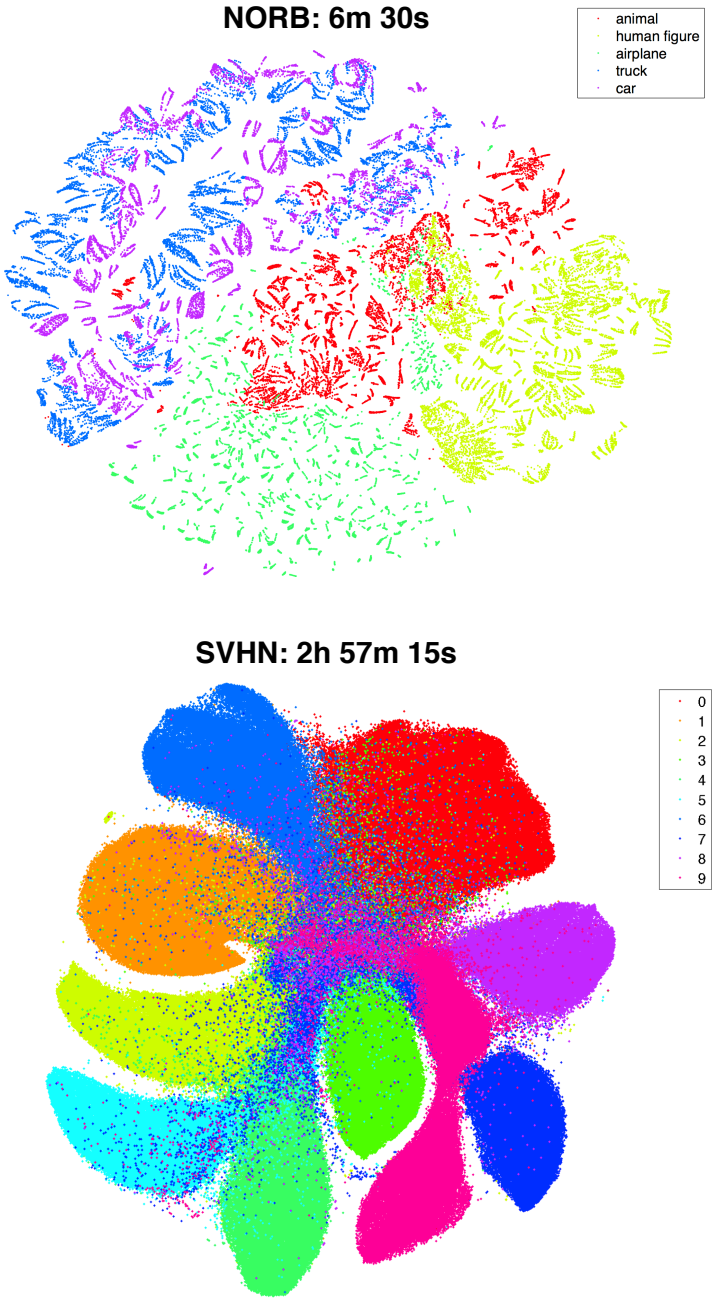


Figure 6: Barnes-Hut t-SNE visualizations obtained with $\theta = 0.5$ of two data sets: NORB object images (top), and street view house numbers (SVHN) data set (bottom). The colors of the points indicate the classes of the corresponding objects. The titles of the figures indicate the computation time that was used to construct the corresponding embeddings. Figure best viewed in color.

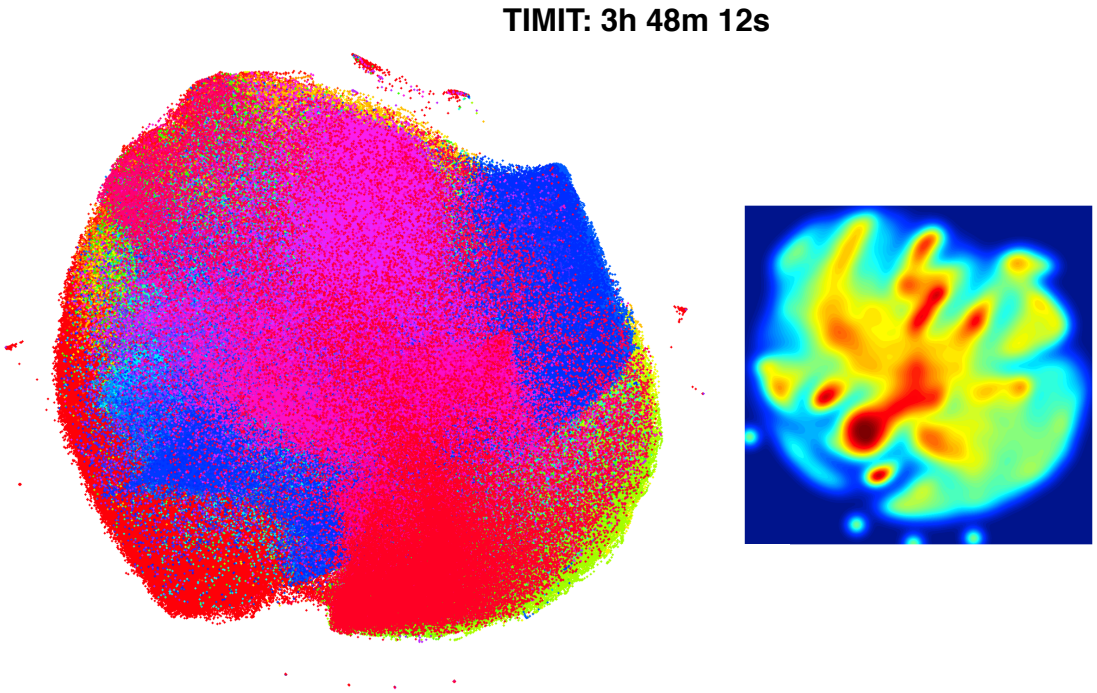


Figure 7: Barnes-Hut t-SNE visualization obtained with $\theta = 0.5$ of the TIMIT speech frames data set. The left figure shows a scatter plot in which the colors of the points indicate the classes of the corresponding objects. The right figure shows a Parzen density estimate of the two-dimensional embedding. The title of the figure indicates the computation time that was used to construct the corresponding embeddings. Figure best viewed in color.

previous experiment 1, we fixed the parameter θ to 0.5 in the experiments with Barnes-Hut t-SNE; in the experiments with dual-tree t-SNE, we fixed θ to 0.2.

The results presented in Figure 4 show that both Barnes-Hut t-SNE and dual-tree t-SNE are indeed orders of magnitude faster than standard t-SNE, whilst the difference in quality of the constructed embeddings (which is measured by the nearest-neighbor errors) is negligible. Most prominently, the computational advantages of Barnes-Hut t-SNE and dual-tree t-SNE rapidly increase as the number of objects in the data set N increases. The results also suggest that a fixed value of $\theta = 0.5$ for Barnes-Hut t-SNE and $\theta = 0.2$ for dual-tree t-SNE appears to work well across a range of data set sizes N. As in the first experiment, the results of this experiment also suggest that Barnes-Hut t-SNE slightly outperforms dual-tree t-SNE in terms of the trade-off between quality of the embedding and the associated computational costs.

Experiment 3. Figure 5, 6, and 7 present embeddings of all five data sets constructed by Barnes-Hut t-SNE with $\theta = 0.5$. The colors of the points indicate the classes of the

corresponding objects; the titles of the plots indicate the computation time that was used to construct the corresponding embeddings.

The visualization in the top part of Figure 5 shows that Barnes-Hut t-SNE can efficiently construct high-quality embeddings of the 70,000 MNIST handwritten digit images: although no supervised information was used, all ten digit classes are clearly separated in an embedding that was constructed in just over 12 minutes. Although our MNIST embedding contains many more points, it may be compared with that presented by van der Maaten and Hinton (2008). Visually, the structure of the two embeddings is very similar.

The results on the CIFAR-10 data set (in the bottom part of Figure 5) show a reasonably good separation of classes; in particular, classes such as *truck* and *ship* are clearly separated from the other classes. To evaluate the quality of the CIFAR-10 embedding, we measured the generalization error of an 11-nearest neighbor classifier that was trained on the 2D representation of the training instances and evaluated on the 2D representation of the test instances (note that the figure shows a joint embedding of training and test data): the generalization error of this classifier 0.2467, which is not much worse than the performance of a logistic regressor trained on the original $D = 1,024$-dimensional features.

The results obtained on the NORB data set are presented in the top part of Figure 6, and reveal a clear separation of the five classes even though supervised information was not used in the construction of the embedding. In addition, the embedding of the NORB images accurately reveals the rotation manifolds that are present in the NORB data set. The different rotation manifolds that belong to the same class correspond to different elevations and lighting conditions.

The results obtained on the street view house numbers (SVHN) data set in the bottom part of Figure 6 show that Barnes-Hut SNE can also model the global structure of the data correctly when the data set becomes very large (recall that there are 630,420 images in the SVHN data set): all classes are quite well separated in the embedding of the SVHN data set, with the exception of a group of images in which the house numbers are difficult to recognize and that are grouped in the center of the embedding. Further analysis of the SVHN embedding revealed that the majority of misclassifications by the convolutional network are indeed located in this central region of the embedding.

The results presented in the Figure 7 show that tree-based variants of t-SNE make it practical to embed data sets with more than a million data points: the TIMIT embedding shows all 1,105,455 speech segments, and was constructed in less than four hours. It should be noted here that scatter plots depicting embeddings of millions of instances may not accurately visualize the underlying (class-conditional) densities. To illustrate this problem, the right part of Figure 7 shows a Parzen density estimate of the two-dimensional embedding. This density estimate clearly shows that the density of points is not nearly uniform over the embedding space, even though the scatter plot does suggest this. In fact, inspection of the density estimates of the individual classes reveals that most classes are in fact modeled by small, dense clusters in the two-dimensional embedding. This suggests the use of class-conditional density maps (van Eck and Waltman, 2010) for the visualization of such large-scale embeddings.

A version of the MNIST embedding in which the original digit images are shown is presented in Figure 8. The insets in this figure reveal that, like standard t-SNE, Barnes-Hut t-SNE is very good at preserving local structure of the data in the embedding: for

instance, the visualization clearly shows that orientation is one of the main sources of variation within the cluster of ones. Embeddings in which the original CIFAR-10, NORB, and SVHN images are presented in the online supplemental material.

6. Conclusion

We investigated two tree-based implementations of t-SNE (van der Maaten and Hinton, 2008), called Barnes-Hut t-SNE and dual-tree t-SNE, that: (1) construct a sparse approximation of the similarities between input objects using vantage-point trees and (2) approximate the t-SNE gradient by computing interactions between groups of points instead of between pairs of points. The new t-SNE variants run in $\mathcal{O}(N \log N)$ rather than $\mathcal{O}(N^2)$, and require only $\mathcal{O}(N)$ memory. Our experimental evaluation of Barnes-Hut t-SNE and dual-tree t-SNE shows that both algorithms are substantially faster than standard t-SNE, and that both facilitate the visualization of data sets with millions of input objects in scatter plots. The results of our experiments suggest that Barnes-Hut t-SNE slightly outperforms dual-tree t-SNE (in terms of the trade-off between accuracy and speed) due to the additional bookkeeping that is required in dual-tree t-SNE.

A drawback of the Barnes-Hut variant of t-SNE is that the gradient approximations do not provide any error bounds and can in fact be unbounded (Salmon and Warren, 1994). By contrast, dual-tree and fast multipole methods do provide such error bounds (e.g., Warren and Salmon, 1993; Gray and Moore, 2001; Baxter and Roussos, 2002; Wan and Karniadakis, 2006). None of these bounds, however, takes into account the iterative nature of t-SNE, i.e., the fact that errors may propagate during learning. Vladymyrov and Carreira-Perpiñán (2014) present an error bound that incorporates the iterative nature of SNE-like embedding techniques, but makes strong assumptions on the error per iteration to achieve this bound. de Freitas et al. (2006) present stability results for Krylov subspace iteration, but it is unclear how these results extend to Barnes-Hut and dual-tree t-SNE. In general, we believe the lack of formal error bounds is acceptable because the t-SNE objective function is non-convex anyway: as long as the inner product between the gradient estimate and the true gradient remains positive, we are still guaranteed to converge to a local minimum of the objective function (assuming the step size is set properly; Zoutendijk, 1960).

Another limitation of Barnes-Hut t-SNE and dual-tree t-SNE is that the algorithms can only be used to embed data in two or three dimensions. Generalizations to higher dimensions are impractical because the size of the tree grows exponentially in the dimensionality of the embedding space. Having said that, this limitation is not very severe since t-SNE is mainly used for visualization of data in scatter plots (i.e., for embedding in two or three dimensions). Moreover, it is straightforward to replace the quadtrees used in this paper by metric trees that scale better to high-dimensional embedding spaces.

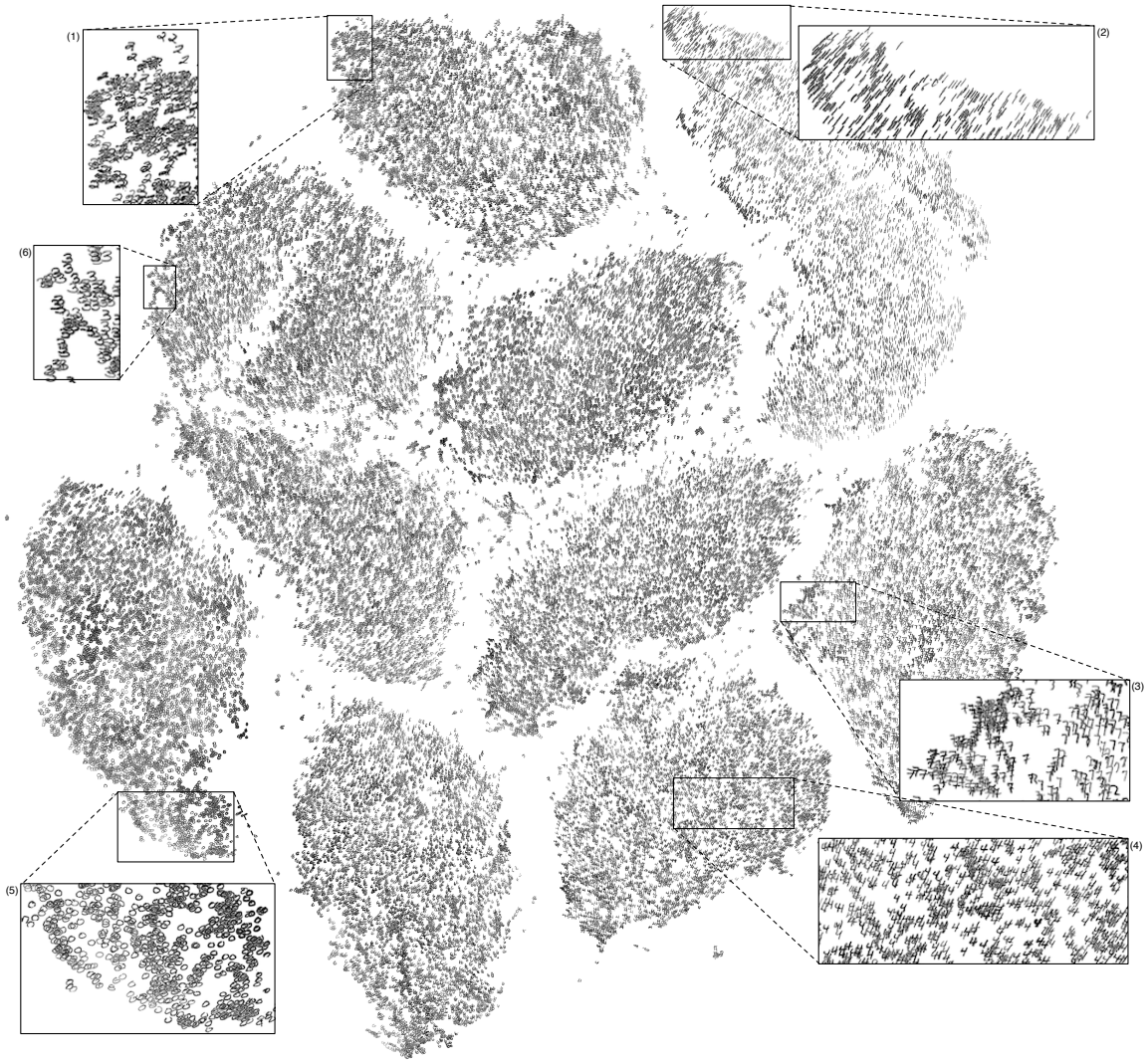


Figure 8: Barnes-Hut t-SNE visualization of all 70,000 MNIST handwritten digit images (constructed in 10 minutes and 45 seconds using $\theta = 0.5$). The insets (from the top left, clockwise) show (1) twos in a curl style grouped together, (2) similarly oriented ones ranging from fat to thin, (3) continental sevens grouped separately from other sevens, (4) similar fours, (5) round zeros ranging from thin to fat, and (6) similar threes. Zoom in on the visualization for more detailed views.

Acknowledgments

The research leading to these results has received funding from the Netherlands Organization for Scientific Research (NWO) under grant agreement n^o 612.001.301, and from the European Union Seventh Framework Program (FP7/2007-2013) under grant agreement n^o 604102. The author thanks Geoffrey Hinton for many helpful discussions, and three anonymous reviewers for suggestions that helped to improve the paper.

References

J. Barnes and P. Hut. A hierarchical O(N log N) force-calculation algorithm. *Nature*, 324 (4):446–449, 1986.

B.J.C. Baxter and G. Roussos. A new error estimate of the fast Gauss transform. *SIAM Journal on Scientific Computation*, 24(1):257–259, 2002.

R. Bayer and E. McCreight. Organization and maintenance of large ordered indexes. *Acta Informatica*, 1(3):173–189, 1972.

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

S. Brin. Near neighbor search in large metric spaces. In *Proceedings of the International Conference on Very Large Data Bases*, pages 574–584, 1995.

C.J.C. Burges. Dimension reduction: A guided tour. *Foundations and Trends in Machine Learning*, 2(4):1–95, 2010.

M.Á. Carreira-Perpiñán. The elastic embedding algorithm for dimensionality reduction. In *Proceedings of the International Conference on Machine Learning*, pages 167–174, 2010.

M. Chalmers. A linear iteration time layout algorithm for visualising high-dimensional data. In *Proceedings of IEEE Visualization*, pages 127–132, 1996.

K. Cho, B. van Merriënboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *arXiv 1406.1078*, 2014.

D.J. Croton, V. Springel, S.D.M. White, G. De Lucia, C.S. Frenk, L. Gao, A. Jenkins, G. Kauffmann, J.F. Navarro, and N. Yoshida. The many lives of active galactic nuclei: cooling flows, black holes and the luminosities and colours of galaxies. *Monthly Notices of the Royal Astronomical Society*, 365(1):11–28, 2006.

N. de Freitas, Y. Wang, M. Mahdaviani, and D. Lang. Fast Krylov methods for N-body learning. In *Advances in Neural Information Processing Systems*, volume 18, pages 251–258, 2006.

J.H. Freidman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3:209–226, 1977.

T.M.J. Fruchterman and E.M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.

K. Fukunaga and P.M. Narendra. A branch and bound algorithm for computing k-nearest neighbors. *IEEE Transactions on Computers*, 24:750–753, 1975.

A.G. Gray. Fast kernel matrix-vector multiplication with application to gaussian process learning. Technical Report CMU-CS-04-110, Carnegie Mellon University, 2004.

A.G. Gray and A.W. Moore. N-body problems in statistical learning. In *Advances in Neural Information Processing Systems*, pages 521–527, 2001.

A.G. Gray and A.W. Moore. Rapid evaluation of multiple density models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2003.

L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, 1987.

J. Heer, M. Bostock, and V. Ogievetsky. A tour through the visualization zoo. *Communications of the ACM*, 53:59–67, 2010.

G.E. Hinton and S.T. Roweis. Stochastic Neighbor Embedding. In *Advances in Neural Information Processing Systems*, volume 15, pages 833–840, 2003.

G.E Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. In *arXiv 1207.0580*, 2012.

Y. Hu. Efficient and high-quality force-directed graph drawing. *The Mathematica Journal*, 10(1):37–71, 2005.

P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of* 30^{th} *Symposium on Theory of Computing*, 1998.

R.A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295–307, 1988.

S. Ji. Computational genetic neuroanatomy of the developing mouse brain: dimensionality reduction, visualization, and clustering. *BMC Bioinformatics*, 14(222):1–14, 2013.

Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. `http://caffe.berkeleyvision.org/`, 2013.

D. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann. *Mastering the Information Age: Solving Problems with Visual Analytics*. Eurographics Association, Germany, 2010.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

C.C. Laczny, N. Pinel, N. Vlassis, and P. Wilmes. Alignment-free visualization of metagenomic data by nonlinear dimension reduction. *Scientific Reports*, 4:1–12, 2014.

D. Lang, M. Klaas, and N. de Freitas. Empirical testing of fast kernel density estimation algorithms. Technical Report TR-2005-03, University of British Columbia, 2005.

N.D. Lawrence. Spectral dimensionality reduction via maximum entropy. *Proceedings of the International Conference on Artificial Intelligence and Statistics, JMLR W&CP*, 15: 51–59, 2011.

Y. LeCun, F.J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 97–104, 2004.

T. Liu, A.W. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In *Advances in Neural Information Processing Systems*, volume 17, pages 825–832, 2004.

M. Mahdaviani, N. de Freitas, B. Fraser, and F. Hamze. Fast computational methods for visually guided robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 138–143, 2005.

M. Muja and D.G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proceedings of the International Conference on Computer Vision Theory and Applications*, 2009.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A.Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, 2006.

A. Quigley and P. Eades. FADE: Graph drawing, clustering, and visual abstraction. In *Proceedings of the International Symposium on Graph Drawing*, pages 197–210, 2000.

V.C. Raykar and R. Duraiswami. Fast optimal bandwidth selection for kernel density estimation. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 524–528, 2006.

V. Rokhlin. Rapid solution of integral equations of classic potential theory. *Journal of Computational Physics*, 60:187–207, 1985.

S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000.

R.R. Salakhutdinov and G.E. Hinton. Semantic hashing. In *Proceedings of the SIGIR Workshop on Information Retrieval and Applications of Graphical Models*, pages 52–63, 2007.

J.K. Salmon and M.S. Warren. Skeletons from the treecode closet. *Journal of Computational Physics*, 111(1):136–155, 1994.

L.K. Saul, K.Q. Weinberger, J.H. Ham, F. Sha, and D.D. Lee. Spectral methods for dimensionality reduction. In *Semisupervised Learning*. The MIT Press, 2006.

P. Sermanet, S. Chintala, and Y. LeCun. Convolutional neural networks applied to house numbers digit classification. In *Proceedings of the International Conference on Pattern Recognition*, pages 3288–3291, 2012.

F. Sha and L.K. Saul. Large margin Gaussian mixture modeling for phonetic classification and recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 265–268, 2006.

C. Silpa-Anan and R. Hartley. Optimised kd-trees for fast image descriptor matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

V. Springel, N. Yoshidaa, and S.D.M. White. GADGET: A code for collisionless and gasdynamical cosmological simulations. *New Astronomy*, 6(2):79–117, 2001.

J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

P. Tiňo and I.T. Nabney. Hierarchical GTM: Constructing localized nonlinear projection manifolds in a principled way. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):639–656, 2002.

A. Torralba, R. Fergus, and W.T. Freeman. 80 million tiny images: A large dataset for non-parametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.

L.J.P. van der Maaten. Learning a parametric embedding by preserving local structure. In *Proceedings of the International Conference on Artificial Intelligence and Statistics, JMLR W&CP*, volume 5, pages 384–391, 2009.

L.J.P. van der Maaten. Barnes-Hut-SNE. In *Proceedings of the International Conference on Learning Representations*, 2013.

L.J.P. van der Maaten and G.E. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2431–2456, 2008.

L.J.P. van der Maaten, E.O. Postma, and H.J. van den Herik. Dimensionality reduction: A comparative review. Technical Report TiCC-TR 2009-005, Tilburg University, 2009.

N.J. van Eck and L. Waltman. Software survey: Vosviewer, a computer program for bibliometric mapping. *Scientometrics*, 84:523–538, 2010.

J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *Journal of Machine Learning Research*, 11(Feb):451–490, 2010.

M. Vladymyrov and M.Á. Carreira-Perpiñán. Partial-Hessian strategies for fast learning of nonlinear embeddings. In *Proceedings of the International Conference on Machine Learning*, pages 345–352, 2012.

M. Vladymyrov and M.Á. Carreira-Perpiñán. Entropic affinities: Properties and efficient numerical computation. *Proceedings of the International Conference on Machine Learning, JMLR W&CP*, 28(3):477–485, 2013.

M. Vladymyrov and M.A. Carreira-Perpiñán. Linear-time training of nonlinear low-dimensional embeddings. In *Proceedings of the International Conference on Artificial Intelligence and Statistics. JMLR: W&CP*, volume 33, pages 968–977, 2014.

X. Wan and G.E. Karniadakis. A sharp error estimate for the fast gauss transform. *Journal of Computational Physics*, 219(1):7–12, 2006.

M.S. Warren and J.K. Salmon. A parallel hashed octtree N-body algorithm. In *Proceedings of the ACM/IEEE Conference on Supercomputing*, pages 12–21, 1993.

Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems*, pages 1753–1760, 2008.

C. Yang, R. Duraiswami, N.A. Gumerov, and L. Davis. Improved fast Gauss transform and efficient kernel density estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 664–671, 2003.

Z. Yang, J. Peltonen, and S. Kaski. Scalable optimization of neighbor embedding for visualization. In *Proc. of the Int. Conf. on Machine Learning*, 2013.

P.N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321, 1993.

G. Zoutendijk. *Methods of Feasible Directions*. Elsevier Publishing Company, Amsterdam, The Netherlands, 1960.

Journal of Machine Learning Research 1 (2014) 3247-3295 Submitted 6/13; Revised 4/14; Published 8/14

Set-Valued Approachability and Online Learning with Partial Monitoring

Shie Mannor SHIE@EE.TECHNION.AC.IL
Israel Institute of Technology (Technion)
Faculty of Electrical Engineering
32000 Haifa, Israel

Vianney Perchet VIANNEY.PERCHET@NORMALESUP.ORG
Université Paris Diderot, LPMA
8 place FM/13
75013 Paris, France

Gilles Stoltz STOLTZ@HEC.FR
GREGHEC: HEC Paris — CNRS
1 rue de la Libération
78351 Jouy-en-Josas, France

Editor: Alexander Rakhlin

Abstract

Approachability has become a standard tool in analyzing learning algorithms in the adversarial online learning setup. We develop a variant of approachability for games where there is ambiguity in the obtained reward: it belongs to a set rather than being a single vector. Using this variant we tackle the problem of approachability in games with partial monitoring and develop a simple and generally efficient strategy (i.e., with constant per-step complexity) for this setup. As an important example, we instantiate our general strategy to the case when external regret or internal regret is to be minimized under partial monitoring.

Keywords: online learning, approachability, regret, partial monitoring

1. Introduction

Blackwell's approachability theory and its variants have become a standard and useful tool in analyzing online learning algorithms (Cesa-Bianchi and Lugosi, 2006) and algorithms for learning in games (Hart and Mas-Colell, 2000, 2001). The first application of Blackwell's approachability to learning in the online setup is due to Blackwell (1956b) himself. Numerous other contributions are summarized in the monograph by Cesa-Bianchi and Lugosi (2006). Blackwell's approachability theory enjoys a natural geometric interpretation that allows it to be used in situations where other learning methods (online convex optimization or exponential weights) do not seem to be easily applicable. In some sense, it can be used to go beyond the minimization of the regret to control quantities of a different flavor. Examples of such uses can be found in Mannor et al. (2009), which minimizes the regret together with path constraints, and in Mannor and Shimkin (2008), which minimizes the regret in games whose stage duration is not fixed. Recently, it has been shown by Abernethy et al.

(2011) that approachability and low regret learning are equivalent in the sense that efficient reductions exist from one problem to the other. Another recent paper by Rakhlin et al. (2011) showed that approachability can be analyzed from the perspective of learnability using tools from learning theory.

In this paper we consider approachability and online learning with partial monitoring in games against an arbitrary opponent. That is, we will obtain worst-case performance guarantees: guarantees that are valid for *all* strategies of the opponent. In partial monitoring the decision maker does not know how much reward was obtained and only gets a (random) signal whose distribution depends on the pair of actions taken by the decision maker and the opponent. There are two extremes of this setup that are well studied. On the one extreme we have the case where the signal includes the reward itself (or a signal that can be used to unbiasedly estimate the reward), which is essentially the celebrated bandits setup. The other extreme is the case where the signal is not informative (i.e., it tells the decision maker nothing about the actual reward obtained); this setting then essentially consists of repeating the same situation over and over again, as no information is gained over time. We consider a setup encompassing these situations and more general ones, in which the signal is indicative of the actual reward, but is not necessarily a sufficient statistic thereof. The difficulty is that the decision maker cannot compute the actual reward obtained nor the actions of the opponent.

Regret minimization with partial monitoring (defined in the general sense of Rustichini, 1999) has been studied in several papers in the learning theory community. Piccolboni and Schindelhauer (2001), Mannor and Shimkin (2003), Cesa-Bianchi et al. (2006), Bartók et al. (2010, 2011), Foster and Rakhlin (2012) study games in which an accurate estimation of the rewards (or worst-case rewards) of the decision maker is possible thanks to some statistically sufficient monitoring; in this case, the notion of regret with partial monitoring reduces to the classical notion of regret with full monitoring. A general policy with vanishing external regret with partial monitoring is presented by Lugosi et al. (2008). This policy is based on exponential weights and a specific estimation procedure for the (worst-case) obtained rewards.

In contrast, we devise a general (efficient) algorithm for the problem of approachability under partial monitoring. We then apply it to the more restricted problem of regret minimization. More precisely, we first define a new type of approachability setup, for set-valued functions, which enables to re-derive the extension of approachability to the partial monitoring vector-valued setting proposed by Perchet (2011a). More importantly, we provide concrete algorithms for this approachability problem that are more efficient in the sense that, unlike previous works in the domain, their complexity is constant over all steps. Moreover, their rates of convergence are independent of the game at hand, as in the seminal paper by Blackwell (1956b) but for the first time in this general framework. For example, the recent theoretical study of approachability by Perchet and Quincampoix (2011), which is based on somehow related arguments, does neither provide rates of convergence nor concrete algorithms.

1.1 Outline and Comparison to Known Results

The paper is organized in three main parts. The first part consists of Section 2, where we recall basic facts from approachability theory, when a decision maker faces an arbitrary opponent in the standard vector-valued games setting.

The second part deals with our first contribution, a novel setup for approachability termed "set-valued approachability", where instead of obtaining a vector-valued reward, the decision maker obtains a set, that represents the ambiguity concerning his reward. In Section 3, we provide a simple characterization of approachable convex sets and an algorithm for the set-valued reward setup under the assumption that the set-valued reward functions are linear. In Section 4 we extend the set-valued approachability setup to problems where the set-valued reward functions are not linear, but rather concave in the mixed action of the decision maker and convex in the mixed action of the opponent. This new concept of set-valued approachability is interesting on its own, as it cannot be directly encompassed into classical vector-valued approachability; yet we retrieve several familiar results (characterization of approachable convex sets, rates of convergence that are independent of the dimension, and so on). More importantly, these results are the key tools for our second series of contributions, which we describe now.

The third part studies approachability in repeated games with partial monitoring. Previous general results in this setup suffered from at least one of the following drawbacks. They were either non-constructive (Rustichini, 1999) or were highly inefficient. The latter drawback refers to strategies that relied on some sort of lifting to the space of probability measures on mixed actions (see e.g., Lehrer and Solan, 2007 and Perchet, 2009, 2011a). They then typically required a fine grid of elements in this lifted space, which had to be progressively refined over time. This construction leads to two main issues: on the one hand, the step complexity continuously increases and becomes prohibitive in the number T of past steps. On the other hand, rates of convergence deteriorate and depend on the dimension. Our aim is therefore to devise algorithms that are efficient (as long as the projection onto some convex set can be done efficiently), with a constant step complexity (although it may depend on parameters of the problem at hand), and with rates of convergence independent of the ambient dimension. Our strategies are the first, to our knowledge, satisfying all of these properties in the general approachability framework. They do so because they do not rely on finer and finer grids; as a byproduct, they can also be considered more natural. Section 5 discusses in greater detail all the points mentioned in this paragraph.

More precisely, we state in Section 5.1 the necessary and sufficient condition for approachability in games with partial monitoring and show in Section 5.2 how to apply set-valued approachability framework to the repeated vector-valued games with partial monitoring. In Section 5.3 we then consider a specific type of games where the signaling structure possesses a special property, called bi-piecewise linearity, that can be exploited to derive simple, constructive and efficient strategies. This type of games is rich enough as it encompasses several useful special cases discussed in the later sections. In Section 5.4, we mention the general signaling case and explain how it is possible to approach certain special sets such as polytopes efficiently (thanks to a reduction to bi-piecewise linearity) with the same dimension-independent rates of convergence—and even general convex sets, although inefficiently in the latter case.

As an important other example of a setting where bi-piecewise linearity holds, we apply in Section 6 the results of Section 5.3 to both external-regret and internal-regret minimization in repeated games with partial monitoring. In this specific case, our algorithms have rates similar to the ones obtained by Lugosi et al. (2008) but slower than Perchet (2011b); however our proof is direct and simpler and the strategy is efficient.

1.2 Mixed Actions versus Pure Actions

Most of Sections 2–4 (classical approachability and set-valued approachability) is concerned with mixed actions, while Sections 5–6 (approachability in games with partial monitoring) are focused on pure actions. The explanation for this is as follows. Even though pure actions are inherent to the model of partial monitoring, the reduction from approachability in games with partial monitoring to set-valued approachability, as described in Section 5.2, is to set-valued approachability with mixed actions.

2. Some Basic Facts from Approachability Theory

In this section we recall the most basic version of Blackwell's approachability theorem for vector-valued payoff functions.

We consider a vector-valued game between two players, a decision maker (first player) and an opponent (second player), with respective finite action sets $\mathcal{A}$ and $\mathcal{B}$, whose cardinalities are referred to as $N_{\mathcal{A}}$ and $N_{\mathcal{B}}$. We denote by d the dimension of the reward vectors and equip $\mathbb{R}^d$ with the ℓ^2–norm $\|\cdot\|_2$. The payoff function of the first player is given by a mapping $m : \mathcal{A} \times \mathcal{B} \to \mathbb{R}^d$, which is multi-linearly extended to $\Delta(\mathcal{A}) \times \Delta(\mathcal{B})$, the set of product-distributions over $\mathcal{A} \times \mathcal{B}$.

We consider a framework in which mixed actions are taken. We denote by $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ and $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots$ the actions in $\Delta(\mathcal{A})$ and $\Delta(\mathcal{B})$ sequentially taken by each player. We assume a full or bandit monitoring for the first player: at the end of round t, when receiving the payoff $m(\boldsymbol{x}_t, \boldsymbol{y}_t)$, either the mixed action $\boldsymbol{y}_t$ (full monitoring) or only the indicated payoff (bandit monitoring) is revealed to him.

Strategies of the players are defined as mappings associating the information available at the beginning of each round $t \geqslant 1$ with a mixed action. In particular, strategies of the first player in the case of full monitoring associate with $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}$ and $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}$ a mixed action $\boldsymbol{x}_t \in \Delta(\mathcal{A})$, while in the case of bandit monitoring, they do this association based on $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}$ and $m(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, m(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1})$. We do not restrict the opponent and assume a full monitoring for him: his strategies associate with $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}$ and $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}$ a mixed action $\boldsymbol{y}_t \in \Delta(\mathcal{B})$.

2.1 Necessary and Sufficient Condition for Approachability

Given a set $\mathcal{C}$, the aim of the first player is to ensure that his average payoff converges to $\mathcal{C}$, while the second player wants to prevent it. This gives rise to Blackwell's classical definition of approachability. (Here, we state it as m–approachability to remind the reader, in the notation, that the underlying payoff function is m.)

Definition 1 *Given a function $m : \mathcal{A} \times \mathcal{B} \to \mathbb{R}^d$, a set $\mathcal{C} \subseteq \mathbb{R}^d$ is* m–approachable *by the first player if he has a strategy such that, for all $\varepsilon > 0$, there exists an integer T_ε such that for all strategies of the second player,*

$$\mathbb{P}\left\{\forall\, T \geqslant T_\varepsilon, \quad \inf_{c\in\mathcal{C}} \left\| c - \frac{1}{T}\sum_{t=1}^{T} m\big(\boldsymbol{x}_t, \boldsymbol{y}_t\big) \right\|_2 \leqslant \varepsilon \right\} \geqslant 1-\varepsilon\,.$$

In particular, the first player has a strategy that ensures that the average of his vector-valued payoffs converges almost surely to the set $\mathcal{C}$, uniformly with respect to the strategies of the second player.

As will be recalled below in Theorem 3, even stronger approachability guarantees can be achieved. Indeed, the first player has deterministic strategies such that, for all (deterministic or randomized) strategies of the second player, with probability 1, for all $T \geqslant 1$,

$$\inf_{c\in\mathcal{C}} \left\| c - \frac{1}{T}\sum_{t=1}^{T} m\big(\boldsymbol{x}_t, \boldsymbol{y}_t\big) \right\|_2 \leqslant \beta(T)\,,$$

where $\beta(\cdot)$ is some decreasing mapping to 0 to be determined later.

For closed convex sets there is a simple characterization of approachability that is a direct consequence of von Neumann's minimax theorem.

Theorem 2 (see Blackwell, 1956a, Theorem 3) *A closed convex set $\mathcal{C} \subseteq \mathbb{R}^d$ is approachable if and only if*

$$\forall\, \boldsymbol{y} \in \Delta(\mathcal{B}), \quad \exists\, \boldsymbol{x} \in \Delta(\mathcal{A}), \qquad\qquad m(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{C}\,.$$

2.2 An Associated Strategy (whose Efficiency Depends on the Geometry of $\mathcal{C}$)

Blackwell suggested a simple strategy with a geometric flavor; it only requires bandit monitoring.

Play an arbitrary $\boldsymbol{x}_1$. For $t \geqslant 1$, given the vector-valued quantity

$$\widehat{m}_t = \frac{1}{t}\sum_{s=1}^{t} m(\boldsymbol{x}_s, \boldsymbol{y}_s)\,,$$

compute the projection c_t (in ℓ^2–norm) of $\widehat{m}_t$ on $\mathcal{C}$. Find a mixed action $\boldsymbol{x}_{t+1}$ that solves the minimax equation

$$\min_{\boldsymbol{x}\in\Delta(\mathcal{A})} \max_{\boldsymbol{y}\in\Delta(\mathcal{B})} \big\langle \widehat{m}_t - c_t,\, m(\boldsymbol{x}, \boldsymbol{y}) - c_t \big\rangle\,, \tag{1}$$

where $\langle\,\cdot\,,\,\cdot\,\rangle$ is the Euclidean inner product in $\mathbb{R}^d$.

The stated minimax problem for determining $\boldsymbol{x}_{t+1}$ can be solved efficiently using, e.g., linear programming: the associated complexity is polynomial in $N_\mathcal{A}$ and $N_\mathcal{B}$. This strategy is efficient if computing the required projections onto $\mathcal{C}$ in ℓ^2–norm can be performed efficiently.

The strategy presented above enjoys the following rates of convergence for approachability, which can be derived as a special case of the results stated and proved in Theorem 25 later in this paper.

Theorem 3 (see Blackwell, 1956a, Theorems 1 and 3) *We consider an approachable closed convex set $\mathcal{C} \subseteq \mathbb{R}^d$ and we denote by M a bound in norm over m, i.e.,*

$$\max_{(a,b)\in\mathcal{A}\times\mathcal{B}} \big\| m(a,b) \big\|_2 \leqslant M \,.$$

The above strategy ensures that for all strategies of the second player, with probability 1, for all $T \geqslant 1$,

$$\inf_{c\in\mathcal{C}} \left\| c - \frac{1}{T}\sum_{t=1}^{T} m\big(\boldsymbol{x}_t, \boldsymbol{y}_t\big) \right\|_2 \leqslant \frac{2M}{\sqrt{T}} \,.$$

3. Set-Valued Approachability for Finite Games

In this section we extend the results from the previous section to set-valued payoff functions in the case of full monitoring. We denote by $\mathcal{S}\big(\mathbb{R}^d\big)$ the set of all subsets of $\mathbb{R}^d$ and consider a set-valued payoff function $\overline{m} : \mathcal{A} \times \mathcal{B} \to \mathcal{S}\big(\mathbb{R}^d\big)$. When the players choose respective actions $a \in \mathcal{A}$ and $b \in \mathcal{B}$, the first player gets the *subset* $\overline{m}(a,b)$ as a payoff. This models the ambiguity or uncertainty associated with some true underlying payoff.

3.1 Mixed Actions Taken and Observed

For the moment, we only consider the case of mixed actions taken and observed, keeping the same definition of a strategy as in the previous section. (The next subsection will briefly explain, for the sake of completeness, how to deal with the case of pure actions taken and observed.)

We extend $\overline{m}$ multi-linearly to $\Delta(\mathcal{A}) \times \Delta(\mathcal{B})$ and even to $\Delta(\mathcal{A} \times \mathcal{B})$, the set of joint probability distributions on $\mathcal{A} \times \mathcal{B}$, as follows. Let

$$\mu = \big(\mu_{a,b}\big)_{(a,b)\in\mathcal{A}\times\mathcal{B}}$$

be such a joint probability distribution; then $\overline{m}(\mu)$ is defined as a finite convex combination[1] of subsets of $\mathbb{R}^d$,

$$\overline{m}(\mu) = \sum_{a\in\mathcal{A}} \sum_{b\in\mathcal{B}} \mu_{a,b}\, \overline{m}(a,b) \,.$$

The product-distribution of two elements $\boldsymbol{x} = (x_a)_{a\in\mathcal{A}} \in \Delta(\mathcal{A})$ and $\boldsymbol{y} = (y_b)_{b\in\mathcal{B}} \in \Delta(\mathcal{B})$ will be denoted by $\boldsymbol{x} \otimes \boldsymbol{y}$; it gives a probability mass of $x_a y_b$ to each pair $(a,b) \in \mathcal{A} \times \mathcal{B}$. When μ is such a product-distribution, we use the notation $\overline{m}(\mu) = \overline{m}(\boldsymbol{x}, \boldsymbol{y})$.

We can now describe how the game proceeds. At each round t, the players choose simultaneously respective mixed actions $\boldsymbol{x}_t \in \Delta(\mathcal{A})$ and $\boldsymbol{y}_t \in \Delta(\mathcal{B})$. Full monitoring takes place for the first player: he observes $\boldsymbol{y}_t$ at the end of round t and he gets the subset $\overline{m}(\boldsymbol{x}_t, \boldsymbol{y}_t)$ as a payoff (which, again, accounts for the uncertainty).

1. For two sets S, T and $\alpha \in [0,1]$, the convex combination $\alpha S + (1-\alpha)T$ is defined as

$$\big\{\alpha s + (1-\alpha)t, \quad s \in S \text{ and } t \in T\big\} \,.$$

3.1.1 Definition of Set-Valued Approachability

We are interested in the behavior of

$$\frac{1}{T}\sum_{t=1}^{T}\overline{m}(\boldsymbol{x}_t,\boldsymbol{y}_t)=\overline{m}(\nu_T)\,,\qquad\text{where}\qquad\nu_T:=\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t\otimes\boldsymbol{y}_t$$

is the empirical joint distribution of mixed actions taken during the first T rounds.

The distance of this set $\overline{m}(\nu_T)$ to the target set $\mathcal{C}$ will be measured in a worst-case sense (à la Hausdorff): we denote by

$$\varepsilon_T=\sup_{\xi\in\overline{m}(\nu_T)}\inf_{c\in\mathcal{C}}\|c-\xi\|_2$$

the smallest value such that $\overline{m}(\nu_T)$ is included in an ε_T–neighborhood of $\mathcal{C}$. Approachability of a set $\mathcal{C}$ with the set-valued payoff function $\overline{m}$ then simply means that the sequence of ε_T tends almost-surely to 0, uniformly with respect to the strategies of the second player. This is made formal in the following definition.

Definition 4 *Given a set-valued payoff function $\overline{m}:\mathcal{A}\times\mathcal{B}\to\mathcal{S}\big(\mathbb{R}^d\big)$, a set $\mathcal{C}\subseteq\mathbb{R}^d$ is $\overline{m}$–approachable by the first player if he has a strategy such that, for all $\varepsilon>0$, there exists an integer T_ε such that for all strategies of the second player,*

$$\mathbb{P}\left\{\forall\,T\geqslant T_\varepsilon,\quad\sup_{\xi\in\overline{m}(\nu_T)}\inf_{c\in\mathcal{C}}\|c-\xi\|_2\leqslant\varepsilon\right\}\geqslant 1-\varepsilon\,.$$

When the set-valued function $\overline{m}$ is clear from the context, we will simply say that $\mathcal{C}$ is set-valued approachable. Actually, just as in the classical case of approachability, the bounds exhibited below in Theorem 8 will be for deterministic strategies of the first player and will read as follows: for all (deterministic or randomized) strategies of the second player, with probability 1, for all $T\geqslant 1$,

$$\sup_{\xi\in\overline{m}(\nu_T)}\inf_{c\in\mathcal{C}}\|c-\xi\|_2\leqslant\beta(T)\,,$$

where $\beta(\cdot)$ is a mapping decreasing to 0 to be determined.

3.1.2 A Useful Continuity Lemma

Before proceeding we provide a continuity lemma. It can be reformulated as indicating that for all joint distributions μ and ν over $\mathcal{A}\times\mathcal{B}$, the set $\overline{m}(\mu)$ is contained in an $M\,\|\mu-\nu\|_1$–neighborhood of $\overline{m}(\nu)$, where M is a bound in ℓ^2–norm on $\overline{m}$. This is a result that we will use repeatedly.

Definition 5 *The set-valued function $\overline{m}:\mathcal{A}\times\mathcal{B}\to\mathcal{S}\big(\mathbb{R}^d\big)$ is bounded in ℓ^2–norm by M if*

$$\forall(a,b)\in\mathcal{A}\times\mathcal{B},\qquad\sup_{\xi\in\overline{m}(a,b)}\|\xi\|_2\leqslant M\,.$$

Lemma 6 *Let μ and ν be two probability distributions over $\mathcal{A} \times \mathcal{B}$. We assume that the set-valued function $\overline{m}$ is bounded in ℓ^2–norm by M. Then*

$$\sup_{\xi \in \overline{m}(\mu)} \inf_{c \in \overline{m}(\nu)} \|\xi - c\|_2 \leqslant M \|\mu - \nu\|_1 \leqslant M\sqrt{N_{\mathcal{A}} N_{\mathcal{B}}} \|\mu - \nu\|_2 ,$$

where the norms in the right-hand side are respectively the ℓ^1 and ℓ^2–norms between probability distributions.

Proof Let ξ be an element of $\overline{m}(\mu)$; it can be written as

$$\xi = \sum_{a \in \mathcal{A}} \sum_{b \in \mathcal{B}} \mu_{a,b} \, \zeta_{a,b}$$

for some elements $\zeta_{a,b} \in \overline{m}(a,b)$. We consider

$$c = \sum_{a \in \mathcal{A}} \sum_{b \in \mathcal{B}} \nu_{a,b} \, \zeta_{a,b} ,$$

which is an element of $\overline{m}(\nu)$. Then by the triangle inequality,

$$\|\xi - c\|_2 = \left\| \sum_{a \in \mathcal{A}} \sum_{b \in \mathcal{B}} \big(\mu_{a,b} - \nu_{a,b}\big) \zeta_{a,b} \right\|_2 \leqslant \sum_{a \in \mathcal{A}} \sum_{b \in \mathcal{B}} \big|\mu_{a,b} - \nu_{a,b}\big| \, \|\zeta_{a,b}\|_2 \leqslant M \sum_{a \in \mathcal{A}} \sum_{b \in \mathcal{B}} \big|\mu_{a,b} - \nu_{a,b}\big| .$$

This entails the first claimed inequality. The second one follows from an application of the Cauchy-Schwarz inequality. ∎

Corollary 7 *If the set-valued function $\overline{m}$ is bounded in norm, then for all $\boldsymbol{y} \in \Delta(\mathcal{B})$, the mappings $D_{\boldsymbol{y}} : \Delta(A) \to \mathbb{R}$ defined, for all $\boldsymbol{x} \in \Delta(\mathcal{A})$, by*

$$D_{\boldsymbol{y}}(\boldsymbol{x}) = \sup_{\xi \in \overline{m}(\boldsymbol{x}, \boldsymbol{y})} \inf_{c \in \mathcal{C}} \|c - \xi\|_2$$

are continuous.

Proof We show that for all $\boldsymbol{x}, \boldsymbol{x}' \in \Delta(\mathcal{A})$, the condition $\|\boldsymbol{x}' - \boldsymbol{x}\|_1 \leqslant \varepsilon$ implies that $D_{\boldsymbol{y}}(\boldsymbol{x}) - D_{\boldsymbol{y}}(\boldsymbol{x}') \leqslant M\varepsilon$, where M is the bound in ℓ^2–norm over $\overline{m}$. Indeed, fix $\delta > 0$ and let $\xi_{\delta,\boldsymbol{x}} \in \overline{m}(\boldsymbol{x}, \boldsymbol{y})$ be such that

$$D_{\boldsymbol{y}}(\boldsymbol{x}) \leqslant \inf_{c \in \mathcal{C}} \big\| c - \xi_{\delta,\boldsymbol{x}} \big\|_2 + \delta . \tag{2}$$

By Lemma 6 (with the choices $\mu = \boldsymbol{x} \otimes \boldsymbol{y}$ and $\nu = \boldsymbol{x}' \otimes \boldsymbol{y}$),

$$\inf_{\xi' \in \overline{m}(\boldsymbol{x}', \boldsymbol{y})} \|\xi_{\delta,\boldsymbol{x}} - \xi'\| \leqslant M\varepsilon ,$$

and therefore, there exists $\xi_{\delta,\boldsymbol{x}'} \in \overline{m}(\boldsymbol{x}', \boldsymbol{y})$ such that $\big\| \xi_{\delta,\boldsymbol{x}} - \xi_{\delta,\boldsymbol{x}'} \big\|_2 \leqslant M\varepsilon + \delta$. The triangle inequality entails that

$$\inf_{c \in \mathcal{C}} \big\| c - \xi_{\delta,\boldsymbol{x}} \big\|_2 \leqslant \inf_{c \in \mathcal{C}} \big\| c - \xi_{\delta,\boldsymbol{x}'} \big\|_2 + M\varepsilon + \delta .$$

Substituting in (2), we get that

$$D_{\boldsymbol{y}}(\boldsymbol{x}) \leqslant M\varepsilon + 2\delta + \inf_{c\in\mathcal{C}} \left\| c - \xi_{\delta,\boldsymbol{x}'} \right\|_2 \leqslant M\varepsilon + 2\delta + D_{\boldsymbol{y}}(\boldsymbol{x}')\,,$$

which, letting $\delta \to 0$, proves our continuity claim. ■

3.1.3 Necessary and Sufficient Condition for Set-Valued Approachability

This condition will be referred to as (SVAC), an acronym that stands for "set-valued approachability condition."

Theorem 8 *Suppose that the set-valued function $\overline{m}$ is bounded in norm. A closed convex set $\mathcal{C} \subseteq \mathbb{R}^d$ is $\overline{m}$–approachable if and only if the following set-valued approachability condition is satisfied,*

$$\forall\, \boldsymbol{y} \in \Delta(\mathcal{B}),\quad \exists\, \boldsymbol{x} \in \Delta(\mathcal{A}), \qquad \overline{m}(\boldsymbol{x},\boldsymbol{y}) \subseteq \mathcal{C}\,. \tag{SVAC}$$

In this case, an $\overline{m}$–approaching strategy for $\mathcal{C}$ is an m–approaching strategy of $\widetilde{\mathcal{C}}$ defined below at (3) *and* (4)*. It satisfies, for all $T \geqslant 1$,*

$$\sup_{\xi\in\overline{m}(\nu_T)} \inf_{c\in\mathcal{C}} \|c-\xi\|_2 \leqslant 2M\sqrt{\frac{N_{\mathcal{A}}N_{\mathcal{B}}}{T}}\,,$$

where M is a bound in ℓ^2–norm on $\overline{m}$.

Proof [of the necessity of Condition (SVAC)] If the condition does not hold, then there exists $\boldsymbol{y}_0 \in \Delta(\mathcal{B})$ such that for every $\boldsymbol{x} \in \mathcal{A}$, the set $\overline{m}(\boldsymbol{x},\boldsymbol{y}_0)$ is not included in $\mathcal{C}$, i.e., it contains at least one point not in $\mathcal{C}$. We consider the mapping $D_{\boldsymbol{y}_0}$ defined in the statement of Corollary 7. Since $\mathcal{C}$ is closed, distances of given individual points to $\mathcal{C}$ are achieved; therefore, by the choice of $\boldsymbol{y}_0$, we get that $D_{\boldsymbol{y}_0}(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in \Delta(\mathcal{A})$. Now, since $D_{\boldsymbol{y}_0}$ is continuous on the compact set $\Delta(\mathcal{A})$, as asserted by the indicated corollary, it attains its minimum, whose value we denote by $D_{\min} > 0$.

Assume now that the second player chooses at each round $\boldsymbol{y}_t = \boldsymbol{y}_0$ as his mixed action. Then, denoting

$$\overline{\boldsymbol{x}}_T = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t\,,$$

we get that $\nu_T = \overline{\boldsymbol{x}}_T \otimes \boldsymbol{y}_0$, and hence, for all strategies of the first player and for all $T \geqslant 1$,

$$\sup_{\xi\in\overline{m}(\nu_T)} \inf_{c\in\mathcal{C}} \|c-\xi\|_2 = D_{\boldsymbol{y}_0}(\overline{\boldsymbol{x}}_T) \geqslant D_{\min} > 0\,,$$

which shows that $\mathcal{C}$ is not approachable. ■

We now prove in a constructive way, by exhibiting a suitable strategy (the one alluded at in the statement of the theorem), that (SVAC) is sufficient for set-valued approachability.

We identify probability distributions over $\mathcal{A} \times \mathcal{B}$ with vectors in $\mathbb{R}^{\mathcal{A}\times\mathcal{B}}$ and consider the vector-valued payoff function

$$m : (a,b) \in \mathcal{A} \times \mathcal{B} \longmapsto \delta_{(a,b)} \in \mathbb{R}^{\mathcal{A}\times\mathcal{B}}, \tag{3}$$

where $\delta_{(a,b)}$ is the point mass on (a,b). We extend m to $\Delta(\mathcal{A}) \times \Delta(\mathcal{B})$ in a multi-linear fashion. The target set will be

$$\widetilde{\mathcal{C}} = \left\{ \mu \in \Delta(\mathcal{A} \times \mathcal{B}) : \quad \overline{m}(\mu) \subseteq \mathcal{C} \right\}. \tag{4}$$

The linearity of the function $\overline{m}$ on $\Delta(\mathcal{A} \times \mathcal{B})$ entails that if $\mathcal{C}$ is a convex set (respectively, a closed set, or a polyhedron), then $\widetilde{\mathcal{C}}$ is a convex set as well (respectively, a closed set, or a polyhedron). In the case where $\widetilde{\mathcal{C}}$ is a polyhedron, it is actually a polytope (that is, a compact polyhedron).

We then consider the m–approaching strategy of $\widetilde{\mathcal{C}}$ described in (1) and now prove that it enjoys the convergence guarantees stated in Theorem 8.

Lemma 9 *Condition* (SVAC) *is equivalent to the m–approachability of $\widetilde{\mathcal{C}}$.*

Proof Since $\mathcal{C}$ and thus $\widetilde{\mathcal{C}}$ are closed and convex sets, we can resort to Theorem 2. The latter states that the m–approachability of $\widetilde{\mathcal{C}}$ is equivalent to the fact that for all $\boldsymbol{y} \in \Delta(\mathcal{B})$, there exists some $\boldsymbol{x} \in \Delta(\mathcal{A})$ such that $\mu = m(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x} \otimes \boldsymbol{y}$, the product-distribution between $\boldsymbol{x}$ and $\boldsymbol{y}$, belongs to $\widetilde{\mathcal{C}}$, i.e., satisfies $\overline{m}(\mu) = \overline{m}(\boldsymbol{x}, \boldsymbol{y}) \subseteq \mathcal{C}$. ■

The definition (3) of m entails the rewriting

$$\nu_T = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{x}_t \otimes \boldsymbol{y}_t = \frac{1}{T} \sum_{t=1}^{T} m(\boldsymbol{x}_t, \boldsymbol{y}_t) \,.$$

Let $P_{\widetilde{\mathcal{C}}}$ denote the projection operator onto $\widetilde{\mathcal{C}}$; the quantities at hand in the definition of m–approachability of $\widetilde{\mathcal{C}}$ are given by

$$\varepsilon_T = \left\| \nu_T - P_{\widetilde{\mathcal{C}}}(\nu_T) \right\|_2 = \inf_{\mu \in \widetilde{\mathcal{C}}} \left\| \nu_T - \mu \right\|_2 .$$

We now relate them to the ones arising in the definition of $\overline{m}$–approachability of $\mathcal{C}$.

Lemma 10 *The following upper bound holds,*

$$\sup_{\xi \in \overline{m}(\nu_T)} \inf_{c \in \mathcal{C}} \left\| c - \xi \right\|_2 \leqslant M \sqrt{N_{\mathcal{A}} N_{\mathcal{B}}} \; \varepsilon_T \,.$$

Proof Lemma 6 entails that the sets $\overline{m}(\nu_T)$ are included in $M\sqrt{N_{\mathcal{A}}N_{\mathcal{B}}}\,\varepsilon_T$–neighborhoods of $\overline{m}\big(P_{\widetilde{\mathcal{C}}}(\nu_T)\big)$. Since by definition of $\widetilde{\mathcal{C}}$, one has $\overline{m}\big(P_{\widetilde{\mathcal{C}}}(\nu_T)\big) \subseteq \mathcal{C}$, we get in particular that the sets $\overline{m}(\nu_T)$ are included in $M\sqrt{N_{\mathcal{A}}N_{\mathcal{B}}}\,\varepsilon_T$–neighborhoods of $\mathcal{C}$, which is exactly the statement of the lemma. ■

Proof [**of the sufficiency of Condition** (SVAC)] First, Lemma 9 and Condition (SVAC) entail, via Theorem 3, that the considered strategy m–approaches $\widetilde{\mathcal{C}}$, at the following rate: $\varepsilon_T \leqslant 2/\sqrt{T}$, with probability 1. Second, Lemma 10 indicates that this strategy also $\overline{m}$–approaches $\mathcal{C}$, at the stated rate of $2M\sqrt{N_{\mathcal{A}}N_{\mathcal{B}}/T}$, with probability 1. ■

3.1.4 Remarks: on Efficiency; on Full versus Bandit Monitorings

Note that, as explained around (1), the considered strategy for m–approaching $\widetilde{\mathcal{C}}$, or equivalently $\overline{m}$–approaching $\mathcal{C}$, is efficient as long as projections in ℓ^2–norm onto the set $\widetilde{\mathcal{C}}$ defined in (4) can be computed efficiently. The latter depends on the respective geometries of $\overline{m}$ and $\mathcal{C}$. We will provide examples of favorable cases (see, e.g., Section 6.1 about minimization of external regret under partial monitoring). In the sequel the notion of "efficiency up to a projection oracle" will refer to this efficiency depending solely on the efficient computation of the needed projections.

The proposed strategy does not require full monitoring, although it seems to rely on the observation of the pair of played mixed actions $m(\boldsymbol{x}_t, \boldsymbol{y}_t)$. With bandit monitoring, only the played sets $\overline{m}(\boldsymbol{x}_t, \boldsymbol{y}_t)$ would be available, not the $\boldsymbol{y}_t$ themselves; in that case, the player can act as if the other player chose any $\boldsymbol{y}'_t$ that generates this set, i.e., such that $\overline{m}(\boldsymbol{x}_t, \boldsymbol{y}'_t) = \overline{m}(\boldsymbol{x}_t, \boldsymbol{y}_t)$.

3.2 Pure Actions Taken and Observed

It is well-known that the basic results recalled in Section 2 extend to the case of pure actions. We briefly explain here how the developed theory of set-valued approachability for games with mixed actions extends as well to the case of pure actions taken (still under full monitoring).

The game goes as follows. At each round t, the players choose simultaneously respective pure actions $A_t \in \mathcal{A}$ and $B_t \in \mathcal{B}$, possibly at random according to distributions $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$. As a result, the first player gets the subset $\overline{m}(A_t, B_t)$ as a payoff and observes B_t. Strategies for the players now associate with $A_1, \ldots, A_{t-1}$ and $B_1, \ldots, B_{t-1}$ mixed actions $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$, according to which A_t and B_t are drawn independently.

We are interested in the behavior of

$$\frac{1}{T}\sum_{t=1}^{T} \overline{m}(A_t, B_t) = \overline{m}(\pi_T)\,, \qquad \text{where} \qquad \pi_T := \frac{1}{T}\sum_{t=1}^{T} \delta_{(A_t, B_t)}$$

is the empirical distribution of the pairs (A_t, B_t) of actions taken during the first T rounds. The definition of set-valued approachability extends as follows.

Definition 11 *A set $\mathcal{C} \subseteq \mathbb{R}^d$ is $\overline{m}$–approachable by the first player with pure actions if he has a strategy such that, for all $\varepsilon > 0$, there exists an integer T_ε such that for all strategies of the second player,*

$$\mathbb{P}\left\{\forall\, T \geqslant T_\varepsilon, \quad \sup_{\xi \in \overline{m}(\pi_T)} \inf_{c \in \mathcal{C}} \|c - \xi\|_2 \leqslant \varepsilon\right\} \geqslant 1 - \varepsilon\,.$$

The fact that Condition (SVAC) is still a necessary and sufficient condition for $\overline{m}$–approachability with pure actions of a closed convex set $\mathcal{C} \subseteq \mathbb{R}^d$ (where $\overline{m}$ is bounded) can be seen as follows.

Concerning the proof of the sufficiency of this condition, first recall that Lemma 9 indicates that Condition (SVAC) is equivalent to the m–approachability of $\widetilde{\mathcal{C}}$. In view of a version of Theorem 3 for pure actions (e.g., Theorem II.4.3 of Mertens et al., 1994) the strategy described around (3) and (4), with the replacement of the ν_T by the π_T and extra random draws of the pure actions A_t according to the mixed distributions $\boldsymbol{x}_t$ thus computed, is such that the quantities

$$\varepsilon'_T = \left\| \pi_T - P_{\widetilde{\mathcal{C}}}(\pi_T) \right\|_2 = \inf_{\mu \in \widetilde{\mathcal{C}}} \|\pi_T - \mu\|_2$$

satisfy the following convergence guarantees. For all $\delta \in (0,1)$, there exists an integer T_δ such that for all strategies of the second player,

$$\mathbb{P}\big\{\forall\, T \geqslant T_\delta, \quad \varepsilon'_T \leqslant \delta\big\} \geqslant 1 - \delta\,.$$

This shows that this strategy also $\overline{m}$–approaches $\mathcal{C}$ since Lemma 10 is valid with the respective replacements of ν_T and ε_T by π_T and ε'_T.

The proof of the necessity of the condition is the same as for mixed actions taken, with the addition of a concentration argument. Indeed, by martingale convergence (e.g., repeated uses of the Hoeffding-Azuma inequality together with an application of the Borel-Cantelli lemma), $\delta_T = \|\pi_T - \nu_T\|_1$ converges to zero almost surely as T goes to infinity. By applying Lemma 6 and by using the notation of the proof of Theorem 8, we get

$$\sup_{\xi \in \overline{m}(\pi_T)} \inf_{c \in \mathcal{C}} \|c - \xi\|_2 \geqslant \sup_{\xi \in \overline{m}(\nu_T)} \inf_{c \in \mathcal{C}} \|c - \xi\|_2 - M\delta_T \geqslant D_{\min} - M\delta_T\,,$$

and we simply take the lim inf in the above inequalities to conclude the argument.

4. Set-Valued Approachability for Concave–Convex Set-Valued Games

We consider in this section the same setting of mixed actions taken and observed as in Section 3.1, that is, we deal with set-valued payoff functions $\overline{m} : \Delta(\mathcal{A}) \times \Delta(\mathcal{B}) \to \mathcal{S}\big(\mathbb{R}^d\big)$ under full monitoring. However, in the previous section $\overline{m}$ was linear on $\Delta(\mathcal{A}) \times \Delta(\mathcal{B})$, an assumption that we now weaken while still having that (SVAC) is the necessary and sufficient condition for set-valued approachability. The price to pay for this is the loss of the exhibited efficiency (up to a projection oracle) of the approaching strategies and an inferior convergence rate.

Formally, the functions $\overline{m} : \Delta(\mathcal{A}) \times \Delta(\mathcal{B}) \to \mathcal{S}\big(\mathbb{R}^d\big)$ that we will consider will satisfy one or several of the following properties.

Definition 12 *The set-valued function $\overline{m} : \Delta(\mathcal{A}) \times \Delta(\mathcal{B}) \to \mathcal{S}\big(\mathbb{R}^d\big)$ is bounded in ℓ^2–norm by M if*

$$\forall (\boldsymbol{x}, \boldsymbol{y}) \in \Delta(\mathcal{A}) \times \Delta(\mathcal{B}), \qquad \sup_{\xi \in \overline{m}(\boldsymbol{x}, \boldsymbol{y})} \|\xi\|_2 \leqslant M\,.$$

Definition 13 *A function $\overline{m} : \Delta(\mathcal{A}) \times \Delta(\mathcal{B}) \to \mathcal{S}\big(\mathbb{R}^d\big)$ is uniformly continuous in its first argument if for every $\varepsilon > 0$, there exists $\eta > 0$ such that for all $\boldsymbol{x}, \boldsymbol{x}' \in \Delta(\mathcal{A})$ satisfying $\|\boldsymbol{x} - \boldsymbol{x}'\|_1 \leqslant \eta$ and for all $\boldsymbol{y} \in \Delta(\mathcal{B})$, the set $\overline{m}(\boldsymbol{x}', \boldsymbol{y})$ is included in an ε–neighborhood of $\overline{m}(\boldsymbol{x}, \boldsymbol{y})$ in the Euclidean norm. Put differently,*

$$\sup_{\xi \in \overline{m}(\boldsymbol{x}', \boldsymbol{y})} \inf_{c \in \overline{m}(\boldsymbol{x}, \boldsymbol{y})} \|\xi - c\|_2 \leqslant \varepsilon \qquad \text{or} \qquad \overline{m}(\boldsymbol{x}', \boldsymbol{y}) \subseteq \overline{m}(\boldsymbol{x}, \boldsymbol{y}) + \varepsilon \boldsymbol{B}\,,$$

where $\boldsymbol{B}$ is the unit Euclidean ball in $\mathbb{R}^d$.

Uniform continuity in the second argument is defined symmetrically.

Definition 14 *A function $\overline{m} : \Delta(\mathcal{A}) \times \Delta(\mathcal{B}) \to \mathcal{S}\big(\mathbb{R}^d\big)$ is concave in its first argument if for all $\boldsymbol{x}, \boldsymbol{x}' \in \Delta(\mathcal{A})$, all $\boldsymbol{y} \in \Delta(\mathcal{B})$, and all $\alpha \in [0, 1]$,*

$$\overline{m}\big(\alpha \boldsymbol{x} + (1 - \alpha)\boldsymbol{x}',\, \boldsymbol{y}\big) \subseteq \alpha\, \overline{m}(\boldsymbol{x}, \boldsymbol{y}) + (1 - \alpha)\, \overline{m}(\boldsymbol{x}', \boldsymbol{y})\,.$$

A function $\overline{m} : \Delta(\mathcal{A}) \times \Delta(\mathcal{B}) \to \mathcal{S}\big(\mathbb{R}^d\big)$ is convex in its second argument if for all $\boldsymbol{x} \in \Delta(\mathcal{A})$, all $\boldsymbol{y}, \boldsymbol{y}' \in \Delta(\mathcal{B})$, and all $\alpha \in [0, 1]$,

$$\alpha\, \overline{m}(\boldsymbol{x}, \boldsymbol{y}) + (1 - \alpha)\, \overline{m}(\boldsymbol{x}, \boldsymbol{y}') \subseteq \overline{m}\big(\boldsymbol{x},\, \alpha \boldsymbol{y} + (1 - \alpha)\boldsymbol{y}'\big)\,.$$

An example of such a concave–convex function $\overline{m}$ is discussed in Lemma 17.

The following theorem indicates that (SVAC) is the necessary and sufficient condition for the $\overline{m}$–approachability of a closed convex set $\mathcal{C}$ when the payoff function $\overline{m}$ satisfies all four properties stated in Definitions 13 and 14. (Boundedness of $\overline{m}$ indeed follows from the continuity of $\overline{m}$ in each variable.)

Theorem 15 *If $\overline{m}$ is bounded, convex, and uniformly continuous in its second argument, then* (SVAC) *entails that a closed convex set $\mathcal{C}$ is $\overline{m}$–approachable.*

If $\overline{m}$ is concave and uniformly continuous in its first argument, then a closed convex set $\mathcal{C}$ is $\overline{m}$–approachable only if (SVAC) *is satisfied.*

The proof of the necessity statement follows closely the arguments used in the proof of Theorem 8. The sufficiency statement relies on the use of what is called a calibrated strategy, where we define calibration in a (slightly) stronger way than Foster and Vohra (1998) did. All the details, including the definition of the stronger notion of calibration and the construction of an algorithm controlling it, can be found in Appendix A.

5. Approachability in Games with Partial Monitoring

A repeated vector-valued game with partial monitoring is described as follows (see, e.g., Mertens et al., 1994, Rustichini, 1999, and references therein). The players have respective finite action sets $\mathcal{I}$ and $\mathcal{J}$. We denote by $r : \mathcal{I} \times \mathcal{J} \to \mathbb{R}^d$ the vector-valued payoff function of the first player and extend it multi-linearly to $\Delta(\mathcal{I}) \times \Delta(\mathcal{J})$. At each round, players simultaneously choose their actions $I_t \in \mathcal{I}$ and $J_t \in \mathcal{J}$, possibly at random according to probability distributions denoted by $\boldsymbol{p}_t \in \Delta(\mathcal{I})$ and $\boldsymbol{q}_t \in \Delta(\mathcal{J})$. At the end of a round, the

first player does not observe J_t nor $r(I_t, J_t)$ but only receives a signal. There is a finite set $\mathcal{H}$ of possible signals; the feedback S_t that is given to the first player is drawn at random according to the distribution $H(I_t, J_t)$, where the mapping $H : \mathcal{I} \times \mathcal{J} \to \Delta(\mathcal{H})$ is known by the first player. We will refer to H as the signaling structure.

Formally, strategies of the first player now associate with $I_1, \ldots, I_{t-1}$ and $S_1, \ldots, S_{t-1}$ a mixed action $\boldsymbol{p}_t \in \Delta(\mathcal{I})$, according to which I_t is drawn independently. We do not impose any restriction on the opponent player, who enjoys a full monitoring: strategies of his associate with $I_1, \ldots, I_{t-1}$, with $J_1, \ldots, J_{t-1}$ and with $S_1, \ldots, S_{t-1}$ a mixed action $\boldsymbol{q}_t \in \Delta(\mathcal{I})$, according to which J_t is drawn independently.

Example 1 *Examples of such partial monitoring games are provided by, e.g., Cesa-Bianchi et al. (2006), among which we can cite the apple tasting problem, the label-efficient prediction constraint, and the multi-armed bandit settings.*

Some additional notation will be useful. We denote by R a bound on the norm of (the linear extension of) r,

$$R = \max_{(i,j) \in \mathcal{I} \times \mathcal{J}} \big\| r(i,j) \big\|_2 \,. \tag{5}$$

The cardinalities of the finite sets $\mathcal{I}$, $\mathcal{J}$, and $\mathcal{H}$ will be referred to as $N_{\mathcal{I}}$, $N_{\mathcal{J}}$, and $N_{\mathcal{H}}$.

The definition of approachability can be extended from the setting of full information to the setting of partial monitoring as follows. The only new ingredient is the signaling structure H, the aim is unchanged.

Definition 16 *Let $\mathcal{C} \subseteq \mathbb{R}^d$ be some set; $\mathcal{C}$ is r–approachable by the first player for the signaling structure H if he has a strategy such that, for all $\varepsilon > 0$, there exists an integer T_ε such that for all strategies of the second player,*

$$\mathbb{P}\left\{ \forall\, T \geqslant T_\varepsilon, \quad \inf_{c \in \mathcal{C}} \left\| c - \frac{1}{T} \sum_{t=1}^{T} r(I_t, J_t) \right\|_2 \leqslant \varepsilon \right\} \geqslant 1 - \varepsilon \,.$$

In particular, the first player has a strategy that ensures that the sequence of his average vector-valued payoffs converges almost surely to the set $\mathcal{C}$ (uniformly with respect to the strategies of the second player), even if he only observes the random signals S_t as a feedback.

Here again, more precise approachability guarantees than the ones required by the definition will be obtained. Indeed, Corollary 27 exhibits bounds of the following form, for a suitable strategy of the first player. For all strategies of the second player and for all $T \geqslant 1$, with probability at least $1 - P_T$,

$$\sup_{\tau \geqslant T} \inf_{c \in \mathcal{C}} \left\| c - \frac{1}{\tau} \sum_{t=1}^{\tau} r(I_t, J_t) \right\|_2 \leqslant R_T \,,$$

where $P_T = O(1/T)$ and $R_T = O\big(T^{-1/5} \ln T\big)$.

Our contributions to approachability in games with partial monitoring: A necessary and sufficient condition for r–approachability with the signaling structure H was already stated

and proved by Perchet (2011a), together with an approaching strategy. We therefore need to detail where our contribution lies.

First, our strategy is efficient (as long as some projection operator can be computed efficiently, e.g., in the case when the target set is a polytope, see Sections 5.4.1–5.4.2 as well as in the cases of external and internal regret minimization described below in Section 6). In contrast, the one of Perchet (2011a) relies on auxiliary strategies that are calibrated and that require a grid that is progressively refined (leading to a step complexity that is prohibitive in the number T of past steps and to rates of convergence that become dependent on the dimension). The latter construction is in essence the one used in Section 4.

Second, we are able, for the first time, to exhibit convergence rates that are independent of the dimension (as is the case with full monitoring). A somehow related result appeared in Perchet (2011b), but only for the special case of regret minimization. The proof techniques used therein are involved and hold only for regret minimization, not for general approachability.

Third, as far as elegance is concerned, our proof of the sufficiency of the condition for r–approachability with the signaling structure H is short, compact, and more direct than the one of Perchet (2011a) or even of Perchet (2011b), which relied on several layers of concepts (for example, calibration or internal regret in games with partial monitoring).

5.1 Statement of the Necessary and Sufficient Condition for Approachability

To recall the mentioned approachability condition of Perchet (2011a) we need some additional notation. For all $\boldsymbol{q} \in \Delta(\mathcal{J})$, we denote by $\widetilde{H}(\boldsymbol{q})$ the element in $\Delta(\mathcal{H})^{\mathcal{I}}$ defined as follows. For all $i \in \mathcal{I}$, its i–th component is given by the convex combination of probability distributions over $\mathcal{H}$

$$\widetilde{H}(\boldsymbol{q})_i = H(i, \boldsymbol{q}) = \sum_{j \in \mathcal{J}} q_j H(i,j)\,.$$

Also, we denote by $\mathcal{F}$ the convex set of feasible vectors of probability distributions over $\mathcal{H}$:

$$\mathcal{F} = \left\{ \widetilde{H}(\boldsymbol{q}) : \quad \boldsymbol{q} \in \Delta(\mathcal{J}) \right\}.$$

A generic element of $\mathcal{F}$ will be denoted by $\sigma \in \mathcal{F}$ and we define the set-valued function $\overline{m}$, for all $\boldsymbol{p} \in \Delta(\mathcal{I})$ and $\sigma \in \mathcal{F}$, by

$$\overline{m}(\boldsymbol{p}, \sigma) = \left\{ r(\boldsymbol{p}, \boldsymbol{q}_{\text{eqv}}) : \quad \boldsymbol{q}_{\text{eqv}} \in \Delta(\mathcal{J}) \text{ such that } \widetilde{H}(\boldsymbol{q}_{\text{eqv}}) = \sigma \right\}.$$

We use in $\boldsymbol{q}_{\text{eqv}}$ the subscript "eqv" (standing for "equivalent") as all considered $\boldsymbol{q}_{\text{eqv}}$ vectors induce the same distributions of signals σ and are thus equivalent from the monitoring perspective.

The necessary and sufficient condition exhibited by Perchet (2011a) for the r–approachability of $\mathcal{C}$ with the signaling structure H can now be recalled. In the sequel we will refer to this condition as Condition (APM), an acronym that stands for "approachability with partial monitoring."

Condition 1 [referred to as Condition (APM)] *The signaling structure H, the vector-payoff function r, and the set $\mathcal{C}$ satisfy*

$$\forall\, \boldsymbol{q} \in \Delta(\mathcal{J}), \quad \exists\, \boldsymbol{p} \in \Delta(\mathcal{I}), \quad \forall\, \boldsymbol{q}' \in \Delta(\mathcal{J}), \qquad \widetilde{H}(\boldsymbol{q}) = \widetilde{H}(\boldsymbol{q}') \quad \Rightarrow \quad r(\boldsymbol{p}, \boldsymbol{q}') \in \mathcal{C}\,.$$

The condition can be equivalently reformulated as

$$\forall\, \sigma \in \mathcal{F}, \quad \exists\, \boldsymbol{p} \in \Delta(\mathcal{I}), \qquad \overline{m}(\boldsymbol{p}, \sigma) \subseteq \mathcal{C}\,. \tag{APM}$$

The subsequent sections show (in a constructive way, with a strategy efficient up to a projection oracle) that Condition (APM) is sufficient for r–approachability of closed convex sets $\mathcal{C}$ given the signaling structure H. That this condition is necessary was already proved in Section 3.1 of Perchet (2011a).

5.2 Links with Set-Valued Approachability

As will become clear in the proof of Theorem 24, the key in our problem will be to ensure the set-valued approachability of $\mathcal{C}$ with the following non-linear set-valued payoff function, that is however concave–convex in the sense of Definition 14.

Lemma 17 *The function*

$$(\boldsymbol{p}, \boldsymbol{q}) \in \Delta(\mathcal{I}) \times \Delta(\mathcal{J}) \;\longmapsto\; \overline{m}\Big(\boldsymbol{p},\, \widetilde{H}(\boldsymbol{q})\Big)$$

is concave in its first argument and convex in its second argument.

Proof For the concavity part, we consider some pair $\boldsymbol{p}, \boldsymbol{p}' \in \Delta(\mathcal{I})$, some $\boldsymbol{q} \in \Delta(\mathcal{J})$ and some $\alpha \in [0,1]$. By the linearity of r, the elements of the set of interest can be written as

$$\begin{aligned}
&\overline{m}\Big(\alpha\boldsymbol{p} + (1-\alpha)\boldsymbol{p}',\, \widetilde{H}(\boldsymbol{q})\Big) \\
&= \Big\{\alpha\, r(\boldsymbol{p}, \boldsymbol{q}_{\mathrm{eqv}}) + (1-\alpha) r(\boldsymbol{p}', \boldsymbol{q}_{\mathrm{eqv}}) : \quad \boldsymbol{q}_{\mathrm{eqv}} \in \Delta(\mathcal{J}) \text{ such that } \widetilde{H}(\boldsymbol{q}_{\mathrm{eqv}}) = \widetilde{H}(\boldsymbol{q})\Big\}\,.
\end{aligned}$$

This set is therefore indeed included in (but in general, not equal to)

$$\begin{aligned}
&\alpha\, \overline{m}\Big(\boldsymbol{p},\, \widetilde{H}(\boldsymbol{q})\Big) + (1-\alpha)\, \alpha\, \overline{m}\Big(\boldsymbol{p},\, \widetilde{H}(\boldsymbol{q})\Big) \\
= \quad & \alpha \Big\{r(\boldsymbol{p}, \boldsymbol{q}_{\mathrm{eqv}}) : \quad \boldsymbol{q}_{\mathrm{eqv}} \in \Delta(\mathcal{J}) \text{ such that } \widetilde{H}(\boldsymbol{q}_{\mathrm{eqv}}) = \widetilde{H}(\boldsymbol{q})\Big\} \\
& + (1-\alpha) \Big\{r(\boldsymbol{p}, \boldsymbol{q}'_{\mathrm{eqv}}) : \quad \boldsymbol{q}'_{\mathrm{eqv}} \in \Delta(\mathcal{J}) \text{ such that } \widetilde{H}(\boldsymbol{q}'_{\mathrm{eqv}}) = \widetilde{H}(\boldsymbol{q})\Big\}\,.
\end{aligned}$$

Similarly, for the convexity part, we consider some pair $\boldsymbol{q}, \boldsymbol{q}' \in \Delta(\mathcal{J})$, some $\boldsymbol{p} \in \Delta(\mathcal{I})$ and some $\alpha \in [0,1]$. Elements of the convex combination of sets

$$\alpha\, \overline{m}\Big(\boldsymbol{p},\, \widetilde{H}\big(\boldsymbol{q}\big)\Big) + (1-\alpha)\, \overline{m}\Big(\boldsymbol{p},\, \widetilde{H}\big(\boldsymbol{q}'\big)\Big)$$

are of the form

$$\alpha\, r(\boldsymbol{p}, \boldsymbol{q}_{\mathrm{eqv}}) + (1-\alpha)\, r(\boldsymbol{p}, \boldsymbol{q}'_{\mathrm{eqv}}) = r\big(\boldsymbol{p},\, \alpha\boldsymbol{q}_{\mathrm{eqv}} + (1-\alpha)\boldsymbol{q}'_{\mathrm{eqv}}\big)\,,$$

where $\boldsymbol{q}_{\mathrm{eqv}}$ and $\boldsymbol{q}'_{\mathrm{eqv}}$ are such that

$$\widetilde{H}(\boldsymbol{q}_{\mathrm{eqv}}) = \widetilde{H}(\boldsymbol{q}) \qquad \text{and} \qquad \widetilde{H}(\boldsymbol{q}'_{\mathrm{eqv}}) = \widetilde{H}(\boldsymbol{q}')\,. \tag{6}$$

In particular, by linearity of $\widetilde{H}$, we have

$$\widetilde{H}\big(\alpha \boldsymbol{q}_{\mathrm{eqv}} + (1-\alpha)\boldsymbol{q}'_{\mathrm{eqv}}\big) = \widetilde{H}\big(\alpha \boldsymbol{q} + (1-\alpha)\boldsymbol{q}'\big)\,,$$

which shows that

$$r\big(\boldsymbol{p},\, \alpha \boldsymbol{q}_{\mathrm{eqv}} + (1-\alpha)\boldsymbol{q}'_{\mathrm{eqv}}\big) \in \overline{m}\Big(\boldsymbol{p},\, \widetilde{H}\big(\alpha \boldsymbol{q} + (1-\alpha)\boldsymbol{q}'\big)\Big)\,.$$

The desired inclusion

$$\alpha\, \overline{m}\Big(\boldsymbol{p},\, \widetilde{H}\big(\boldsymbol{q}\big)\Big) + (1-\alpha)\, \overline{m}\Big(\boldsymbol{p},\, \widetilde{H}\big(\boldsymbol{q}'\big)\Big) \subseteq \overline{m}\Big(\boldsymbol{p},\, \widetilde{H}\big(\alpha \boldsymbol{q} + (1-\alpha)\boldsymbol{q}'\big)\Big)$$

follows. Note that this inclusion is not an equality in general, as it cannot be guaranteed that any $\boldsymbol{q}''_{\mathrm{eqv}}$ such that

$$\widetilde{H}\big(\alpha \boldsymbol{q} + (1-\alpha)\boldsymbol{q}'\big) = \widetilde{H}(\boldsymbol{q}''_{\mathrm{eqv}})$$

can be decomposed under the form $\alpha \boldsymbol{q}_{\mathrm{eqv}} + (1-\alpha)\boldsymbol{q}'_{\mathrm{eqv}}$, where $\boldsymbol{q}_{\mathrm{eqv}}$ and $\boldsymbol{q}'_{\mathrm{eqv}}$ satisfy (6). ■

Unfortunately, efficient strategies for set-valued approachability were only proposed in the linear case (Section 3), not in the concave–convex case (Section 4), and the proof of Lemma 17 shows that linearity cannot be guaranteed per se. However, we illustrate in the next example (and provide a general theory in the next section) how working in lifted spaces can lead to linearity and hence to efficiency.

Example 2 *We consider a game in which the second player (the column player) can force the first player (the row player) to play a game of matching pennies in the dark by choosing actions L or M. More formally, in the matrix below, the real numbers denote the payoff while ♣ and ♡ denote the two possible signals. The respective sets of actions are $\mathcal{I} = \{T,\, B\}$ and $\mathcal{J} = \{L,\, M,\, R\}$.*

	L	M	R
T	1 / ♣	-1 / ♣	2 / ♡
B	-1 / ♣	1 / ♣	3 / ♡

In this example we only study the mapping $\boldsymbol{p} \mapsto \overline{m}(\boldsymbol{p}, ♣)$ and show that it is piecewise linear on $\Delta(\mathcal{I})$, thus, is induced by a linear mapping defined on a lifted space.

We introduce a set $\mathcal{A} = \{\boldsymbol{p}_T,\, \boldsymbol{p}_B,\, \boldsymbol{p}_{1/2}\}$ of possibly mixed actions extending the set $\mathcal{I} = \{T,\, B\}$ of pure actions; the set $\mathcal{A}$ is composed of

$$\boldsymbol{p}_T = \delta_T, \quad \boldsymbol{p}_B = \delta_B, \quad \text{and} \quad \boldsymbol{p}_{1/2} = \frac{1}{2}\delta_T + \frac{1}{2}\delta_B\,.$$

Each mixed action in $\Delta(\mathcal{I})$ can be uniquely written as $\boldsymbol{p}_\lambda = \lambda\, \delta_B + (1-\lambda)\, \delta_T$ for some $\lambda \in [0,1]$. Now, for $\lambda \geqslant 1/2$, first,

$$\boldsymbol{p}_\lambda = (2\lambda - 1)\, \delta_B + \big(1 - (2\lambda - 1)\big)\, \boldsymbol{p}_{1/2}\,;$$

second, by definition of $\overline{m}$,

$$\overline{m}\big(\boldsymbol{p}_\lambda, \clubsuit\big) = [1-2\lambda,\, 2\lambda-1]\,;$$

since in particular $\overline{m}\big(\boldsymbol{p}_{1/2}, \clubsuit\big) = \{0\}$ and $\overline{m}(\delta_B, \clubsuit) = [-1, 1]$, we have the convex decomposition

$$\overline{m}\big(\boldsymbol{p}_\lambda, \clubsuit\big) = (2\lambda-1)\,\overline{m}(\delta_B, \clubsuit) + \big(1-(2\lambda-1)\big)\,\overline{m}(\boldsymbol{p}_{1/2}, \clubsuit)\,,$$

that can be restated as

$$\overline{m}\Big((2\lambda-1)\,\delta_B + \big(1-(2\lambda-1)\big)\,\boldsymbol{p}_{1/2}, \clubsuit\Big) = (2\lambda-1)\,\overline{m}(\delta_B, \clubsuit) + \big(1-(2\lambda-1)\big)\,\overline{m}(\boldsymbol{p}_{1/2}, \clubsuit)\,.$$

That is, $\overline{m}(\,\cdot\,, \clubsuit)$ is linear on the subset of $\Delta(\mathcal{I})$ corresponding to mixed actions $\boldsymbol{p}_\lambda$ with $\lambda \geqslant 1/2$.

A similar property holds on the subset of distributions with $\lambda \leqslant 1/2$, so that we proved that $\overline{m}(\,\cdot\,, \clubsuit)$ is piecewise linear on $\Delta(\mathcal{I})$.

The linearity on a lifted space comes from the following observation: $\overline{m}$ is induced by the linear extension to $\Delta(\mathcal{A})$ of the restriction of $\overline{m}$ to $\mathcal{A}$ (see Definition 21 for a more formal statement).

5.3 A Particular Class of Games, Encompassing Regret Minimization

In this section we consider the case where the payoff function and the signaling structure have some special properties described below (linked to linearity properties on lifted spaces and called "bi-piecewise linearity") and that can be exploited to get efficient strategies. The case of general games with partial monitoring is then considered in Section 5.4 but the particular class of games considered here is already rich enough to encompass the minimization of external and internal regret, as will be seen in Section 6.

To define bi-piecewise linearity of a game with partial monitoring, we start from a technical lemma that shows that $\overline{m}(\boldsymbol{p}, \sigma)$ can be written as a *finite* convex combination of sets of the form $\overline{m}(\boldsymbol{p}, b)$, where b belongs to some finite set $\mathcal{B} \subseteq \mathcal{F}$ that depends on the game. Under the additional assumption of the so-called piecewise linearity of the thus-defined mappings $\overline{m}(\,\cdot\,, b)$, we then describe an efficient strategy for approachability (up to a projection oracle) followed by convergence rate guarantees.

Definition 18 *Let P be a polytope and let $\mathcal{X}$ be a convex set. A mapping $f : P \to \mathcal{X}$ is piecewise linear if f is continuous and*

- *there exist finitely many sub-polytopes $P_1, \dots, P_K$ covering P and such that two different sub-polytopes P_k, $P_{k'}$ have an intersection with empty interior; we call these sub-polytopes a decomposition of P;*
- *f is linear on each sub-polytope P_k.*

5.3.1 Bi-Piecewise Linearity of a Game — a Preliminary Technical Result

Lemma 19 *For any game with partial monitoring, there exists a finite set $\mathcal{B} \subset \mathcal{F}$ and a piecewise-linear (injective) mapping $\Phi : \mathcal{F} \to \Delta(\mathcal{B})$ such that*

$$\forall\, \sigma \in \mathcal{F}, \quad \forall\, \boldsymbol{p} \in \Delta(\mathcal{I}), \qquad \overline{m}(\boldsymbol{p}, \sigma) = \sum_{b \in \mathcal{B}} \Phi_b(\sigma)\, \overline{m}(\boldsymbol{p}, b)\,,$$

where we denoted the convex weight vector $\Phi(\sigma) \in \Delta(\mathcal{B})$ by $\big(\Phi_b(\sigma)\big)_{b \in \mathcal{B}}$.

Proof $\widetilde{H}$ is linear on the polytope $\Delta(\mathcal{J})$; Proposition 2.4 in Rambau and Ziegler (1996) thus implies that its inverse mapping $\widetilde{H}^{-1}$ is a piecewise linear mapping of $\mathcal{F}$ into the set of the subsets of $\Delta(\mathcal{J})$. (Note that the latter set has a structure of a convex set, see Footnote 1.) This means by definition that there exists a finite decomposition of $\mathcal{F}$ into polytopes $P_1, \ldots, P_K$ each on which $\widetilde{H}^{-1}$ is linear. Up to a triangulation (see, e.g., Goodman and O'Rourke, 2004, Chapter 14), we can assume that each P_k is a simplex. Denote by $\mathcal{B}_k \subseteq \mathcal{F}$ the set of vertices of P_k; then, the finite subset stated in the lemma is

$$\mathcal{B} = \bigcup_{k=1}^{K} \mathcal{B}_k\,,$$

the set of all vertices of all the simplices.

Fix any $\sigma \in \mathcal{F}$. It belongs to some simplex P_k, so that there exists a convex decomposition $\sigma = \sum_{b \in \mathcal{B}_k} \lambda_b\, b$; this decomposition is unique within the simplex P_k. If σ belongs to two different simplices, then it actually belongs to their common face and the two possible decompositions coincide (some coefficients λ_b in the above decomposition are null). All in all, with each $\sigma \in \mathcal{F}$, we can associate a unique decomposition in $\mathcal{B}$,

$$\sigma = \sum_{b \in \mathcal{B}} \Phi_b(\sigma)\, b\,,$$

where the coefficients $\big(\Phi_b(\sigma)\big)_{b \in \mathcal{B}}$ form a convex weight vector over $\mathcal{B}$, i.e., belong to $\Delta(\mathcal{B})$; in addition, $\Phi_b(\sigma) > 0$ only if $b \in \mathcal{B}_k$, where k is such that $\sigma \in P_k$.

Since $\widetilde{H}^{-1}$ is linear on each simplex $P_1, \ldots, P_K$, we therefore get

$$\widetilde{H}^{-1}(\sigma) = \sum_{b \in \mathcal{B}} \Phi_b(\sigma)\, \widetilde{H}^{-1}(b)\,.$$

Finally, the result is a consequence of the fact that

$$\overline{m}(\boldsymbol{p}, \sigma) = r\Big(\boldsymbol{p},\, \widetilde{H}^{-1}(\sigma)\Big) = r\left(\boldsymbol{p}, \sum_{b \in \mathcal{B}} \Phi_b(\sigma)\, \widetilde{H}^{-1}(b)\right)\,,$$

that implies, by the linearity of r, that

$$\overline{m}(\boldsymbol{p}, \sigma) = \sum_{b \in \mathcal{B}} \Phi_b(\sigma)\, r\Big(\boldsymbol{p}, \widetilde{H}^{-1}(b)\Big) = \sum_{b \in \mathcal{B}} \Phi_b(\sigma)\, \overline{m}(\boldsymbol{p}, b)\,,$$

which concludes the proof. ■

Remark 20 *The proof shows that Φ is piecewise linear on a finite decomposition of $\mathcal{F}$; it is therefore Lipschitz on $\mathcal{F}$. We denote by κ_Φ its Lipschitz constant with respect to the ℓ^2–norms.*

The main contribution of this subsection (Definition 21) relies on the following additional assumption.

Assumption 1 *We assume that $\overline{m}(\,\cdot\,, b)$ is piecewise linear on $\Delta(\mathcal{I})$ for every $b \in \mathcal{B}$. We then call the corresponding game (r, H) a bi-piecewise linear game.*

Assumption 1 means that for each $b \in \mathcal{B}$ there exists a decomposition of $\Delta(\mathcal{I})$ into polytopes each on which $\overline{m}(\,\cdot\,, b)$ is linear. Since $\mathcal{B}$ is finite, there exist finitely many such decompositions to consider, and thus there exists a decomposition to polytopes that refines all of them. (The latter is generated by the intersection of all considered polytopes as b varies.) By construction, every $\overline{m}(\,\cdot\,, b)$ is linear on any of the polytopes of this common decomposition. We denote by $\mathcal{A} \subset \Delta(\mathcal{I})$ the finite subset of all their vertices. A construction similar to the one used in the proof of Lemma 19 leads to a piecewise linear (injective) mapping $\Theta : \Delta(\mathcal{I}) \to \Delta(\mathcal{A})$, where $\Theta(\boldsymbol{p})$ is the decomposition of $\boldsymbol{p}$ on the vertices of the polytope(s) of the decomposition to which it belongs, satisfying

$$\forall\, b \in \mathcal{B}, \quad \forall\, \boldsymbol{p} \in \Delta(\mathcal{I}), \qquad \overline{m}(\boldsymbol{p}, b) = \sum_{a \in \mathcal{A}} \Theta_a(\boldsymbol{p})\, \overline{m}(a, b)\,,$$

where we denoted the convex weight vector $\Theta(\boldsymbol{p}) \in \Delta(\mathcal{A})$ by $\big(\Theta_a(\boldsymbol{p})\big)_{a \in \mathcal{A}}$. This, Lemma 19, and Assumption 1 show that on a lifted space, $\overline{m}$ coincides with a bi-linear mapping $\overline{\overline{m}}$, as is made formal in the next definition.

Definition 21 *For a bi-piecewise linear game, we denote by $\overline{\overline{m}}$ the linear extension to $\Delta(\mathcal{A} \times \mathcal{B})$ of the restriction of $\overline{m}$ to $\mathcal{A} \times \mathcal{B}$, so that for all $\boldsymbol{p} \in \Delta(\mathcal{I})$ and $\sigma \in \mathcal{F}$,*

$$\overline{m}(\boldsymbol{p}, \sigma) = \overline{\overline{m}}\big(\Theta(\boldsymbol{p}),\, \Phi(\sigma)\big)\,.$$

5.3.2 Construction of a Strategy to Approach $\mathcal{C}$

The approaching strategy for the original problem is based on a strategy Ψ for $\overline{\overline{m}}$–approachability of $\mathcal{C}$, provided by Theorem 8; we therefore first need to prove the existence of such a Ψ.

Lemma 22 *Under Condition* (APM)*, the closed convex set $\mathcal{C}$ is $\overline{\overline{m}}$–approachable.*

Proof We show that Condition (SVAC) in Theorem 8 is satisfied, that is, that for all $\boldsymbol{y} \in \Delta(\mathcal{B})$, there exists some $\boldsymbol{x} \in \Delta(\mathcal{A})$ such that $\overline{\overline{m}}(\boldsymbol{x}, \boldsymbol{y}) \subseteq \mathcal{C}$. With such a given $\boldsymbol{y} \in \Delta(\mathcal{B})$, we associate[2] the feasible vector of signals $\sigma = \sum_{b \in \mathcal{B}} y_b\, b \in \mathcal{F}$ and let $\boldsymbol{p}$ be given by Condition (APM), so that $\overline{m}(\boldsymbol{p}, \sigma) \subseteq \mathcal{C}$. By linearity of $\overline{\overline{m}}$ (for the first equality), by

2. Note, however, that we do not necessarily have that $\Phi(\sigma)$ and $\boldsymbol{y}$ are equal, as Φ is not a one-to-one mapping (it is injective but not surjective).

Approaching Strategy in Games with Partial Monitoring

Parameters: an integer block length $L \geqslant 1$, an exploration parameter $\gamma \in [0,1]$, a strategy Ψ for $\overline{\overline{m}}$–approachability of $\mathcal{C}$

Notation: $\boldsymbol{u} \in \Delta(\mathcal{I})$ is the uniform distribution over $\mathcal{I}$, $P_{\mathcal{F}}$ denotes the projection operator in ℓ^2–norm of $\mathbb{R}^{\mathcal{H}\times\mathcal{I}}$ onto $\mathcal{F}$

Initialization: compute the finite set $\mathcal{B}$ and the mapping $\Phi : \mathcal{F} \to \Delta(\mathcal{B})$ of Lemma 19, compute the finite set $\mathcal{A}$ and the mapping $\Theta : \Delta(\mathcal{I}) \to \Delta(\mathcal{A})$ defined based on Assumption 1, pick an arbitrary $\boldsymbol{\theta}_1 \in \Delta(\mathcal{A})$

For all blocks $n = 1, 2, \ldots,$

1. define $\boldsymbol{x}_n = \sum_{a\in\mathcal{A}} \theta_{n,a}\, a$ and $\boldsymbol{p}_n = (1-\gamma)\,\boldsymbol{x}_n + \gamma\,\boldsymbol{u}$; refer to the components of $\boldsymbol{p}_n$ as $(p_{i,n})_{i\in\mathcal{I}}$;
2. for rounds $t = (n-1)L+1, \ldots, nL$,
   2.1 draw an action $I_t \in \mathcal{I}$ at random according to $\boldsymbol{p}_n$;
   2.2 get the signal S_t;
3. form the estimated vector of probability distributions over signals,
$$\widetilde{\sigma}_n = \left(\frac{1}{L} \sum_{t=(n-1)L+1}^{nL} \frac{\mathbb{I}_{\{S_t=s\}}\mathbb{I}_{\{I_t=i\}}}{p_{I_t,n}} \right)_{(i,s)\in\mathcal{I}\times\mathcal{H}};$$
4. compute the projection $\widehat{\sigma}_n = P_{\mathcal{F}}\big(\widetilde{\sigma}_n\big)$;
5. choose $\boldsymbol{\theta}_{n+1} = \Psi\Big(\boldsymbol{\theta}_1, \Phi\big(\widehat{\sigma}_1\big), \ldots, \boldsymbol{\theta}_n, \Phi\big(\widehat{\sigma}_n\big)\Big)$.

Figure 1: The proposed strategy, which plays in blocks.

convexity of $\overline{m}$ in its second argument (for the first inclusion), by Lemma 19 (for the second and fourth equalities), by construction of $\mathcal{A}$ (for the third equality),

$$\begin{aligned}\overline{\overline{m}}\big(\Theta(\boldsymbol{p}), \boldsymbol{y}\big) = \sum_{a\in\mathcal{A}} \Theta_a(\boldsymbol{p}) \sum_{b\in\mathcal{B}} y_b\, \overline{m}(a,b) \quad &\subseteq \quad \sum_{a\in\mathcal{A}} \Theta_a(\boldsymbol{p})\, \overline{m}(a,\sigma) = \sum_{a\in\mathcal{A}} \Theta_a(\boldsymbol{p}) \sum_{b\in\mathcal{B}} \Phi_b(\sigma)\, \overline{m}(a,b) \\ &= \quad \sum_{b\in\mathcal{B}} \Phi_b(\sigma)\, \overline{m}(\boldsymbol{p},b) = \overline{m}(\boldsymbol{p},\sigma) \subseteq \mathcal{C}\,,\end{aligned}$$

which concludes the proof. ■

We consider the strategy described in Figure 1 (and the notation introduced therein). It forces exploration at a γ rate, as is usual in situations with partial monitoring. One of its key ingredients, that conditionally unbiased estimators are available, is extracted from Lugosi et al. (2008, Section 6): in block n we consider sums of elements of the form

$$\widehat{H}_t = \left(\frac{\mathbb{I}_{\{S_t=s\}}\mathbb{I}_{\{I_t=i\}}}{p_{I_t,n}} \right)_{(i,s)\in\mathcal{I}\times\mathcal{H}} \in \mathbb{R}^{\mathcal{H}\times\mathcal{I}}\,.$$

Averaging over the respective random draws of I_t and S_t according to $\boldsymbol{p}_n$ and $H(I_t, J_t)$, i.e., taking the conditional expectation $\mathbb{E}_t$ with respect to $\boldsymbol{p}_n$ and J_t, we get

$$\mathbb{E}_t\big[\widehat{H}_t\big] = \widetilde{H}\big(\delta_{J_t}\big)\,. \tag{7}$$

Indeed, the conditional expectation of the component i of $\widehat{H}_t$ equals

$$\mathbb{E}_t\left[\left(\frac{\mathbb{I}_{\{S_t=s\}}\mathbb{I}_{\{I_t=i\}}}{p_{I_t,n}}\right)_{s\in\mathcal{H}}\right] = \mathbb{E}_t\left[\frac{H(I_t,J_t)\,\mathbb{I}_{\{I_t=i\}}}{p_{I_t,n}}\right] = \frac{H(i,J_t)}{p_{i,n}}\;\mathbb{E}_t\big[\mathbb{I}_{\{I_t=i\}}\big] = H(i,J_t)\,,$$

where we first took the expectation over the random draw of S_t (conditionally on $\boldsymbol{p}_n$, J_t, and I_t) and then over the one of I_t. Consequently, concentration arguments show that for L large enough,

$$\widetilde{\sigma}_n = \frac{1}{L}\sum_{t=(n-1)L+1}^{nL}\widehat{H}_t \qquad \text{is close to} \qquad \widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)\,, \qquad \text{where} \quad \widehat{\boldsymbol{q}}_n = \frac{1}{L}\sum_{t=(n-1)L+1}^{nL}\delta_{J_t}\,. \tag{8}$$

Actually, since $\mathcal{F}\subseteq\Delta(\mathcal{H})^{\mathcal{I}}$, we have a natural embedding of $\mathcal{F}$ into $\mathbb{R}^{\mathcal{H}\times\mathcal{I}}$ and we can define $P_{\mathcal{F}}$, the convex projection operator onto $\mathcal{F}$ (in ℓ^2–norm). Instead of using directly $\widetilde{\sigma}_n$, we consider in our strategy $\widehat{\sigma}_n = P_{\mathcal{F}}\big(\widetilde{\sigma}_n\big)$, which is even closer to $\widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)$.

More precisely, the following result can be extracted from the proof of Theorem 6.1 in Lugosi et al. (2008). For the convenience of the reader, a self-contained proof is provided in Appendix C.

Lemma 23 *For all $\delta\in(0,1)$, for all blocks $n\geqslant 1$, with probability at least $1-\delta$,*

$$\left\|\widehat{\sigma}_n - \widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)\right\|_2 \leqslant \sqrt{N_{\mathcal{I}}N_{\mathcal{H}}}\left(\sqrt{\frac{2N_{\mathcal{I}}}{\gamma L}\ln\frac{2N_{\mathcal{I}}N_{\mathcal{H}}}{\delta}} + \frac{1}{3}\frac{N_{\mathcal{I}}}{\gamma L}\ln\frac{2N_{\mathcal{I}}N_{\mathcal{H}}}{\delta}\right).$$

5.3.3 A Performance Guarantee for the Strategy of Figure 1

For the sake of simplicity, we provide first a performance bound for fixed parameters γ and L tuned as functions of a known horizon T. We then obtain a bound holding only for the specific round T. Adaptation to $T\to\infty$ (and the obtention of bounds for all $T\geqslant 1$) are then described in the next section. We recall that R was defined in (5) as a bound in norm on r.

Theorem 24 *Consider a closed convex set $\mathcal{C}$ and a game (r,H) for which Condition* (APM) *is satisfied and that is bi-piecewise linear in the sense of Assumption 1. Then, for all strategies of the second player, the strategy of Figure 1, run with parameters $\gamma\in[0,1]$ and $L\geqslant 1$ and fed with a strategy Ψ for $\overline{\overline{m}}$–approachability of $\mathcal{C}$ (provided by Lemma 22) is such that, for all $T\geqslant L+1$, for all $\delta\in(0,1)$, with probability at least $1-\delta$,*

$$\begin{aligned}\inf_{c\in\mathcal{C}}\left\|c-\frac{1}{T}\sum_{t=1}^{T}r(I_t,J_t)\right\|_2 \;\leqslant\; &\frac{2L}{T}R + 4R\sqrt{\frac{\ln\big(2(T+L)/(L\delta)\big)}{T-L}} + 2\gamma R + \frac{2R}{\sqrt{T/L-1}}\sqrt{N_{\mathcal{A}}N_{\mathcal{B}}} \\ &+ R\kappa_\Phi\sqrt{N_{\mathcal{I}}N_{\mathcal{H}}N_{\mathcal{A}}}\left(\sqrt{\frac{2N_{\mathcal{I}}}{\gamma L}\ln\frac{2N_{\mathcal{I}}N_{\mathcal{H}}(T+L)}{L\delta}} + \frac{1}{3}\frac{N_{\mathcal{I}}}{\gamma L}\ln\frac{2N_{\mathcal{I}}N_{\mathcal{H}}(T+L)}{L\delta}\right).\end{aligned}$$

In particular, for all $T \geqslant 1$, the choices of $L = \lceil T^{3/5} \rceil$ and $\gamma = T^{-1/5}$ imply that for all strategies of the second player, for all $\delta \in (0,1)$, with probability at least $1-\delta$,

$$\inf_{c \in \mathcal{C}} \left\| c - \frac{1}{T} \sum_{t=1}^{T} r(I_t, J_t) \right\|_2 \leqslant \Xi \left(T^{-1/5} \sqrt{\ln \frac{T}{\delta}} + T^{-2/5} \ln \frac{T}{\delta} \right)$$

for some constant Ξ depending only on $\mathcal{C}$ and on the game (r, H) at hand.

The efficiency of the strategy of Figure 1 depends on whether it can be fed with an efficient approaching strategy Ψ, which in turn depends on the respective geometries of $\overline{m}$ and $\mathcal{C}$, as was indicated at the end of Section 3.1. (Note that the projection onto $\mathcal{F}$ can be performed in polynomial time, as the latter closed convex set is defined by finitely many linear constraints, and that the computation of $\mathcal{A}$, $\mathcal{B}$, and $\overline{\overline{m}}$ can be performed beforehand.) In any case, the per-round complexity is constant (though possibly large).

Proof We write T as $T = NL + k$ where N is an integer and $0 \leqslant k \leqslant L-1$ and will show successively that (possibly with overwhelming probability only) the following statements hold.

$$\frac{1}{T} \sum_{t=1}^{T} r(I_t, J_t) \qquad \text{is close to} \qquad \frac{1}{NL} \sum_{t=1}^{NL} r(I_t, J_t)\,; \tag{9}$$

$$\frac{1}{NL} \sum_{t=1}^{NL} r(I_t, J_t) \qquad \text{is close to} \qquad \frac{1}{N} \sum_{n=1}^{N} r\big(\boldsymbol{p}_n, \widehat{\boldsymbol{q}}_n\big)\,; \tag{10}$$

$$\frac{1}{N} \sum_{n=1}^{N} r\big(\boldsymbol{p}_n, \widehat{\boldsymbol{q}}_n\big) \qquad \text{is close to} \qquad \frac{1}{N} \sum_{n=1}^{N} r\big(\boldsymbol{x}_n, \widehat{\boldsymbol{q}}_n\big)\,; \tag{11}$$

$$\frac{1}{N} \sum_{n=1}^{N} r\big(\boldsymbol{x}_n, \widehat{\boldsymbol{q}}_n\big)$$

$$= \frac{1}{N} \sum_{n=1}^{N} \sum_{a \in \mathcal{A}} \theta_{n,a}\, r\big(a, \widehat{\boldsymbol{q}}_n\big) \qquad \text{belongs to the set} \qquad \frac{1}{N} \sum_{n=1}^{N} \sum_{a \in \mathcal{A}} \theta_{n,a}\, \overline{m}\Big(a, \widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)\Big)\,;$$

$$\frac{1}{N} \sum_{n=1}^{N} \sum_{a \in \mathcal{A}} \theta_{n,a}\, \overline{m}\Big(a, \widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)\Big) \qquad \text{is equal to the set} \qquad \frac{1}{N} \sum_{n=1}^{N} \overline{\overline{m}}\Big(\boldsymbol{\theta}_n, \Phi\Big(\widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)\Big)\Big)\,;$$

$$\frac{1}{N} \sum_{n=1}^{N} \overline{\overline{m}}\Big(\boldsymbol{\theta}_n, \Phi\Big(\widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)\Big)\Big) \qquad \text{is close to the set} \qquad \frac{1}{N} \sum_{n=1}^{N} \overline{\overline{m}}\Big(\boldsymbol{\theta}_n, \Phi\big(\widehat{\sigma}_n\big)\Big)\,; \tag{12}$$

$$\frac{1}{N} \sum_{n=1}^{N} \overline{\overline{m}}\Big(\boldsymbol{\theta}_n, \Phi\big(\widehat{\sigma}_n\big)\Big) \qquad \text{is close to the set} \qquad \mathcal{C}\,; \tag{13}$$

where we recall that the notation $\widehat{\boldsymbol{q}}_n$ was defined in (8) and is referring to the empirical distribution of the J_t in the n–th block. Actually, we will show below the numbered statements only. The first unnumbered statement is immediate by the definition of $\boldsymbol{x}_n$, the linearity of

r, and the very definition of $\overline{m}$; while the second one follows from Definition 21:

$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N}\sum_{a\in\mathcal{A}}\theta_{n,a}\,\overline{m}\Big(a,\,\widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)\Big) &= \frac{1}{N}\sum_{n=1}^{N}\sum_{(a,b)\in\mathcal{A}\times\mathcal{B}}\theta_{n,a}\,\Phi_b\Big(\widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)\Big)\,\overline{m}(a,b)\\ &= \frac{1}{N}\sum_{n=1}^{N}\overline{\overline{m}}\Big(\boldsymbol{\theta}_n,\,\Phi\Big(\widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)\Big)\Big)\,.\end{aligned}$$

Step 1: Assertion (9)**.** A direct calculation decomposing the sum over T elements into a sum over the NL first elements and the k remaining ones shows that

$$\left\|\frac{1}{T}\sum_{t=1}^{T}r(I_t,J_t)-\frac{1}{NL}\sum_{t=1}^{NL}r(I_t,J_t)\right\|_2\leqslant R\left(\frac{k}{T}+\left(\frac{1}{NL}-\frac{1}{T}\right)NL\right)=\frac{2k}{T}R\leqslant\frac{2L}{T}R\,.$$

Step 2: Assertion (10)**.** We note that by defining $\mathbb{E}_t$ as the conditional expectation with respect to (I_1,S_1,J_1), …, $(I_{t-1},S_{t-1},J_{t-1})$ and J_t, which fixes the values of the distribution $\boldsymbol{p}'_t$ of I_t and the value of J_t, we have

$$\mathbb{E}_t\big[r(I_t,J_t)\big]=r(\boldsymbol{p}'_t,J_t)\,.$$

We note that by definition of the forecaster, $\boldsymbol{p}'_t=\boldsymbol{p}_n$ if t belongs to the n–th block. By a version of the Hoeffding-Azuma inequality for sums of Hilbert space-valued martingale differences stated as[3] Lemma 3.2 in Chen and White (1996), we therefore get that with probability at least $1-\delta$,

$$\left\|\frac{1}{NL}\sum_{t=1}^{NL}r(I_t,J_t)-\frac{1}{N}\sum_{n=1}^{N}r\big(\boldsymbol{p}_n,\,\widehat{\boldsymbol{q}}_n\big)\right\|_2\leqslant 4R\sqrt{\frac{\ln(2/\delta)}{NL}}\leqslant 4R\sqrt{\frac{\ln(2/\delta)}{T-L}}\,.$$

The second inequality comes from $NL=T-k\geqslant T-L$.

Step 3: Assertion (11)**.** Since by definition $\boldsymbol{p}_n=(1-\gamma)\,\boldsymbol{x}_n+\gamma\,\boldsymbol{u}$, we get

$$\left\|\frac{1}{N}\sum_{n=1}^{N}r\big(\boldsymbol{p}_n,\,\widehat{\boldsymbol{q}}_n\big)-\frac{1}{N}\sum_{n=1}^{N}r\big(\boldsymbol{x}_n,\,\widehat{\boldsymbol{q}}_n\big)\right\|_2\leqslant 2\gamma R\,.$$

Step 4: Assertion (12)**.** We fix a given block n. Lemma 23 indicates that with probability $1-\delta$,

$$\left\|\widehat{\sigma}_n-\widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)\right\|_2\leqslant\sqrt{N_{\mathcal{I}}N_{\mathcal{H}}}\left(\sqrt{\frac{2N_{\mathcal{I}}}{\gamma L}\ln\frac{2N_{\mathcal{I}}N_{\mathcal{H}}}{\delta}}+\frac{1}{3}\frac{N_{\mathcal{I}}}{\gamma L}\ln\frac{2N_{\mathcal{I}}N_{\mathcal{H}}}{\delta}\right).\tag{14}$$

Since Φ is Lipschitz (see Remark 20), with a Lipschitz constant in ℓ^2–norms denoted by κ_Φ, we get that with probability $1-\delta$,

$$\left\|\Phi\big(\widehat{\sigma}_n\big)-\Phi\Big(\widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)\Big)\right\|_2\leqslant\kappa_\Phi\sqrt{N_{\mathcal{I}}N_{\mathcal{H}}}\left(\sqrt{\frac{2N_{\mathcal{I}}}{\gamma L}\ln\frac{2N_{\mathcal{I}}N_{\mathcal{H}}}{\delta}}+\frac{1}{3}\frac{N_{\mathcal{I}}}{\gamma L}\ln\frac{2N_{\mathcal{I}}N_{\mathcal{H}}}{\delta}\right).$$

3. Note that the martingale increments are bounded in norm by $2R$ in our framework and that $\sqrt{u}\,e^{-u}\leqslant e^{-u/2}$ for all $u\geqslant 0$.

By a union bound, the above bound holds for all blocks $n = 1, \dots, N$ with probability at least $1 - N\delta$. Finally, an application of Lemma 6 shows that

$$\frac{1}{N} \sum_{n=1}^{N} \overline{\overline{m}}\Big(\boldsymbol{\theta}_n, \Phi\Big(\widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)\Big)\Big)$$

is in a ε_T–neighborhood (in ℓ^2–norm) of

$$\frac{1}{N} \sum_{n=1}^{N} \overline{\overline{m}}\Big(\boldsymbol{\theta}_n, \Phi\big(\widehat{\sigma}_n\big)\Big)\,,$$

where

$$\varepsilon_T = R\sqrt{N_{\mathcal{B}}}\left(\kappa_\Phi \sqrt{N_{\mathcal{I}} N_{\mathcal{H}}}\left(\sqrt{\frac{2N_{\mathcal{I}}}{\gamma L}\ln\frac{2N_{\mathcal{I}}N_{\mathcal{H}}}{\delta}} + \frac{1}{3}\frac{N_{\mathcal{I}}}{\gamma L}\ln\frac{2N_{\mathcal{I}}N_{\mathcal{H}}}{\delta}\right)\right).$$

Step 5: Assertion (13)**.** Since $\mathcal{C}$ is $\overline{\overline{m}}$–approachable and by definition of the choices of the $\boldsymbol{\theta}_n$ in Figure 1, we get by Theorem 8, with probability 1,

$$\inf_{c\in\mathcal{C}} \left\| c - \frac{1}{N}\sum_{n=1}^{N} \overline{\overline{m}}\Big(\boldsymbol{\theta}_n, \Phi\big(\widehat{\sigma}_n\big)\Big)\right\|_2 \leqslant \frac{2R}{\sqrt{N}}\sqrt{N_{\mathcal{A}}N_{\mathcal{B}}} \leqslant \frac{2R}{\sqrt{T/L-1}}\sqrt{N_{\mathcal{A}}N_{\mathcal{B}}}\,,$$

since, as used already at the end of step 2, $N \geqslant T/L - 1$.

Conclusion of the proof. The proof is concluded by putting the pieces together, thanks to a triangle inequality. By a union bound, the obtained bound holds however only with probability at least $1 - (N+1)\delta \geqslant 1 - \big((T+L)/L\big)\delta$, where we used $N \leqslant T/L$. The stated bound follows by replacing all occurrences of δ in the previous steps by $\delta L/(T+L)$. ■

5.3.4 Uniform Guarantees over Time

We present here a variant of the strategy of Figure 1 that r–approaches $\mathcal{C}$ for the signaling structure H. This is achieved by making the strategy independent of the horizon T. (The strategy of the previous section depended on the knowledge of T, via suitable choices for L and γ.) Two options could have been worked out: resorting to some "doubling trick" or having the parameters L and γ vary over time. In the latter option, the lengths L_n of blocks n and the exploration rates γ_n used therein are no longer constant but of lengths polynomial in n. We chose the latter option for the sake of elegance and because it relies on a result of independent interest, namely a generalization of Theorem 3 to polynomial averages. We only state this generalization for mixed actions taken and observed, but the adaptation for pure actions follows easily.

Consider the setting of Theorem 3. The studied strategy relies on a parameter $\alpha \geqslant 0$. It plays an arbitrary $\boldsymbol{x}_1$. For $t \geqslant 1$, it forms at stage $t+1$ the vector-valued polynomial average

$$\widehat{m}_t^\alpha = \frac{1}{T_t^\alpha}\sum_{s=1}^{t} s^\alpha\, m(\boldsymbol{x}_s, \boldsymbol{y}_s) \qquad \text{where} \qquad T_t^\alpha = \sum_{s=1}^{t} s^\alpha\,,$$

computes its projection c_t^α onto $\mathcal{C}$, and resorts to a mixed action $\boldsymbol{x}_{t+1}$ solving the minimax equation

$$\min_{\boldsymbol{x}\in\Delta(\mathcal{A})} \max_{\boldsymbol{y}\in\Delta(\mathcal{B})} \left\langle \widehat{m}_t^\alpha - c_t^\alpha,\, m(\boldsymbol{x},\boldsymbol{y}) - c_t^\alpha \right\rangle .$$

Theorem 25 *We denote by M a bound in norm over m, i.e.,*

$$\max_{(a,b)\in\mathcal{A}\times\mathcal{B}} \big\| m(a,b) \big\|_2 \leqslant M .$$

For all $\alpha \geqslant 0$, when $\mathcal{C}$ is an approachable closed convex set, the above strategy ensures that for all strategies of the second player, with probability 1, for all $T \geqslant 1$,

$$\inf_{c\in\mathcal{C}} \left\| c - \frac{1}{\sum_{t=1}^T t^\alpha} \sum_{t=1}^T t^\alpha\, m(\boldsymbol{x}_t,\boldsymbol{y}_t) \right\|_2 \leqslant 2M \frac{\sqrt{\sum_{t=1}^T t^{2\alpha}}}{\sum_{t=1}^T t^\alpha} \leqslant \frac{2M(\alpha+1)}{\sqrt{T}} . \tag{15}$$

It is interesting to note that the convergence rate is independent of α and is the same as standard approachability ($1/\sqrt{T}$). The proof of this theorem is a slight modification of the proof of Theorem 3 and is hence deferred to Appendix D.

The extension to polynomially weighted averages can also be obtained in the context of set-valued approachability. This is because the key to Theorem 8 is Lemma 10, which indicates that to get set-valued approachability, it suffices to approach, in the usual sense, $\widetilde{\mathcal{C}}$. Both can thus be performed with polynomially weighted averages.

Consider now the variant of the strategy of Figure 1 for which the length of the n-th block, denoted by L_n, is equal to n^α, the exploration rate on this block comes at a rate $\gamma_n = n^{-\alpha/3}$, and Ψ is an $\overline{\overline{m}}$–approaching strategy of $\mathcal{C}$ with respect to polynomially weighted averages with parameter $\alpha = 3/2$. We call it a time-adaptive version of the strategy of Figure 1; indeed, this choice of α ensures that there are roughly $T^{2/5}$ blocks and that the length of the last one is of the order of $T^{3/5}$. Note that it does not depend anymore on any time horizon T, hence guarantees can be obtained for all T.

Theorem 26 *In the same setting and under the same assumptions as in Theorem 24, the time-adaptive version of the strategy described in Figure 1 (with $L_n = n^\alpha$ and $\gamma_n = n^{-\alpha/3}$ for $\alpha = 3/2$) ensures that, for all strategies of the second player, for all $T \geqslant 1$ and all $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$\inf_{c\in\mathcal{C}} \left\| c - \frac{1}{T} \sum_{t=1}^T r(I_t,J_t) \right\|_2 \leqslant \Xi \left(T^{-1/5} \sqrt{\ln \frac{T}{\delta}} + T^{-2/5} \ln \frac{T}{\delta} \right)$$

for some constant Ξ depending only on $\mathcal{C}$ and on the game (r, H) at hand.

The proof follows closely the one of Theorem 24 and is presented in Appendix E.

Corollary 27 *In the same setting and under the same assumptions as in Theorem 24, the time-adaptive version of the strategy described in Figure 1 (with $L_n = n^\alpha$ and $\gamma_n = n^{-\alpha/3}$ for $\alpha = 3/2$) indeed r–approaches $\mathcal{C}$ for the signalling structure H.*

Proof The strategy at hand is such that for all $T \geqslant 1$, with probability at least $1 - 1/T^2$,

$$\inf_{c \in \mathcal{C}} \left\| c - \frac{1}{T} \sum_{t=1}^{T} r(I_t, J_t) \right\|_2 \leqslant \Xi\big(T^{-1/5} \ln T^3 + T^{-2/5} \ln T^3\big) .$$

In particular, a union bound shows that for all $T \geqslant 2$,

$$\sup_{\tau \geqslant T} \inf_{c \in \mathcal{C}} \left\| c - \frac{1}{\tau} \sum_{t=1}^{\tau} r(I_t, J_t) \right\|_2 \leqslant R_T \stackrel{\text{def}}{=} \sup_{\tau \geqslant T} \Xi\big(\tau^{-1/5} \ln \tau^3 + \tau^{-2/5} \ln \tau^3\big) ,$$

with probability at least $1 - P_T$, where $P_T = \sum_{\tau \geqslant T} 1/\tau^2$. We note that $P_T \to 0$ and $R_T \to 0$ as $T \to \infty$. To see that the definition of approachability is satisfied, given $\varepsilon > 0$, it suffices to define T_ε as the minimal T such that $R_T \leqslant \varepsilon$ and $P_T \leqslant \varepsilon$. ■

5.4 Approachability in the Case of General Games with Partial Monitoring

Unfortunately, as is illustrated in the example below, there exist games with partial monitoring that are not bi-piecewise linear. However, we will show that if Condition (APM) holds there exist strategies with a constant per-round complexity that approach polytopes even when the game is not bi-piecewise linear. That is, by considering simpler closed convex sets $\mathcal{C}$, no assumption is needed on the pair (r, H).

We will conclude this main part of the paper by re-proving, using a doubling trick, that Condition (APM) is still sufficient for approachability in the most general case when no assumption is made neither on (r, H) nor on $\mathcal{C}$, at the cost of inefficiency.

Example 3 *The following game (with the same action and signal sets as in Example 2) is not bi-piecewise linear.*

	L	M	R
T	$(1,0,0,0)$ / ♣	$(0,0,1,0)$ / ♣	$(2,0,4,0)$ / ♡
B	$(0,1,0,0)$ / ♣	$(0,0,0,1)$ / ♣	$(0,3,0,5)$ / ♡

Proof We denote mixed actions of the first player by $(p, 1-p)$, where $p \in [0,1]$ denotes the probability of playing T and $1-p$ is the probability of playing B. It is immediate that $\overline{m}\big((p,1-p), \clubsuit\big)$ can be identified with the set of all product distributions on 2×2 elements with first marginal distribution $(p, 1-p)$. The proof of Lemma 19 shows that the set $\mathcal{B}$ associated with any game always contains the Dirac masses on each signal; that is, $\delta_\clubsuit \in \mathcal{B}$. But for $p \neq p'$ and $\lambda \in (0,1)$, denoting $\overline{p} = \lambda\, p + (1-\lambda) p'$, one necessarily has that

$$\overline{m}\big((\overline{p}, 1-\overline{p}), \clubsuit\big) \subsetneq \lambda\, \overline{m}\big((p,1-p), \clubsuit\big) + (1-\lambda)\, \overline{m}\big((p',1-p'), \clubsuit\big) ;$$

the inclusion $\subseteq$ holds by concavity of $\overline{m}$ in its first argument (Lemma 17) but this inclusion is always strict here since the left-hand side is formed by product distributions while the right-hand side also contains distributions with correlations. Hence, bi-piecewise linearity cannot hold for this game. ■

5.4.1 Approachability of the Negative Orthant in General Games

For the sake of simplicity, we start with the case of the negative orthant $\mathbb{R}^d_-$ and prove the following result. Note that for the first time in this general framework, the rates for approachability of polytopes are independent of the dimension (as is the case with full monitoring).

Theorem 28 *If Condition* (APM) *is satisfied for $\overline{m}$ and $\mathbb{R}^d_-$, then there exists a strategy for (r, H)–approaching $\mathbb{R}^d_-$ at a rate of the order of $T^{-1/5}$, with a constant per-round complexity.*

Our argument is based on Lemma 19; we use in the sequel the objects and notation introduced therein. We denote by $r = (r_k)_{1 \leqslant k \leqslant d}$ the components of the d–dimensional payoff function r and introduce, for all $k \in \{1, \dots, d\}$, the set-valued mapping $\widetilde{m}_k$ defined by

$$\widetilde{m}_k : \quad (\boldsymbol{p}, b) \in \Delta(\mathcal{I}) \times \mathcal{B} \; \longmapsto \; \widetilde{m}_k(\boldsymbol{p}, b) = \Big\{ r_k(\boldsymbol{p}, \boldsymbol{q}) : \quad \boldsymbol{q} \in \Delta(\mathcal{J}) \text{ such that } \widetilde{H}(\boldsymbol{q}) = b \Big\} .$$

The mapping $\widetilde{m}$ is then defined as the Cartesian product of the $\widetilde{m}_k$; formally, for all $\boldsymbol{p} \in \Delta(\mathcal{I})$ and $b \in \mathcal{B}$,

$$\widetilde{m}(\boldsymbol{p}, b) = \Big\{ (z_1, \dots, z_d) : \qquad \forall k \in \{1, \dots, d\}, \quad z_k \in \widetilde{m}_k(\boldsymbol{p}, b) \Big\} .$$

We then linearly extend this mapping to a set-valued mapping $\widetilde{m}$ defined on $\Delta(\mathcal{I}) \times \Delta(\mathcal{B})$ and finally consider the set-valued mapping $\breve{m}$ defined on $\Delta(\mathcal{I}) \times \mathcal{F}$ by

$$\forall \, \sigma \in \mathcal{F}, \quad \forall \, \boldsymbol{p} \in \Delta(\mathcal{I}), \qquad \breve{m}(\boldsymbol{p}, \sigma) = \widetilde{m}\big(\boldsymbol{p}, \Phi(\sigma)\big) = \sum_{b \in \mathcal{B}} \Phi_b(\sigma) \, \widetilde{m}(\boldsymbol{p}, b) \, ,$$

where Φ refers to the mapping defined in Lemma 19 (based on $\overline{m}$). The lemma below indicates why $\breve{m}$ is an excellent substitute to $\overline{m}$ in the case of the approachability of the orthant $\mathbb{R}^d_-$.

Lemma 29 *The set-valued mappings $\breve{m}$ and $\overline{m}$ satisfy that for all $p \in \Delta(\mathcal{I})$ and $\sigma \in \mathcal{F}$,*

1. *the inclusion $\overline{m}(\boldsymbol{p}, \sigma) \subseteq \breve{m}(\boldsymbol{p}, \sigma)$ holds;*
2. *if $\overline{m}(\boldsymbol{p}, \sigma) \subseteq \mathbb{R}^d_-$, then one also has $\breve{m}(\boldsymbol{p}, \sigma) \subseteq \mathbb{R}^d_-$.*

The interpretation of these two properties is: 1. $\breve{m}$–approaching a set $\mathcal{C}$ is more difficult than $\overline{m}$–approaching it; and 2. that if Condition (APM) holds for $\overline{m}$ and $\mathbb{R}^d_-$, it also holds for $\breve{m}$ and $\mathbb{R}^d_-$.

Proof For the first property, note that by the component-wise construction of $\widetilde{m}$,

$$\forall \, b \in \mathcal{B}, \quad \forall \, \boldsymbol{p} \in \Delta(\mathcal{I}), \qquad \overline{m}(\boldsymbol{p}, b) \subseteq \widetilde{m}(\boldsymbol{p}, b) \, .$$

Lemma 19, the linear extension of $\widetilde{m}$, and the definition of $\breve{m}$ then show that

$$\forall \, \sigma \in \mathcal{F}, \quad \forall \, \boldsymbol{p} \in \Delta(\mathcal{I}), \qquad \overline{m}(\boldsymbol{p}, \sigma) = \sum_{b \in \mathcal{B}} \Phi_b(\sigma) \, \overline{m}(\boldsymbol{p}, b) \subseteq \widetilde{m}\big(\boldsymbol{p}, \, \Phi(\sigma)\big) = \breve{m}(\boldsymbol{p}, \sigma) \, .$$

As for the second property, it suffices to work component-wise. Note that (by Lemma 19 again) the stated assumption exactly means that $\sum_{b\in\mathcal{B}} \Phi_b(\sigma)\, \overline{m}(\boldsymbol{p}, b) \subset \mathbb{R}^d_-$. In particular, rewriting the non-positivity constraint for each of the d components of the payoff vectors, we get

$$\sum_{b\in\mathcal{B}} \Phi_b(\sigma)\, \widetilde{m}_k(\boldsymbol{p}, b) \subseteq \mathbb{R}_- \,,$$

for all $k \in \{1, \dots, d\}$; thus, in particular, $\sum_{b\in\mathcal{B}} \Phi_b(\sigma)\, \widetilde{m}(\boldsymbol{p}, b) = \breve{m}(\boldsymbol{p}, \sigma) \subseteq \mathbb{R}^d_-$. ■

We can then extend the result of the previous section without the bi-piecewise linearity assumption.

Proof [of Theorem 28] The assumption of the theorem and the second property of Lemma 29 imply that Condition (APM) holds for $\mathbb{R}^d_-$ and $\breve{m}$. Furthermore, the latter corresponds to a bi-piecewise linear game, i.e., Assumption 1 is satisfied. Indeed, we show below that each $\widetilde{m}_k(\,\cdot\,, b)$ is a piecewise linear function. As a consequence, each $\breve{m}(\,\cdot\,, b)$ is also a piecewise linear function.

Each $\widetilde{m}_k(\,\cdot\,, b)$ is piecewise linear since $\widetilde{m}_k$ is based on the scalar payoff function r_k. Indeed, since $\widetilde{H}$ is linear, the set

$$\Big\{ \boldsymbol{q} \in \Delta(\mathcal{J}) \text{ such that } \widetilde{H}(\boldsymbol{q}) = b \Big\}$$

is a polytope, thus, the convex hull of some finite set $\{\boldsymbol{q}_{b,1}, \dots, \boldsymbol{q}_{b,M}\} \subset \Delta(\mathcal{J})$. Therefore, for every $\boldsymbol{p} \in \Delta(\mathcal{I})$, by linearity of r_k (and by the fact that it takes one-dimensional values),

$$\begin{aligned}
\widetilde{m}_k(\boldsymbol{p}, b) &= \Big\{ r_k(\boldsymbol{p}, \boldsymbol{q}) : \quad \boldsymbol{q} \in \Delta(\mathcal{J}) \text{ such that } \widetilde{H}(\boldsymbol{q}) = b \Big\} \\
&= \mathrm{co}\, \Big\{ r_k(\boldsymbol{p}, \boldsymbol{q}_{b,1}), \dots, r(\boldsymbol{p}, \boldsymbol{q}_{b,M}) \Big\} = \Big[\min_{k\in\{1,\dots,M\}} r_k(\boldsymbol{p}, \boldsymbol{q}_{b,k}) \,,\ \max_{k'\in\{1,\dots,M\}} r_k(\boldsymbol{p}, \boldsymbol{q}_{b,k'}) \Big] \,, \quad (16)
\end{aligned}$$

where co stands for the convex hull. Since all mappings $r_k(\,\cdot\,, \boldsymbol{q}_{b,k})$ are linear, their minimum and their maximum are piecewise linear functions, therefore $\widetilde{m}_k(\,\cdot\,, b)$ is also piecewise linear.

Therefore, the steps between Equations (11)–(13) of the proof of Theorem 24 (or the corresponding statements in the proof of Theorem 26) can be adapted by replacing $\overline{m}$ and $\overline{\overline{m}}$ by, respectively, $\widetilde{m}$, $\breve{m}$, and its extension corresponding to Definition 21. The result about approachability rates follows.

We now prove that the strategy constructed here is efficient. Indeed, recall that this is always the case as long as the projection onto the associated convex set $\widetilde{\mathcal{C}}$ defined by (4) with the linear function $\overline{\overline{m}}$ is also efficient. But this follows from the fact that as $\mathbb{R}^d_-$ is a polyhedron, the set $\widetilde{\mathcal{C}}$ is a polytope. ■

We now generalize the above ideas to more complex sets.

5.4.2 Approachability of Polytopes for General Games

If the target set $\mathcal{C}$ is a polytope, then $\mathcal{C}$ can be written as the intersection of a finite number of half-planes, i.e., there exists a finite family $\big\{(e_k,\, f_k) \in \mathbb{R}^d \times \mathbb{R},\ k \in \mathcal{K}\big\}$ such that

$$\mathcal{C} = \big\{z \in \mathbb{R}^d : \quad \langle z, e_k\rangle \leqslant f_k, \quad \forall\, k \in \mathcal{K}\big\}.$$

Given the original (not necessarily bi-piecewise linear) game (r, H), we introduce another game $(r_{\mathcal{C}}, H)$, whose payoff function $r_{\mathcal{C}} : \mathcal{I} \times \mathcal{J} \to \mathbb{R}^{\mathcal{K}}$ is defined as

$$\forall\, i \in \mathcal{I}, \quad \forall\, j \in \mathcal{J}, \qquad r_{\mathcal{C}}(i,j) = \Big[\langle r(i,j), e_k\rangle - f_k\Big]_{k\in\mathcal{K}}.$$

The following lemma is a mere exercise of rewriting.

Lemma 30 *Given a polytope $\mathcal{C}$, the (r, H)–approachability of $\mathcal{C}$ and the $\big(r_{\mathcal{C}}, H\big)$–approachability of $\mathbb{R}^d_-$ are equivalent in the sense that every strategy for solving one problem translates to a strategy for solving the other problem. In addition, Condition* (APM) *holds for (r, H) and $\mathcal{C}$ if and only if it holds for $\big(r_{\mathcal{C}}, H\big)$ and $\mathbb{R}^d_-$.*

Via the lemma above, Theorem 28 indicates that Condition (APM) for (r, H) and $\mathcal{C}$ is a sufficient condition for the (r, H)–approachability of $\mathcal{C}$ and provides an efficient strategy to do so. (The per-round complexity of this strategy depends in particular at least linearly on the cardinality of $\mathcal{K}$.) Again, rates of convergence are also, for the first time, independent of the dimensions (yet the question of their optimality remains open).

5.4.3 Approachability of General Convex Sets in the Case of General Games

In the above, we provided efficient strategies in the following cases:

- Up to projection oracles, when the games (r, H) are bi-piecewise linear, with no assumption on the target set $\mathcal{C}$; see Section 5.3. This includes at least the minimization of external and internal regret, for which the projections can indeed be performed efficiently; see the upcoming Section 6.
- When the target set $\mathcal{C}$ is a polytope, with no assumption on the game (r, H); see Sections 5.4.1 and 5.4.2.

We only mention the case of general games (r, H) and general closed convex target sets $\mathcal{C}$ in this section to have a complete, self-contained, and constructive proof of the sufficiency of Condition (APM) for (r, H)–approachability. (Perchet, 2011a already proved the latter.)

Theorem 15 and Lemma 17 show that Condition (APM) is indeed sufficient to (r, H)–approach any general closed convex set $\mathcal{C}$. However, the computational complexity of the resulting strategy is much larger as the per-round complexity increases over time. Another way to deal with a general closed convex set is based on the fact that it can be approximated arbitrarily well by a polytope (where the number of vertices of the latter increases as the quality of the approximation does). Playing in regimes approachability strategies of such a sequence of approximations also gives an approachability strategy of the original set $\mathcal{C}$. However, the per-round complexity increases over time (as the numbers of vertices of the approximating polytopes do).

6. Application to Regret Minimization

In this section we analyze external and internal regret minimization in repeated games with partial monitoring from the perspective of approachability. We show how to efficiently minimize regret in both setups using the results developed for vector-valued games with partial monitoring. To do so, we indicate why the assumption of bi-piecewise linearity (Assumption 1) is satisfied.

The results instantiated below are not necessarily new in terms of efficiency or convergence rates, and some are even slightly suboptimal. However, our point is that all previous good strategies were specifically designed for the problem of regret minimization, while we introduced above a general strategy for all approachability problems, including regret minimization. And what we gained in generality (the wider range of problems that we can deal with) has no impact (or little impact only) on the efficiency or on the rates, which we think is an important contribution.

6.1 External Regret

We consider in this section the framework and aim introduced by Rustichini (1999) and studied, sometimes for restricted classes of games, by Piccolboni and Schindelhauer (2001), Mannor and Shimkin (2003), Cesa-Bianchi et al. (2006), Lugosi et al. (2008), Bartók et al. (2010, 2011), Foster and Rakhlin (2012). We show that our general strategy can be used for regret minimization.

Scalar payoffs are obtained (but not observed) by the first player, i.e., $d = 1$: the payoff function r is a mapping $\mathcal{I} \times \mathcal{J} \to \mathbb{R}$. We still denote by R a bound on $|r|$. We define in this section

$$\widehat{\boldsymbol{q}}_T = \frac{1}{T} \sum_{t=1}^{T} \delta_{J_T}$$

as the empirical distribution of the actions taken by the second player during the first T rounds. (This is in slight contrast with the notation $\widehat{\boldsymbol{q}}_n$ used in Section 5.3 to denote such an empirical distribution, but only taken within regime n.)

The external regret of the first player at round T equals by definition

$$R_T^{\text{ext}} = \max_{\boldsymbol{p} \in \Delta(\mathcal{I})} \rho\Big(\boldsymbol{p}, \widetilde{H}\big(\widehat{\boldsymbol{q}}_T\big)\Big) - \frac{1}{T} \sum_{t=1}^{T} r(I_t, J_t)\,,$$

where $\rho : \Delta(\mathcal{I}) \times \mathcal{F}$ is defined as follows: for all $\boldsymbol{p} \in \Delta(\mathcal{I})$ and $\sigma \in \mathcal{F}$,

$$\rho(\boldsymbol{p}, \sigma) = \min\Big\{ r(\boldsymbol{p}, \boldsymbol{q}) \,:\, \boldsymbol{q} \text{ such that } \widetilde{H}(\boldsymbol{q}) = \sigma \Big\}\,. \tag{17}$$

The function ρ is continuous in its first argument and therefore the supremum in the defining expression of R_T^{ext} is a maximum.

We recall briefly why, intuitively, this is the natural notion of external regret to consider in this case (more formal arguments are given in Rustichini, 1999). Indeed, the first term in the definition of R_T^{ext} is (close to) the worst-case average payoff obtained by the first player when playing consistently a mixed action $\boldsymbol{p}$ against a sequence of mixed actions inducing on average the same laws on the signals as the sequence of actions actually played.

Rustichini (1999) calls the partial monitoring in the game (r, H) statistically sufficient when

$$\max_{\boldsymbol{p}\in\Delta(\mathcal{I})} r\big(\boldsymbol{p}, \widehat{\boldsymbol{q}}_T\big) = \max_{\boldsymbol{p}\in\Delta(\mathcal{I})} \rho\Big(\boldsymbol{p}, \widetilde{H}\big(\widehat{\boldsymbol{q}}_T\big)\Big)\,.$$

In general, only an inequality $\geqslant$ holds between the two quantities. A line of research initiated by Piccolboni and Schindelhauer (2001) first studied efficient strategies to minimize the regret in the said case of a statistically sufficient monitoring.

6.1.1 A Strategy Minimizing External Regret

The following result is a consequence of Theorem 26, as its proof shows; it corresponds to the main result of Lugosi et al. (2008), with the same convergence rate but with a different strategy. (However, Perchet, 2011b, Section 2.3 exhibited an efficient strategy achieving a convergence rate of order $T^{-1/3}$, which is optimal; this strategy was an ad hoc strategy for regret minimization. Nonetheless, a question that remains open is thus whether the rates exhibited in Theorem 26 could be improved.)

Corollary 31 *The first player can apply the strategy of Theorem 25 such that for all strategies of the second player, for all T and all $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$R_T^{\mathrm{ext}} \;\leqslant\; \Xi\left(T^{-1/5}\sqrt{\ln\frac{T}{\delta}} + T^{-2/5}\ln\frac{T}{\delta}\right)$$

for some constant Ξ depending only on the game (r, H) at hand.

The discussion about the efficiency of the strategy is postponed to the end of this section, as it relies on some objects that will be introduced in the proof of the corollary. The latter proof is an extension to the setting of partial monitoring of the original proof and strategy of Blackwell (1956b) for the case of external regret under full monitoring: in the latter case the vector-payoff function $\underline{r}$ and the set $\mathcal{C}$ considered in our proof are equal to the ones considered by Blackwell.

Proof We embed $\mathcal{F}$ into $\mathbb{R}^{\mathcal{I}\times\mathcal{H}}$ so that in this proof we will be working in the vector space $\mathbb{R}^d = \mathbb{R}\times\mathbb{R}^{\mathcal{I}\times\mathcal{H}}$. We consider the closed convex set $\mathcal{C}$ and the vector-valued payoff function $\underline{r}$ respectively defined by

$$\mathcal{C} = \left\{(z,\sigma)\in\mathbb{R}\times\mathcal{F}:\quad z \geqslant \max_{\boldsymbol{p}\in\Delta(\mathcal{I})}\rho(\boldsymbol{p},\,\sigma)\right\} \qquad \text{and} \qquad \underline{r}(i,j) = \begin{bmatrix} r(i,j) \\ \widetilde{H}(\delta_j) \end{bmatrix},$$

for all $(i,j)\in\mathcal{I}\times\mathcal{J}$.

We first show that Condition (APM) is satisfied for the considered convex set $\mathcal{C}$ and game $(\underline{r}, H)$. To do so, by continuity of ρ in its first argument, we associate with each $\boldsymbol{q}\in\Delta(\mathcal{J})$ an element $\phi(\boldsymbol{q})\in\Delta(\mathcal{I})$ such that

$$\phi(\boldsymbol{q}) \in \operatorname*{argmax}_{\boldsymbol{p}\in\Delta(\mathcal{I})} \rho\big(\boldsymbol{p},\,\widetilde{H}(\boldsymbol{q})\big)\,.$$

Then, given any $\boldsymbol{q}\in\Delta(\mathcal{J})$, we note that for all $\boldsymbol{q}'$ satisfying $\widetilde{H}(\boldsymbol{q}') = \widetilde{H}(\boldsymbol{q})$, we have by definition of ρ,

$$r\big(\phi(\boldsymbol{q}),\,\boldsymbol{q}'\big) \geqslant \rho\big(\phi(\boldsymbol{q}),\,\widetilde{H}(\boldsymbol{q}')\big) = \max_{\boldsymbol{p}\in\Delta(\mathcal{I})}\rho\big(\boldsymbol{p},\,\widetilde{H}(\boldsymbol{q}')\big)\,,$$

which shows that $\underline{r}\big(\phi(\boldsymbol{q}),\, \boldsymbol{q}'\big) \in \mathcal{C}$. The required condition is thus satisfied.

We then show that Assumption 1 is satisfied. To do so, we will use the same arguments as around (16) and actually prove the stronger property that the mappings $\overline{m}(\,\cdot\,,\sigma)$ are piecewise linear for all $\sigma \in \mathcal{F}$; we fix such a σ in the sequel. Only the first coordinate r of $\underline{r}$ depends on $\boldsymbol{p}$, so the desired property is true if and only if the mapping $\overline{m}_1(\,\cdot\,,\sigma)$ defined by

$$\boldsymbol{p} \in \Delta(\mathcal{I}) \;\longmapsto\; \overline{m}_1(\boldsymbol{p},\sigma) = \Big\{ r(\boldsymbol{p},\boldsymbol{q}) : \quad \boldsymbol{q} \in \Delta(\mathcal{J}) \text{ such that } \widetilde{H}(\boldsymbol{q}) = \sigma \Big\}$$

is piecewise linear. But this is true because r takes scalar values, as indicated around (16).

Theorem 26 can then be applied to exhibit the convergence rates; we simply need to relate the quantity of interest here to the one considered therein. To that end we use the fact that the mapping

$$\sigma \in \mathcal{F} \longmapsto \max_{\boldsymbol{p} \in \Delta(\mathcal{I})} \rho(\boldsymbol{p},\sigma)$$

is Lipschitz, with Lipschitz constant in ℓ^2–norm denoted by L_ρ; the proof of this fact is detailed in the last paragraph of this proof.

Now, the regret is non-positive if $\sum_{t=1}^T \underline{r}(I_t,J_t)/T$ belongs to $\mathcal{C}$; we therefore only need to consider the case when this average is not in $\mathcal{C}$. In the latter case, we denote by $(\widetilde{r}_T, \widetilde{\sigma}_T)$ its projection in ℓ^2–norm onto $\mathcal{C}$. We have first that the defining inequality of $\mathcal{C}$ is an equality on its border, so that

$$\widetilde{r}_T = \max_{\boldsymbol{p} \in \Delta(\mathcal{I})} \rho\big(\boldsymbol{p}, \widetilde{\sigma}_T\big)\,;$$

and second, that

$$\begin{aligned}
R_T^{\mathrm{ext}} &= \max_{\boldsymbol{p} \in \Delta(\mathcal{I})} \rho\Big(\boldsymbol{p}, \widetilde{H}\big(\widehat{\boldsymbol{q}}_T\big)\Big) - \frac{1}{T}\sum_{t=1}^T r(I_t,J_t) \\
&\leqslant \left| \max_{\boldsymbol{p} \in \Delta(\mathcal{I})} \rho\Big(\boldsymbol{p}, \widetilde{H}\big(\widehat{\boldsymbol{q}}_T\big)\Big) - \max_{\boldsymbol{p} \in \Delta(\mathcal{I})} \rho\big(\boldsymbol{p}, \widetilde{\sigma}_T\big) \right| + \left| \widetilde{r}_T - \frac{1}{T}\sum_{t=1}^T r(I_t,J_t) \right| \\
&\leqslant L_\rho \left\| \widetilde{\sigma}_T - \widetilde{H}\big(\widehat{\boldsymbol{q}}_T\big) \right\|_2 + \left| \widetilde{r}_T - \frac{1}{T}\sum_{t=1}^T r(I_t,J_t) \right| \\
&\leqslant \sqrt{2}\, \max\big\{L_\rho, 1\big\} \left\| \begin{bmatrix} \widetilde{r}_T \\ \widetilde{\sigma}_T \end{bmatrix} - \frac{1}{T}\sum_{t=1}^T \underline{r}(I_t,J_t) \right\|_2 \\
&= \sqrt{2}\, \max\big\{L_\rho, 1\big\} \inf_{c \in \mathcal{C}} \left\| c - \frac{1}{T}\sum_{t=1}^T \underline{r}(I_t,J_t) \right\|_2 .
\end{aligned}$$

The claimed rates are now seen to follow from the ones indicated in Theorem 26.

It only remains to prove the indicated Lipschitz continuity. (All Lipschitz continuity statements that follow will be with respect to the ℓ^2–norms.) We have by Definition 21 that for all $\boldsymbol{p} \in \Delta(\mathcal{I})$ and $\sigma \in \mathcal{F}$,

$$\rho(\boldsymbol{p},\sigma) = \min\, \overline{\overline{m}}_1\big(\boldsymbol{p}, \Phi(\sigma)\big)\,,$$

where the linear $\overline{\overline{m}}_1$ indifferently either is relative to $\overline{m}_1$ or is the projection onto the first component of the function $\overline{\overline{m}}$ relative to $\overline{m}$. By Remark 20 the mapping $\sigma \in \mathcal{F} \mapsto \Phi(\sigma)$ is κ_Φ–Lipschitz; this entails, by Lemma 6, that for all $\boldsymbol{p} \in \Delta(\mathcal{I})$, the mapping $\sigma \in \mathcal{F} \mapsto \rho(\boldsymbol{p}, \sigma)$ is $R\sqrt{N_\mathcal{B}}\ \kappa_\Phi$–Lipschitz. In particular, since the latter Lipschitz constant is independent of $\boldsymbol{p}$, the mapping

$$\sigma \in \mathcal{F} \ \longmapsto \ \max_{\boldsymbol{p} \in \Delta(\mathcal{I})} \rho(\boldsymbol{p}, \sigma)$$

is $R\sqrt{N_\mathcal{B}}\ \kappa_\Phi$–Lipschitz as well, which concludes the proof. ■

6.1.2 Discussion about Efficiency

An argument similar to the one in Perchet (2011b) shows that the convex set $\mathcal{C}$ is defined by a finite number of piecewise linear equations, it is therefore a polyhedron; so that the projection onto it, as well as the computation of the strategy, can be done efficiently. We only sketch here the argument. The argument used when referring to (16) indicates a priori that for each $\sigma \in \mathcal{F}$, there exist a finite number K_σ (depending on σ) of mixed actions $\boldsymbol{q}_{\sigma,1}, \ldots, \boldsymbol{q}_{\sigma,M_\sigma}$ such that for all $\boldsymbol{p} \in \Delta(\mathcal{I})$, we have $\rho(\boldsymbol{p}, \sigma) = \min\bigl\{r(\boldsymbol{p}, \boldsymbol{q}_{\sigma,1}), \ldots, r(\boldsymbol{p}, \boldsymbol{q}_{\sigma,M_\sigma})\bigr\}$. But by an argument stated in Perchet (2011b),

$$\sigma \ \longmapsto \ \Bigl\{\boldsymbol{q} \in \Delta(\mathcal{J}) \text{ such that } \widetilde{H}(\boldsymbol{q}) = \sigma \Bigr\}$$

evolves in a piecewise linear way and thus there exist a finite number K of piecewise linear functions $\sigma \mapsto q'_{\sigma,k}$, with $k = 1, \ldots, K$, such that, for all $\sigma \in \mathcal{F}$,

$$\bigl\{\boldsymbol{q}_{\sigma,1}, \ldots, \boldsymbol{q}_{\sigma,K_\sigma}\bigr\} = \bigl\{\boldsymbol{q}'_{\sigma,1}, \ldots, \boldsymbol{q}'_{\sigma,K}\bigr\}.$$

(There can be some redundancies between the $\boldsymbol{q}'_{\sigma,k}$.) Because of this, we have that for all $\boldsymbol{p} \in \Delta(\mathcal{I})$ and $\sigma \in \mathcal{F}$,

$$\rho(\boldsymbol{p}, \sigma) = \min\bigl\{r(\boldsymbol{p}, \boldsymbol{q}'_{\sigma,1}), \ldots, r(\boldsymbol{p}, \boldsymbol{q}'_{\sigma,K})\bigr\}.$$

Each function $\sigma \mapsto \boldsymbol{q}'_{\sigma,k}$ being piecewise linear, one can construct a finite set $\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{\tilde{K}}\} \subset \Delta(\mathcal{I})$ such that, for any $\sigma \in \mathcal{F}$, the mapping $\boldsymbol{p} \mapsto \rho(\boldsymbol{p}, \sigma)$ is maximized at one of these $\boldsymbol{p}_k$. The convex set $\mathcal{C}$ is therefore defined by a finite number of piecewise linear equations, it is a polyhedron. Its lifted image $\widetilde{\mathcal{C}}$ is a then a polytope: thus, the projection onto it, hence the computation of the proposed strategy, can be done efficiently.

6.2 Internal / Swap Regret

Foster and Vohra (1999) defined internal regret with full monitoring as follows. A player has no internal regret if, for every action $i \in \mathcal{I}$, he has no external regret on the stages when this specific action i was played (if there are enough such stages). In other words, i is the best response to the empirical distribution of actions of the other player on these stages.

With partial monitoring, the first player evaluates his payoffs in a pessimistic way through the function ρ defined in (17). This function is not linear over $\Delta(\mathcal{I})$ in general (it is concave), so that the best responses are not necessarily pure actions $i \in \mathcal{I}$ but mixed

actions, i.e., elements of $\Delta(\mathcal{I})$. Following Lehrer and Solan (2007) one therefore can partition the stages not depending on the pure actions actually played but on the mixed actions $\boldsymbol{p}_t \in \Delta(\mathcal{I})$ used to draw them. To this end, it is convenient to assume that the strategies of the first player need to pick these mixed actions in a finite grid of $\Delta(\mathcal{I})$, which we denote by $\{\boldsymbol{p}_g,\ g \in \mathcal{G}\}$, where $\mathcal{G}$ is a finite set. At each round t, the first player picks an index $G_t \in \mathcal{G}$ and uses the distribution $\boldsymbol{p}_{G_t}$ to draw his action I_t. A discussion about the choice of $\mathcal{G}$ is provided below. For now, we define formally $\mathcal{G}$–internal regret as internal regret with respect to the set of mixed actions $\mathcal{G}$.

Up to a standard concentration-of-the-measure argument, we will measure the payoff at round t with $r\big(\boldsymbol{p}_{G_t}, J_t\big)$ rather than with $r(I_t, J_t)$. For each $g \in \mathcal{G}$, we denote by $N_T(g)$ the number of stages in $\{1, \dots, T\}$ for which we had $G_t = g$ and, whenever $N_T(g) > 0$,

$$\widehat{\boldsymbol{q}}_{T,g} = \frac{1}{N_T(g)} \sum_{t: G_t = g} \delta_{J_t}\,.$$

We define $\widehat{\boldsymbol{q}}_{T,g}$ in an arbitrary way when $N_T(g) = 0$. The $\mathcal{G}$–internal regret of the first player at round T is measured as

$$R_T^{\mathrm{int}} = \max_{g,g' \in \mathcal{G}} \frac{N_T(g)}{T} \left(\rho\Big(\boldsymbol{p}_{g'}, \widetilde{H}\big(\widehat{\boldsymbol{q}}_{T,g}\big)\Big) - r\big(\boldsymbol{p}_g, \widehat{\boldsymbol{q}}_{T,g}\big) \right).$$

Actually, our proof technique rather leads to the minimization of some $\mathcal{G}$–swap regret (see Blum and Mansour, 2007, for the definition of swap regret in full monitoring):

$$R_T^{\mathrm{swap}} = \sum_{g \in \mathcal{G}} \frac{N_T(g)}{T} \left(\max_{g' \in \mathcal{G}} \rho\Big(\boldsymbol{p}_{g'}, \widetilde{H}\big(\widehat{\boldsymbol{q}}_{T,g}\big)\Big) - r\big(\boldsymbol{p}_g, \widehat{\boldsymbol{q}}_{T,g}\big) \right)_+ .$$

At first sight, to handle all possible alternatives one should take $\mathcal{G}$ as a thin grid in $\Delta(\mathcal{I})$, i.e., some ε–discretization of the latter. This is what Lehrer and Solan (2007) do. However, Perchet (2011b) showed that there exists a finite subset $\mathcal{G}_0$ of $\Delta(\mathcal{I})$ such that $\mathcal{G}_0$ contains a best response to any mixed action of the second player: for all $\boldsymbol{q} \in \Delta(\mathcal{J})$,

$$\left(\operatorname*{argmin}_{\boldsymbol{p} \in \Delta(\mathcal{I})}\ \rho\big(\boldsymbol{p},\, \widetilde{H}(\boldsymbol{q})\big) \right) \cap \mathcal{G}_0 \neq \emptyset\,.$$

The strategy we discuss below will have a complexity polynomial in the size of $\mathcal{G}$. We thus advise to take $\mathcal{G} = \mathcal{G}_0$ for the sake of efficiency.

Again, the following bound on the swap regret easily follows from Theorem 24. The latter constructs a simple and direct strategy to control the swap regret, thus also the internal regret. It therefore improves on the results of Lehrer and Solan (2007) and Perchet (2009, 2011b), three papers that presented more involved and less efficient strategies to do so. These strategies were indeed based on auxiliary strategies using thin grids that need to be refined over time; this resulted in complexities that were at least exponential in the number of rounds. (The ideas used therein bear some resemblance with what is done in calibration, see the references provided in Section 4.) In contrast, our strategy can have a constant per-round complexity (when used with the grid $\mathcal{G}_0$). This is a major improvement in efficiency. However, as far as convergence rates are concerned, we must note that again, as in the case of external regret, Perchet (2011b) obtained rates of the faster order $T^{-1/3}$, for an ad hoc (inefficient) strategy. We thus sacrifice efficiency for rates.

Corollary 32 *The first player has an explicit strategy such that for all strategies of the second player, for all T and all $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$R_T^{\mathrm{swap}} \leqslant \Xi \left(T^{-1/5} \sqrt{\ln \frac{T}{\delta}} + T^{-2/5} \ln \frac{T}{\delta} \right)$$

for some constant Ξ depending only on the game (r, H) at hand and on the size of the finite grid $\mathcal{G}$.

Proof The proof of this corollary is based on ideas similar to the ones used in the proof of Corollary 31; $\mathcal{G}$ will play the role of the action set of the first player. The proof proceeds in four steps. In the first step, we construct an approachability setup and show that Condition (APM) applies. In the second step, we show that Assumption 1 is satisfied. In the third step we analyze the convergence rates of the swap regret. In the fourth and final step, we show that the set we are approaching possesses some smoothness properties by providing a uniform Lipschitz bound on certain functions.

Step 1: We denote by

$$\mathcal{F}_{\mathrm{cone}} = \big\{ \lambda\sigma, \quad \sigma \in \mathcal{F}, \ \lambda \in \mathbb{R}_+ \big\}$$

the cone generated by $\mathcal{F}$ and extend linearly $\rho : \Delta(\mathcal{I}) \times \mathcal{F} \to \mathbb{R}$ into a mapping $\rho : \Delta(\mathcal{I}) \times \mathcal{F}_{\mathrm{cone}} \to R$ as follows: for all $\boldsymbol{p} \in \Delta(\mathcal{I})$, for all $\lambda \geqslant 0$ with $\lambda \neq 1$, and all $\sigma \in \mathcal{F}$,

$$\rho(\boldsymbol{p}, \lambda\sigma) = \begin{cases} 0 & \text{if } \lambda = 0, \\ \lambda\, \rho(\boldsymbol{p}, \sigma) & \text{if } \lambda > 0. \end{cases}$$

In the sequel, we embed $\mathcal{F}_{\mathrm{cone}}$ into $\mathbb{R}^{\mathcal{I}\times\mathcal{H}}$.

The closed convex set $\mathcal{C}$ and the vector-valued payoff function $\underline{r}$ are then respectively defined by

$$\mathcal{C} = \left\{ (z_g, \boldsymbol{v}_g)_{g\in\mathcal{G}} \in \big(\mathbb{R} \times \mathcal{F}_{\mathrm{cone}}\big)^{\mathcal{G}} : \quad \forall\, g \in \mathcal{G}, \quad z_g \geqslant \max_{g' \in \mathcal{G}} \rho\big(\boldsymbol{p}_{g'}, \boldsymbol{v}_g\big) \right\}$$

and, for all $(g, j) \in \mathcal{G} \times \mathcal{J}$,

$$\underline{r}(g,j) = \left[\begin{array}{c} r\big(\boldsymbol{p}_g, j\big)\, \mathbb{I}_{\{g'=g\}} \\ \widetilde{H}(\delta_j)\, \mathbb{I}_{\{g'=g\}} \end{array} \right]_{g' \in \mathcal{G}}.$$

To show that $\mathcal{C}$ is $\underline{r}$–approachable, we associate with each $\boldsymbol{q} \in \Delta(\mathcal{J})$ an element $g^\star(\boldsymbol{q}) \in \mathcal{G}$ such that

$$g^\star(\boldsymbol{q}) \in \operatorname*{argmax}_{g\in\mathcal{G}} \rho\big(\boldsymbol{p}_g, \widetilde{H}(\boldsymbol{q})\big)\,.$$

Then, given any $\boldsymbol{q} \in \Delta(\mathcal{J})$, we note that for all $\boldsymbol{q}'$ satisfying $\widetilde{H}(\boldsymbol{q}') = \widetilde{H}(\boldsymbol{q})$, the components of the vector $\underline{r}\big(g^\star(\boldsymbol{q}), \boldsymbol{q}'\big)$ are all null but the ones corresponding to $g^\star(\boldsymbol{q})$, for which we have

$$\begin{aligned} &r\big(\boldsymbol{p}_{g^\star(\boldsymbol{q})}, \boldsymbol{q}'\big) \\ &\quad \geqslant \rho\Big(\boldsymbol{p}_{g^\star(\boldsymbol{q})}, \widetilde{H}\big(\boldsymbol{q}'\big)\Big) = \rho\Big(\boldsymbol{p}_{g^\star(\boldsymbol{q})}, \widetilde{H}\big(\boldsymbol{q}\big)\Big) = \max_{g'\in\mathcal{G}} \rho\Big(\boldsymbol{p}_{g'}, \widetilde{H}\big(\boldsymbol{q}\big)\Big) = \max_{g'\in\mathcal{G}} \rho\Big(\boldsymbol{p}_{g'}, \widetilde{H}\big(\boldsymbol{q}'\big)\Big)\,, \end{aligned}$$

where the first inequality is by definition of ρ. Therefore, $\underline{r}\big(g^\star(\boldsymbol{q}), \boldsymbol{q}'\big) \in \mathcal{C}$. Condition (APM) in Lemma 22 and Theorem 24 is thus satisfied, so that we have approachability.

Step 2: We then show that Assumption 1 is satisfied. It suffices to show that for all $\sigma \in \mathcal{F}$, the mapping

$$\pi = (\pi_g)_{g\in\mathcal{G}} \in \Delta(\mathcal{G}) \;\longmapsto\; \overline{m}_1(\pi,\sigma) = \Big\{ \big(\pi_g\, r(\boldsymbol{p}_g, \boldsymbol{q})\big)_{g\in\mathcal{G}} : \quad \boldsymbol{q} \in \Delta(\mathcal{J}) \text{ such that } \widetilde{H}(\boldsymbol{q}) = \sigma \Big\}$$

is piecewise linear (as the other components in the definition of $\overline{m}$ are linear in π). This is the case since for each g, the mapping

$$\pi \in \Delta(\mathcal{G}) \;\longmapsto\; \Big\{ \pi_g\, r(\boldsymbol{p}_g, \boldsymbol{q}) : \quad \boldsymbol{q} \in \Delta(\mathcal{J}) \text{ such that } \widetilde{H}(\boldsymbol{q}) = \sigma \Big\}$$

is seen to be piecewise linear, by using the same one-dimensional argument as the one stated around (16) and also used in the proof of Corollary 31.

Step 3: We now exhibit the convergence rates. In view of the form of the defining set of constraints for $\mathcal{C}$, the coordinates of the elements in $\mathcal{C}$ can be grouped according to each $g \in \mathcal{G}$ and projections onto $\mathcal{C}$ can therefore be done separately for each such subset of coordinates. The subset of coordinates of $\sum_{t=1}^T \underline{r}(G_t, J_t)/T$ corresponding to a given g is formed by

$$\frac{N_T(g)}{T}\, r\big(\boldsymbol{p}_g, \widehat{\boldsymbol{q}}_{T,g}\big) \qquad \text{and} \qquad \frac{N_T(g)}{T}\, \widetilde{H}\big(\widehat{\boldsymbol{q}}_{T,g}\big)\,.$$

When

$$\frac{N_T(g)}{T}\, r\big(\boldsymbol{p}_g, \widehat{\boldsymbol{q}}_{T,g}\big) \geqslant \max_{g'\in\mathcal{G}}\; \rho\bigg(\boldsymbol{p}_{g'},\, \frac{N_T(g)}{T}\, \widetilde{H}\big(\widehat{\boldsymbol{q}}_{T,g}\big)\bigg)\,,$$

we denote these quantities by $\widetilde{r}_{T,g}$ and $\widetilde{\boldsymbol{v}}_{T,g}$. Otherwise, we project this pair on the set

$$\mathcal{C}_g = \Big\{ (z_g, \boldsymbol{v}_g) \in \mathbb{R} \times \mathcal{F}_{\text{cone}} : \quad z_g \geqslant \max_{g'\in\mathcal{G}} \rho\big(\boldsymbol{p}_{g'}, \boldsymbol{v}_g\big) \Big\}$$

and denote by $\widetilde{r}_{T,g}$ and $\widetilde{\boldsymbol{v}}_{T,g}$ the coordinates of the projection; they satisfy the defining inequality of $\mathcal{C}_g$ with equality,

$$\widetilde{r}_{T,g} = \max_{g'\in\mathcal{G}} \rho\big(\boldsymbol{p}_{g'}, \widetilde{\boldsymbol{v}}_{T,g}\big)\,.$$

By distinguishing for each g according to which of the two cases above arose (for the first inequality), we may decompose and upper bound the swap regret as follows,

$$\begin{aligned}
&R_T^{\text{swap}} \\
&= \sum_{g\in\mathcal{G}} \frac{N_T(g)}{T} \bigg(\max_{g'\in\mathcal{G}} \rho\Big(\boldsymbol{p}_{g'}, \widetilde{H}\big(\widehat{\boldsymbol{q}}_{T,g}\big)\Big) - r\big(\boldsymbol{p}_g, \widehat{\boldsymbol{q}}_{T,g}\big) \bigg)_+ \\
&= \sum_{g\in\mathcal{G}} \bigg(\max_{g'\in\mathcal{G}} \rho\bigg(\boldsymbol{p}_{g'}, \frac{N_T(g)}{T}\, \widetilde{H}\big(\widehat{\boldsymbol{q}}_{T,g}\big)\bigg) - \frac{N_T(g)}{T}\, r\big(\boldsymbol{p}_g, \widehat{\boldsymbol{q}}_{T,g}\big) \bigg)_+ \\
&\leqslant \sum_{g\in\mathcal{G}} \bigg| \max_{g'\in\mathcal{G}} \rho\bigg(\boldsymbol{p}_{g'}, \frac{N_T(g)}{T}\, \widetilde{H}\big(\widehat{\boldsymbol{q}}_{T,g}\big)\bigg) - \max_{g'\in\mathcal{G}} \rho\big(\boldsymbol{p}_{g'}, \widetilde{\boldsymbol{v}}_{g,T}\big) \bigg| + \sum_{g\in\mathcal{G}} \bigg| \widetilde{r}_{T,g} - \frac{N_T(g)}{T}\, r\big(\boldsymbol{p}_g, \widehat{\boldsymbol{q}}_{T,g}\big) \bigg| \\
&\leqslant \sum_{g\in\mathcal{G}} \overline{L}_\rho \bigg\| \frac{N_T(g)}{T}\, \widetilde{H}\big(\widehat{\boldsymbol{q}}_{T,g}\big) - \widetilde{\boldsymbol{v}}_{g,T} \bigg\|_2 + \sum_{g\in\mathcal{G}} \bigg| \widetilde{r}_{T,g} - \frac{N_T(g)}{T}\, r\big(\boldsymbol{p}_g, \widehat{\boldsymbol{q}}_{T,g}\big) \bigg|\,,
\end{aligned}$$

where we used a fact proved in step 4, that the mapping

$$\boldsymbol{v} \in \mathcal{F}_{\text{cone}} \longmapsto \max_{g' \in \mathcal{G}} \rho\big(\boldsymbol{p}_{g'}, \boldsymbol{v}\big) \tag{18}$$

is $\overline{L}_\rho$–Lipschitz. In the last inequality we had a sum of ℓ^2–norms, which can be bounded by a single ℓ^2–norm,

$$\begin{aligned} R_T^{\text{swap}} &\leqslant \max\big\{\overline{L}_\rho, 1\big\} \sqrt{2N_{\mathcal{G}}} \left\| \left[\begin{array}{c} \widetilde{r}_{T,g} \\ \widetilde{\boldsymbol{v}}_{T,g} \end{array} \right]_{g \in \mathcal{G}} - \frac{1}{T} \sum_{t=1}^T \underline{r}(I_t, J_t) \right\|_2 \\ &\leqslant \max\big\{\overline{L}_\rho, 1\big\} \sqrt{2N_{\mathcal{G}}} \inf_{c \in \mathcal{C}} \left\| c - \frac{1}{T} \sum_{t=1}^T \underline{r}(I_t, J_t) \right\|_2 , \end{aligned}$$

where we denoted by $N_{\mathcal{G}}$ the cardinality of $\mathcal{G}$. Resorting to the convergence rate stated in Theorem 26 concludes the proof.

Step 4: It only remains to prove the claimed Lipschitzness of the mapping (18). (All Lipschitzness statements that follow will be with respect to the ℓ^2–norms.) To do so, it suffices to show that for all fixed elements $\boldsymbol{p} \in \Delta(\mathcal{I})$, the functions $\boldsymbol{v} \in \mathcal{F}_{\text{cone}} \mapsto \rho(\boldsymbol{p}, \boldsymbol{v})$ are Lipschitz, with a Lipschitz constant $\overline{L}_\rho$ that is independent of $\boldsymbol{p}$. Note that we already proved at the end of the proof of Corollary 31 that $\sigma \in \mathcal{F} \mapsto \rho(\boldsymbol{p}, \sigma)$ is Lipschitz, with a Lipschitz constant L_ρ independent of $\boldsymbol{p}$. Consider now two elements $\boldsymbol{v}$, $\boldsymbol{v}' \in \mathcal{F}_{\text{cone}}$, which we write as $\boldsymbol{v} = \lambda\sigma$ and $\boldsymbol{v}' = \lambda'\sigma'$, with σ, $\sigma' \in \mathcal{F}$ and λ, $\lambda' \in \mathbb{R}_+$. Using triangle inequalities, the Lipschitzness of ρ on $\mathcal{F}$, and the fact that r thus ρ are bounded by R,

$$\begin{aligned} \big|\rho(\boldsymbol{p}, \lambda\sigma) - \rho(\boldsymbol{p}, \lambda'\sigma')\big| &\leqslant \big|\lambda\big(\rho(\boldsymbol{p}, \sigma) - \rho(\boldsymbol{p}, \sigma')\big)\big| + \big|(\lambda - \lambda')\rho(\boldsymbol{p}, \sigma')\big| \\ &\leqslant \lambda\, L_\rho \big\| \sigma - \sigma' \big\|_2 + R \big|\lambda - \lambda'\big| \\ &\leqslant L_\rho \big\| \lambda\sigma - \lambda'\sigma' + (\lambda' - \lambda)\sigma' \big\|_2 + R \big|\lambda - \lambda'\big| \\ &\leqslant L_\rho \big\| \lambda\sigma - \lambda'\sigma' \big\|_2 + \big(R + L_\rho N_{\mathcal{I}}\big) \big|\lambda - \lambda'\big| , \end{aligned}$$

where we used also for the last inequality that since σ is a vector of $N_{\mathcal{I}}$ probability distributions over the signals, $\|\sigma\|_2 \leqslant \|\sigma\|_1 = N_{\mathcal{I}}$. To conclude the argument, we simply need to show that $\big|\lambda - \lambda'\big|$ can be bounded by $\big\| \lambda\sigma - \lambda'\sigma' \big\|_2$ up to some universal constant, which we do now. We resort again to the fact that $\|\sigma\|_1 = \|\sigma'\|_1 = N_{\mathcal{I}}$ and can thus write, thanks to a triangle inequality and assuming with no loss of generality that $\lambda' < \lambda$, that

$$\big|\lambda - \lambda'\big| = \frac{1}{N_{\mathcal{I}}} \Big(\lambda \|\sigma\|_1 - \lambda' \big\| \sigma' \big\|_1 \Big) \leqslant \frac{1}{N_{\mathcal{I}}} \big\| \lambda\sigma - \lambda'\sigma' \big\|_1 \leqslant \frac{\sqrt{N_{\mathcal{H}} N_{\mathcal{I}}}}{N_{\mathcal{I}}} \big\| \lambda\sigma - \lambda'\sigma' \big\|_2 ,$$

where we used the Cauchy-Schwarz inequality for the final step. One can thus take, for instance,

$$\overline{L}_\rho = L_\rho + \big(R + L_\rho N_{\mathcal{I}}\big) \sqrt{\frac{N_{\mathcal{H}}}{N_{\mathcal{I}}}} .$$

This concludes the proof. ■

7. Summary of the Results

This paper extended Blackwell's classical approachability theory to the case where set-valued functions are considered, which models ambiguity in the obtained reward. In the case of mixed actions taken, this extension was provided in the case of linear (Section 3) and concave–convex (Section 4) set-valued functions; only in the former case efficient strategies (up to a projection oracle) could be constructed.

The second part of this paper (Section 5) applies this theory of set-valued approachability to approachability with partial monitoring. The necessary and sufficient Condition (APM) for this was exhibited by Perchet (2011a) and was recalled in Section 5.1; its link with the necessary and sufficient condition for set-valued approachability was discussed in Section 5.2. Then, under a so-called assumption of bi-piecewise linearity of the game (r, H) at hand, an efficient strategy (up to a projection oracle) was constructed and studied in Section 5.3, for the approachability of any closed convex set $\mathcal{C}$. Alternatively, Section 5.4 showed that for any game (r, H) at hand but under the constraint that the target set $\mathcal{C}$ is a polytope, the above efficient construction could still be used. In both cases, the novelty also relies not only the gained efficiency with respect to the construction by Perchet (2011a) but also on getting for the first time rates of convergence that are independent of the ambient dimension. The case of any game (r, H) and any closed target set $\mathcal{C}$ was discussed, for the sake of completeness, at then end of Section 5.4, so that the present article contains a complete and self-contained constructive proof of the sufficiency of Condition (APM).

Finally, Section 6 showed that the well-studied case of regret minimization, a special case of approachability, could fall under the umbrella of bi-piecewise linearity, and hence be performed efficiently, as was already known.

Acknowledgments

Shie Mannor was partially supported by the ISF under contract 890015 and the Google Inter-university center for Electronic Markets and Auctions. Vianney Perchet benefited from the support of the ANR under grants ANR-10-BLAN 0112 and ANR-13-JS01-0004-01. Gilles Stoltz acknowledges support from Investissements d'Avenir (ANR-11-IDEX-0003/Labex Ecodec/ANR-11-LABX-0047).

An extended abstract of this paper appeared in the *Proceedings of the 24th Annual Conference on Learning Theory* (COLT'11), JMLR Workshop and Conference Proceedings, Volume 19, pages 515–536, 2011.

Appendix A. Proof of Theorem 15

Proof [of the second statement of Theorem 15] The proof of Corollary 7 extends to the case considered here and shows, thanks to the ad hoc consideration of the result stated in Lemma 6 as following from Definition 13, that for all $\boldsymbol{y} \in \Delta(\mathcal{B})$, the mapping $D_{\boldsymbol{y}}$ is still continuous over $\Delta(\mathcal{A})$. We now proceed by contradiction and assume that (SVAC) is not satisfied. The first part of the proof of the necessity of (SVAC) in Theorem 8 also applies to the present case: there exists $\boldsymbol{y}_0$ such that $D_{\boldsymbol{y}_0} \geqslant D_{\min} > 0$ over $\Delta(\mathcal{A})$. It then suffices to note that whenever the second player resorts to $\boldsymbol{y}_t = \boldsymbol{y}_0$ at all rounds $t \geqslant 1$, then for all

strategies of the first player, the quantity of interest in the set-valued approachability can be lower bounded as follows. Thanks to the concavity in the first argument,

$$\sup\left\{\inf_{c\in\mathcal{C}}\ \|\xi-c\|_2:\quad \xi\in\frac{1}{T}\sum_{t=1}^{T}\overline{m}(\boldsymbol{x}_t,\boldsymbol{y}_0)\right\}$$
$$\geqslant \sup\left\{\inf_{c\in\mathcal{C}}\ \|\xi-c\|_2:\quad \xi\in\overline{m}\left(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t,\ \boldsymbol{y}_0\right)\right\}=D_{\boldsymbol{y}_0}\left(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t\right)\geqslant D_{\min}>0\,.$$

Therefore, $\mathcal{C}$ is $\overline{m}$–approachable by no strategy of the first player. ■

The proof of the first statement of Theorem 15 relies on the use of approximately calibrated strategies of the first player, as introduced and studied (among others) by Dawid (1982), Foster and Vohra (1998), Mannor and Stoltz (2010). Formally, given $\eta>0$, an η–calibrated strategy of the first player considers some finite covering of $\Delta(\mathcal{B})$ by N_η balls of radius η and abides by the following constraints. Denoting by $\boldsymbol{y}^1,\dots,\boldsymbol{y}^{N_\eta}$ the centers of the balls in the covering (they form what will be referred to later on as an η–grid), such a strategy chooses only forecasts in $\left\{\boldsymbol{y}^1,\dots,\boldsymbol{y}^{N_\eta}\right\}$. We thus denote by L_t the index chosen in $\left\{1,\dots,N_\eta\right\}$ at round t and by

$$N_T(\ell)=\sum_{t=1}^{T}\mathbb{I}_{\{L_t=\ell\}}$$

the total number of rounds within the first T ones when the element ℓ of the grid was chosen. We denote by $(\,\cdot\,)_+$ the function that gives the nonnegative part of a real number. The final condition to be satisfied is that for all $\delta\in(0,1)$, there exists an integer T_δ such that for all strategies of the second player, with probability at least $1-\delta$, for all $T\geqslant T_\delta$,

$$\sum_{\ell=1}^{N_\eta}\frac{N_T(\ell)}{T}\left(\left\|\boldsymbol{y}^\ell-\frac{1}{N_T(\ell)}\sum_{t=1}^{T}\boldsymbol{y}_t\mathbb{I}_{\{L_t=\ell\}}\right\|_1-\eta\right)_+\leqslant\delta\,. \tag{19}$$

This calibration criterion is slightly stronger than the classical η–calibration score usually considered in the literature, which consists of omitting nonnegative parts in the criterion above and ensuring that for all strategies of the second player, with probability at least $1-\delta$, for all $T\geqslant T_\delta$,

$$\sum_{\ell=1}^{N_\eta}\frac{N_T(\ell)}{T}\left\|\boldsymbol{y}^\ell-\frac{1}{N_T(\ell)}\sum_{t=1}^{T}\boldsymbol{y}_t\mathbb{I}_{\{L_t=\ell\}}\right\|_1\leqslant\eta+\delta\,. \tag{20}$$

The existence of a calibrated strategy in the sense of (19) however follows from the same approachability-based construction studied in Mannor and Stoltz (2010) to get (20) and is detailed below in Section B. In the sequel we will only use the following consequence of calibration: that for all strategies of the second player, with probability at least $1-\delta$, for all $T\geqslant T_\delta$,

$$\max_{\ell=1,\dots,N_\eta}\frac{N_T(\ell)}{T}\left(\left\|\boldsymbol{y}^\ell-\frac{1}{N_T(\ell)}\sum_{t=1}^{T}\boldsymbol{y}_t\mathbb{I}_{\{L_t=\ell\}}\right\|_1-\eta\right)_+\leqslant\delta\,. \tag{21}$$

Proof [of the first statement of Theorem 15] The insight of this proof is similar to the one illustrated in Perchet (2009). We first note that it suffices to prove that for all $\varepsilon > 0$, the set $\mathcal{C}_\varepsilon$ defined as the ε–neighborhood of $\mathcal{C}$ is $\overline{m}$–approachable. This is so up to proceeding in regimes $r = 1, 2, \ldots$ each corresponding to a dyadic value $\varepsilon_r = 2^{-r}$ and lasting for a number of rounds carefully chosen in terms of the length of the previous regimes.

Therefore, we fix $\varepsilon > 0$ and associate with it a modulus of continuity $\eta > 0$ given by the uniform continuity of $\overline{m}$ in its second argument. We consider an $\eta/2$–calibrated strategy of the first player, which we will use as an auxiliary strategy. Since (SVAC) is satisfied, we may associate with each element $\boldsymbol{y}^\ell$ of the underlying $\eta/2$–grid a mixed action $\boldsymbol{x}^\ell \in \Delta(\mathcal{A})$ such that $\overline{m}\big(\boldsymbol{x}^\ell, \boldsymbol{y}^\ell\big) \subseteq \mathcal{C}$. The main strategy of the first player then prescribes the use of $\boldsymbol{x}_t = \boldsymbol{x}^{L_t}$ at each round $t \geqslant 1$. The intuition behind this definition is that if $\boldsymbol{y}^{L_t}$ is forecasted by the auxiliary strategy, then since the latter is calibrated, one should play as good as possible against $\boldsymbol{y}^{L_t}$. In view of the aim at hand, which is approaching $\mathcal{C}$, such a good reply is given by $\boldsymbol{x}^{L_t}$.

To assess the constructed strategy, we group rounds according to the values ℓ taken by the L_t. To that end, we recall that $N_T(\ell)$ denotes the number of rounds in which $\boldsymbol{y}^\ell$ was forecasted and $\boldsymbol{x}^\ell$ was played. The average payoff up to round T is then rewritten as

$$\frac{1}{T}\sum_{t=1}^{T} \overline{m}(\boldsymbol{x}_t, \boldsymbol{y}_t) = \sum_{\ell=1}^{N_{\eta/2}} \frac{N_T(\ell)}{T} \left(\frac{1}{N_T(\ell)} \sum_{t=1}^{T} \overline{m}\big(\boldsymbol{x}^\ell, \boldsymbol{y}_t\big) \mathbb{I}_{\{L_t = \ell\}} \right) .$$

We denote for all ℓ such that $N_T(\ell) > 0$ the average of their corresponding mixed actions $\boldsymbol{y}_t$ by

$$\overline{\boldsymbol{y}}_T^\ell = \frac{1}{N_T(\ell)} \sum_{t=1}^{T} \boldsymbol{y}_t \mathbb{I}_{\{L_t = \ell\}} .$$

The convexity of $\overline{m}$ in its second argument leads to the inclusion

$$\frac{1}{T}\sum_{t=1}^{T} \overline{m}(\boldsymbol{x}_t, \boldsymbol{y}_t) = \sum_{\ell=1}^{N_{\eta/2}} \frac{N_T(\ell)}{T} \left(\frac{1}{N_T(\ell)} \sum_{t=1}^{T} \overline{m}\big(\boldsymbol{x}^\ell, \boldsymbol{y}_t\big) \mathbb{I}_{\{L_t = \ell\}} \right) \subseteq \sum_{\ell=1}^{N_{\eta/2}} \frac{N_T(\ell)}{T}\, \overline{m}\big(\boldsymbol{x}^\ell, \overline{\boldsymbol{y}}_T^\ell\big) .$$

We recall that $\boldsymbol{B}$ denotes the unit Euclidean ball in $\mathbb{R}^d$. To show that the above-defined strategy $\overline{m}$–approaches $\mathcal{C}_\varepsilon = \mathcal{C} + \varepsilon \boldsymbol{B}$, it suffices to show that for all $\delta \in (0,1)$, there exists an integer T'_δ such that for all strategies of the second player,

$$\mathbb{P}\left\{ \forall\, T \geqslant T'_\delta, \quad \sum_{\ell=1}^{N_{\eta/2}} \frac{N_T(\ell)}{T}\, \overline{m}\big(\boldsymbol{x}^\ell, \overline{\boldsymbol{y}}_T^\ell\big) \subseteq \mathcal{C} + (\varepsilon + \delta)\boldsymbol{B} \right\} \geqslant 1 - \delta .$$

We denote by M a bound in ℓ^2–norm on $\overline{m}$, i.e., for all $\boldsymbol{x} \in \Delta(\mathcal{A})$ and $\boldsymbol{y} \in \Delta(\mathcal{B})$, the inclusion $\overline{m}(\boldsymbol{x}, \boldsymbol{y}) \subseteq M\boldsymbol{B}$ holds. We let $\delta' = \delta(\eta/2)/\big(M\, N_{\eta/2}\big)$ and define T'_δ as the time $T_{\delta'}$ corresponding to (21). All statements that follow will be for all strategies of the second player and with probability at least $1 - \delta' \geqslant 1 - \delta$, for all $T \geqslant T'_\delta$, as required. For each index

ℓ of the grid, either $\delta' T/N_T(\ell) \leqslant \eta/2$ or $\delta' T/N_T(\ell) > \eta/2$. In the first case, following (21), $\left\| \boldsymbol{y}^\ell - \overline{\boldsymbol{y}}_T^\ell \right\| \leqslant \eta/2 + \delta' T/N_T(\ell) \leqslant \eta$; since η is the modulus of continuity for ε, we get that

$$\frac{N_T(\ell)}{T}\,\overline{m}\big(\boldsymbol{x}^\ell, \overline{\boldsymbol{y}}_T^\ell\big) \subseteq \frac{N_T(\ell)}{T}\left(\overline{m}\big(\boldsymbol{x}^\ell, \boldsymbol{y}^\ell\big) + \varepsilon \boldsymbol{B}\right) \subseteq \frac{N_T(\ell)}{T}\big(\mathcal{C} + \varepsilon \boldsymbol{B}\big)\,,$$

where we used the definition of $\boldsymbol{x}^\ell$ to get the second inclusion. In the second case, using the boundedness of $\overline{m}$, we simply write

$$\frac{N_T(\ell)}{T}\,\overline{m}\big(\boldsymbol{x}^\ell, \overline{\boldsymbol{y}}_T^\ell\big) \subseteq \frac{N_T(\ell)}{T}\,M\boldsymbol{B} \subseteq \frac{\delta'}{\eta/2}\,M\boldsymbol{B}\,.$$

Summing these bounds over ℓ yields

$$\sum_{\ell=1}^{N_{\eta/2}} \frac{N_T(\ell)}{T}\,\overline{m}\big(\boldsymbol{x}^\ell, \overline{\boldsymbol{y}}_T^\ell\big) \subseteq \mathcal{C} + \varepsilon\boldsymbol{B} + \frac{N_{\eta/2}\delta'}{\eta/2}\,M\,\boldsymbol{B} = \mathcal{C} + (\varepsilon + \delta)\,\boldsymbol{B}\,,$$

where we used the definition of δ' in terms of δ. This concludes the proof. ∎

Appendix B. An Auxiliary Result of Calibration

We prove here (19) for a given $\eta > 0$ and do so by following closely the methodology of Mannor and Stoltz (2010). (Note that this result is of independent interest.)

We actually assume that the covering $\boldsymbol{y}^1, \ldots, \boldsymbol{y}^{N_\eta}$ is slightly finer than what was required around (19) and that it forms an $\eta/N_{\mathcal{B}}$–grid of $\Delta(\mathcal{B})$, i.e., that for all $\boldsymbol{y} \in \Delta(\mathcal{B})$, there exists $\ell \in \{1, \ldots, N_\eta\}$ such that $\left\| \boldsymbol{y} - \boldsymbol{y}^\ell \right\|_1 \leqslant \eta/N_{\mathcal{B}}$.

We recall that elements $\boldsymbol{y} \in \mathcal{B}$ are denoted by $\boldsymbol{y} = (y_b)_{b\in\mathcal{B}}$ and we identify $\Delta(\mathcal{B})$ with a subset of $\mathbb{R}^{N_\mathcal{B}}$. In particular, $\mathbb{I}_b$, the Dirac mass on a given $b \in \mathcal{B}$, is a binary vector whose only non-null component is the one indexed by b. Finally, we denote by

$$\underline{0} = (0, \ldots, 0) \qquad \text{and} \qquad \underline{1} = (1, \ldots, 1)$$

the elements of $\mathbb{R}^\mathcal{B}$ respectively formed by zeros and ones only.

We consider a vector-valued payoff function $C : \{1, \ldots, N_\eta\} \times \mathcal{B} \to \mathbb{R}^{2N_\eta N_\mathcal{B}}$ defined as follows; for all $\ell \in \{1, \ldots, N_\eta\}$ and for all $b \in \mathcal{B}$,

$$C(\ell, b) = \left(\underline{0},\ \ldots,\ \underline{0},\ \ \boldsymbol{y}^\ell - \mathbb{I}_b - \frac{\eta}{N_\mathcal{B}}\underline{1},\ \ \mathbb{I}_b - \boldsymbol{y}^\ell - \frac{\eta}{N_\mathcal{B}}\underline{1},\ \ \underline{0},\ \ldots,\ \underline{0}\right),$$

which is a vector of $2N_\eta$ elements of $\mathbb{R}^\mathcal{B}$ composed by $2(N_\eta - 1)$ occurrences of the zero element $\underline{0} \in \mathbb{R}^\mathcal{B}$ and two non-zero elements, located in the positions indexed by $2\ell - 1$ and 2ℓ.

We now show that the closed convex set $(\mathbb{R}_-)^{2N_\eta N_\mathcal{B}}$ is C–approachable; to do so, we resort to the characterization stated in Theorem 2. To each $\boldsymbol{y} \in \Delta(\mathcal{B})$ we will associate a pure action $\ell_{\boldsymbol{y}}$ in $\{1, \ldots, N_\eta\}$ so that $C\big(\ell_{\boldsymbol{y}}, \boldsymbol{y}\big) \in (\mathbb{R}_-)^{2N_\eta N_\mathcal{B}}$; note that to satisfy the necessary and sufficient condition, it is not necessary here to resort to mixed actions of the

first player. The index $\ell_{\boldsymbol{y}}$ is any index ℓ such that $\left\| \boldsymbol{y} - \boldsymbol{y}^{\ell} \right\|_1 \leqslant \eta / N_{\mathcal{B}}$; such an index always exists as noted at the beginning of this proof. Indeed, one then has in particular that for each component $b \in \mathcal{B}$,

$$\left| y_b^{\ell_{\boldsymbol{y}}} - y_b \right| \leqslant \left\| \boldsymbol{y}^{\ell_{\boldsymbol{y}}} - \boldsymbol{y} \right\|_1 \leqslant \eta / N_{\mathcal{B}} \,.$$

A straightforward adaptation of the proof of Theorem 3 (see, e.g., Mertens et al., 1994) then yields a strategy such that for all $\delta \in (0,1)$ and for all strategies of the second player, with probability at least $1 - \delta$,

$$\sup_{\tau \geqslant T} \inf_{c \in (\mathbb{R}_-)^{2N_\eta N_{\mathcal{B}}}} \left\| c - \frac{1}{\tau} \sum_{t=1}^{\tau} C(L_t, \boldsymbol{y}_t) \right\|_2 \leqslant 2M \sqrt{\frac{2}{\delta T}} \,, \tag{22}$$

where M is a bound in Euclidean norm over C, e.g., $M = 4 + 2\eta$. The quantities of interest can be rewritten as

$$\frac{1}{\tau} \sum_{t=1}^{\tau} C(L_t, \boldsymbol{y}_t) = \left(\frac{N_\tau(\ell)}{\tau} \big(\boldsymbol{y}^{\ell} - \overline{\boldsymbol{y}}_\tau^{\ell} \big) - \frac{N_\tau(\ell)}{\tau} \frac{\eta}{N_{\mathcal{B}}} \underline{1}, \; \frac{N_\tau(\ell)}{\tau} \big(\overline{\boldsymbol{y}}_\tau^{\ell} - \boldsymbol{y}^{\ell} \big) - \frac{N_\tau(\ell)}{\tau} \frac{\eta}{N_{\mathcal{B}}} \underline{1} \right)_{\ell \in \{1, \dots, N_\eta\}} ,$$

where we recall that we denoted for all ℓ such that $N_\tau(\ell) > 0$ the average of their corresponding mixed actions $\boldsymbol{y}_t$ by

$$\overline{\boldsymbol{y}}_\tau^{\ell} = \frac{1}{N_\tau(\ell)} \sum_{t=1}^{\tau} \boldsymbol{y}_t \mathbb{I}_{\{L_t = \ell\}} \,.$$

The projection in ℓ^2–norm of quantity of interest onto $(\mathbb{R}_-)^{2N_\eta N_{\mathcal{B}}}$ is formed by its non-positive components, so that its square distance to $(\mathbb{R}_-)^{2N_\eta N_{\mathcal{B}}}$ equals

$$\begin{aligned} &\inf_{c \in (\mathbb{R}_-)^{2N_\eta N_{\mathcal{B}}}} \left\| c - \frac{1}{\tau} \sum_{t=1}^{\tau} C(L_t, \boldsymbol{y}_t) \right\|_2^2 \\ &\qquad = \sum_{\ell=1}^{N_\eta} \left(\frac{N_\tau(\ell)}{\tau} \right)^2 \sum_{b \in \mathcal{B}} \underbrace{\left(\left(y_b^{\ell} - \overline{y}_{\tau,b}^{\ell} - \frac{\eta}{N_{\mathcal{B}}} \right)_+^2 + \left(\overline{y}_{\tau,b}^{\ell} - y_b^{\ell} - \frac{\eta}{N_{\mathcal{B}}} \right)_+^2 \right)}_{= \left(|\overline{y}_{\tau,b}^{\ell} - y_b^{\ell}| - \eta / N_{\mathcal{B}} \right)_+^2} . \end{aligned}$$

Therefore, our target is achieved: using the fact that $(\,\cdot\,)_+$ is subadditive first, and then applying the Cauchy-Schwarz inequality,

$$\begin{aligned} \sum_{\ell=1}^{N_\eta} \frac{N_\tau(\ell)}{\tau} \Big(\left\| \boldsymbol{y}^{\ell} - \overline{\boldsymbol{y}}_\tau \right\|_1 - \eta \Big)_+ &\leqslant \sum_{\ell=1}^{N_\eta} \frac{N_\tau(\ell)}{\tau} \sum_{b \in \mathcal{B}} \left(\left| y_b^{\ell} - \overline{y}_{\tau,b}^{\ell} \right| - \frac{\eta}{N_{\mathcal{B}}} \right)_+ \\ &\leqslant \sqrt{N_\eta N_{\mathcal{B}}} \sqrt{\sum_{\ell=1}^{N_\eta} \left(\frac{N_\tau(\ell)}{\tau} \right)^2 \sum_{b \in \mathcal{B}} \left(\left| y_b^{\ell} - \overline{y}_{\tau,b}^{\ell} \right| - \frac{\eta}{N_{\mathcal{B}}} \right)_+^2} \\ &\leqslant 2M \sqrt{N_\eta N_{\mathcal{B}}} \sqrt{\frac{2}{\delta T}} \,, \end{aligned}$$

where the last inequality holds, by (22), for all $\tau \geqslant T$ with probability at least $1-\delta$. Choosing an integer T_δ sufficiently large so that

$$2M\sqrt{N_\eta N_{\mathcal{B}}}\sqrt{\frac{2}{\delta\, T_\delta}} \leqslant \delta$$

concludes the proof of the property stated in (19).

Appendix C. Proof of Lemma 23

Proof For all $(i,j) \in \mathcal{I}\times\mathcal{J}$, the quantity $H(i,j)$ is a probability distribution over the set of signals $\mathcal{H}$; we denote by $H_s(i,j)$ the probability mass that it puts on some signal $s \in \mathcal{H}$.

Equation (7) indicates that for each pair $(i,s) \in \mathcal{I}\times\mathcal{H}$,

$$\sum_{t=(n-1)L+1}^{nL} \left(\frac{\mathbb{I}_{\{S_t=s\}}\mathbb{I}_{\{I_t=i\}}}{p_{I_t,n}} - H_s(i,J_t) \right)$$

is a sum of L elements of a martingale difference sequence, with respect to the filtration whose t-th element is generated by $\boldsymbol{p}_n$, the pairs $(I_s,\, S_s)$ for $s \leqslant t$, and J_s for $s \leqslant t+1$. The conditional variances of the increments are bounded by

$$\mathbb{E}_t\left[\left(\frac{\mathbb{I}_{\{S_t=s\}}\mathbb{I}_{\{I_t=i\}}}{p_{I_t,n}} \right)^2 \right] \leqslant \frac{1}{p_{i,n}^2}\,\mathbb{E}_t\big[\mathbb{I}_{\{I_t=i\}}\big] = \frac{1}{p_{i,n}}\,;$$

since by definition of the strategy, $\boldsymbol{p}_n = (1-\gamma)\,\boldsymbol{x}_n + \gamma\,\boldsymbol{u}$, we have that $p_{i,n} \geqslant \gamma/N_{\mathcal{I}}$, which shows that the sum of the conditional variances is bounded by

$$\sum_{t=(n-1)L+1}^{nL} \mathrm{Var}_t\left(\frac{\mathbb{I}_{\{S_t=s\}}\mathbb{I}_{\{I_t=i\}}}{p_{I_t,n}} \right) \leqslant \frac{LN_{\mathcal{I}}}{\gamma}\,.$$

The Bernstein-Freedman inequality (see Freedman, 1975 or Cesa-Bianchi et al., 2006, Lemma A.1) therefore indicates that with probability at least $1-\delta$,

$$\left| \frac{1}{L}\sum_{t=(n-1)L+1}^{nL} \frac{\mathbb{I}_{\{S_t=s\}}\mathbb{I}_{\{I_t=i\}}}{p_{I_t,n}} - \underbrace{\frac{1}{L}\sum_{t=(n-1)L+1}^{nL} H_s(i,J_t)}_{=\, H_s(i,\widehat{\boldsymbol{q}}_n)} \right| \leqslant \sqrt{2\frac{N_{\mathcal{I}}}{\gamma L}\ln\frac{2}{\delta}} + \frac{1}{3}\frac{N_{\mathcal{I}}}{\gamma L}\ln\frac{2}{\delta}\,.$$

Therefore, by summing the above inequalities over $i \in \mathcal{I}$ and $s \in \mathcal{H}$, we get (after a union bound) that with probability at least $1 - N_{\mathcal{I}}N_{\mathcal{H}}\delta$,

$$\left\| \widetilde{\sigma}_n - \widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big) \right\|_2 \leqslant \sqrt{N_{\mathcal{I}}N_{\mathcal{H}}} \left(\sqrt{\frac{2N_{\mathcal{I}}}{\gamma L}\ln\frac{2}{\delta}} + \frac{1}{3}\frac{N_{\mathcal{I}}}{\gamma L}\ln\frac{2}{\delta} \right).$$

Finally, since $\widehat{\sigma}_n$ is the projection in the ℓ^2–norm of $\widetilde{\sigma}_n$ onto the convex set $\mathcal{F}$, to which $\widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)$ belongs, we have that

$$\left\| \widehat{\sigma}_n - \widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big) \right\|_2 \leqslant \left\| \widetilde{\sigma}_n - \widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big) \right\|_2,$$

and this concludes the proof. ■

Appendix D. Proof of Theorem 25

Proof We denote by d_t^α the squared distance of $\widehat{m}_t^\alpha$ to $\mathcal{C}$,

$$d_t^\alpha = \inf_{c\in\mathcal{C}} \left\| c - \widehat{m}_t^\alpha \right\|^2 = \left\| c_t^\alpha - \widehat{m}_t^\alpha \right\|^2$$

and use the shortcut notation $m_t = m(\boldsymbol{x}_t, \boldsymbol{y}_t)$ for all $t \geqslant 1$. Then,

$$\begin{aligned}
d_{t+1}^\alpha &\leqslant \left\| \widehat{m}_{t+1}^\alpha - c_t^\alpha \right\|^2 = \left\| \widehat{m}_t^\alpha - c_t^\alpha + \frac{(t+1)^\alpha}{T_{t+1}^\alpha}\Big(m_{t+1} - \widehat{m}_t^\alpha\Big) \right\|^2 \\
&\leqslant \left\| \widehat{m}_t^\alpha - c_t^\alpha \right\|^2 + \frac{2(t+1)^\alpha}{T_{t+1}^\alpha}\big\langle \widehat{m}_t^\alpha - c_t^\alpha,\, m_{t+1} - m_t^\alpha \big\rangle + \left(\frac{(t+1)^\alpha}{T_{t+1}^\alpha}\right)^2 \left\| m_{t+1} - \widehat{m}_t^\alpha \right\|^2 \\
&\leqslant d_t^\alpha + \frac{2(t+1)^\alpha}{T_{t+1}^\alpha}\Big(\underbrace{\big\langle \widehat{m}_t^\alpha - c_t^\alpha,\, m_{t+1} - c_t^\alpha \big\rangle}_{\leqslant 0} + \big\langle \widehat{m}_t^\alpha - c_t^\alpha,\, c_t^\alpha - m_t^\alpha \big\rangle \Big) + \left(\frac{(t+1)^\alpha}{T_{t+1}^\alpha}\right)^2 4M^2 \\
&\leqslant d_t^\alpha \left(1 - \frac{2(t+1)^\alpha}{T_{t+1}^\alpha}\right) + \left(\frac{(t+1)^\alpha}{T_{t+1}^\alpha}\right)^2 4M^2,
\end{aligned}$$

where we used in the third inequality the same convex projection inequality as in the proof of Theorem 3.

The first inequality in (15) then follows by induction: the bound $2M$ for $t = 1$ is by boundedness of m. If the stated bound holds for d_t^α, then

$$d_{t+1}^\alpha \leqslant \left(2M \frac{\sqrt{\sum_{s=1}^t s^{2\alpha}}}{\sum_{s=1}^t s^\alpha}\right)^2 \left(1 - \frac{2(t+1)^\alpha}{T_{t+1}^\alpha}\right) + \left(\frac{(t+1)^\alpha}{T_{t+1}^\alpha}\right)^2 4M^2 \leqslant 4M^2 \frac{\sum_{s=1}^{t+1} s^{2\alpha}}{\left(T_{t+1}^\alpha\right)^2},$$

as desired, since

$$\frac{1}{\left(T_t^\alpha\right)^2}\left(1 - \frac{2(t+1)^\alpha}{T_{t+1}^\alpha}\right) = \frac{T_t^\alpha - (t+1)^\alpha}{T_{t+1}^\alpha \left(T_t^\alpha\right)^2} = \frac{1}{T_{t+1}^\alpha \left(T_t^\alpha\right)^2} \frac{\left(T_t^\alpha\right)^2 - (t+1)^{2\alpha}}{T_t^\alpha + (t+1)^\alpha} \leqslant \frac{1}{\left(T_{t+1}^\alpha\right)^2}.$$

The second inequality in (15) is straightforward for $\alpha = 0$ and is proved for $\alpha > 0$ as follows. First, by comparing sums and integrals, we get that for all $t \geqslant 1$,

$$\frac{t^{\alpha+1}}{\alpha+1} = \int_0^t s^\alpha \,\mathrm{d}s \leqslant \sum_{s=1}^t s^\alpha \leqslant t \times t^\alpha = t^{\alpha+1}.$$

Therefore,

$$\frac{\sqrt{\sum_{s=1}^t s^{2\alpha}}}{\sum_{s=1}^t s^\alpha} \leqslant (\alpha+1)\frac{\sqrt{t^{2\alpha+1}}}{t^{\alpha+1}} = \frac{\alpha+1}{\sqrt{t}}.$$

This concludes the proof. Note for later purposes that upper bounding above the sum of the s^α as

$$\sum_{s=1}^t s^\alpha \leqslant t^\alpha + \int_1^t s^\alpha \,\mathrm{d}s \leqslant t^\alpha + \frac{t^{\alpha+1}}{\alpha+1}$$

shows that

$$\sum_{s=1}^{t} s^\alpha \sim \frac{t^{\alpha+1}}{\alpha+1}\,.$$

■

Appendix E. Proof of Theorem 26

Proof The proof follows closely the proof of Theorem 24. We choose N so as to write $T = T_N^\alpha + k$ where $0 \leqslant k \leqslant L_{N+1} - 1$. We adapt step 1 as follows,

$$\left\| \frac{1}{T}\sum_{t=1}^{T} r(I_t, J_t) - \frac{1}{T_N^\alpha}\sum_{t=1}^{T_N^\alpha} r(I_t, J_t) \right\|_2 \leqslant R\left(\frac{k}{T} + \left(\frac{1}{T_N^\alpha} - \frac{1}{T}\right) T_N^\alpha\right) = \frac{2k}{T} R \leqslant \frac{2L_{N+1}}{T} R\,.$$

Second, as in step 2, we resort again to the Hoeffding-Azuma inequality for sums of Hilbert space-valued martingale differences; with probability at least $1-\delta$,

$$\left\| \frac{1}{T_N^\alpha}\sum_{t=1}^{T_N^\alpha} r(I_t, J_t) - \frac{1}{T_N^\alpha}\sum_{n=1}^{N} n^\alpha\, r\big(\boldsymbol{p}_n,\, \widehat{\boldsymbol{q}}_n\big) \right\|_2 \leqslant 4R\sqrt{\frac{\ln(2/\delta)}{T_N^\alpha}}\,.$$

In view of the choice $\gamma_n = n^{-\alpha/3}$, step 3 translates here to

$$\left\| \frac{1}{T_N^\alpha}\sum_{n=1}^{N} n^\alpha\, r\big(\boldsymbol{p}_n,\, \widehat{\boldsymbol{q}}_n\big) - \frac{1}{T_N^\alpha}\sum_{n=1}^{N} n^\alpha\, r\big(\boldsymbol{x}_n,\, \widehat{\boldsymbol{q}}_n\big) \right\|_2 \leqslant 2R\,\frac{\sum_{n=1}^N n^\alpha \gamma_n}{T_N^\alpha} = 2R\,\frac{\sum_{n=1}^N n^{2\alpha/3}}{T_N^\alpha} = 2R\frac{T_N^{(2\alpha/3)}}{T_N^\alpha}\,.$$

The same argument as the one at the beginning of the proof of Theorem 24 shows that

$$\frac{1}{T_N^\alpha}\sum_{n=1}^{N} n^\alpha\, r\big(\boldsymbol{x}_n,\, \widehat{\boldsymbol{q}}_n\big) \;\in\; \frac{1}{T_N^\alpha}\sum_{n=1}^{N} n^\alpha\, \overline{\overline{m}}\Big(\boldsymbol{\theta}_n,\, \Phi\Big(\widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)\Big)\Big).$$

Step 4 starts also by an application of Lemma 23 together with the Lipschitzness of Φ to get that for all regimes $n = 1, \ldots, N$, with probability at least $1-\delta$,

$$\left\| \Phi\big(\widehat{\sigma}_n\big) - \Phi\Big(\widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)\Big) \right\|_2 \leqslant \kappa_\Phi \sqrt{N_{\mathcal{I}} N_{\mathcal{H}}} \left(\sqrt{\frac{2N_{\mathcal{I}}}{\gamma_n L_n} \ln \frac{2N_{\mathcal{I}} N_{\mathcal{H}}}{\delta}} + \frac{1}{3}\frac{N_{\mathcal{I}}}{\gamma_n L_n} \ln \frac{2N_{\mathcal{I}} N_{\mathcal{H}}}{\delta}\right).$$

By a union bound, the above bound holds for all regimes $n = 1, \ldots, N$ with probability at least $1 - N\delta$. Then, an application of Lemma 6 shows that

$$\frac{1}{T_N^\alpha}\sum_{n=1}^{N} n^\alpha\, \overline{\overline{m}}\Big(\boldsymbol{\theta}_n,\, \Phi\Big(\widetilde{H}\big(\widehat{\boldsymbol{q}}_n\big)\Big)\Big) \quad \text{is in a } \varepsilon_N\text{–neighborhood of} \quad \frac{1}{T_N^\alpha}\sum_{n=1}^{N} n^\alpha\, \overline{\overline{m}}\Big(\boldsymbol{\theta}_n,\, \Phi\big(\widehat{\sigma}_n\big)\Big)\,,$$

where, substituting the values of $L_n = n^\alpha$ and $\gamma_n = n^{-\alpha/3}$,

$$\begin{aligned}\varepsilon_N &= R\sqrt{N_{\mathcal{B}}}\left(\kappa_\Phi\sqrt{N_{\mathcal{I}}N_{\mathcal{H}}}\;\frac{1}{T_N^\alpha}\sum_{n=1}^N n^\alpha\left(\sqrt{\frac{2N_{\mathcal{I}}}{\gamma_n L_n}\ln\frac{2N_{\mathcal{I}}N_{\mathcal{H}}}{\delta}}+\frac{1}{3}\frac{N_{\mathcal{I}}}{\gamma_n L_n}\ln\frac{2N_{\mathcal{I}}N_{\mathcal{H}}}{\delta}\right)\right)\\ &= R\sqrt{N_{\mathcal{B}}}\left(\kappa_\Phi\sqrt{N_{\mathcal{I}}N_{\mathcal{H}}}\left(\frac{T_N^{(2\alpha/3)}}{T_N^\alpha}\sqrt{2N_{\mathcal{I}}\ln\frac{2N_{\mathcal{I}}N_{\mathcal{H}}}{\delta}}+\frac{T_N^{(\alpha/3)}}{T_N^\alpha}\,\frac{N_{\mathcal{I}}}{3}\ln\frac{2N_{\mathcal{I}}N_{\mathcal{H}}}{\delta}\right)\right).\end{aligned}$$

It then suffices, as in step 5 of the original proof, to write the convergence rates for set-valued approachability guaranteed by the strategy Ψ. By combining the result of Lemma 10 with Theorem 25 and Lemma 6, we get

$$\inf_{c\in\mathcal{C}}\;\left\|c-\frac{1}{T_n^\alpha}\sum_{n=1}^N n^\alpha\,\overline{\overline{m}}\Big(\boldsymbol{\theta}_n,\,\Phi\big(\widehat{\sigma}_n\big)\Big)\right\|_2\leqslant\frac{2R\,(\alpha+1)}{\sqrt{N}}\sqrt{N_{\mathcal{A}}N_{\mathcal{B}}}\,.$$

Putting all things together and applying a union bound, we obtain that with probability at least $1-(N+1)\delta$,

$$\begin{aligned}&\inf_{c\in\mathcal{C}}\;\left\|c-\frac{1}{T}\sum_{t=1}^T r(I_t,J_t)\right\|_2\\ &\quad=O\left(\frac{(N+1)^\alpha}{T}+\sqrt{\frac{\ln(1/\delta)}{T_N^\alpha}}+\frac{T_N^{(2\alpha/3)}}{T_N^\alpha}+\frac{T_N^{(2\alpha/3)}}{T_N^\alpha}\sqrt{\ln\frac{1}{\delta}}+\frac{T_N^{(\alpha/3)}}{T_N^\alpha}\ln\frac{1}{\delta}+\frac{1}{\sqrt{N}}\right).\end{aligned}$$

Since (as proved at the end of the proof of Theorem 25) $T_N^\beta\sim N^{\beta+1}/(\beta+1)$ for all $\beta\geqslant 0$, we get that

$$N\sim\big((\alpha+1)T\big)^{1/(\alpha+1)}\qquad\text{and}\qquad T_N^\beta\sim\frac{N^{\beta+1}}{\beta+1}\sim\kappa_{\alpha,\beta}\,T^{(\beta+1)/(\alpha+1)}\,,$$

where $\kappa_{\alpha,\beta}$ is a constant that only depends on α and β. Replacing δ by $\delta/(N+1)$ as we did in step 5 of the proof of Theorem 24, choosing $\alpha=3/2$ and substituting the equivalences above ensures the result. ■

References

J. Abernethy, P. L. Bartlett, and E. Hazan. Blackwell approachability and low-regret learning are equivalent. In *Proceedings of the Twenty-Fourth Annual Conference on Learning Theory (COLT'11)*. JMLR Workshop and Conference Proceedings, 2011.

G. Bartók, D. Pál, and C. Szepesvári. Toward a classification of finite partial-monitoring games. In *Proceedings of the Twenty-First International Conference on Algorithmic Learning Theory (ALT'10)*, pages 224–238. Springer, 2010.

G. Bartók, D. Pál, and C. Szepesvári. Minimax regret of finite partial-monitoring games in stochastic environments. In *Proceedings of the Twenty-Fourth Annual Conference on Learning Theory (COLT'11)*. JMLR Workshop and Conference Proceedings, 2011.

D. Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6:1–8, 1956a.

D. Blackwell. Controlled random walks. In *Proceedings of the International Congress of Mathematicians, 1954, Amsterdam, vol. III*, pages 336–338, 1956b.

A. Blum and Y. Mansour. From external to internal regret. *Journal of Machine Learning Research*, 8:1307–1324, 2007.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

N. Cesa-Bianchi, G. Lugosi, and G. Stoltz. Regret minimization under partial monitoring. *Mathematics of Operations Research*, 31:562–580, 2006.

X. Chen and H. White. Laws of large numbers for Hilbert space-valued mixingales with applications. *Econometric Theory*, 12:284–304, 1996.

A.P. Dawid. The well-calibrated Bayesian. *Journal of the American Statistical Association*, 77:605–613, 1982.

D. Foster and R. Vohra. Asymptotic calibration. *Biometrika*, 85:379–390, 1998.

D. Foster and R. Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29:7–36, 1999.

D.P. Foster and A. Rakhlin. No internal regret via neighborhood watch. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS'12)*, 2012.

D.A. Freedman. On tail probabilities for martingales. *Annals of Probability*, 3:100–118, 1975.

J.E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry*. Discrete Mathematics and its Applications. Chapman & Hall/CRC, Boca Raton, FL, second edition, 2004.

S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68:1127–1150, 2000.

S. Hart and A. Mas-Colell. A general class of adaptive strategies. *Journal of Economic Theory*, 98:26–54, 2001.

E. Lehrer and E. Solan. Learning to play partially-specified equilibrium. Mimeo, 2007.

G. Lugosi, S. Mannor, and G. Stoltz. Strategies for prediction under imperfect monitoring. *Mathematics of Operations Research*, 33:513–528, 2008. An extended abstract was presented at COLT'07.

S. Mannor and N. Shimkin. On-line learning with imperfect monitoring. In *Proceedings of the Sixteenth Annual Conference on Learning Theory (COLT'03)*, pages 552–567. Springer, 2003.

S. Mannor and N. Shimkin. Regret minimization in repeated matrix games with variable stage duration. *Games and Economic Behavior*, 63(1):227–258, 2008.

S. Mannor and G. Stoltz. A geometric proof of calibration. *Mathematics of Operations Research*, 35:721–727, 2010.

S. Mannor, J. Tsitsiklis, and J. Y. Yu. Online learning with sample path constraints. *Journal of Machine Learning Research*, 10(Mar):569–590, 2009.

J.-F. Mertens, S. Sorin, and S. Zamir. Repeated games. Technical Report no. 9420, 9421, 9422, Université de Louvain-la-Neuve, 1994.

V. Perchet. Calibration and internal no-regret with random signals. In *Proceedings of the Twentieth International Conference on Algorithmic Learning Theory (ALT'09)*, pages 68–82, 2009.

V. Perchet. Approachability of convex sets in games with partial monitoring. *Journal of Optimization Theory and Applications*, 149:665–677, 2011a.

V. Perchet. Internal regret with partial monitoring calibration-based optimal algorithms. *Journal of Machine Learning Research*, 12:1893–1921, 2011b.

V. Perchet and M. Quincampoix. On an unified framework for approachability in games with or without signals. Mimeo, 2011.

A. Piccolboni and C. Schindelhauer. Discrete prediction games with arbitrary feedback and loss. In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory (COLT'01)*, pages 208–223, 2001.

A. Rakhlin, K. Sridharan, and A. Tewari. Online learning: Beyond regret. In *Proceedings of the Twenty-Fourth Annual Conference on Learning Theory (COLT'11)*. JMLR Workshop and Conference Proceedings, 2011.

J. Rambau and G. Ziegler. Projections of polytopes and the generalized Baues conjecture. *Discrete and Computational Geometry*, 16:215–237, 1996.

A. Rustichini. Minimizing regret: The general case. *Games and Economic Behavior*, 29: 224–243, 1999.

Journal of Machine Learning Research 15 (2014) 3297-3331 Submitted 9/13; Revised 5/14; Published 10/14

Learning Graphical Models With Hubs

Kean Ming Tan KEANMING@UW.EDU
Department of Biostatistics
University of Washington
Seattle WA, 98195

Palma London PALONDON@UW.EDU
Karthik Mohan KARNA@UW.EDU
Department of Electrical Engineering
University of Washington
Seattle WA, 98195

Su-In Lee SUINLEE@CS.WASHINGTON.EDU
Department of Computer Science and Engineering, Genome Sciences
University of Washington
Seattle WA, 98195

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

Editor: Xiaotong Shen

Abstract

We consider the problem of learning a high-dimensional graphical model in which there are a few *hub* nodes that are *densely-connected* to many other nodes. Many authors have studied the use of an ℓ_1 penalty in order to learn a sparse graph in the high-dimensional setting. However, the ℓ_1 penalty implicitly assumes that each edge is equally likely and independent of all other edges. We propose a general framework to accommodate more realistic networks with hub nodes, using a convex formulation that involves a row-column overlap norm penalty. We apply this general framework to three widely-used probabilistic graphical models: the Gaussian graphical model, the covariance graph model, and the binary Ising model. An alternating direction method of multipliers algorithm is used to solve the corresponding convex optimization problems. On synthetic data, we demonstrate that our proposed framework outperforms competitors that do not explicitly model hub nodes. We illustrate our proposal on a webpage data set and a gene expression data set.

Keywords: Gaussian graphical model, covariance graph, binary network, *lasso*, hub, alternating direction method of multipliers

1. Introduction

Graphical models are used to model a wide variety of systems, such as gene regulatory networks and social interaction networks. A graph consists of a set of p nodes, each representing a variable, and a set of edges between pairs of nodes. The presence of an edge between two nodes indicates a relationship between the two variables. In this manuscript, we consider two types of graphs: conditional independence graphs and marginal independence graphs. In a conditional independence graph, an edge connects a pair of variables if and only if they are conditionally dependent—dependent conditional upon the other variables. In a marginal independence graph, two nodes are joined by an edge if and only if they are marginally dependent—dependent without conditioning on the other variables.

In recent years, many authors have studied the problem of learning a graphical model in the high-dimensional setting, in which the number of variables p is larger than the number of observations n. Let $\mathbf{X}$ be a $n \times p$ matrix, with rows $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Throughout the rest of the text, we will focus on three specific types of graphical models:

1. A *Gaussian graphical model*, where $\mathbf{x}_1, \ldots, \mathbf{x}_n \overset{\text{i.i.d.}}{\sim} N(\mathbf{0}, \mathbf{\Sigma})$. In this setting, $(\mathbf{\Sigma}^{-1})_{jj'} = 0$ for some $j \neq j'$ if and only if the jth and j'th variables are conditionally independent (Mardia et al., 1979); therefore, the sparsity pattern of $\mathbf{\Sigma}^{-1}$ determines the conditional independence graph.

2. A *Gaussian covariance graph model*, where $\mathbf{x}_1, \ldots, \mathbf{x}_n \overset{\text{i.i.d.}}{\sim} N(\mathbf{0}, \mathbf{\Sigma})$. Then $\Sigma_{jj'} = 0$ for some $j \neq j'$ if and only if the jth and j'th variables are marginally independent. Therefore, the sparsity pattern of $\mathbf{\Sigma}$ determines the marginal independence graph.

3. A *binary Ising graphical model*, where $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are i.i.d. with density function

$$p(\mathbf{x}, \mathbf{\Theta}) = \frac{1}{Z(\mathbf{\Theta})} \exp\left[\sum_{j=1}^{p} \theta_{jj} x_j + \sum_{1 \leq j < j' \leq p} \theta_{jj'} x_j x_{j'}\right],$$

   where $\mathbf{\Theta}$ is a $p \times p$ symmetric matrix, and $Z(\mathbf{\Theta})$ is the partition function, which ensures that the density sums to one. Here, $\mathbf{x}$ is a binary vector, and $\theta_{jj'} = 0$ if and only if the jth and j'th variables are conditionally independent. The sparsity pattern of $\mathbf{\Theta}$ determines the conditional independence graph.

To construct an interpretable graph when $p > n$, many authors have proposed applying an ℓ_1 penalty to the parameter encoding each edge, in order to encourage sparsity. For instance, such an approach is taken by Yuan and Lin (2007a), Friedman et al. (2007), Rothman et al. (2008), and Yuan (2008) in the Gaussian graphical model; El Karoui (2008), Bickel and Levina (2008), Rothman et al. (2009), Bien and Tibshirani (2011), Cai and Liu (2011), and Xue et al. (2012) in the covariance graph model; and Lee et al. (2007), Höfling and Tibshirani (2009), and Ravikumar et al. (2010) in the binary model.

However, applying an ℓ_1 penalty to each edge can be interpreted as placing an independent double-exponential prior on each edge. Consequently, such an approach implicitly assumes that each edge is equally likely and independent of all other edges; this corresponds to an Erdős-Rényi graph in which most nodes have approximately the same number

of edges (Erdős and Rényi, 1959). This is unrealistic in many real-world networks, in which we believe that certain nodes (which, unfortunately, are not known *a priori*) have a lot more edges than other nodes. An example is the network of webpages in the World Wide Web, where a relatively small number of webpages are connected to many other webpages (Barabási and Albert, 1999). A number of authors have shown that real-world networks are *scale-free*, in the sense that the number of edges for each node follows a power-law distribution; examples include gene-regulatory networks, social networks, and networks of collaborations among scientists (among others, Barabási and Albert, 1999; Barabási, 2009; Liljeros et al., 2001; Jeong et al., 2001; Newman, 2000; Li et al., 2005). More recently, Hao et al. (2012) have shown that certain genes, referred to as *super hubs*, regulate hundreds of downstream genes in a gene regulatory network, resulting in far denser connections than are typically seen in a scale-free network.

In this paper, we refer to very densely-connected nodes, such as the "super hubs" considered in Hao et al. (2012), as *hubs*. When we refer to hubs, we have in mind nodes that are connected to a very substantial number of other nodes in the network—and in particular, we are referring to nodes that are much more densely-connected than even the most highly-connected node in a scale-free network. An example of a network containing hub nodes is shown in Figure 1.

Here we propose a convex penalty function for estimating graphs containing hubs. Our formulation simultaneously identifies the hubs and estimates the entire graph. The penalty function yields a convex optimization problem when combined with a convex loss function. We consider the application of this hub penalty function in modeling Gaussian graphical models, covariance graph models, and binary Ising models. Our formulation does not require that we know *a priori* which nodes in the network are hubs.

In related work, several authors have proposed methods to estimate a scale-free Gaussian graphical model (Liu and Ihler, 2011; Defazio and Caetano, 2012). However, those methods do not model hub nodes—the most highly-connected nodes that arise in a scale-free network are far less connected than the hubs that we consider in our formulation. Under a different framework, some authors proposed a screening-based procedure to identify hub nodes in the context of Gaussian graphical models (Hero and Rajaratnam, 2012; Firouzi and Hero, 2013). Our proposal outperforms such approaches when hub nodes are present (see discussion in Section 3.5.4).

In Figure 1, the performance of our proposed approach is shown in a toy example in the context of a Gaussian graphical model. We see that when the true network contains hub nodes (Figure 1(a)), our proposed approach (Figure 1(b)) is much better able to recover the network than is the graphical lasso (Figure 1(c)), a well-studied approach that applies an ℓ_1 penalty to each edge in the graph (Friedman et al., 2007).

We present the hub penalty function in Section 2. We then apply it to the Gaussian graphical model, the covariance graph model, and the binary Ising model in Sections 3, 4, and 5, respectively. In Section 6, we apply our approach to a webpage data set and a gene expression data set. We close with a discussion in Section 7.

2. The General Formulation

In this section, we present a general framework to accommodate network with hub nodes.

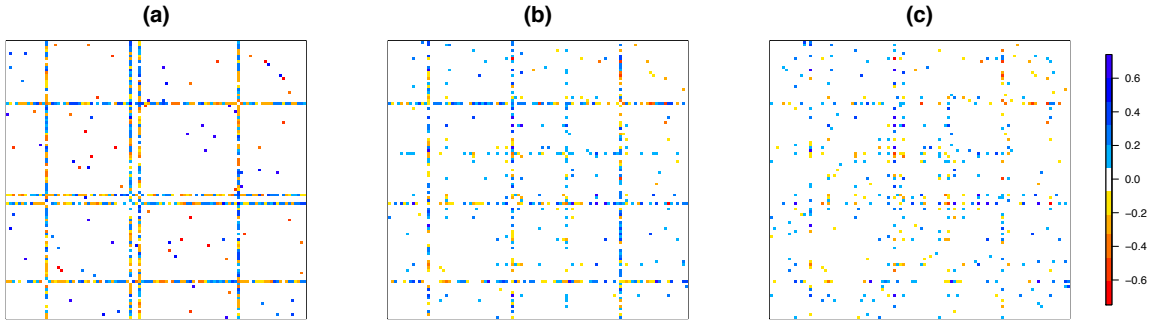


Figure 1: (a): Heatmap of the inverse covariance matrix in a toy example of a Gaussian graphical model with four hub nodes. White elements are zero and colored elements are non-zero in the inverse covariance matrix. Thus, colored elements correspond to edges in the graph. (b): Estimate from the *hub graphical lasso*, proposed in this paper. (c): Graphical lasso estimate.

2.1 The Hub Penalty Function

Let $\mathbf{X}$ be a $n \times p$ data matrix, $\mathbf{\Theta}$ a $p \times p$ symmetric matrix containing the parameters of interest, and $\ell(\mathbf{X}, \mathbf{\Theta})$ a loss function (assumed to be convex in $\mathbf{\Theta}$). In order to obtain a sparse and interpretable graph estimate, many authors have considered the problem

$$\underset{\mathbf{\Theta} \in \mathcal{S}}{\text{minimize}} \qquad \{\ell(\mathbf{X}, \mathbf{\Theta}) + \lambda \|\mathbf{\Theta} - \text{diag}(\mathbf{\Theta})\|_1\}, \tag{1}$$

where λ is a non-negative tuning parameter, $\mathcal{S}$ is some set depending on the loss function, and $\|\cdot\|_1$ is the sum of the absolute values of the matrix elements. For instance, in the case of a Gaussian graphical model, we could take $\ell(\mathbf{X}, \mathbf{\Theta}) = -\log\det\mathbf{\Theta} + \text{trace}(\mathbf{S\Theta})$, the negative log-likelihood of the data, where $\mathbf{S}$ is the empirical covariance matrix and $\mathcal{S}$ is the set of $p \times p$ positive definite matrices. The solution to (1) can then be interpreted as an estimate of the inverse covariance matrix. The ℓ_1 penalty in (1) encourages zeros in the solution. But it typically does not yield an estimate that contains hubs.

In order to explicitly model hub nodes in a graph, we wish to replace the ℓ_1 penalty in (1) with a convex penalty that encourages a solution that can be decomposed as $\mathbf{Z}+\mathbf{V}+\mathbf{V}^T$, where $\mathbf{Z}$ is a sparse symmetric matrix, and $\mathbf{V}$ is a matrix whose columns are either entirely zero or almost entirely non-zero (see Figure 2). The sparse elements of $\mathbf{Z}$ represent edges between non-hub nodes, and the non-zero columns of $\mathbf{V}$ correspond to hub nodes. We achieve this goal via the *hub penalty function*, which takes the form

$$\text{P}(\mathbf{\Theta}) = \min_{\mathbf{V},\mathbf{Z}:\ \mathbf{\Theta}=\mathbf{V}+\mathbf{V}^T+\mathbf{Z}} \left\{ \lambda_1\|\mathbf{Z} - \text{diag}(\mathbf{Z})\|_1 + \lambda_2\|\mathbf{V} - \text{diag}(\mathbf{V})\|_1 + \lambda_3 \sum_{j=1}^{p} \|(\mathbf{V} - \text{diag}(\mathbf{V}))_j\|_q \right\}. \tag{2}$$

Here λ_1, λ_2, and λ_3 are nonnegative tuning parameters. Sparsity in $\mathbf{Z}$ is encouraged via the ℓ_1 penalty on its off-diagonal elements, and is controlled by the value of λ_1. The ℓ_1 and ℓ_1/ℓ_q norms on the columns of $\mathbf{V}$ induce group sparsity when $q = 2$ (Yuan and Lin, 2007b; Simon et al., 2013); λ_3 controls the selection of hub nodes, and λ_2 controls the sparsity of

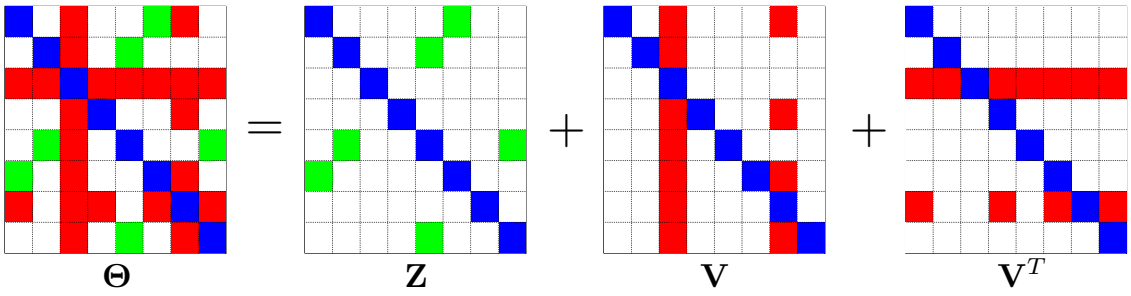


Figure 2: Decomposition of a symmetric matrix $\mathbf{\Theta}$ into $\mathbf{Z} + \mathbf{V} + \mathbf{V}^T$, where $\mathbf{Z}$ is sparse, and most columns of $\mathbf{V}$ are entirely zero. Blue, white, green, and red elements are diagonal, zero, non-zero in $\mathbf{Z}$, and non-zero due to two hubs in $\mathbf{V}$, respectively.

each hub node's connections to other nodes. The convex penalty (2) can be combined with $\ell(\mathbf{X}, \mathbf{\Theta})$ to yield the convex optimization problem

$$\underset{\mathbf{\Theta}\in\mathcal{S},\mathbf{V},\mathbf{Z}}{\text{minimize}} \quad \Big\{ \ell(\mathbf{X}, \mathbf{\Theta}) + \lambda_1 \|\mathbf{Z} - \text{diag}(\mathbf{Z})\|_1 + \lambda_2 \|\mathbf{V} - \text{diag}(\mathbf{V})\|_1 + \lambda_3 \sum_{j=1}^{p} \|(\mathbf{V} - \text{diag}(\mathbf{V}))_j\|_q \Big\} \text{ subject to } \mathbf{\Theta} = \mathbf{V} + \mathbf{V}^T + \mathbf{Z}, \tag{3}$$

where the set $\mathcal{S}$ depends on the loss function $\ell(\mathbf{X}, \mathbf{\Theta})$.

Note that when $\lambda_2 \to \infty$ or $\lambda_3 \to \infty$, then (3) reduces to (1). In this paper, we take $q = 2$, which leads to estimation of a network containing dense hub nodes. Other values of q such as $q = \infty$ are also possible (see, e.g., Mohan et al., 2014). We note that the hub penalty function is closely related to recent work on overlapping group lasso penalties in the context of learning multiple sparse precision matrices (Mohan et al., 2014).

2.2 Algorithm

In order to solve (3) with $q = 2$, we use an *alternating direction method of multipliers* (ADMM) algorithm (see, e.g., Eckstein and Bertsekas, 1992; Boyd et al., 2010; Eckstein, 2012). ADMM is an attractive algorithm for this problem, as it allows us to decouple some of the terms in (3) that are difficult to optimize jointly. In order to develop an ADMM algorithm for (3) with guaranteed convergence, we reformulate it as a consensus problem, as in Ma et al. (2013). The convergence of the algorithm to the optimal solution follows from classical results (see, e.g., the review papers Boyd et al., 2010; Eckstein, 2012).

In greater detail, we let $\mathbf{B} = (\mathbf{\Theta}, \mathbf{V}, \mathbf{Z})$, $\tilde{\mathbf{B}} = (\tilde{\mathbf{\Theta}}, \tilde{\mathbf{V}}, \tilde{\mathbf{Z}})$,

$$f(\mathbf{B}) = \ell(\mathbf{X}, \mathbf{\Theta}) + \lambda_1 \|\mathbf{Z} - \text{diag}(\mathbf{Z})\|_1 + \lambda_2 \|\mathbf{V} - \text{diag}(\mathbf{V})\|_1 + \lambda_3 \sum_{j=1}^{p} \|(\mathbf{V} - \text{diag}(\mathbf{V}))\|_2,$$

Algorithm 1 ADMM Algorithm for Solving (3).

1. **Initialize** the parameters:
   (a) primal variables $\mathbf{\Theta}, \mathbf{V}, \mathbf{Z}, \tilde{\mathbf{\Theta}}, \tilde{\mathbf{V}}$, and $\tilde{\mathbf{Z}}$ to the $p \times p$ identity matrix.
   (b) dual variables $\mathbf{W}_1, \mathbf{W}_2$, and $\mathbf{W}_3$ to the $p \times p$ zero matrix.
   (c) constants $\rho > 0$ and $\tau > 0$.

2. **Iterate** until the stopping criterion $\frac{\|\mathbf{\Theta}_t - \mathbf{\Theta}_{t-1}\|_F^2}{\|\mathbf{\Theta}_{t-1}\|_F^2} \leq \tau$ is met, where $\mathbf{\Theta}_t$ is the value of $\mathbf{\Theta}$ obtained at the tth iteration:
   (a) Update $\mathbf{\Theta}, \mathbf{V}, \mathbf{Z}$:
      i. $\mathbf{\Theta} = \underset{\mathbf{\Theta} \in \mathcal{S}}{\arg\min} \left\{ \ell(\mathbf{X}, \mathbf{\Theta}) + \frac{\rho}{2} \|\mathbf{\Theta} - \tilde{\mathbf{\Theta}} + \mathbf{W}_1\|_F^2 \right\}$.
      ii. $\mathbf{Z} = S(\tilde{\mathbf{Z}} - \mathbf{W}_3, \frac{\lambda_1}{\rho})$, $\mathrm{diag}(\mathbf{Z}) = \mathrm{diag}(\tilde{\mathbf{Z}} - \mathbf{W}_3)$. Here S denotes the soft-thresholding operator, applied element-wise to a matrix: $S(A_{ij}, b) = \mathrm{sign}(A_{ij}) \max(|A_{ij}| - b, 0)$.
      iii. $\mathbf{C} = \tilde{\mathbf{V}} - \mathbf{W}_2 - \mathrm{diag}(\tilde{\mathbf{V}} - \mathbf{W}_2)$.
      iv. $\mathbf{V}_j = \max\left(1 - \frac{\lambda_3}{\rho \|S(\mathbf{C}_j, \lambda_2/\rho)\|_2}, 0\right) \cdot S(\mathbf{C}_j, \lambda_2/\rho)$ for $j = 1, \ldots, p$.
      v. $\mathrm{diag}(\mathbf{V}) = \mathrm{diag}(\tilde{\mathbf{V}} - \mathbf{W}_2)$.
   (b) Update $\tilde{\mathbf{\Theta}}, \tilde{\mathbf{V}}, \tilde{\mathbf{Z}}$:
      i. $\mathbf{\Gamma} = \frac{\rho}{6} \left[(\mathbf{\Theta} + \mathbf{W}_1) - (\mathbf{V} + \mathbf{W}_2) - (\mathbf{V} + \mathbf{W}_2)^T - (\mathbf{Z} + \mathbf{W}_3) \right]$.
      ii. $\tilde{\mathbf{\Theta}} = \mathbf{\Theta} + \mathbf{W}_1 - \frac{1}{\rho}\mathbf{\Gamma}$; iii. $\tilde{\mathbf{V}} = \frac{1}{\rho}(\mathbf{\Gamma} + \mathbf{\Gamma}^T) + \mathbf{V} + \mathbf{W}_2$; iv. $\tilde{\mathbf{Z}} = \frac{1}{\rho}\mathbf{\Gamma} + \mathbf{Z} + \mathbf{W}_3$.
   (c) Update $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$:
      i. $\mathbf{W}_1 = \mathbf{W}_1 + \mathbf{\Theta} - \tilde{\mathbf{\Theta}}$; ii. $\mathbf{W}_2 = \mathbf{W}_2 + \mathbf{V} - \tilde{\mathbf{V}}$; iii. $\mathbf{W}_3 = \mathbf{W}_3 + \mathbf{Z} - \tilde{\mathbf{Z}}$.

and

$$g(\tilde{\mathbf{B}}) = \begin{cases} 0 & \text{if } \tilde{\mathbf{\Theta}} = \tilde{\mathbf{V}} + \tilde{\mathbf{V}}^T + \tilde{\mathbf{Z}} \\ \infty & \text{otherwise.} \end{cases}$$

Then, we can rewrite (3) as

$$\underset{\mathbf{B},\tilde{\mathbf{B}}}{\text{minimize}} \left\{ f(\mathbf{B}) + g(\tilde{\mathbf{B}}) \right\} \qquad \text{subject to } \mathbf{B} = \tilde{\mathbf{B}}. \tag{4}$$

The scaled augmented Lagrangian for (4) takes the form

$$\begin{aligned} L(\mathbf{B}, \tilde{\mathbf{B}}, \mathbf{W}) = \ell(\mathbf{X}, \mathbf{\Theta}) + \lambda_1 \|\mathbf{Z} - \text{diag}(\mathbf{Z})\|_1 + \lambda_2 \|\mathbf{V} - \text{diag}(\mathbf{V})\|_1 \\ + \lambda_3 \sum_{j=1}^{p} \|(\mathbf{V} - \text{diag}(\mathbf{V}))_j\|_2 + g(\tilde{\mathbf{B}}) + \frac{\rho}{2} \|\mathbf{B} - \tilde{\mathbf{B}} + \mathbf{W}\|_F^2, \end{aligned}$$

where $\mathbf{B}$ and $\tilde{\mathbf{B}}$ are the primal variables, and $\mathbf{W} = (\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3)$ is the dual variable. Note that the scaled augmented Lagrangian can be derived from the usual Lagrangian by adding a quadratic term and completing the square (Boyd et al., 2010).

A general algorithm for solving (3) is provided in Algorithm 1. The derivation is in Appendix A. Note that only the update for $\mathbf{\Theta}$ (Step 2(a)i) depends on the form of the convex loss function $\ell(\mathbf{X}, \mathbf{\Theta})$. In the following sections, we consider special cases of (3) that lead to estimation of Gaussian graphical models, covariance graph models, and binary networks with hub nodes.

3. The Hub Graphical Lasso

Assume that $\mathbf{x}_1, \ldots, \mathbf{x}_n \overset{\text{i.i.d.}}{\sim} N(\mathbf{0}, \mathbf{\Sigma})$. The well-known *graphical lasso* problem (see, e.g., Friedman et al., 2007) takes the form of (1) with $\ell(\mathbf{X}, \mathbf{\Theta}) = -\log\det\mathbf{\Theta} + \text{trace}(\mathbf{S\Theta})$, and $\mathbf{S}$ the empirical covariance matrix of $\mathbf{X}$:

$$\underset{\mathbf{\Theta} \in \mathcal{S}}{\text{minimize}} \quad \left\{ -\log\det\mathbf{\Theta} + \text{trace}(\mathbf{S\Theta}) + \lambda \sum_{j \neq j'} |\Theta_{jj'}| \right\}, \tag{5}$$

where $\mathcal{S} = \{\mathbf{\Theta} : \mathbf{\Theta} \succ 0 \text{ and } \mathbf{\Theta} = \mathbf{\Theta}^T\}$. The solution to this optimization problem serves as an estimate for $\mathbf{\Sigma}^{-1}$. We now use the hub penalty function to extend the graphical lasso in order to accommodate hub nodes.

3.1 Formulation and Algorithm

We propose the *hub graphical lasso* (HGL) optimization problem, which takes the form

$$\underset{\mathbf{\Theta} \in \mathcal{S}}{\text{minimize}} \qquad \{-\log\det\mathbf{\Theta} + \text{trace}(\mathbf{S\Theta}) + \text{P}(\mathbf{\Theta})\}. \tag{6}$$

Again, $\mathcal{S} = \{\mathbf{\Theta} : \mathbf{\Theta} \succ 0 \text{ and } \mathbf{\Theta} = \mathbf{\Theta}^T\}$. It encourages a solution that contains hub nodes, as well as edges that connect non-hubs (Figure 1). Problem (6) can be solved using Algorithm 1. The update for $\mathbf{\Theta}$ in Algorithm 1 (Step 2(a)i) can be derived by minimizing

$$-\log\det\mathbf{\Theta} + \text{trace}(\mathbf{S\Theta}) + \frac{\rho}{2} \|\mathbf{\Theta} - \tilde{\mathbf{\Theta}} + \mathbf{W}_1\|_F^2 \tag{7}$$

with respect to $\mathbf{\Theta}$ (note that the constraint $\mathbf{\Theta} \in \mathcal{S}$ in (6) is treated as an implicit constraint, due to the domain of definition of the log det function). This can be shown to have the solution

$$\mathbf{\Theta} = \frac{1}{2}\mathbf{U}\left(\mathbf{D} + \sqrt{\mathbf{D}^2 + \frac{4}{\rho}\mathbf{I}}\right)\mathbf{U}^T,$$

where $\mathbf{UDU}^T$ denotes the eigen-decomposition of $\tilde{\mathbf{\Theta}} - \mathbf{W}_1 - \frac{1}{\rho}\mathbf{S}$.

The complexity of the ADMM algorithm for HGL is $O(p^3)$ per iteration; this is the complexity of the eigen-decomposition for updating $\mathbf{\Theta}$. We now briefly compare the computational time for the ADMM algorithm for solving (6) to that of an interior point method (using the solver `Sedumi` called from `cvx`). On a 1.86 GHz Intel Core 2 Duo machine, the interior point method takes ~ 3 minutes, while ADMM takes only 1 second, on a data set with $p = 30$. We present a more extensive run time study for the ADMM algorithm for HGL in Appendix E.

3.2 Conditions for HGL Solution to be Block Diagonal

In order to reduce computations for solving the HGL problem, we now present a necessary condition and a sufficient condition for the HGL solution to be block diagonal, subject to some permutation of the rows and columns. The conditions depend only on the tuning parameters λ_1, λ_2, and λ_3. These conditions build upon similar results in the context of Gaussian graphical models from the recent literature (see, e.g., Witten et al., 2011; Mazumder and Hastie, 2012; Yang et al., 2012b; Danaher et al., 2014; Mohan et al., 2014). Let $C_1, C_2, \ldots, C_K$ denote a partition of the p features.

Theorem 1 *A sufficient condition for the HGL solution to be block diagonal with blocks given by $C_1, C_2, \ldots, C_K$ is that* $\min\left\{\lambda_1, \frac{\lambda_2}{2}\right\} > |S_{jj'}|$ *for all $j \in C_k, j' \in C_{k'}, k \neq k'$.*

Theorem 2 *A necessary condition for the HGL solution to be block diagonal with blocks given by $C_1, C_2, \ldots, C_K$ is that* $\min\left\{\lambda_1, \frac{\lambda_2+\lambda_3}{2}\right\} > |S_{jj'}|$ *for all $j \in C_k, j' \in C_{k'}, k \neq k'$.*

Theorem 1 implies that one can screen the empirical covariance matrix $\mathbf{S}$ to check if the HGL solution is block diagonal (using standard algorithms for identifying the connected components of an undirected graph; see, e.g., Tarjan, 1972). Suppose that the HGL solution is block diagonal with K blocks, containing $p_1, \ldots, p_K$ features, and $\sum_{k=1}^K p_k = p$. Then, one can simply solve the HGL problem on the features within each block separately. Recall that the bottleneck of the HGL algorithm is the eigen-decomposition for updating $\mathbf{\Theta}$. The block diagonal condition leads to massive computational speed-ups for implementing the HGL algorithm: instead of computing an eigen-decomposition for a $p \times p$ matrix in each iteration of the HGL algorithm, we compute the eigen-decomposition of K matrices of dimensions $p_1 \times p_1, \ldots, p_K \times p_K$. The computational complexity per-iteration is reduced from $O(p^3)$ to $\sum_{k=1}^K O(p_k^3)$.

We illustrate the reduction in computational time due to these results in an example with $p = 500$. Without exploiting Theorem 1, the ADMM algorithm for HGL (with a particular value of λ) takes 159 seconds; in contrast, it takes only 22 seconds when Theorem 1 is applied. The estimated precision matrix has 107 connected components, the largest of which contains 212 nodes.

3.3 Some Properties of HGL

We now present several properties of the HGL optimization problem (6), which can be used to provide guidance on the suitable range for the tuning parameters λ_1, λ_2, and λ_3. In what follows, $\mathbf{Z}^*$ and $\mathbf{V}^*$ denote the optimal solutions for $\mathbf{Z}$ and $\mathbf{V}$ in (6). Let $\frac{1}{s} + \frac{1}{q} = 1$ (recall that q appears in Equation 2).

Lemma 3 *A sufficient condition for $\mathbf{Z}^*$ to be a diagonal matrix is that $\lambda_1 > \frac{\lambda_2+\lambda_3}{2}$.*

Lemma 4 *A sufficient condition for $\mathbf{V}^*$ to be a diagonal matrix is that $\lambda_1 < \frac{\lambda_2}{2} + \frac{\lambda_3}{2(p-1)^{1/s}}$.*

Corollary 5 *A necessary condition for both $\mathbf{V}^*$ and $\mathbf{Z}^*$ to be non-diagonal matrices is that $\frac{\lambda_2}{2} + \frac{\lambda_3}{2(p-1)^{1/s}} \leq \lambda_1 \leq \frac{\lambda_2+\lambda_3}{2}$.*

Furthermore, (6) reduces to the graphical lasso problem (5) under a simple condition.

Lemma 6 *If $q = 1$, then (6) reduces to (5) with tuning parameter $\min\left\{\lambda_1, \frac{\lambda_2+\lambda_3}{2}\right\}$.*

Note also that when $\lambda_2 \to \infty$ or $\lambda_3 \to \infty$, (6) reduces to (5) with tuning parameter λ_1. However, throughout the rest of this paper, we assume that $q = 2$, and λ_2 and λ_3 are finite.

The solution $\hat{\mathbf{\Theta}}$ of (6) is unique, since (6) is a strictly convex problem. We now consider the question of whether the decomposition $\hat{\mathbf{\Theta}} = \hat{\mathbf{V}} + \hat{\mathbf{V}}^T + \hat{\mathbf{Z}}$ is unique. We see that the decomposition is unique in a certain regime of the tuning parameters. For instance, according to Lemma 3, when $\lambda_1 > \frac{\lambda_2+\lambda_3}{2}$, $\hat{\mathbf{Z}}$ is a diagonal matrix and hence $\hat{\mathbf{V}}$ is unique. Similarly, according to Lemma 4, when $\lambda_1 < \frac{\lambda_2}{2} + \frac{\lambda_3}{2(p-1)^{1/s}}$, $\hat{\mathbf{V}}$ is a diagonal matrix and hence $\hat{\mathbf{Z}}$ is unique. Studying more general conditions on $\mathbf{S}$ and on λ_1, λ_2, and λ_3 such that the decomposition is guaranteed to be unique is a challenging problem and is outside of the scope of this paper.

3.4 Tuning Parameter Selection

In this section, we propose a *Bayesian information criterion* (BIC)-type quantity for tuning parameter selection in (6). Recall from Section 2 that the hub penalty function (2) decomposes the parameter of interest into the sum of three matrices, $\mathbf{\Theta} = \mathbf{Z} + \mathbf{V} + \mathbf{V}^T$, and places an ℓ_1 penalty on $\mathbf{Z}$, and an ℓ_1/ℓ_2 penalty on $\mathbf{V}$.

For the graphical lasso problem in (5), many authors have proposed to select the tuning parameter λ such that $\hat{\mathbf{\Theta}}$ minimizes the following quantity:

$$-n \cdot \log\det(\hat{\mathbf{\Theta}}) + n \cdot \text{trace}(\mathbf{S}\hat{\mathbf{\Theta}}) + \log(n) \cdot |\hat{\mathbf{\Theta}}|,$$

where $|\hat{\mathbf{\Theta}}|$ is the cardinality of $\hat{\mathbf{\Theta}}$, that is, the number of unique non-zeros in $\hat{\mathbf{\Theta}}$ (see, e.g., Yuan and Lin, 2007a).[1]

1. The term $\log(n) \cdot |\hat{\mathbf{\Theta}}|$ is motivated by the fact that the degrees of freedom for an estimate involving the ℓ_1 penalty can be approximated by the cardinality of the estimated parameter (Zou et al., 2007).

Using a similar idea, we propose the following BIC-type quantity for selecting the set of tuning parameters $(\lambda_1, \lambda_2, \lambda_3)$ for (6):

$$\text{BIC}(\hat{\mathbf{\Theta}}, \hat{\mathbf{V}}, \hat{\mathbf{Z}}) = -n \cdot \log\det(\hat{\mathbf{\Theta}}) + n \cdot \text{trace}(\mathbf{S}\hat{\mathbf{\Theta}}) + \log(n) \cdot |\hat{\mathbf{Z}}| + \log(n) \cdot \left(\nu + c \cdot [|\hat{\mathbf{V}}| - \nu]\right),$$

where ν is the number of estimated hub nodes, that is, $\nu = \sum_{j=1}^{p} 1_{\{\|\hat{\mathbf{V}}_j\|_0 > 0\}}$, c is a constant between zero and one, and $|\hat{\mathbf{Z}}|$ and $|\hat{\mathbf{V}}|$ are the cardinalities (the number of unique non-zeros) of $\hat{\mathbf{Z}}$ and $\hat{\mathbf{V}}$, respectively.[2] We select the set of tuning parameters $(\lambda_1, \lambda_2, \lambda_3)$ for which the quantity $\text{BIC}(\hat{\mathbf{\Theta}}, \hat{\mathbf{V}}, \hat{\mathbf{Z}})$ is minimized. Note that when the constant c is small, $\text{BIC}(\hat{\mathbf{\Theta}}, \hat{\mathbf{V}}, \hat{\mathbf{Z}})$ will favor more hub nodes in $\hat{\mathbf{V}}$. In this manuscript, we take $c = 0.2$.

3.5 Simulation Study

In this section, we compare HGL to two sets of proposals: proposals that learn an Erdős-Rényi Gaussian graphical model, and proposals that learn a Gaussian graphical model in which some nodes are highly-connected.

3.5.1 Notation and Measures of Performance

We start by defining some notation. Let $\hat{\mathbf{\Theta}}$ be the estimate of $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$ from a given proposal, and let $\hat{\mathbf{\Theta}}_j$ be its jth column. Let $\mathcal{H}$ denote the set of indices of the hub nodes in $\mathbf{\Theta}$ (that is, this is the set of true hub nodes in the graph), and let $|\mathcal{H}|$ denote the cardinality of the set. In addition, let $\hat{\mathcal{H}}_r$ be the set of *estimated hub nodes*: the set of nodes in $\hat{\mathbf{\Theta}}$ that are among the $|\mathcal{H}|$ most highly-connected nodes, and that have at least r edges. The values chosen for $|\mathcal{H}|$ and r depend on the simulation set-up, and will be specified in each simulation study.

We now define several measures of performance that will be used to evaluate the various methods.

- Number of correctly estimated edges: $\sum_{j<j'} \left(1_{\{|\hat{\Theta}_{jj'}| > 10^{-5} \text{ and } |\Theta_{jj'}| \neq 0\}}\right)$.
- Proportion of correctly estimated hub edges:

$$\frac{\sum_{j \in \mathcal{H}, j' \neq j} \left(1_{\{|\hat{\Theta}_{jj'}| > 10^{-5} \text{ and } |\Theta_{jj'}| \neq 0\}}\right)}{\sum_{j \in \mathcal{H}, j' \neq j} \left(1_{\{|\Theta_{jj'}| \neq 0\}}\right)}.$$

- Proportion of correctly estimated hub nodes: $\frac{|\hat{\mathcal{H}}_r \cap \mathcal{H}|}{|\mathcal{H}|}$.
- Sum of squared errors: $\sum_{j<j'} \left(\hat{\Theta}_{jj'} - \Theta_{jj'}\right)^2$.

2. The term $\log(n) \cdot |\hat{\mathbf{Z}}|$ is motivated by the degrees of freedom from the ℓ_1 penalty, and the term $\log(n) \cdot \left(\nu + c \cdot [|\hat{\mathbf{V}}| - \nu]\right)$ is motivated by an approximation of the degrees of freedom of the ℓ_2 penalty proposed in Yuan and Lin (2007b).

3.5.2 Data Generation

We consider three set-ups for generating a $p \times p$ adjacency matrix $\mathbf{A}$.

I - Network with hub nodes: for all $i < j$, we set $A_{ij} = 1$ with probability 0.02, and zero otherwise. We then set A_{ji} equal to A_{ij}. Next, we randomly select $|\mathcal{H}|$ hub nodes and set the elements of the corresponding rows and columns of $\mathbf{A}$ to equal one with probability 0.7 and zero otherwise.

II - Network with two connected components and hub nodes: the adjacency matrix is generated as $\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & 0 \\ 0 & \mathbf{A}_2 \end{pmatrix}$, with $\mathbf{A}_1$ and $\mathbf{A}_2$ as in Set-up I, each with $|\mathcal{H}|/2$ hub nodes.

III - Scale-free network:[3] the probability that a given node has k edges is proportional to $k^{-\alpha}$. Barabási and Albert (1999) observed that many real-world networks have $\alpha \in [2.1, 4]$; we took $\alpha = 2.5$. Note that there is no natural notion of hub nodes in a scale-free network. While some nodes in a scale-free network have more edges than one would expect in an Erdős-Rényi graph, there is no clear distinction between "hub" and "non-hub" nodes, unlike in Set-ups I and II. In our simulation settings, we consider any node that is connected to more than 5% of all other nodes to be a hub node.[4]

We then use the adjacency matrix $\mathbf{A}$ to create a matrix $\mathbf{E}$, as

$$E_{ij} \overset{\text{i.i.d.}}{\sim} \begin{cases} 0 & \text{if } A_{ij} = 0 \\ \text{Unif}([-0.75, -0.25] \cup [0.25, 0.75]) & \text{otherwise,} \end{cases}$$

and set $\bar{\mathbf{E}} = \frac{1}{2}(\mathbf{E} + \mathbf{E}^T)$. Given the matrix $\bar{\mathbf{E}}$, we set $\mathbf{\Sigma}^{-1}$ equal to $\bar{\mathbf{E}} + (0.1 - \Lambda_{\min}(\bar{\mathbf{E}}))\mathbf{I}$, where $\Lambda_{\min}(\bar{\mathbf{E}})$ is the smallest eigenvalue of $\bar{\mathbf{E}}$. We generate the data matrix $\mathbf{X}$ according to $\mathbf{x}_1, \ldots, \mathbf{x}_n \overset{\text{i.i.d.}}{\sim} N(\mathbf{0}, \mathbf{\Sigma})$. Then, variables are standardized to have standard deviation one.

3.5.3 Comparison to Graphical Lasso and Neighbourhood Selection

In this subsection, we compare the performance of HGL to two proposals that learn a sparse Gaussian graphical model.

- The graphical lasso (5), implemented using the `R` package `glasso`.
- The neighborhood selection approach of Meinshausen and Bühlmann (2006), implemented using the `R` package `glasso`. This approach involves performing p ℓ_1-penalized regression problems, each of which involves regressing one feature onto the others.

3. Recall that our proposal is not intended for estimating a scale-free network.
4. The cutoff threshold of 5% is chosen in order to capture the most highly-connected nodes in the scale-free network. In our simulation study, around three nodes are connected to at least $0.05 \times p$ other nodes in the network. The precise choice of cut-off threshold has little effect on the results obtained in the figures that follow.

We consider the three simulation set-ups described in the previous section with $n = 1000$, $p = 1500$, and $|\mathcal{H}| = 30$ hub nodes in Set-ups I and II. Figure 3 displays the results, averaged over 100 simulated data sets. Note that the sum of squared errors is not computed for Meinshausen and Bühlmann (2006), since it does not directly yield an estimate of $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$.

HGL has three tuning parameters. To obtain the curves shown in Figure 3, we fixed $\lambda_1 = 0.4$, considered three values of λ_3 (each shown in a different color in Figure 3), and used a fine grid of values of λ_2. The solid black circle in Figure 3 corresponds to the set of tuning parameters $(\lambda_1, \lambda_2, \lambda_3)$ for which the BIC as defined in Section 3.4 is minimized. The graphical lasso and Meinshausen and Bühlmann (2006) each involves one tuning parameter; we applied them using a fine grid of the tuning parameter to obtain the curves shown in Figure 3.

Results for Set-up I are displayed in Figures 3-I(a) through 3-I(d), where we calculate the proportion of correctly estimated hub nodes as defined in Section 3.5.1 with $r = 300$. Since this simulation set-up exactly matches the assumptions of HGL, it is not surprising that HGL outperforms the other methods. In particular, HGL is able to identify most of the hub nodes when the number of estimated edges is approximately equal to the true number of edges. We see similar results for Set-up II in Figures 3-II(a) through 3-II(d), where the proportion of correctly estimated hub nodes is as defined in Section 3.5.1 with $r = 150$.

In Set-up III, recall that we define a node that is connected to at least 5% of all nodes to be a hub. The proportion of correctly estimated hub nodes is then as defined in Section 3.5.1 with $r = 0.05 \times p$. The results are presented in Figures 3-III(a) through 3-III(d). In this set-up, only approximately three of the nodes (on average) have more than 50 edges, and the hub nodes are not as highly-connected as in Set-up I or Set-up II. Nonetheless, HGL outperforms the graphical lasso and Meinshausen and Bühlmann (2006).

Finally, we see from Figure 3 that the set of tuning parameters $(\lambda_1, \lambda_2, \lambda_3)$ selected using BIC performs reasonably well. In particular, the graphical lasso solution always has BIC larger than HGL, and hence, is not selected.

3.5.4 Comparison to Additional Proposals

In this subsection, we compare the performance of HGL to three additional proposals:

- The partial correlation screening procedure of Hero and Rajaratnam (2012). The elements of the partial correlation matrix (computed using a pseudo-inverse when $p > n$) are thresholded based on their absolute value, and a hub node is declared if the number of nonzero elements in the corresponding column of the thresholded partial correlation matrix is sufficiently large. Note that the purpose of Hero and Rajaratnam (2012) is to screen for hub nodes, rather than to estimate the individual edges in the network.

- The scale-free network estimation procedure of Liu and Ihler (2011). This is the solution to the non-convex optimization problem

$$\underset{\mathbf{\Theta} \in \mathcal{S}}{\text{minimize}} \quad \left\{ -\log\det \mathbf{\Theta} + \text{trace}(\mathbf{S}\mathbf{\Theta}) + \alpha \sum_{j=1}^{p} \log(\|\theta_{\setminus j}\|_1 + \epsilon_j) + \sum_{j=1}^{p} \beta_j |\theta_{jj}| \right\}, \qquad (8)$$

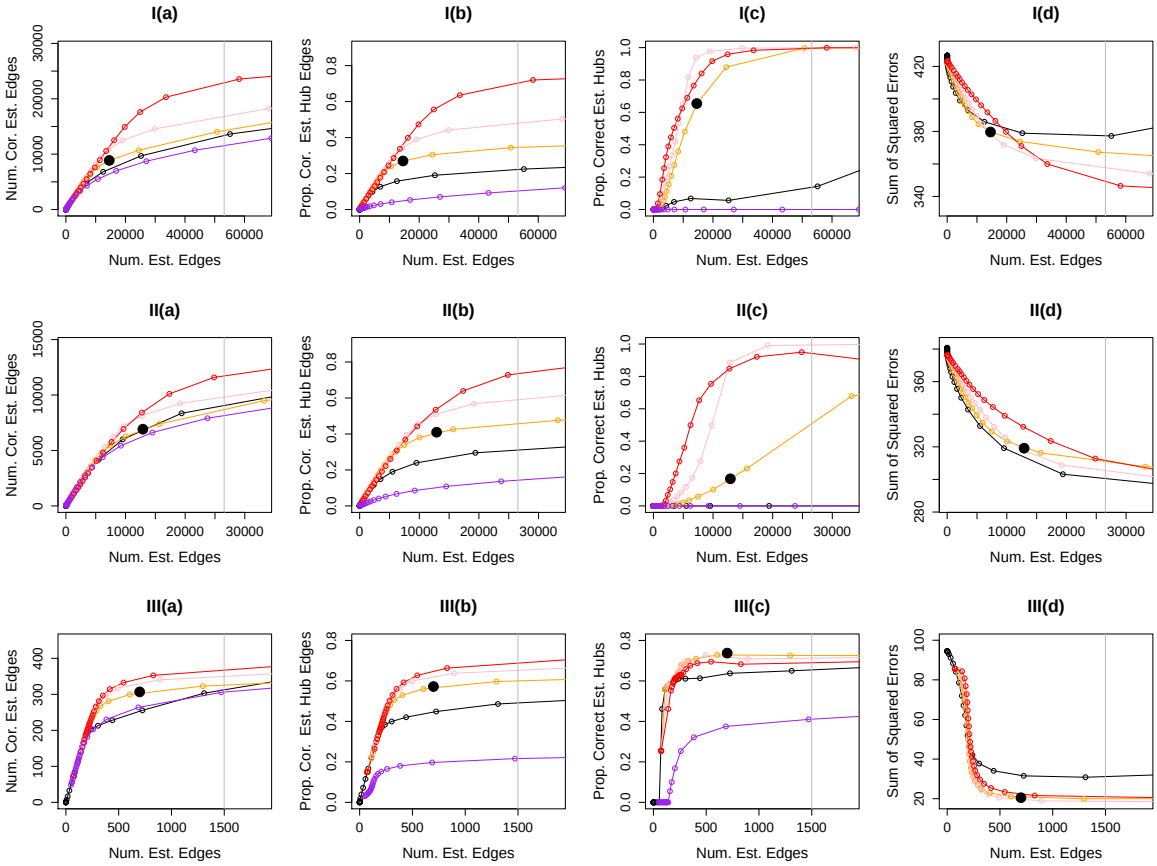


Figure 3: Simulation for Gaussian graphical model. Row I: Results for Set-up I. Row II: Results for Set-up II. Row III: Results for Set-up III. The results are for $n = 1000$ and $p = 1500$. In each panel, the x-axis displays the number of estimated edges, and the vertical gray line is the number of edges in the true network. The y-axes are as follows: Column (a): Number of correctly estimated edges; Column (b): Proportion of correctly estimated hub edges; Column (c): Proportion of correctly estimated hub nodes; Column (d): Sum of squared errors. The black solid circles are the results for HGL based on tuning parameters selected using the BIC-type criterion defined in Section 3.4. Colored lines correspond to the graphical lasso (Friedman et al., 2007) (——); HGL with $\lambda_3 = 0.5$ (——), $\lambda_3 = 1$ (——), and $\lambda_3 = 2$ (——); neighborhood selection (Meinshausen and Bühlmann, 2006) (——).

where $\theta_{\setminus j} = \{\theta_{jj'} | j' \neq j\}$, and ϵ_j, β_j, and α are tuning parameters. Here, $\mathcal{S} = \{\mathbf{\Theta} : \mathbf{\Theta} \succ 0 \text{ and } \mathbf{\Theta} = \mathbf{\Theta}^T\}$.

- Sparse partial correlation estimation procedure of Peng et al. (2009), implemented using the `R` package `space`. This is an extension of the neighborhood selection approach of Meinshausen and Bühlmann (2006) that combines p ℓ_1-penalized regression problems in order to obtain a symmetric estimator. The authors claimed that the proposal performs well in estimating a scale-free network.

We generated data under Set-ups I and III (described in Section 3.5.2) with $n = 250$ and $p = 500$,[5] and with $|\mathcal{H}| = 10$ for Set-up I. The results, averaged over 100 data sets, are displayed in Figures 4 and 5.

To obtain Figures 4 and 5, we applied Liu and Ihler (2011) using a fine grid of α values, and using the choices for β_j and ϵ_j specified by the authors: $\beta_j = 2\alpha/\epsilon_j$, where ϵ_j is a small constant specified in Liu and Ihler (2011). There are two tuning parameters in Hero and Rajaratnam (2012): (1) ρ, the value used to threshold the partial correlation matrix, and (2) d, the number of non-zero elements required for a column of the thresholded matrix to be declared a hub node. We used $d = \{10, 20\}$ in Figures 4 and 5, and used a fine grid of values for ρ. Note that the value of d has no effect on the results for Figures 4(a)-(b) and Figures 5(a)-(b), and that larger values of d tend to yield worse results in Figures 4(c) and 5(c). For Peng et al. (2009), we used a fine grid of tuning parameter values to obtain the curves shown in Figures 4 and 5. The sum of squared errors was not reported for Peng et al. (2009) and Hero and Rajaratnam (2012) since they do not directly yield an estimate of the precision matrix. As a baseline reference, the graphical lasso is included in the comparison.

We see from Figure 4 that HGL outperforms the competitors when the underlying network contains hub nodes. It is not surprising that Liu and Ihler (2011) yields better results than the graphical lasso, since the former approach is implemented via an iterative procedure: in each iteration, the graphical lasso is performed with an updated tuning parameter based on the estimate obtained in the previous iteration. Hero and Rajaratnam (2012) has the worst results in Figures 4(a)-(b); this is not surprising, since the purpose of Hero and Rajaratnam (2012) is to screen for hub nodes, rather than to estimate the individual edges in the network.

From Figure 5, we see that the performance of HGL is comparable to that of Liu and Ihler (2011) and Peng et al. (2009) under the assumption of a scale-free network; note that this is the precise setting for which Liu and Ihler (2011)'s proposal is intended, and Peng et al. (2009) reported that their proposal performs well in this setting. In contrast, HGL is not intended for the scale-free network setting (as mentioned in the Introduction, it is intended for a setting with hub nodes). Again, Liu and Ihler (2011) and Peng et al. (2009) outperform the graphical lasso, and Hero and Rajaratnam (2012) has the worst results in Figures 5(a)-(b). Finally, we see from Figures 4 and 5 that the BIC-type criterion for HGL proposed in Section 3.4 yields good results.

5. In this subsection, a small value of p was used due to the computations required to run the R package space, as well as computational demands of the Liu and Ihler (2011) algorithm.

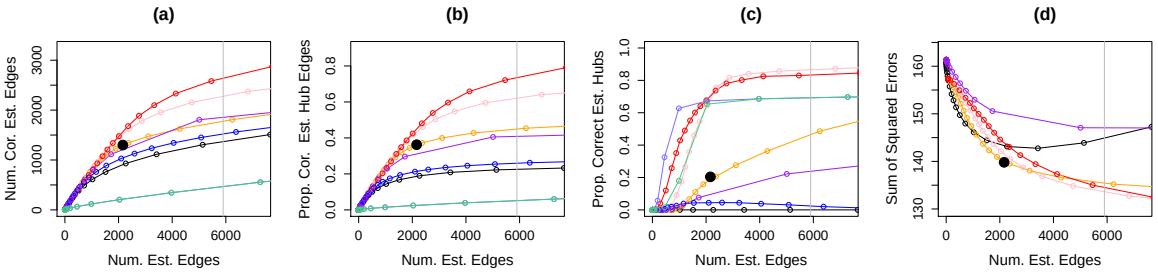


Figure 4: Simulation for the Gaussian graphical model. Set-up I was applied with $n = 250$ and $p = 500$. Details of the axis labels and the solid black circles are as in Figure 3. The colored lines correspond to the graphical lasso (Friedman et al., 2007) (——); HGL with $\lambda_3 = 1$ (——), $\lambda_3 = 2$ (——), and $\lambda_3 = 3$ (——); the hub screening procedure (Hero and Rajaratnam, 2012) with $d = 10$ (——) and $d = 20$ (——); the scale-free network approach (Liu and Ihler, 2011) (——); sparse partial correlation estimation (Peng et al., 2009) (——).

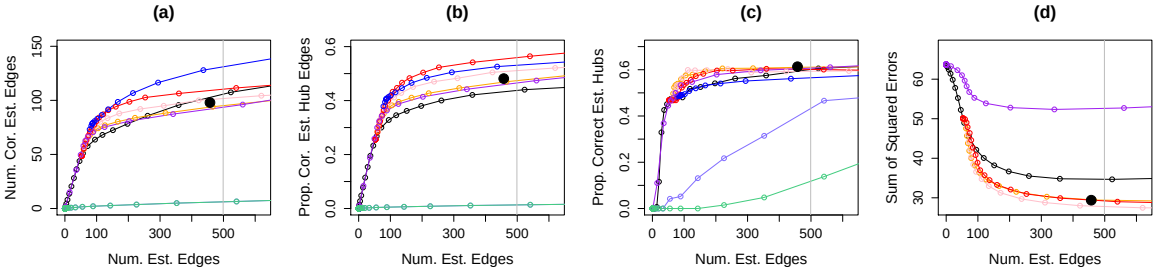


Figure 5: Simulation for the Gaussian graphical model. Set-up III was applied with $n = 250$ and $p = 500$. Details of the axis labels and the solid black circles are as in Figure 3. The colored lines correspond to the graphical lasso (Friedman et al., 2007) (——); HGL with $\lambda_3 = 1$ (——), $\lambda_3 = 2$ (——), and $\lambda_3 = 3$ (——); the hub screening procedure (Hero and Rajaratnam, 2012) with $d = 10$ (——) and $d = 20$ (——); the scale-free network approach (Liu and Ihler, 2011) (——); sparse partial correlation estimation (Peng et al., 2009) (——).

4. The Hub Covariance Graph

In this section, we consider estimation of a covariance matrix under the assumption that $\mathbf{x}_1, \ldots, \mathbf{x}_n \overset{\text{i.i.d.}}{\sim} N(\mathbf{0}, \mathbf{\Sigma})$; this is of interest because the sparsity pattern of $\mathbf{\Sigma}$ specifies the structure of the marginal independence graph (see, e.g., Drton and Richardson, 2003; Chaudhuri et al., 2007; Drton and Richardson, 2008). We extend the covariance estimator of Xue et al. (2012) to accommodate hub nodes.

4.1 Formulation and Algorithm

Xue et al. (2012) proposed to estimate $\mathbf{\Sigma}$ using

$$\hat{\mathbf{\Sigma}} = \underset{\mathbf{\Sigma} \in \mathcal{S}}{\arg\min} \left\{ \frac{1}{2} \|\mathbf{\Sigma} - \mathbf{S}\|_F^2 + \lambda \|\mathbf{\Sigma}\|_1 \right\}, \tag{9}$$

where $\mathbf{S}$ is the empirical covariance matrix, $\mathcal{S} = \{\mathbf{\Sigma} : \mathbf{\Sigma} \succeq \epsilon \mathbf{I} \text{ and } \mathbf{\Sigma} = \mathbf{\Sigma}^T\}$, and ϵ is a small positive constant; we take $\epsilon = 10^{-4}$. We extend (9) to accommodate hubs by imposing the hub penalty function (2) on $\mathbf{\Sigma}$. This results in the *hub covariance graph* (HCG) optimization problem,

$$\underset{\mathbf{\Sigma} \in \mathcal{S}}{\text{minimize}} \qquad \left\{ \frac{1}{2} \|\mathbf{\Sigma} - \mathbf{S}\|_F^2 + \mathrm{P}(\mathbf{\Sigma}) \right\},$$

which can be solved via Algorithm 1. To update $\mathbf{\Theta} = \mathbf{\Sigma}$ in Step 2(a)i, we note that

$$\underset{\mathbf{\Sigma} \in \mathcal{S}}{\arg\min} \left\{ \frac{1}{2} \|\mathbf{\Sigma} - \mathbf{S}\|_F^2 + \frac{\rho}{2} \|\mathbf{\Sigma} - \tilde{\mathbf{\Sigma}} + \mathbf{W}_1\|_F^2 \right\} = \frac{1}{1+\rho} (\mathbf{S} + \rho \tilde{\mathbf{\Sigma}} - \rho \mathbf{W}_1)^+,$$

where $(\mathbf{A})^+$ is the projection of a matrix $\mathbf{A}$ onto the convex cone $\{\mathbf{\Sigma} \succeq \epsilon \mathbf{I}\}$. That is, if $\sum_{j=1}^p d_j \mathbf{u}_j \mathbf{u}_j^T$ denotes the eigen-decomposition of the matrix $\mathbf{A}$, then $(\mathbf{A})^+$ is defined as $\sum_{j=1}^p \max(d_j, \epsilon) \mathbf{u}_j \mathbf{u}_j^T$. The complexity of the ADMM algorithm is $O(p^3)$ per iteration, due to the complexity of the eigen-decomposition for updating $\mathbf{\Sigma}$.

4.2 Simulation Study

We compare HCG to two competitors for obtaining a sparse estimate of $\mathbf{\Sigma}$:

1. The non-convex ℓ_1-penalized log-likelihood approach of Bien and Tibshirani (2011), using the `R` package `spcov`. This approach solves

$$\underset{\mathbf{\Sigma} \succ 0}{\text{minimize}} \left\{ \log\det \mathbf{\Sigma} + \mathrm{trace}(\mathbf{\Sigma}^{-1}\mathbf{S}) + \lambda \|\mathbf{\Sigma}\|_1 \right\}.$$

2. The convex ℓ_1-penalized approach of Xue et al. (2012), given in (9).

We first generated an adjacency matrix $\mathbf{A}$ as in Set-up I in Section 3.5.2, modified to have $|\mathcal{H}| = 20$ hub nodes. Then $\bar{\mathbf{E}}$ was generated as described in Section 3.5.2, and we set $\mathbf{\Sigma}$ equal to $\bar{\mathbf{E}} + (0.1 - \Lambda_{\min}(\bar{\mathbf{E}}))\mathbf{I}$. Next, we generated $\mathbf{x}_1, \ldots, \mathbf{x}_n \overset{\text{i.i.d.}}{\sim} N(\mathbf{0}, \mathbf{\Sigma})$. Finally, we standardized the variables to have standard deviation one. In this simulation study, we set $n = 500$ and $p = 1000$.

Figure 6 displays the results, averaged over 100 simulated data sets. We calculated the proportion of correctly estimated hub nodes as defined in Section 3.3.1 with $r = 200$. We used a fine grid of tuning parameters for Xue et al. (2012) in order to obtain the curves shown in each panel of Figure 6. HCG involves three tuning parameters, λ_1, λ_2, and λ_3. We fixed $\lambda_1 = 0.2$, considered three values of λ_3 (each shown in a different color), and varied λ_2 in order to obtain the curves shown in Figure 6.

Figure 6 does not display the results for the proposal of Bien and Tibshirani (2011), due to computational constraints in the `spcov` `R` package. Instead, we compared our proposal to that of Bien and Tibshirani (2011) using $n = 100$ and $p = 200$; those results are presented in Figure 10 in Appendix D.

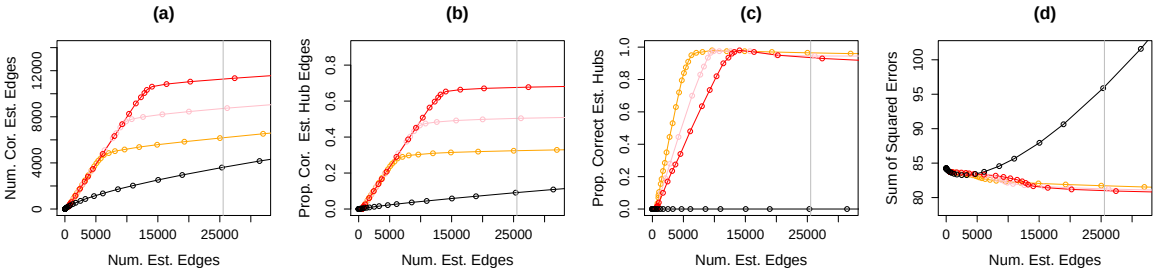


Figure 6: Covariance graph simulation with $n = 500$ and $p = 1000$. Details of the axis labels are as in Figure 3. The colored lines correspond to the proposal of Xue et al. (2012) (——); HCG with $\lambda_3 = 1$ (——), $\lambda_3 = 1.5$ (——), and $\lambda_3 = 2$ (——).

We see that HCG outperforms the proposals of Xue et al. (2012) (Figures 6 and 10) and Bien and Tibshirani (2011) (Figure 10). These results are not surprising, since those other methods do not explicitly model the hub nodes.

5. The Hub Binary Network

In this section, we focus on estimating a binary Ising Markov random field, which we refer to as a binary network. We refer the reader to Ravikumar et al. (2010) for an in-depth discussion of this type of graphical model and its applications.

In this set-up, each entry of the $n \times p$ data matrix $\mathbf{X}$ takes on a value of zero or one. We assume that the observations $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are i.i.d. with density

$$p(\mathbf{x}, \mathbf{\Theta}) = \frac{1}{Z(\mathbf{\Theta})} \exp\left[\sum_{j=1}^{p} \theta_{jj} x_j + \sum_{1 \le j < j' \le p} \theta_{jj'} x_j x_{j'}\right], \tag{10}$$

where $Z(\mathbf{\Theta})$ is the partition function, which ensures that the density sums to one. Here $\mathbf{\Theta}$ is a $p \times p$ symmetric matrix that specifies the network structure: $\theta_{jj'} = 0$ implies that the jth and j'th variables are conditionally independent.

In order to obtain a sparse graph, Lee et al. (2007) considered maximizing an ℓ_1-penalized log-likelihood under this model. Due to the difficulty in computing the log-

partition function, several authors have considered alternative approaches. For instance, Ravikumar et al. (2010) proposed a neighborhood selection approach. The proposal of Ravikumar et al. (2010) involves solving p logistic regression separately, and hence, the estimated parameter matrix is not symmetric. In contrast, several authors considered maximizing an ℓ_1-penalized pseudo-likelihood with a symmetric constraint on $\mathbf{\Theta}$ (see, e.g., Höfling and Tibshirani, 2009; Guo et al., 2010, 2011).

5.1 Formulation and Algorithm

Under the model (10), the log-pseudo-likelihood for n observations takes the form

$$\sum_{j=1}^{p}\sum_{j'=1}^{p}\theta_{jj'}(\mathbf{X}^T\mathbf{X})_{jj'} - \sum_{i=1}^{n}\sum_{j=1}^{p}\log\left(1+\exp\left[\theta_{jj}+\sum_{j'\neq j}\theta_{jj'}x_{ij'}\right]\right), \tag{11}$$

where $\mathbf{x}_i$ is the ith row of the $n\times p$ matrix $\mathbf{X}$. The proposal of Höfling and Tibshirani (2009) involves maximizing (11) subject to an ℓ_1 penalty on $\mathbf{\Theta}$. We propose to instead impose the hub penalty function (2) on $\mathbf{\Theta}$ in (11) in order to estimate a sparse binary network with hub nodes. This leads to the optimization problem

$$\underset{\mathbf{\Theta}\in\mathcal{S}}{\text{minimize}} \quad \left\{-\sum_{j=1}^{p}\sum_{j'=1}^{p}\theta_{jj'}(\mathbf{X}^T\mathbf{X})_{jj'} + \sum_{i=1}^{n}\sum_{j=1}^{p}\log\left(1+\exp\left[\theta_{jj}+\sum_{j'\neq j}\theta_{jj'}x_{ij'}\right]\right) + \mathrm{P}(\mathbf{\Theta})\right\}, \tag{12}$$

where $\mathcal{S}=\{\mathbf{\Theta}:\mathbf{\Theta}=\mathbf{\Theta}^T\}$. We refer to the solution to (12) as the *hub binary network* (HBN). The ADMM algorithm for solving (12) is given in Algorithm 1. We solve the update for $\mathbf{\Theta}$ in Step 2(a)i using the Barzilai-Borwein method (Barzilai and Borwein, 1988). The details are given in Appendix F.

5.2 Simulation Study

Here we compare the performance of HBN to the proposal of Höfling and Tibshirani (2009), implemented using the `R` package `BMN`.

We simulated a binary network with $p=50$ and $|\mathcal{H}|=5$ hub nodes. To generate the parameter matrix $\mathbf{\Theta}$, we created an adjacency matrix $\mathbf{A}$ as in Set-up I of Section 3.5.2 with five hub nodes. Then $\bar{\mathbf{E}}$ was generated as in Section 3.5.2, and we set $\mathbf{\Theta}=\bar{\mathbf{E}}$.

Each of $n=100$ observations was generated using Gibbs sampling (Ravikumar et al., 2010; Guo et al., 2010). Suppose that $x_1^{(t)},\ldots,x_p^{(t)}$ is obtained at the tth iteration of the Gibbs sampler. Then, the $(t+1)$th iteration is obtained according to

$$x_j^{(t+1)} \sim \text{Bernoulli}\left(\frac{\exp(\theta_{jj}+\sum_{j\neq j'}\theta_{jj'}x_{j'}^{(t)})}{1+\exp(\theta_{jj}+\sum_{j\neq j'}\theta_{jj'}x_{j'}^{(t)})}\right) \qquad \text{for } j=1,\ldots,p.$$

We took the first 10^5 iterations as our burn-in period, and then collected an observation every 10^4 iterations, such that the observations were nearly independent (Guo et al., 2010).

The results, averaged over 100 data sets, are shown in Figure 7. We used a fine grid of values for the ℓ_1 tuning parameter for Höfling and Tibshirani (2009), resulting in curves

shown in each panel of the figure. For HBN, we fixed $\lambda_1 = 5$, considered $\lambda_3 = \{15, 25, 30\}$, and used a fine grid of values of λ_2. The proportion of correctly estimated hub nodes was calculated using the definition in Section 3.5.1 with $r = 20$. Figure 7 indicates that HBN consistently outperforms the proposal of Höfling and Tibshirani (2009).

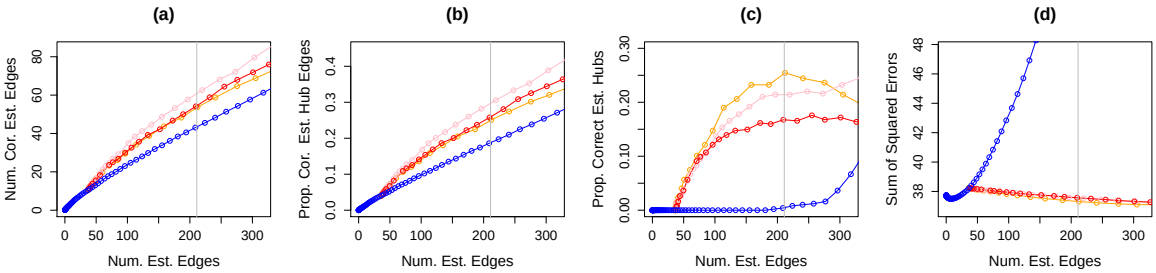


Figure 7: Binary network simulation with $n = 100$ and $p = 50$. Details of the axis labels are as in Figure 3. The colored lines correspond to the ℓ_1-penalized pseudo-likelihood proposal of Höfling and Tibshirani (2009) (——); and HBN with $\lambda_3 = 15$ (——), $\lambda_3 = 25$ (——), and $\lambda_3 = 30$ (——).

6. Real Data Application

We now apply HGL to a university webpage data set, and a brain cancer data set.

6.1 Application to University Webpage Data

We applied HGL to the university webpage 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 consists of the occurrences of various terms (words) on webpages from four computer science departments at Cornell, Texas, Washington and Wisconsin. We consider only the 544 student webpages, and select 100 terms with the largest entropy for our analysis. In what follows, we model these 100 terms as the nodes in a Gaussian graphical model.

The goal of the analysis is to understand the relationships among the terms that appear on the student webpages. In particular, we wish to identify terms that are hubs. We are not interested in identifying edges between non-hub nodes. For this reason, we fix the tuning parameter that controls the sparsity of $\mathbf{Z}$ at $\lambda_1 = 0.45$ such that the matrix $\mathbf{Z}$ is sparse. In the interest of a graph that is interpretable, we fix $\lambda_3 = 1.5$ to obtain only a few hub nodes, and then select a value of λ_2 ranging from 0.1 to 0.5 using the BIC-type criterion presented in Section 3.4. We performed HGL with the selected tuning parameters $\lambda_1 = 0.45$, $\lambda_2 = 0.25$, and $\lambda_3 = 1.5$.[6] The estimated matrices are shown in Figure 8.

Figure 8(a) indicates that six hub nodes are detected: *comput*, *research*, *scienc*, *software*, *system*, and *work*. For instance, the fact that *comput* is a hub indicates that many terms'

6. The results are qualitatively similar for different values of λ_1.

occurrences are explained by the occurrence of the word *comput*. From Figure 8(b), we see that several pairs of terms take on non-zero values in the matrix ($\mathbf{Z}-\text{diag}(\mathbf{Z})$). These include *(depart, univers)*; *(home, page)*; *(institut, technolog)*; *(graduat, student)*; *(univers, scienc)*, and *(languag,program)*. These results provide an intuitive explanation of the relationships among the terms in the webpages.

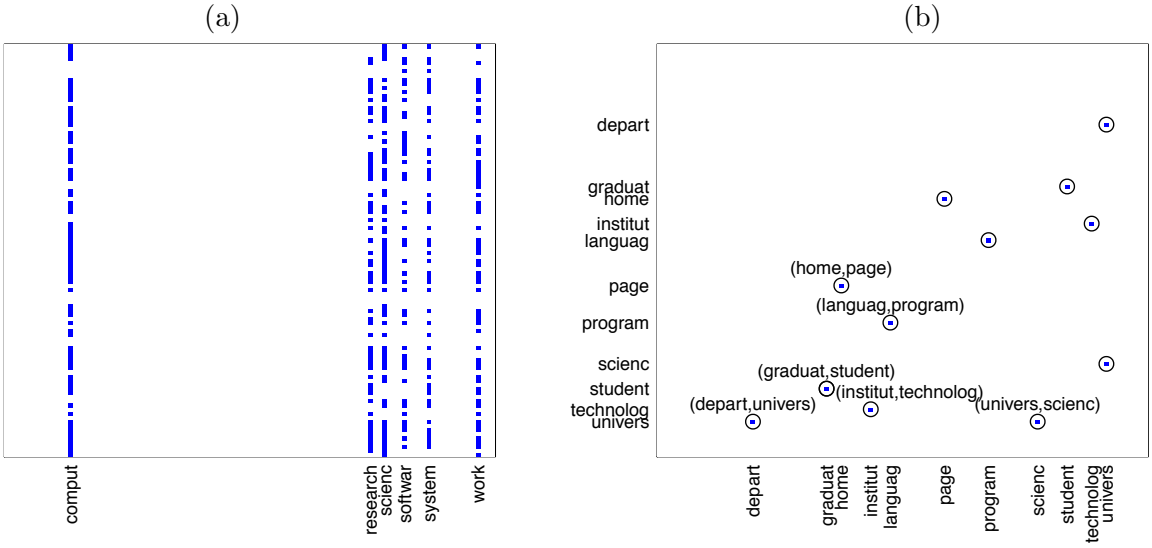


Figure 8: Results for HGL on the webpage data with tuning parameters selected using BIC: $\lambda_1 = 0.45$, $\lambda_2 = 0.25$, $\lambda_3 = 1.5$. Non-zero estimated values are shown, for *(a):* $(\mathbf{V} - \text{diag}(\mathbf{V}))$, and *(b):* $(\mathbf{Z} - \text{diag}(\mathbf{Z}))$.

6.2 Application to Gene Expression Data

We applied HGL to a publicly available cancer gene expression data set (Verhaak et al., 2010). The data set consists of mRNA expression levels for 17,814 genes in 401 patients with glioblastoma multiforme (GBM), an extremely aggressive cancer with very poor patient prognosis. Among 7,462 genes known to be associated with cancer (Rappaport et al., 2013), we selected 500 genes with the highest variance.

We aim to reconstruct the gene regulatory network that represents the interactions among the genes, as well as to identify hub genes that tend to have many interactions with other genes. Such genes likely play an important role in regulating many other genes in the network. Identifying such regulatory genes will lead to a better understanding of brain cancer, and eventually may lead to new therapeutic targets. Since we are interested in identifying hub genes, and not as interested in identifying edges between non-hub nodes, we fix $\lambda_1 = 0.6$ such that the matrix $\mathbf{Z}$ is sparse. We fix $\lambda_3 = 6.5$ to obtain a few hub nodes, and we select λ_2 ranging from 0.1 to 0.7 using the BIC-type criterion presented in Section 3.4.

We applied HGL with this set of tuning parameters to the empirical covariance matrix corresponding to the 401×500 data matrix, after standardizing each gene to have variance one. In Figure 9, we plotted the resulting network (for simplicity, only the 438 genes with at least two neighbors are displayed). We found that five genes are identified as hubs. These genes are TRIM48, TBC1D2B, PTPN2, ACRC, and ZNF763, in decreasing order of estimated edges.

Interestingly, some of these genes have known regulatory roles. PTPN2 is known to be a signaling molecule that regulates a variety of cellular processes including cell growth, differentiation, mitotic cycle, and oncogenic transformation (Maglott et al., 2004). ZNF763 is a DNA-binding protein that regulates the transcription of other genes (Maglott et al., 2004). These genes do not appear to be highly-connected to many other genes in the estimate that results from applying the graphical lasso (5) to this same data set (results not shown). These results indicate that HGL can be used to recover known regulators, as well as to suggest other potential regulators that may be targets for follow-up analysis.

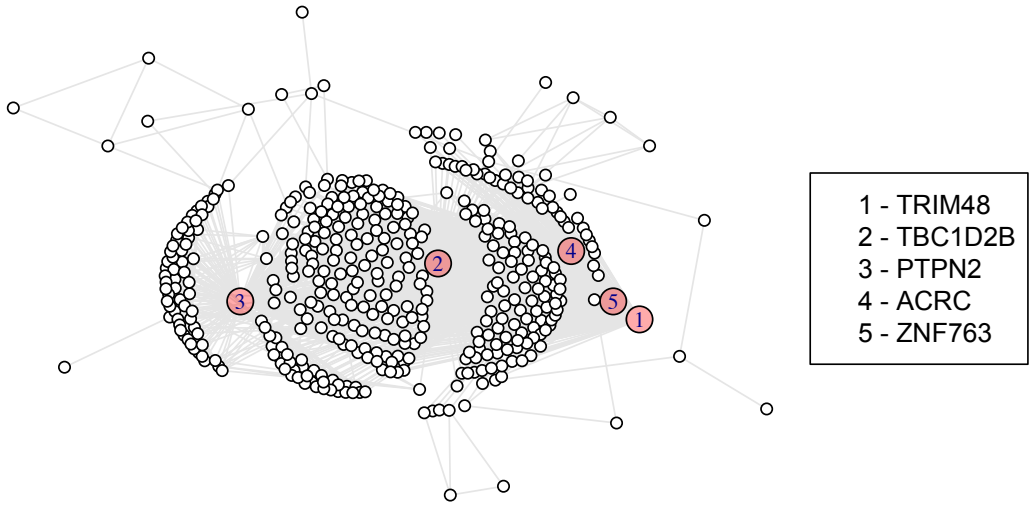


Figure 9: Results for HGL on the GBM data with tuning parameters selected using BIC: $\lambda_1 = 0.6$, $\lambda_2 = 0.4$, $\lambda_3 = 6.5$. Only nodes with at least two edges in the estimated network are displayed. Nodes displayed in pink were found to be hubs by the HGL algorithm.

7. Discussion

We have proposed a general framework for estimating a network with hubs by way of a convex penalty function. The proposed framework has three tuning parameters, so that it can flexibly accommodate different numbers of hubs, sparsity levels within a hub, and connectivity levels among non-hubs. We have proposed a BIC-type quantity to select tuning parameters for our proposal. We note that tuning parameter selection in unsupervised

settings remains a challenging open problem (see, e.g., Foygel and Drton, 2010; Meinshausen and Bühlmann, 2010). In practice, tuning parameters could also be set based on domain knowledge or a desire for interpretability of the resulting estimates.

The framework proposed in this paper assumes an underlying model involving a set of edges between non-hub nodes, as well as a set of hub nodes. For instance, it is believed that such hub nodes arise in biology, in which "super hubs" in transcriptional regulatory networks may play important roles (Hao et al., 2012). We note here that the underlying model of hub nodes assumed in this paper differs fundamentally from a scale-free network in which the degree of connectivity of the nodes follows a power law distribution—scale-free networks simply do not have such very highly-connected hub nodes. In fact, we have shown that existing techniques for estimating a scale-free network, such as Liu and Ihler (2011) and Defazio and Caetano (2012), cannot accommodate the very dense hubs for which our proposal is intended.

As discussed in Section 2, the hub penalty function involves decomposing a parameter matrix $\mathbf{\Theta}$ into $\mathbf{Z} + \mathbf{V} + \mathbf{V}^T$, where $\mathbf{Z}$ is a sparse matrix, and $\mathbf{V}$ is a matrix whose columns are entirely zero or (almost) entirely non-zero. In this paper, we used an ℓ_1 penalty on $\mathbf{Z}$ in order to encourage it to be sparse. In effect, this amounts to assuming that the non-hub nodes obey an Erdős-Rényi network. But our formulation could be easily modified to accommodate a different network prior for the non-hub nodes. For instance, we could assume that the non-hub nodes obey a scale-free network, using the ideas developed in Liu and Ihler (2011) and Defazio and Caetano (2012). This would amount to modeling a scale-free network with hub nodes.

In this paper, we applied the proposed framework to the tasks of estimating a Gaussian graphical model, a covariance graph model, and a binary network. The proposed framework can also be applied to other types of graphical models, such as the Poisson graphical model (Allen and Liu, 2012) or the exponential family graphical model (Yang et al., 2012a).

In future work, we will study the theoretical statistical properties of the HGL formulation. For instance, in the context of the graphical lasso, it is known that the rate of statistical convergence depends upon the maximal degree of any node in the network (Ravikumar et al., 2011). It would be interesting to see whether HGL theoretically outperforms the graphical lasso in the setting in which the true underlying network contains hubs. Furthermore, it will be of interest to study HGL's hub recovery properties from a theoretical perspective.

An `R` package `hglasso` is publicly available on the authors' websites and on `CRAN`.

Acknowledgments

We thank three reviewers for helpful comments that improved the quality of this manuscript. We thank Qiang Liu helpful responses to our inquiries regarding Liu and Ihler (2011). The authors acknowledge funding from the following sources: NIH DP5OD009145 and NSF CAREER DMS-1252624 and Sloan Research Fellowship to DW, NSF CAREER ECCS-0847077 to MF, and Univ. Washington Royalty Research Fund to DW, MF, and SL.

Appendix A. Derivation of Algorithm 1

Recall that the scaled augmented Lagrangian for (4) takes the form

$$L(\mathbf{B}, \tilde{\mathbf{B}}, \mathbf{W}) = \ell(\mathbf{X}, \mathbf{\Theta}) + \lambda_1 \|\mathbf{Z} - \mathrm{diag}(\mathbf{Z})\|_1 + \lambda_2 \|\mathbf{V} - \mathrm{diag}(\mathbf{V})\|_1 + \lambda_3 \sum_{j=1}^{p} \|(\mathbf{V} - \mathrm{diag}(\mathbf{V}))_j\|_2 + g(\tilde{\mathbf{B}}) + \frac{\rho}{2}\|\mathbf{B} - \tilde{\mathbf{B}} + \mathbf{W}\|_F^2. \tag{13}$$

The proposed ADMM algorithm requires the following updates:

1. $\mathbf{B}^{(t+1)} \leftarrow \underset{\mathbf{B}}{\mathrm{argmin}}\ L(\mathbf{B}, \tilde{\mathbf{B}}^t, \mathbf{W}^t)$,
2. $\tilde{\mathbf{B}}^{(t+1)} \leftarrow \underset{\tilde{\mathbf{B}}}{\mathrm{argmin}}\ L(\mathbf{B}^{(t+1)}, \tilde{\mathbf{B}}, \mathbf{W}^t)$,
3. $\mathbf{W}^{(t+1)} \leftarrow \mathbf{W}^t + \mathbf{B}^{(t+1)} - \tilde{\mathbf{B}}^{(t+1)}$.

We now proceed to derive the updates for $\mathbf{B}$ and $\tilde{\mathbf{B}}$.

Updates for B

To obtain updates for $\mathbf{B} = (\mathbf{\Theta}, \mathbf{V}, \mathbf{Z})$, we exploit the fact that (13) is separable in $\mathbf{\Theta}, \mathbf{V}$, and $\mathbf{Z}$. Therefore, we can simply update with respect to $\mathbf{\Theta}, \mathbf{V}$, and $\mathbf{Z}$ one-at-a-time. Update for $\mathbf{\Theta}$ depends on the form of the convex loss function, and is addressed in the main text. Updates for $\mathbf{V}$ and $\mathbf{Z}$ can be easily seen to take the form given in Algorithm 1.

Updates for $\tilde{\mathbf{B}}$

Minimizing the function in (13) with respect to $\tilde{\mathbf{B}}$ is equivalent to

$$\begin{aligned} \underset{\tilde{\mathbf{\Theta}}, \tilde{\mathbf{V}}, \tilde{\mathbf{Z}}}{\text{minimize}} \quad & \left\{ \frac{\rho}{2}\|\mathbf{\Theta} - \tilde{\mathbf{\Theta}} + \mathbf{W}_1\|_F^2 + \frac{\rho}{2}\|\mathbf{V} - \tilde{\mathbf{V}} + \mathbf{W}_2\|_F^2 + \frac{\rho}{2}\|\mathbf{Z} - \tilde{\mathbf{Z}} + \mathbf{W}_3\|_F^2 \right\} \\ \text{subject to} \quad & \tilde{\mathbf{\Theta}} = \tilde{\mathbf{Z}} + \tilde{\mathbf{V}} + \tilde{\mathbf{V}}^T. \end{aligned} \tag{14}$$

Let $\mathbf{\Gamma}$ be the $p \times p$ Lagrange multiplier matrix for the equality constraint. Then, the Lagrangian for (14) is

$$\frac{\rho}{2}\|\mathbf{\Theta} - \tilde{\mathbf{\Theta}} + \mathbf{W}_1\|_F^2 + \frac{\rho}{2}\|\mathbf{V} - \tilde{\mathbf{V}} + \mathbf{W}_2\|_F^2 + \frac{\rho}{2}\|\mathbf{Z} - \tilde{\mathbf{Z}} + \mathbf{W}_3\|_F^2 + \langle \mathbf{\Gamma}, \tilde{\mathbf{\Theta}} - \tilde{\mathbf{Z}} - \tilde{\mathbf{V}} - \tilde{\mathbf{V}}^T \rangle.$$

A little bit of algebra yields

$$\tilde{\mathbf{\Theta}} = \mathbf{\Theta} + \mathbf{W}_1 - \frac{1}{\rho}\mathbf{\Gamma},$$

$$\tilde{\mathbf{V}} = \frac{1}{\rho}(\mathbf{\Gamma} + \mathbf{\Gamma}^T) + \mathbf{V} + \mathbf{W}_2,$$

$$\tilde{\mathbf{Z}} = \frac{1}{\rho}\mathbf{\Gamma} + \mathbf{Z} + \mathbf{W}_3,$$

where $\mathbf{\Gamma} = \frac{\rho}{6}[(\mathbf{\Theta} + \mathbf{W}_1) - (\mathbf{V} + \mathbf{W}_2) - (\mathbf{V} + \mathbf{W}_2)^T - (\mathbf{Z} + \mathbf{W}_3)]$.

Appendix B. Conditions for HGL Solution to be Block-Diagonal

We begin by introducing some notation. Let $\|\mathbf{V}\|_{u,v}$ be the ℓ_u/ℓ_v norm of a matrix $\mathbf{V}$. For instance, $\|\mathbf{V}\|_{1,q} = \sum_{j=1}^{p} \|\mathbf{V}_j\|_q$. We define the support of a matrix $\mathbf{\Theta}$ as follows: $\text{supp}(\mathbf{\Theta}) = \{(i,j) : \Theta_{ij} \neq 0\}$. We say that $\mathbf{\Theta}$ is supported on a set $\mathcal{G}$ if $\text{supp}(\mathbf{\Theta}) \subseteq \mathcal{G}$. Let $\{C_1, \ldots, C_K\}$ be a partition of the index set $\{1, \ldots, p\}$, and let $\mathcal{T} = \cup_{k=1}^{K} \{C_k \times C_k\}$. We let $\mathbf{A}_{\mathcal{T}}$ denote the restriction of the matrix $\mathbf{A}$ to the set $\mathcal{T}$: that is, $(\mathbf{A}_{\mathcal{T}})_{ij} = 0$ if $(i,j) \notin \mathcal{T}$ and $(\mathbf{A}_{\mathcal{T}})_{ij} = A_{ij}$ if $(i,j) \in \mathcal{T}$. Note that any matrix supported on $\mathcal{T}$ is block-diagonal with K blocks, subject to some permutation of its rows and columns. Also, let $S_{\max} = \max_{(i,j) \in \mathcal{T}^c} |S_{ij}|$. Define

$$\begin{aligned} \tilde{\mathbf{P}}(\mathbf{\Theta}) \quad = \quad & \min_{\mathbf{V},\mathbf{Z}} \quad \|\mathbf{Z} - \text{diag}(\mathbf{Z})\|_1 + \hat{\lambda}_2 \|\mathbf{V} - \text{diag}(\mathbf{V})\|_1 + \hat{\lambda}_3 \|\mathbf{V} - \text{diag}(\mathbf{V})\|_{1,q} \\ & \text{subject to} \quad \mathbf{\Theta} = \mathbf{Z} + \mathbf{V} + \mathbf{V}^T, \end{aligned} \tag{15}$$

where $\hat{\lambda}_2 = \frac{\lambda_2}{\lambda_1}$ and $\hat{\lambda}_3 = \frac{\lambda_3}{\lambda_1}$. Then, optimization problem (6) is equivalent to

$$\underset{\mathbf{\Theta} \in \mathcal{S}}{\text{minimize}} \quad -\log\det(\mathbf{\Theta}) + \langle \mathbf{\Theta}, \mathbf{S} \rangle + \lambda_1 \tilde{\mathbf{P}}(\mathbf{\Theta}), \tag{16}$$

where $\mathcal{S} = \{\mathbf{\Theta} : \mathbf{\Theta} \succ 0, \mathbf{\Theta} = \mathbf{\Theta}^T\}$.

Proof of Theorem 1 (Sufficient Condition)

Proof First, we note that if $(\mathbf{\Theta}, \mathbf{V}, \mathbf{Z})$ is a feasible solution to (6), then $(\mathbf{\Theta}_{\mathcal{T}}, \mathbf{V}_{\mathcal{T}}, \mathbf{Z}_{\mathcal{T}})$ is also a feasible solution to (6). Assume that $(\mathbf{\Theta}, \mathbf{V}, \mathbf{Z})$ is not supported on $\mathcal{T}$. We want to show that the objective value of (6) evaluated at $(\mathbf{\Theta}_{\mathcal{T}}, \mathbf{V}_{\mathcal{T}}, \mathbf{Z}_{\mathcal{T}})$ is smaller than the objective value of (6) evaluated at $(\mathbf{\Theta}, \mathbf{V}, \mathbf{Z})$. By Fischer's inequality (Horn and Johnson, 1985),

$$-\log\det(\mathbf{\Theta}) \geq -\log\det(\mathbf{\Theta}_{\mathcal{T}}).$$

Therefore, it remains to show that

$$\begin{aligned} &\langle \mathbf{\Theta}, \mathbf{S} \rangle + \lambda_1 \|\mathbf{Z} - \text{diag}(\mathbf{Z})\|_1 + \lambda_2 \|\mathbf{V} - \text{diag}(\mathbf{V})\|_1 + \lambda_3 \|\mathbf{V} - \text{diag}(\mathbf{V})\|_{1,q} \quad > \\ &\langle \mathbf{\Theta}_{\mathcal{T}}, \mathbf{S} \rangle + \lambda_1 \|\mathbf{Z}_{\mathcal{T}} - \text{diag}(\mathbf{Z}_{\mathcal{T}})\|_1 + \lambda_2 \|\mathbf{V}_{\mathcal{T}} - \text{diag}(\mathbf{V}_{\mathcal{T}})\|_1 + \lambda_3 \|\mathbf{V}_{\mathcal{T}} - \text{diag}(\mathbf{V}_{\mathcal{T}})\|_{1,q}, \end{aligned}$$

or equivalently, that

$$\langle \mathbf{\Theta}_{\mathcal{T}^c}, \mathbf{S} \rangle + \lambda_1 \|\mathbf{Z}_{\mathcal{T}^c}\|_1 + \lambda_2 \|\mathbf{V}_{\mathcal{T}^c}\|_1 + \lambda_3 (\|\mathbf{V} - \text{diag}(\mathbf{V})\|_{1,q} - \|\mathbf{V}_{\mathcal{T}} - \text{diag}(\mathbf{V}_{\mathcal{T}})\|_{1,q}) > 0.$$

Since $\|\mathbf{V} - \text{diag}(\mathbf{V})\|_{1,q} \geq \|\mathbf{V}_{\mathcal{T}} - \text{diag}(\mathbf{V}_{\mathcal{T}})\|_{1,q}$, it suffices to show that

$$\langle \mathbf{\Theta}_{\mathcal{T}^c}, \mathbf{S} \rangle + \lambda_1 \|\mathbf{Z}_{\mathcal{T}^c}\|_1 + \lambda_2 \|\mathbf{V}_{\mathcal{T}^c}\|_1 > 0. \tag{17}$$

Note that $\langle \mathbf{\Theta}_{\mathcal{T}^c}, \mathbf{S} \rangle = \langle \mathbf{\Theta}_{\mathcal{T}^c}, \mathbf{S}_{\mathcal{T}^c} \rangle$. By the sufficient condition, $S_{\max} < \lambda_1$ and $2S_{\max} < \lambda_2$.

In addition, we have that

$$\begin{aligned}
|\langle \mathbf{\Theta}_{\mathcal{T}^c}, \mathbf{S} \rangle| &= |\langle \mathbf{\Theta}_{\mathcal{T}^c}, \mathbf{S}_{\mathcal{T}^c} \rangle| \\
&= |\langle \mathbf{V}_{\mathcal{T}^c} + \mathbf{V}_{\mathcal{T}^c}^T + \mathbf{Z}_{\mathcal{T}^c}, \mathbf{S}_{\mathcal{T}^c} \rangle| \\
&= |\langle 2\mathbf{V}_{\mathcal{T}^c} + \mathbf{Z}_{\mathcal{T}^c}, \mathbf{S}_{\mathcal{T}^c} \rangle| \\
&\leq (2\|\mathbf{V}_{\mathcal{T}^c}\|_1 + \|\mathbf{Z}_{\mathcal{T}^c}\|_1) S_{\max} \\
&< \ \lambda_2 \|\mathbf{V}_{\mathcal{T}^c}\|_1 + \lambda_1 \|\mathbf{Z}_{\mathcal{T}^c}\|_1,
\end{aligned}$$

where the last inequality follows from the sufficient condition. We have shown (17) as desired.

■

Proof of Theorem 2 (Necessary Condition)

We first present a simple lemma for proving Theorem 2. Throughout the proof of Theorem 2, $\|\cdot\|_\infty$ indicates the maximal absolute element of a matrix and $\|\cdot\|_{\infty,s}$ indicates the dual norm of $\|\cdot\|_{1,q}$.

Lemma 7 *The dual representation of* $\tilde{\mathbf{P}}(\mathbf{\Theta})$ *in (15) is*

$$\begin{array}{rcll}
\tilde{\mathbf{P}}^*(\mathbf{\Theta}) & = & \max\limits_{\mathbf{X},\mathbf{Y},\mathbf{\Lambda}} & \langle \mathbf{\Lambda}, \mathbf{\Theta} \rangle \\
& & \text{subject to} & \mathbf{\Lambda} + \mathbf{\Lambda}^T = \hat{\lambda}_2 \mathbf{X} + \hat{\lambda}_3 \mathbf{Y} \\
& & & \|\mathbf{X}\|_\infty \leq 1, \|\mathbf{\Lambda}\|_\infty \leq 1, \|\mathbf{Y}\|_{\infty,s} \leq 1 \\
& & & X_{ii} = 0, Y_{ii} = 0, \Lambda_{ii} = 0 \textit{ for } i = 1, \ldots, p,
\end{array} \tag{18}$$

where $\frac{1}{s} + \frac{1}{q} = 1$.

Proof We first state the dual representations for the norms in (15):

$$\begin{array}{rcll}
\|\mathbf{Z} - \mathrm{diag}(\mathbf{Z})\|_1 & = & \max\limits_{\mathbf{\Lambda}} & \langle \mathbf{\Lambda}, \mathbf{Z} \rangle \\
& & \text{subject to} & \|\mathbf{\Lambda}\|_\infty \leq 1, \Lambda_{ii} = 0 \text{ for } i = 1, \ldots, p,
\end{array}$$

$$\begin{array}{rcll}
\|\mathbf{V} - \mathrm{diag}(\mathbf{V})\|_1 & = & \max\limits_{\mathbf{X}} & \langle \mathbf{X}, \mathbf{V} \rangle \\
& & \text{subject to} & \|\mathbf{X}\|_\infty \leq 1, X_{ii} = 0 \text{ for } i = 1, \ldots, p,
\end{array}$$

$$\begin{array}{rcll}
\|\mathbf{V} - \mathrm{diag}(\mathbf{V})\|_{1,q} & = & \max\limits_{\mathbf{Y}} & \langle \mathbf{Y}, \mathbf{V} \rangle \\
& & \text{subject to} & \|\mathbf{Y}\|_{\infty,s} \leq 1, Y_{ii} = 0 \text{ for } i = 1, \ldots, p.
\end{array}$$

Then,

$$\begin{aligned}
\tilde{\mathbf{P}}(\boldsymbol{\Theta}) &= \min_{\mathbf{V},\mathbf{Z}} && \|\mathbf{Z}-\mathrm{diag}(\mathbf{Z})\|_1 + \hat{\lambda}_2\|\mathbf{V}-\mathrm{diag}(\mathbf{V})\|_1 + \hat{\lambda}_3\|\mathbf{V}-\mathrm{diag}(\mathbf{V})\|_{1,q} \\
&\quad \text{subject to} && \boldsymbol{\Theta} = \mathbf{Z}+\mathbf{V}+\mathbf{V}^T \\
&= \min_{\mathbf{V},\mathbf{Z}} && \max_{\boldsymbol{\Lambda},\mathbf{X},\mathbf{Y}} \langle\boldsymbol{\Lambda},\mathbf{Z}\rangle + \hat{\lambda}_2\langle\mathbf{X},\mathbf{V}\rangle + \hat{\lambda}_3\langle\mathbf{Y},\mathbf{V}\rangle \\
&\quad \text{subject to} && \|\boldsymbol{\Lambda}\|_\infty \le 1, \|\mathbf{X}\|_\infty \le 1, \|\mathbf{Y}\|_{\infty,s} \le 1 \\
&&& \Lambda_{ii} = 0, X_{ii} = 0, Y_{ii} = 0 \text{ for } i = 1,\ldots,p \\
&&& \boldsymbol{\Theta} = \mathbf{Z}+\mathbf{V}+\mathbf{V}^T \\
&= \max_{\boldsymbol{\Lambda},\mathbf{X},\mathbf{Y}} && \min_{\mathbf{V},\mathbf{Z}} \langle\boldsymbol{\Lambda},\mathbf{Z}\rangle + \hat{\lambda}_2\langle\mathbf{X},\mathbf{V}\rangle + \hat{\lambda}_3\langle\mathbf{Y},\mathbf{V}\rangle \\
&\quad \text{subject to} && \|\boldsymbol{\Lambda}\|_\infty \le 1, \|\mathbf{X}\|_\infty \le 1, \|\mathbf{Y}\|_{\infty,s} \le 1 \\
&&& \Lambda_{ii} = 0, X_{ii} = 0, Y_{ii} = 0 \text{ for } i = 1,\ldots,p \\
&&& \boldsymbol{\Theta} = \mathbf{Z}+\mathbf{V}+\mathbf{V}^T \\
&= \max_{\boldsymbol{\Lambda},\mathbf{X},\mathbf{Y}} && \langle\boldsymbol{\Lambda},\boldsymbol{\Theta}\rangle \\
&\quad \text{subject to} && \boldsymbol{\Lambda}+\boldsymbol{\Lambda}^T = \hat{\lambda}_2\mathbf{X} + \hat{\lambda}_3\mathbf{Y} \\
&&& \|\mathbf{X}\|_\infty \le 1, \|\boldsymbol{\Lambda}\|_\infty \le 1, \|\mathbf{Y}\|_{\infty,s} \le 1 \\
&&& X_{ii} = 0, Y_{ii} = 0, \Lambda_{ii} = 0 \text{ for } i = 1,\ldots,p.
\end{aligned}$$

The third equality holds since the constraints on $(\mathbf{V},\mathbf{Z})$ and on $(\boldsymbol{\Lambda},\mathbf{X},\mathbf{Y})$ are both compact convex sets and so by the minimax theorem, we can swap max and min. The last equality follows from the fact that

$$\begin{aligned}
&\min_{\mathbf{V},\mathbf{Z}} && \langle\boldsymbol{\Lambda},\mathbf{Z}\rangle + \hat{\lambda}_2\langle\mathbf{X},\mathbf{V}\rangle + \hat{\lambda}_3\langle\mathbf{Y},\mathbf{V}\rangle \\
&\text{subject to} && \boldsymbol{\Theta} = \mathbf{Z}+\mathbf{V}+\mathbf{V}^T \\
&= && \begin{cases} \langle\boldsymbol{\Lambda},\boldsymbol{\Theta}\rangle & \text{if } \boldsymbol{\Lambda}+\boldsymbol{\Lambda}^T = \hat{\lambda}_2\mathbf{X}+\hat{\lambda}_3\mathbf{Y} \\ -\infty & \text{otherwise.} \end{cases}
\end{aligned}$$

■

We now present the proof of Theorem 2.

Proof The optimality condition for (16) is given by

$$\mathbf{0} = -\boldsymbol{\Theta}^{-1} + \mathbf{S} + \lambda_1\boldsymbol{\Lambda}, \tag{19}$$

where $\boldsymbol{\Lambda}$ is a subgradient of $\tilde{\mathbf{P}}(\boldsymbol{\Theta})$ in (15) and the left-hand side of the above equation is a zero matrix of size $p \times p$.

Now suppose that $\boldsymbol{\Theta}^*$ that solves (19) is supported on $\mathcal{T}$, i.e., $\boldsymbol{\Theta}^*_{\mathcal{T}^c} = 0$. Then for any $(i,j) \in \mathcal{T}^c$, we have that

$$0 = S_{ij} + \lambda_1\Lambda^*_{ij}, \tag{20}$$

where $\boldsymbol{\Lambda}^*$ is a subgradient of $\tilde{\mathbf{P}}(\boldsymbol{\Theta}^*)$. Note that $\boldsymbol{\Lambda}^*$ must be an optimal solution to the optimization problem (18). Therefore, it is also a feasible solution to (18), implying that

$$\begin{aligned}
|\Lambda^*_{ij} + \Lambda^*_{ji}| &\le \hat{\lambda}_2 + \hat{\lambda}_3, \\
|\Lambda^*_{ij}| &\le 1.
\end{aligned}$$

From (20), we have that $\Lambda^*_{ij} = -\frac{S_{ij}}{\lambda_1}$ and thus,

$$\begin{aligned}\lambda_1 &\geq \lambda_1 \max_{(i,j)\in\mathcal{T}^c} |\Lambda^*_{ij}| \\ &= \lambda_1 \max_{(i,j)\in\mathcal{T}^c} \frac{|S_{ij}|}{\lambda_1} \\ &= S_{\max}.\end{aligned}$$

Also, recall that $\hat{\lambda}_2 = \frac{\lambda_2}{\lambda_1}$ and $\hat{\lambda}_3 = \frac{\lambda_3}{\lambda_1}$. We have that

$$\begin{aligned}\lambda_2 + \lambda_3 &\geq \lambda_1 \max_{(i,j)\in\mathcal{T}^c} |\Lambda^*_{ij} + \Lambda^*_{ji}| \\ &= \lambda_1 \max_{(i,j)\in\mathcal{T}^c} \frac{2|S_{ij}|}{\lambda_1} \\ &= 2S_{\max}.\end{aligned}$$

Hence, we obtain the desired result. ■

Appendix C. Some Properties of HGL

Proof of Lemma 3

Proof Let $(\mathbf{\Theta}^*, \mathbf{Z}^*, \mathbf{V}^*)$ be the solution to (6) and suppose that $\mathbf{Z}^*$ is not a diagonal matrix. Note that $\mathbf{Z}^*$ is symmetric since $\mathbf{\Theta} \in \mathcal{S} \equiv \{\mathbf{\Theta} : \mathbf{\Theta} \succ 0 \text{ and } \mathbf{\Theta} = \mathbf{\Theta}^T\}$. Let $\hat{\mathbf{Z}} = \text{diag}(\mathbf{Z}^*)$, a matrix that contains the diagonal elements of the matrix $\mathbf{Z}^*$. Also, construct $\hat{\mathbf{V}}$ as follows,

$$\hat{\mathbf{V}}_{ij} = \begin{cases} \mathbf{V}^*_{ij} + \frac{\mathbf{Z}^*_{ij}}{2} & \text{if } i \neq j \\ \mathbf{V}^*_{jj} & \text{otherwise.} \end{cases}$$

Then, we have that $\mathbf{\Theta}^* = \hat{\mathbf{Z}} + \hat{\mathbf{V}} + \hat{\mathbf{V}}^T$. Thus, $(\mathbf{\Theta}^*, \hat{\mathbf{Z}}, \hat{\mathbf{V}})$ is a feasible solution to (6). We now show that $(\mathbf{\Theta}^*, \hat{\mathbf{Z}}, \hat{\mathbf{V}})$ has a smaller objective than $(\mathbf{\Theta}^*, \mathbf{Z}^*, \mathbf{V}^*)$ in (6), giving us a contradiction. Note that

$$\begin{aligned}\lambda_1\|\hat{\mathbf{Z}} - \text{diag}(\hat{\mathbf{Z}})\|_1 + \lambda_2\|\hat{\mathbf{V}} - \text{diag}(\hat{\mathbf{V}})\|_1 &= \lambda_2\|\hat{\mathbf{V}} - \text{diag}(\hat{\mathbf{V}})\|_1 \\ &= \lambda_2 \textstyle\sum_{i\neq j} |\mathbf{V}^*_{ij} + \frac{\mathbf{Z}^*_{ij}}{2}| \\ &\leq \lambda_2\|\mathbf{V}^* - \text{diag}(\mathbf{V}^*)\|_1 + \tfrac{\lambda_2}{2}\|\mathbf{Z}^* - \text{diag}(\mathbf{Z}^*)\|_1,\end{aligned}$$

and

$$\begin{aligned}&\lambda_3 \textstyle\sum_{j=1}^p \|(\hat{\mathbf{V}} - \text{diag}(\hat{\mathbf{V}}))_j\|_q \\ \leq\ &\lambda_3 \textstyle\sum_{j=1}^p \|(\mathbf{V}^* - \text{diag}(\mathbf{V}^*))_j\|_q + \frac{\lambda_3}{2}\sum_{j=1}^p \|(\mathbf{Z}^* - \text{diag}(\mathbf{Z}^*))_j\|_q \\ \leq\ &\lambda_3 \textstyle\sum_{j=1}^p \|(\mathbf{V}^* - \text{diag}(\mathbf{V}^*))_j\|_q + \frac{\lambda_3}{2}\|\mathbf{Z}^* - \text{diag}(\mathbf{Z}^*)\|_1,\end{aligned}$$

where the last inequality follows from the fact that for any vector $\mathbf{x} \in \mathbb{R}^p$ and $q \geq 1$, $\|\mathbf{x}\|_q$ is a nonincreasing function of q (Gentle, 2007).

Summing up the above inequalities, we get that

$$\begin{array}{rl} \lambda_1\|\hat{\mathbf{Z}} - \mathrm{diag}(\hat{\mathbf{Z}})\|_1 + \lambda_2\|\hat{\mathbf{V}} - \mathrm{diag}(\hat{\mathbf{V}})\|_1 + \lambda_3 \sum_{j=1}^p \|(\hat{\mathbf{V}} - \mathrm{diag}(\hat{\mathbf{V}}))_j\|_q & \leq \\ \frac{\lambda_2+\lambda_3}{2}\|\mathbf{Z}^* - \mathrm{diag}(\mathbf{Z}^*)\|_1 + \lambda_2\|\mathbf{V}^* - \mathrm{diag}(\mathbf{V}^*)\|_1 + \lambda_3 \sum_{j=1}^p \|(\mathbf{V}^* - \mathrm{diag}(\mathbf{V}^*))_j\|_q & < \\ \lambda_1\|\mathbf{Z}^* - \mathrm{diag}(\mathbf{Z}^*)\|_1 + \lambda_2\|\mathbf{V}^* - \mathrm{diag}(\mathbf{V}^*)\|_1 + \lambda_3 \sum_{j=1}^p \|(\mathbf{V}^* - \mathrm{diag}(\mathbf{V}^*))_j\|_q, & \end{array}$$

where the last inequality uses the assumption that $\lambda_1 > \frac{\lambda_2+\lambda_3}{2}$. We arrive at a contradiction and therefore the result holds. ■

Proof of Lemma 4

Proof Let $(\mathbf{\Theta}^*, \mathbf{Z}^*, \mathbf{V}^*)$ be the solution to (6) and suppose $\mathbf{V}^*$ is not a diagonal matrix. Let $\hat{\mathbf{V}} = \mathrm{diag}(\mathbf{V}^*)$, a diagonal matrix that contains the diagonal elements of $\mathbf{V}^*$. Also construct $\hat{\mathbf{Z}}$ as follows,

$$\hat{\mathbf{Z}}_{ij} = \begin{cases} \mathbf{Z}^*_{ij} + \mathbf{V}^*_{ij} + \mathbf{V}^*_{ji} & \text{if } i \neq j \\ \mathbf{Z}^*_{ij} & \text{otherwise.} \end{cases}$$

Then, we have that $\mathbf{\Theta}^* = \hat{\mathbf{V}} + \hat{\mathbf{V}}^T + \hat{\mathbf{Z}}$. We now show that $(\mathbf{\Theta}^*, \hat{\mathbf{Z}}, \hat{\mathbf{V}})$ has a smaller objective value than $(\mathbf{\Theta}^*, \mathbf{Z}^*, \mathbf{V}^*)$ in (6), giving us a contradiction. We start by noting that

$$\begin{array}{rcl} \lambda_1\|\hat{\mathbf{Z}} - \mathrm{diag}(\hat{\mathbf{Z}})\|_1 + \lambda_2\|\hat{\mathbf{V}} - \mathrm{diag}(\hat{\mathbf{V}})\|_1 & = & \lambda_1\|\hat{\mathbf{Z}} - \mathrm{diag}(\hat{\mathbf{Z}})\|_1 \\ & \leq & \lambda_1\|\mathbf{Z}^* - \mathrm{diag}(\mathbf{Z}^*)\|_1 + 2\lambda_1\|\mathbf{V}^* - \mathrm{diag}(\mathbf{V}^*)\|_1. \end{array}$$

By Holder's Inequality, we know that $\mathbf{x}^T\mathbf{y} \leq \|\mathbf{x}\|_q\|\mathbf{y}\|_s$ where $\frac{1}{s} + \frac{1}{q} = 1$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{p-1}$. Setting $\mathbf{y} = \mathrm{sign}(\mathbf{x})$, we have that $\|\mathbf{x}\|_1 \leq (p-1)^{\frac{1}{s}}\|\mathbf{x}\|_q$. Consequently,

$$\frac{\lambda_3}{(p-1)^{\frac{1}{s}}}\|\mathbf{V}^* - \mathrm{diag}(\mathbf{V}^*)\|_1 \leq \lambda_3 \sum_{j=1}^p \|(\mathbf{V}^* - \mathrm{diag}(\mathbf{V}^*))_j\|_q.$$

Combining these results, we have that

$$\begin{aligned} &\lambda_1\|\hat{\mathbf{Z}} - \mathrm{diag}(\hat{\mathbf{Z}})\|_1 + \lambda_2\|\hat{\mathbf{V}} - \mathrm{diag}(\hat{\mathbf{V}})\|_1 + \lambda_3 \sum_{j=1}^p \|(\hat{\mathbf{V}} - \mathrm{diag}(\hat{\mathbf{V}}))_j\|_q \\ &\leq \lambda_1\|\mathbf{Z}^* - \mathrm{diag}(\mathbf{Z}^*)\|_1 + 2\lambda_1\|\mathbf{V}^* - \mathrm{diag}(\mathbf{V}^*)\|_1 \\ &< \lambda_1\|\mathbf{Z}^* - \mathrm{diag}(\mathbf{Z}^*)\|_1 + \left(\lambda_2 + \frac{\lambda_3}{(p-1)^{\frac{1}{s}}}\right)\|\mathbf{V}^* - \mathrm{diag}(\mathbf{V}^*)\|_1 \\ &\leq \lambda_1\|\mathbf{Z}^* - \mathrm{diag}(\mathbf{Z}^*)\|_1 + \lambda_2\|\mathbf{V}^* - \mathrm{diag}(\mathbf{V}^*)\|_1 + \lambda_3 \sum_{j=1}^p \|(\mathbf{V}^* - \mathrm{diag}(\mathbf{V}^*))_j\|_q, \end{aligned}$$

where we use the assumption that $\lambda_1 < \frac{\lambda_2}{2} + \frac{\lambda_3}{2(p-1)^{\frac{1}{s}}}$. This leads to a contradiction. ■

Proof of Lemma 6

In this proof, we consider the case when $\lambda_1 > \frac{\lambda_2+\lambda_3}{2}$. A similar proof technique can be used to prove the case when $\lambda_1 < \frac{\lambda_2+\lambda_3}{2}$.

Proof Let $f(\mathbf{\Theta}, \mathbf{V}, \mathbf{Z})$ denote the objective of (6) with $q = 1$, and $(\mathbf{\Theta}^*, \mathbf{V}^*, \mathbf{Z}^*)$ the optimal solution. By Lemma 3, the assumption that $\lambda_1 > \frac{\lambda_2+\lambda_3}{2}$ implies that $\mathbf{Z}^*$ is a diagonal matrix. Now let $\hat{\mathbf{V}} = \frac{1}{2}\left(\mathbf{V}^* + (\mathbf{V}^*)^T\right)$. Then

$$\begin{aligned}
& f(\mathbf{\Theta}^*, \hat{\mathbf{V}}, \mathbf{Z}^*) \\
= \; & -\log\det\mathbf{\Theta}^* + \langle\mathbf{\Theta}^*, \mathbf{S}\rangle + \lambda_1\|\mathbf{Z}^* - \mathrm{diag}(\mathbf{Z}^*)\|_1 + (\lambda_2+\lambda_3)\|\hat{\mathbf{V}} - \mathrm{diag}(\hat{\mathbf{V}})\|_1 \\
= \; & -\log\det\mathbf{\Theta}^* + \langle\mathbf{\Theta}^*, \mathbf{S}\rangle + \frac{\lambda_2+\lambda_3}{2}\|\mathbf{V}^* + \mathbf{V}^{*T} - \mathrm{diag}(\mathbf{V}^* + \mathbf{V}^{*T})\|_1 \\
\leq \; & -\log\det\mathbf{\Theta}^* + \langle\mathbf{\Theta}^*, \mathbf{S}\rangle + (\lambda_2+\lambda_3)\|\mathbf{V}^* - \mathrm{diag}(\mathbf{V}^*)\|_1 \\
= \; & f(\mathbf{\Theta}^*, \mathbf{V}^*, \mathbf{Z}^*) \\
\leq \; & f(\mathbf{\Theta}^*, \hat{\mathbf{V}}, \mathbf{Z}^*),
\end{aligned}$$

where the last inequality follows from the assumption that $(\mathbf{\Theta}^*, \mathbf{V}^*, \mathbf{Z}^*)$ solves (6). By strict convexity of f, this means that $\mathbf{V}^* = \hat{\mathbf{V}}$, i.e., $\mathbf{V}^*$ is symmetric. This implies that

$$\begin{aligned}
f(\mathbf{\Theta}^*, \mathbf{V}^*, \mathbf{Z}^*) &= -\log\det\mathbf{\Theta}^* + \langle\mathbf{\Theta}^*, \mathbf{S}\rangle + \frac{\lambda_2+\lambda_3}{2}\|\mathbf{V}^* + \mathbf{V}^{*T} - \mathrm{diag}(\mathbf{V}^* + \mathbf{V}^{*T})\|_1 \\
&= -\log\det\mathbf{\Theta}^* + \langle\mathbf{\Theta}^*, \mathbf{S}\rangle + \frac{\lambda_2+\lambda_3}{2}\|\mathbf{\Theta}^* - \mathrm{diag}(\mathbf{\Theta}^*)\|_1 \qquad (21) \\
&= g(\mathbf{\Theta}^*),
\end{aligned}$$

where $g(\mathbf{\Theta})$ is the objective of the graphical lasso optimization problem, evaluated at $\mathbf{\Theta}$, with tuning parameter $\frac{\lambda_2+\lambda_3}{2}$. Suppose that $\tilde{\mathbf{\Theta}}$ minimizes $g(\mathbf{\Theta})$, and $\mathbf{\Theta}^* \neq \tilde{\mathbf{\Theta}}$. Then, by (21) and strict convexity of g, $g(\mathbf{\Theta}^*) = f(\mathbf{\Theta}^*, \mathbf{V}^*, \mathbf{Z}^*) \leq f(\tilde{\mathbf{\Theta}}, \tilde{\mathbf{\Theta}}/2, \mathbf{0}) = g(\tilde{\mathbf{\Theta}}) < g(\mathbf{\Theta}^*)$, giving us a contradiction. Thus it must be that $\tilde{\mathbf{\Theta}} = \mathbf{\Theta}^*$.

■

Appendix D. Simulation Study for Hub Covariance Graph

In this section, we present the results for the simulation study described in Section 4.2 with $n = 100$, $p = 200$, and $|\mathcal{H}| = 4$. We calculate the proportion of correctly estimated hub nodes with $r = 40$. The results are shown in Figure 10. As we can see from Figure 10, our proposal outperforms Bien and Tibshirani (2011). In particular, we can see from Figure 10(c) that Bien and Tibshirani (2011) fails to identify hub nodes.

Appendix E. Run Time Study for the ADMM algorithm for HGL

In this section, we present a more extensive run time study for the ADMM algorithm for HGL. We ran experiments with $p = 100, 200, 300$ and with $n = p/2$ on a 2.26GHz Intel Core

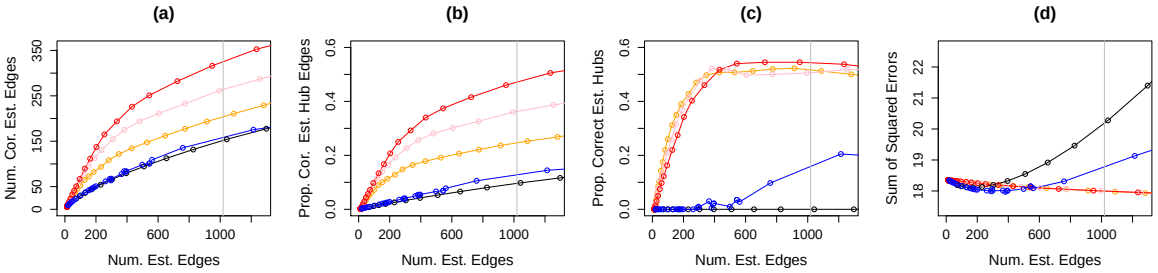


Figure 10: Covariance graph simulation with $n = 100$ and $p = 200$. Details of the axis labels are as in Figure 3. The colored lines correspond to the proposal of Xue et al. (2012) (——); HCG with $\lambda_3 = 1$ (——), $\lambda_3 = 1.5$ (——), and $\lambda_3 = 2$ (——); and the proposal of Bien and Tibshirani (2011) (——).

2 Duo machine. Results averaged over 10 replications are displayed in Figures 11(a)-(b), where the panels depict the run time and number of iterations required for the algorithm to converge, as a function of λ_1, with $\lambda_2 = 0.5$ and $\lambda_3 = 2$ fixed. The number of iterations required for the algorithm to converge is computed as the total number of iterations in Step 2 of Algorithm 1. We see from Figure 11(a) that as p increases from 100 to 300, the run times increase substantially, but never exceed several minutes. Note that these results are without using the block diagonal condition in Theorem 1.

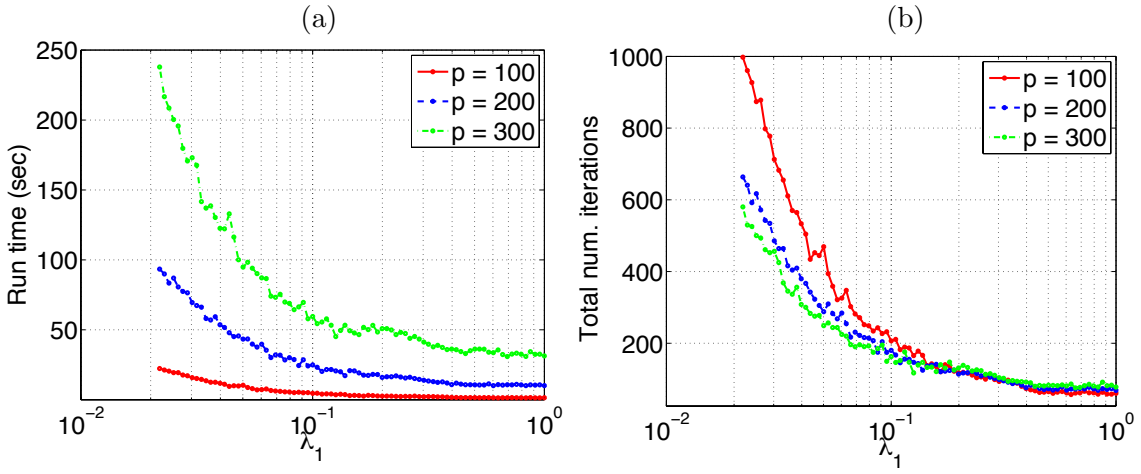


Figure 11: (a): Run time (in seconds) of the ADMM algorithm for HGL, as a function of λ_1, for fixed values of λ_2 and λ_3. (b): The total number of iterations required for the ADMM algorithm for HGL to converge, as a function of λ_1. All results are averaged over 10 simulated data sets. These results are without using the block diagonal condition in Theorem 1.

Appendix F. Update for Θ in Step 2(a)i for Binary Ising Model using Barzilai-Borwein Method

We consider updating $\mathbf{\Theta}$ in Step 2(a)i of Algorithm 1 for binary Ising model. Let

$$h(\mathbf{\Theta}) = -\sum_{j=1}^{p}\sum_{j'=1}^{p}\theta_{jj'}(\mathbf{X}^T\mathbf{X})_{jj'} + \sum_{i=1}^{p}\sum_{j=1}^{p}\log\left(1+\exp\left[\theta_{jj}+\sum_{j'\neq j}\theta_{jj'}x_{ij'}\right]\right) + \frac{\rho}{2}\|\mathbf{\Theta}-\tilde{\mathbf{\Theta}}+\mathbf{W}_1\|_F^2.$$

Then, the optimization problem for Step 2(a)i of Algorithm 1 is

$$\underset{\mathbf{\Theta}\in\mathcal{S}}{\text{minimize}} \quad h(\mathbf{\Theta}), \tag{22}$$

where $\mathcal{S} = \{\mathbf{\Theta} : \mathbf{\Theta} = \mathbf{\Theta}^T\}$. In solving (22), we will treat $\mathbf{\Theta} \in \mathcal{S}$ as an implicit constraint.

The Barzilai-Borwein method is a gradient descent method with the step-size chosen to mimic the secant condition of the BFGS method (see, e.g., Barzilai and Borwein, 1988; Nocedal and Wright, 2006). The convergence of the Barzilai-Borwein method for unconstrained minimization using a non-monotone line search was shown in Raydan (1997). Recent convergence results for a quadratic cost function can be found in Dai (2013). To implement the Barzilai-Borwein method, we need to evaluate the gradient of $h(\mathbf{\Theta})$. Let $\nabla h(\mathbf{\Theta})$ be a $p \times p$ matrix, where the (j, j') entry is the gradient of $h(\mathbf{\Theta})$ with respect to $\theta_{jj'}$, computed under the constraint $\mathbf{\Theta} \in \mathcal{S}$, that is, $\theta_{jj'} = \theta_{j'j}$. Then,

$$(\nabla h(\mathbf{\Theta}))_{jj} = -(\mathbf{X}^T\mathbf{X})_{jj} + \sum_{i=1}^{n}\left[\frac{\exp(\theta_{jj}+\sum_{j'\neq j}\theta_{jj'}x_{ij'})}{1+\exp(\theta_{jj}+\sum_{j'\neq j}\theta_{jj'}x_{ij'})}\right] + \rho(\theta_{jj}-\tilde{\theta}_{jj}+(\mathbf{W}_1)_{jj}),$$

and

$$(\nabla h(\mathbf{\Theta}))_{jj'} = -2(\mathbf{X}^T\mathbf{X})_{jj} + 2\rho(\theta_{jj'}-\tilde{\theta}_{jj'}+(\mathbf{W}_1)_{jj'}) + \sum_{i=1}^{n}\left[\frac{x_{ij'}\exp(\theta_{jj}+\sum_{j'\neq j}\theta_{jj'}x_{ij'})}{1+\exp(\theta_{jj}+\sum_{j'\neq j}\theta_{jj'}x_{ij'})} + \frac{x_{ij}\exp(\theta_{j'j'}+\sum_{j\neq j'}\theta_{jj'}x_{ij})}{1+\exp(\theta_{j'j'}+\sum_{j\neq j'}\theta_{jj'}x_{ij})}\right].$$

A simple implementation of the Barzilai-Borwein algorithm for solving (22) is detailed in Algorithm 2. We note that the Barzilai-Borwein algorithm can be improved (see, e.g., Barzilai and Borwein, 1988; Wright et al., 2009). We leave such improvement for future work.

Algorithm 2 Barzilai-Borwein Algorithm for Solving (22).

1. **Initialize** the parameters:
   (a) $\mathbf{\Theta}_1 = \mathbf{I}$ and $\mathbf{\Theta}_0 = 2\mathbf{I}$.
   (b) constant $\tau > 0$.
2. **Iterate** until the stopping criterion $\frac{\|\mathbf{\Theta}_t - \mathbf{\Theta}_{t-1}\|_F^2}{\|\mathbf{\Theta}_{t-1}\|_F^2} \leq \tau$ is met, where $\mathbf{\Theta}_t$ is the value of $\mathbf{\Theta}$ obtained at the tth iteration:
   (a) $\alpha_t = \text{trace}\left[(\mathbf{\Theta}_t - \mathbf{\Theta}_{t-1})^T(\mathbf{\Theta}_t - \mathbf{\Theta}_{t-1})\right] / \text{trace}\left[(\mathbf{\Theta}_t - \mathbf{\Theta}_{t-1})^T(\nabla h(\mathbf{\Theta}_t) - \nabla h(\mathbf{\Theta}_{t-1}))\right]$.
   (b) $\mathbf{\Theta}_{t+1} = \mathbf{\Theta}_t - \alpha_t \nabla h(\mathbf{\Theta}_t)$.

References

G.I. Allen and Z. Liu. A log-linear graphical model for inferring genetic networks from high-throughput sequencing data. *IEEE International Conference on Bioinformatics and Biomedicine*, 2012.

A.L. Barabási. Scale-free networks: A decade and beyond. *Science*, 325:412–413, 2009.

A.L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286: 509–512, 1999.

J. Barzilai and J.M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8:141–148, 1988.

P.J. Bickel and E. Levina. Regularized estimation of large covariance matrices. *Annals of Statistics*, 36(1):199–227, 2008.

J. Bien and R. Tibshirani. Sparse estimation of a covariance matrix. *Biometrika*, 98(4): 807–820, 2011.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the ADMM. *Foundations and Trends in Machine Learning*, 3(1): 1–122, 2010.

T. Cai and W. Liu. Adaptive thresholding for sparse covariance matrix estimation. *Journal of the American Statistical Association*, 106(494):672–684, 2011.

A. Cardoso-Cachopo. 2009. "http://web.ist.utl.pt/acardoso/datasets/".

S. Chaudhuri, M. Drton, and T. Richardson. Estimation of a covariance matrix with zeros. *Biometrika*, 94:199–216, 2007.

Y. Dai. A new analysis on the Barzilai-Borwein gradient method. *Journal of the Operations Research Society of China*, 1(2):187–198, 2013.

P. Danaher, P. Wang, and D.M. Witten. The joint graphical lasso for inverse covariance estimation across multiple classes. *Journal of the Royal Statistical Society: Series B*, 76 (2):373–397, 2014.

A. Defazio and T.S. Caetano. A convex formulation for learning scale-free network via submodular relaxation. *Advances in Neural Information Processing Systems*, 2012.

M. Drton and T.S. Richardson. A new algorithm for maximum likelihood estimation in Gaussian graphical models for marginal independence. *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, pages 184–191, 2003.

M. Drton and T.S. Richardson. Graphical methods for efficient likelihood inference in Gaussian covariance models. *Journal of Machine Learning Research*, 9:893–914, 2008.

J. Eckstein. Augmented Lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results. *RUTCOR Research Reports*, 32, 2012.

J. Eckstein and D.P. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(3, Ser. A):293–318, 1992.

N. El Karoui. Operator norm consistent estimation of large-dimensional sparse covariance matrices. *The Annals of Statistics*, 36(6):2717–2756, 2008.

P. Erdős and A. Rényi. On random graphs I. *Publ. Math. Debrecen*, 6:290–297, 1959.

H. Firouzi and A.O. Hero. Local hub screening in sparse correlation graphs. *Proceedings of SPIE, volume 8858, Wavelets and Sparsity XV, 88581H*, 2013.

R. Foygel and M. Drton. Extended Bayesian information criteria for Gaussian graphical models. *Advances in Neural Information Processing Systems*, 2010.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9:432–441, 2007.

J. E. Gentle. *Matrix Algebra: Theory, Computations, and Applications in Statistics*. Springer, New York, 2007.

J. Guo, E. Levina, G. Michailidis, and J. Zhu. Joint structure estimation for categorical Markov networks. Submitted, available at http://www.stat.lsa.umich.edu/~elevina, 2010.

J. Guo, E. Levina, G. Michailidis, and J. Zhu. Asymptotic properties of the joint neighborhood selection method for estimating categorical Markov networks. `arXiv: math.PR/0000000`, 2011.

D. Hao, C. Ren, and C. Li. Revisiting the variation of clustering coefficient of biological networks suggests new modular structure. *BMC System Biology*, 6(34):1–10, 2012.

A. Hero and B. Rajaratnam. Hub discovery in partial correlation graphs. *IEEE Transactions on Information Theory*, 58:6064–6078, 2012.

H. Höfling and R. Tibshirani. Estimation of sparse binary pairwise Markov networks using pseudo-likelihoods. *Journal of Machine Learning Research*, 10:883–906, 2009.

R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, New York, NY, 1985.

H. Jeong, S.P. Mason, A.L. Barabási, and Z.N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411:41–42, 2001.

S.-I. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using ℓ_1-regularization. *Advances in Neural Information Processing Systems*, 2007.

L. Li, D. Alderson, J.C. Doyle, and W. Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications. *Internet Mathematics*, 2(4):431–523, 2005.

F. Liljeros, C.R. Edling, L.A.N. Amaral, H.E. Stanley, and Aberg Y. The web of human sexual contacts. *Nature*, 411:907–908, 2001.

Q. Liu and A.T. Ihler. Learning scale free networks by reweighed ℓ_1 regularization. *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 15: 40–48, 2011.

S. Ma, L. Xue, and H. Zou. Alternating direction methods for latent variable Gaussian graphical model selection. *Neural Computation*, 2013.

Maglott et al. Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Research*, 33(D):54–58, 2004.

K.V. Mardia, J. Kent, and J.M. Bibby. *Multivariate Analysis*. Academic Press, 1979.

R. Mazumder and T. Hastie. Exact covariance thresholding into connected components for large-scale graphical lasso. *Journal of Machine Learning Research*, 13:781–794, 2012.

N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.

N. Meinshausen and P. Bühlmann. Stability selection (with discussion). *Journal of the Royal Statistical Society, Series B*, 72:417–473, 2010.

K. Mohan, P. London, M. Fazel, D.M. Witten, and S.-I. Lee. Node-based learning of Gaussian graphical models. *Journal of Machine Learning Research*, 15:445–488, 2014.

M.E.J. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of the United States of America*, 98:404–409, 2000.

J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, 2006.

J. Peng, P. Wang, N. Zhou, and J. Zhu. Partial correlation estimation by joint sparse regression model. *Journal of the American Statistical Association*, 104(486):735–746, 2009.

Rappaport et al. MalaCards: an integrated compendium for diseases and their annotation. *Database (Oxford)*, 2013.

P. Ravikumar, M.J. Wainwright, and J.D. Lafferty. High-dimensional Ising model selection using ℓ_1-regularized logistic regression. *The 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.

M. Raydan. The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. *SIAM Journal on Optimization*, 7:26–33, 1997.

A. Rothman, P.J. Bickel, E. Levina, and J. Zhu. Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2:494–515, 2008.

A. Rothman, E. Levina, and J. Zhu. Generalized thresholding of large covariance matrices. *Journal of the American Statistical Association*, 104:177–186, 2009.

N. Simon, J.H. Friedman, T. Hastie, and R. Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 2013.

R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1 (2):146–160, 1972.

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.

S.J. Wright, R.D. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

L. Xue, S. Ma, and H. Zou. Positive definite ℓ_1 penalized estimation of large covariance matrices. *Journal of the American Statistical Association*, 107(500):1480–1491, 2012.

E. Yang, G.I. Allen, Z. Liu, and P.K. Ravikumar. Graphical models via generalized linear models. *Advances in Neural Information Processing Systems*, 2012a.

S. Yang, Z. Pan, X. Shen, P. Wonka, and J. Ye. Fused multiple graphical lasso. `arXiv:1209.2139 [cs.LG]`, 2012b.

M. Yuan. Efficient computation of ℓ_1 regularized estimates in Gaussian graphical models. *Journal of Computational and Graphical Statistics*, 17(4):809–826, 2008.

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.

H. Zou, T. Hastie, and R. Tibshirani. On the "degrees of freedom" of the lasso. *The Annals of Statistics*, 35(5):2173–2192, 2007.

Journal of Machine Learning Research 15 (2014) 3333-3370 Submitted 9/13; Revised 5/14; Published 10/14

Inconsistency of Pitman–Yor Process Mixtures for the Number of Components

Jeffrey W. Miller JEFFREY_MILLER@BROWN.EDU
Matthew T. Harrison MATTHEW_HARRISON@BROWN.EDU
Division of Applied Mathematics
Brown University
Providence, RI 02912, USA

Editor: Yee Whye Teh

Abstract

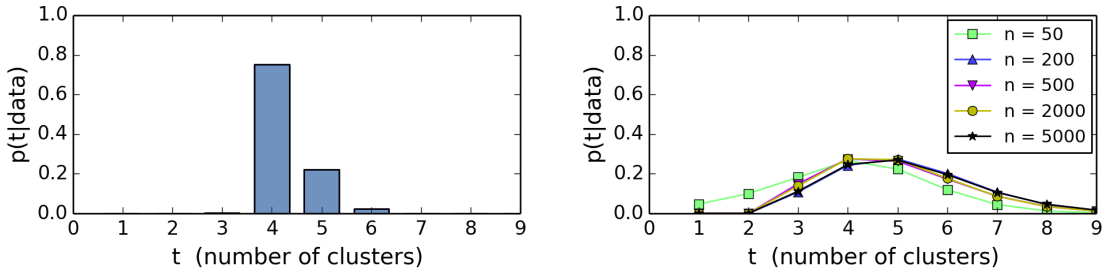
In many applications, a finite mixture is a natural model, but it can be difficult to choose an appropriate number of components. To circumvent this choice, investigators are increasingly turning to Dirichlet process mixtures (DPMs), and Pitman–Yor process mixtures (PYMs), more generally. While these models may be well-suited for Bayesian density estimation, many investigators are using them for inferences about the number of components, by considering the posterior on the number of components represented in the observed data. We show that this posterior is not consistent—that is, on data from a finite mixture, it does not concentrate at the true number of components. This result applies to a large class of nonparametric mixtures, including DPMs and PYMs, over a wide variety of families of component distributions, including essentially all discrete families, as well as continuous exponential families satisfying mild regularity conditions (such as multivariate Gaussians).

Keywords: consistency, Dirichlet process mixture, number of components, finite mixture, Bayesian nonparametrics

1. Introduction

We begin with a motivating example. In population genetics, determining the "population structure" is an important step in the analysis of sampled data. To illustrate, consider the impala, a species of antelope in southern Africa. Impalas are divided into two subspecies: the common impala occupying much of the eastern half of the region, and the black-faced impala inhabiting a small area in the west. While common impalas are abundant, the number of black-faced impalas has been decimated by drought, poaching, and declining resources due to human and livestock expansion. To assist conservation efforts, Lorenzen et al. (2006) collected samples from 216 impalas, and analyzed the genetic variation between/within the two subspecies.

A key part of their analysis consisted of inferring the population structure—that is, partitioning the data into distinct populations, and in particular, determining how many such populations there are. To infer the impala population structure, Lorenzen et al. employed a widely-used tool called STRUCTURE (Pritchard et al., 2000) which, in the simplest version, models the data as a finite mixture, with each component in the mixture corresponding



(a) Posterior for impala data

(b) Posterior for Gaussian mixture data

Figure 1: Estimated DPM posterior distributions of the number of clusters, with concentration parameter 1: (a) For the impala data of Lorenzen et al. ($n = 216$ data points); we use the same base measure as Huelsenbeck and Andolfatto, and our empirical results, shown here, agree with theirs. (b) For data from the three-component univariate Gaussian mixture $\sum_{i=1}^{3} \pi_i \mathcal{N}(x|\mu_i, \sigma_i^2)$ with $\pi = (0.45, 0.3, 0.25)$, $\mu = (4, 6, 8)$, and $\sigma = (1, 0.2, 0.6)$; we use a base measure with the same parameters as Richardson and Green (1997); each plot is the average over 10 independently-drawn data sets. For both (a) and (b), estimates were made via Gibbs sampling (MacEachern, 1994; Neal, 2000), with 10^5 burn-in sweeps and 2×10^5 sample sweeps.

to a distinct population. Structure uses an *ad hoc* method to choose the number of components, but this comes with no guarantees.

Seeking a more principled approach, Pella and Masuda (2006) proposed using a Dirichlet process mixture (DPM). Now, in a DPM, the number of components is infinite with probability 1, and thus the posterior on the number of components is always, trivially, a point mass at infinity. Consequently, Pella and Masuda instead employed the posterior on the number of clusters (that is, the number of components used in generating the data observed so far) for inferences about the number of components. (The terms "component" and "cluster" are often used interchangeably, but we make the following crucial distinction: a component is part of a mixture distribution, while a cluster is the set of indices of data points coming from a given component.) This DPM approach was implemented in a software tool called Structurama (Huelsenbeck and Andolfatto, 2007), and demonstrated on the impala data of Lorenzen et al.; see Figure 1(a).

Structurama has gained acceptance within the population genetics community, and has been used in studies of a variety of organisms, from apples and avocados, to sardines and geckos (Richards et al., 2009; Chen et al., 2009; Gonzalez and Zardoya, 2007; Leaché and Fujita, 2010). Studies such as these can carry significant weight, since they may be used by officials to make informed policy decisions regarding agriculture, conservation, and public health.

More generally, in a number of applications the same scenario has played out: a finite mixture seems to be a natural model, but requires the user to choose the number of components, while a Dirichlet process mixture offers a convenient way to avoid this choice. For nonparametric Bayesian density estimation, DPMs are indeed attractive, since the posterior on the density exhibits nice convergence properties; see Section 1.2. However, in several applications, investigators have drawn inferences from the posterior on the number

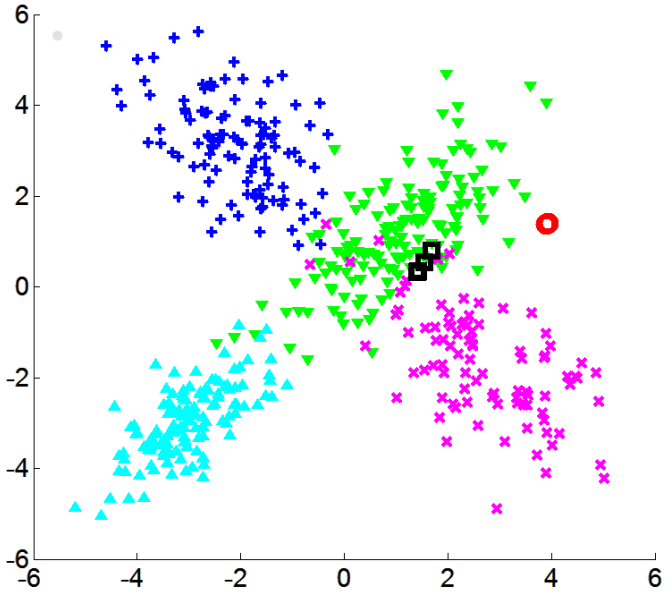


Figure 2: A typical partition sampled from the posterior of a Dirichlet process mixture of bivariate Gaussians, on simulated data from a four-component mixture. Different clusters have different marker shapes ($+, \times, \triangledown, \triangle, \circ, \square$) and different colors. Note the tiny "extra" clusters ($\circ$ and $\square$), in addition to the four dominant clusters.

of clusters—not just the density—on the assumption that this is informative about the number of components. Further examples include gene expression profiling (Medvedovic and Sivaganesan, 2002), haplotype inference (Xing et al., 2006), survival analysis (Argiento et al., 2009), econometrics (Otranto and Gallo, 2002), and evaluation of inference algorithms (Fearnhead, 2004). Of course, if the data-generating process is well-modeled by a DPM, then it is sensible to use this posterior for inference about the number of components represented so far in the data—but that does not seem to be the perspective of these investigators, since they measure performance on simulated data coming from finitely many components or populations.

Therefore, it is important to understand the properties of this procedure. Simulation results give some cause for concern; for instance, Figure 1(b) displays results on data from a mixture of univariate Gaussians with three components. The posterior on the number of clusters does not appear to be concentrating as the number of data points n increases. Empirically, it seems that this is because partitions sampled from the posterior often have tiny, transient "extra" clusters (as has been noted before, see Section 1.2); for instance, see Figure 2, showing a typical posterior sample on data from a four-component mixture of bivariate Gaussians. This raises a fundamental question that has not been addressed in the literature: With enough data, will this posterior eventually concentrate at the true number of components? In other words, is it consistent?

It is well-known that under the prior, the number of clusters goes to infinity as $n \to \infty$, with probability 1. However, this does not necessarily imply that the same is true under the

posterior—it may be that the likelihood is strong enough to overcome this prior tendency. Of course, in a typical Bayesian setting, the prior is fixed, and as n increases the likelihood overwhelms it. In the present situation, though, both the prior (on the number of clusters) and the likelihood (given the number of clusters) are changing with n, and the resulting behavior of the posterior is far from obvious.

1.1 Overview of Results

In this manuscript, we prove that under fairly general conditions, when using a Dirichlet process mixture, the posterior on the number of clusters will not concentrate at any finite value, and therefore will not be consistent for the number of components in a finite mixture. In fact, our results apply to a large class of nonparametric mixtures including DPMs, and Pitman–Yor process mixtures (PYMs) more generally, over a wide variety of families of component distributions.

Before treating our general results and their prerequisite technicalities, we would like to highlight a few interesting special cases that can be succinctly stated. The terminology and notation used below will be made precise in later sections. To reiterate, our results are considerably more general than the following corollary, which is simply presented for the reader's convenience.

Corollary 1 *Consider a Pitman–Yor process mixture with component distributions from one of the following families:*

(a) Normal(μ, Σ) *(multivariate Gaussian),*

(b) Exponential(θ)*,*

(c) Gamma(a, b)*,*

(d) Log-Normal(μ, σ^2)*, or*

(e) Weibull(a, b) *with fixed shape* $a > 0$*,*

along with a base measure that is a conjugate prior of the form in Section 5.2, or

(f) any discrete family $\{P_\theta\}$ *such that* $\bigcap_\theta \{x : P_\theta(x) > 0\} \neq \varnothing$ *(e.g., Poisson, Geometric, Negative Binomial, Binomial, Multinomial, etc.),*

along with any continuous base measure. Consider any $t \in \{1, 2, \dots\}$*, except for* $t = N$ *in the case of a Pitman–Yor process with parameters* $\sigma < 0$ *and* $\vartheta = N|\sigma|$*. If* $X_1, X_2, \dots$ *are i.i.d. from a mixture with* t *components from the family used in the model, then the posterior on the number of clusters* T_n *is not consistent for* t*, and in fact,*

$$\limsup_{n\to\infty} p(T_n = t \mid X_{1:n}) < 1$$

with probability 1*.*

This is implied by Theorems 6, 7, and 11. These more general theorems apply to a broad class of partition distributions, handling Pitman–Yor processes as a special case, and they apply to many other families of component distributions: Theorem 11 covers a large class of exponential families, and Theorem 7 covers families satisfying a certain boundedness condition on the densities (including any case in which the model and data distributions have one or more point masses in common, as well as many location–scale families with scale bounded away from zero). Dirichlet processes are subsumed as a further special case, being Pitman–Yor processes with parameters $\sigma = 0$ and $\vartheta > 0$. Also, the assumption of i.i.d. data from a finite mixture is much stronger than what is required by these results.

For PYMs with $\sigma \in [0, 1)$ (including DPMs), our results show that $p(T_n = t \mid X_{1:n})$ does not concentrate at any finite value, however, we have not been able to determine the precise limiting behavior of this posterior; the two most plausible outcomes are that it diverges, or stabilizes at some limiting distribution.

Regarding the exception of $t = N$ when $\sigma < 0$ in Corollary 1: posterior consistency at $t = N$ is possible, however, this could only occur if the chosen parameter N just happens to be equal to the actual number of components, t. On the other hand, consistency at any t can (in principle) be obtained by putting a prior on N; see Section 1.2.1 below. In a similar vein, some investigators place a prior on the concentration parameter ϑ in a DPM, or allow ϑ to depend on n; we conjecture that inconsistency can still occur in these cases, but in this paper, we examine only the case of fixed σ and ϑ.

Truncated stick-breaking processes (Ishwaran and James, 2001) are sometimes used to approximate nonparametric models. In a very limited case—see Section 2.1—our results show that on data from a one-component mixture, such a process truncated at two components will be inconsistent for the number of components. It seems likely that this will extend to truncations at any number of components.

1.2 Discussion / Related Work

We would like to emphasize that this inconsistency should not be viewed as a deficiency of DPMs and PYMs, but is simply due to a misapplication of them. As flexible priors on densities, DPMs are superb, and there are strong results showing that in many cases the posterior on the density converges in L_1 to the true density at the minimax-optimal rate, up to logarithmic factors (Ghosal et al., 1999; Ghosal and Van der Vaart, 2001; Lijoi et al., 2005; Tokdar, 2006; Ghosh and Ghosal, 2006; Tang and Ghosal, 2007; Ghosal and Van der Vaart, 2007; Walker et al., 2007; James, 2008; Wu and Ghosal, 2010; Bhattacharya and Dunson, 2010; Khazaei et al., 2012; Scricciolo, 2012; Pati et al., 2013); for a general overview, see Ghosal (2010).

We would also like to stress that we do not intend to discourage the use of DPMs and PYMs for clustering—provided that the model is indeed well-suited to the application. In some situations, however, it may be that a finite mixture model with an unknown number of components is more appropriate—in particular, for cluster sizes that are all the same order of magnitude—and in such cases, one would expect to get better clustering results by using a variable-dimension mixture model (see Section 1.2.1 below) rather than a DPM or PYM.

Existing work on posterior consistency of nonparametric mixtures has been primarily focused on the density estimation problem (as mentioned above), although recently, Nguyen (2013) has shown that the DPM posterior on the mixing distribution converges in the Wasserstein metric to the true mixing distribution. These existing results do not necessarily imply consistency for the number of components, since any mixture can be approximated arbitrarily well in these metrics by another mixture with a larger number of components (for instance, by making the weights of the extra components infinitesimally small). There seems to be no prior work on consistency of DPMs or PYMs for the number of components in a finite mixture (aside from Miller and Harrison, 2013, in which we discuss the very special case of a DPM on data from a univariate Gaussian "mixture" with one component of known variance).

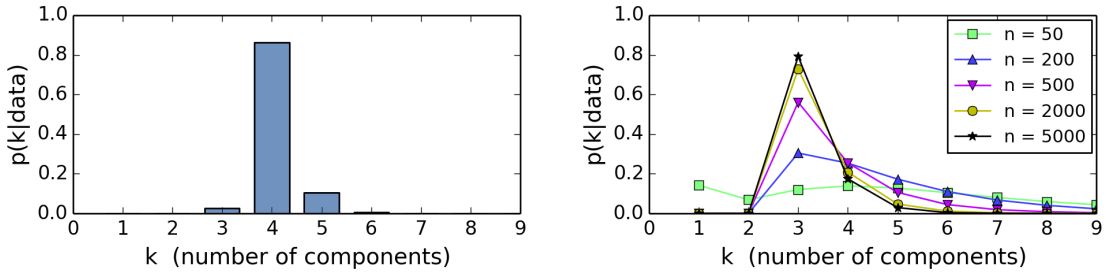
In the context of "species sampling", several authors have studied the Pitman–Yor process posterior (Pitman, 1996; Hansen and Pitman, 2000; James, 2008; Jang et al., 2010; Lijoi et al., 2007, 2008), and interestingly, James (2008) and Jang et al. (2010) have shown that on data from a continuous distribution, the posterior of a Pitman–Yor process with $\sigma > 0$ is inconsistent in the sense that it does not converge weakly to the true distribution. (In contrast, the Dirichlet process is consistent in this sense.) However, this is very different from our situation—in a species sampling model, the observed data is drawn directly from a discrete measure with a Pitman–Yor process prior, while in a PYM model, the observed data is drawn from a mixture with such a measure as the mixing distribution.

Rousseau and Mengersen (2011) proved an interesting result on "overfitted" mixtures, in which data from a finite mixture is modeled by a finite mixture with too many components. In cases where this approximates a DPM, their result implies that the posterior weight of the extra components goes to zero. In a rough sense, this is complementary to our results, which involve showing that there are always some nonempty (but perhaps small) extra clusters.

Empirically, many investigators have noticed that the DPM posterior tends to overestimate the number of components (e.g., West et al., 1994; Ji et al., 2010; Argiento et al., 2009; Lartillot and Philippe, 2004; Onogi et al., 2011, and others), and such observations are consistent with our theoretical results. This overestimation seems to occur because there are typically a few tiny "extra" clusters, and among researchers using DPMs for clustering, this is an annoyance that is sometimes dealt with by pruning such clusters—that is, by removing them before calculating statistics such as the number of clusters (e.g., West et al., 1994; Fox et al., 2007). It may be possible to obtain consistent estimators in this way, but this remains an open question; Rousseau and Mengersen's (2011) results may be applicable here. Other possibilities are using a maximum *a posteriori* (MAP) partition or posterior "mean" partition (Dahl, 2006; Huelsenbeck and Andolfatto, 2007; Onogi et al., 2011) to estimate the number of components; again, the consistency of such approaches remains an open question to our knowledge.

1.2.1 Estimating the Number of Components

A variety of methods for estimating the number of components in a finite mixture have been developed, and many of them come with guarantees of consistency (Henna, 1985; Keribin,



(a) Posterior for impala data

(b) Posterior for Gaussian mixture data

Figure 3: Estimated posterior distributions of the number of components for variable-dimension mixture models applied to the same data sets as in Figure 1. The same priors on component parameters (base measures) were used as in the DPM models.

2000; Nobile, 1994; Leroux, 1992; Ishwaran et al., 2001; James et al., 2001; Henna, 2005; Woo and Sriram, 2006, 2007).

From the Bayesian perspective, perhaps the most natural approach is simply to take a finite mixture model and put a prior on the number of components. For instance, draw the number of components k from a prior which is positive on all positive integers (so there is no *a priori* upper bound), draw mixture weights $(\pi_1, \ldots, \pi_k)$ from, say, a k-dimensional Dirichlet distribution, draw component parameters $\theta_1, \ldots, \theta_k$, and draw the data $X_1, \ldots, X_n$ from the resulting mixture. (Interestingly, it turns out that putting a prior on N in a PYM with $\sigma < 0$ and $\vartheta = N|\sigma|$ is a special case of this; see Gnedin and Pitman, 2006.) Such variable-dimension mixture models have been widely used (Nobile, 1994; Phillips and Smith, 1996; Richardson and Green, 1997; Stephens, 2000; Green and Richardson, 2001; Nobile and Fearnside, 2007), and for density estimation, they have been shown to have posterior rates of concentration similar to Dirichlet process mixtures (Kruijer, 2008; Kruijer et al., 2010). Under the (strong) assumption that the family of component distributions is correctly specified, it has been proven that such models exhibit posterior consistency for the number of components (as well as for the mixing measure and the density) under very general conditions (Nobile, 1994).

Figure 3 shows the posterior on the number of components k for variable-dimension mixture models applied to the same impala data and Gaussian mixture data as in Figure 1. In Figure 3(b), the posterior on k seems to be concentrating at the true number of components (as expected, due to Nobile, 1994), while in Figure 1(b) the DPM posterior does not appear to be concentrating (as expected, due to our results). There is enough information in the data to make the posterior concentrate at the true value; the problem with the DPM posterior is not that estimating the number of components is inherently difficult, but that the DPM posterior is simply the wrong tool for this job.

However, it should be emphasized that this guarantee of posterior consistency for the number of components is contingent upon correct specification of the family of component distributions. In most applications, it seems unreasonable to expect that the data would come from a mixture over a known parametric family, and unfortunately, the posterior on the number of components can be highly sensitive to this type of misspecification—for instance,

since any sufficiently regular density can be approximated arbitrarily well by a mixture of Gaussians, if the data distribution is close to but not exactly a finite mixture of Gaussians, a Gaussian mixture model will introduce more and more components as the amount of data increases. It seems that in order to obtain reliable assessments of heterogeneity using mixture models, one needs to carefully consider the effects of potential misspecification. Steps toward addressing this robustness issue have been taken by Woo and Sriram (2006, 2007).

1.3 Idea of the Proof

Roughly speaking, the reason why the posterior on the number of clusters does not concentrate for PYMs with $\sigma \in [0, 1)$ (the $\sigma < 0$ case is somewhat different) is that under the prior, the partition distribution strongly prefers that some of the clusters be very small, and the likelihood is not significantly decreased by splitting off such small clusters. Handling the likelihood—in a general setting—is the challenging part of the proof.

The proof involves showing that $p(T_n = t + 1 \mid X_{1:n})$ is at least the same order of magnitude (asymptotically with respect to n) as $p(T_n = t \mid X_{1:n})$. To get the basic idea of why this occurs, write

$$p(T_n = t \mid X_{1:n}) = \frac{p(X_{1:n}, T_n = t)}{p(X_{1:n})} = \frac{1}{p(X_{1:n})} \sum_{A \in \mathcal{A}_t(n)} p(X_{1:n}|A)p(A), \tag{1}$$

where the sum is over all partitions A of $\{1, \ldots, n\}$ into t parts.

Now, given some t-part partition A, suppose B is a $(t+1)$-part partition obtained from A by splitting off a single element j to be in its own cluster. For Pitman–Yor processes, $p(B)$ is at least the same order of magnitude as $p(A)/n$. In Section 3, this property is encapsulated in Condition 3, which is simple to check for any closed-form partition distribution.

Similarly, it turns out that typically, for a non-negligible fraction of the elements j, the likelihood $p(X_{1:n}|B)$ is at least the same order of magnitude as $p(X_{1:n}|A)$; in Section 3, this is made precise in Condition 4. This is trivial in discrete cases (see Section 4), and often is easy to show in any particular continuous case, but establishing this condition in a general setting requires some work, and it is this that occupies the bulk of the proof (Section 8 and the appendices).

When both of these conditions are satisfied, we show that in the expression for $p(T_n = t \mid X_{1:n})$ in Equation 1, for each term $p(X_{1:n}|A)p(A)$ there are on the order of n terms $p(X_{1:n}|B)p(B)$ in the corresponding expression for $p(T_n = t+1 \mid X_{1:n})$ that collectively are at least the same order of magnitude as $p(X_{1:n}|A)p(A)$.

1.4 Organization of the Paper

In Section 2, we define the family of partition-based mixture models under consideration, which includes Pitman–Yor and Dirichlet process mixtures as special cases. In Section 3, we state a general inconsistency theorem for partition-based mixtures satisfying certain conditions. In Section 4, we apply the theorem to cases satisfying a certain boundedness condition on the densities, including discrete families as a special case. In Section 5, we introduce notation for exponential families and conjugate priors, and in Section 6, we apply

the theorem to cases in which the mixture is over an exponential family satisfying some regularity conditions. The rest of the paper contains proofs of the results described in the previous sections: Section 7 contains the proof of the general theorem and its application to discrete or bounded cases, Section 8 contains the proof of the application to exponential families, and the appendices contain a number of supporting results for the exponential family case.

2. Model Distribution

Our analysis involves two probability distributions: one which is defined by the model, and another which gives rise to the data. In this section, we describe the model distribution.

Building upon the Dirichlet process (Ferguson, 1973; Blackwell and MacQueen, 1973; Antoniak, 1974), Dirichlet process mixtures were first studied by Antoniak (1974), Berry and Christensen (1979), Ferguson (1983), and Lo (1984), and were later made practical through the efforts of a number of authors (Escobar, 1988; MacEachern, 1994; Escobar and West, 1995; West, 1992; West et al., 1994; Neal, 1992; Liu, 1994; Bush and MacEachern, 1996; MacEachern and Müller, 1998; MacEachern, 1998; Escobar and West, 1998; MacEachern, 1999; Neal, 2000). Pitman–Yor process mixtures (Ishwaran and James, 2001, 2003) are a generalization of DPMs based on the Pitman–Yor process (Perman et al., 1992; Pitman and Yor, 1997), also known as the two-parameter Poisson–Dirichlet process. We consider a general class of partition-based mixture models that includes DPMs and PYMs.

2.1 Partition Distribution

We will use $p(\cdot)$ to denote probabilities and probability densities under the model. Our model specification begins with a distribution on partitions, or more precisely, on *ordered* partitions. Given $n \in \{1, 2, \dots\}$ and $t \in \{1, \dots, n\}$, let $\mathcal{A}_t(n)$ denote the set of all ordered partitions $A = (A_1, \dots, A_t)$ of $\{1, \dots, n\}$ into t nonempty sets (or "parts"). In other words,

$$\mathcal{A}_t(n) = \Big\{(A_1, \dots, A_t) : A_1, \dots, A_t \text{ are disjoint, } \bigcup_{i=1}^{t} A_i = \{1, \dots, n\},\ |A_i| \geq 1\ \forall i\Big\}.$$

For each $n \in \{1, 2, \dots\}$, consider a probability mass function (p.m.f.) $p(A)$ on $\bigcup_{t=1}^{n} \mathcal{A}_t(n)$. This induces a distribution on t in the natural way, via $p(t \mid A) = I(A \in \mathcal{A}_t(n))$. (Throughout, we use I to denote the indicator function: $I(E)$ is 1 if E is true, and 0 otherwise.) It follows that $p(A) = p(A, t)$ when $A \in \mathcal{A}_t(n)$.

Although it is more common to use a distribution on *unordered* partitions $\{A_1, \dots, A_t\}$, for our purposes it is more convenient to work with the corresponding distribution on ordered partitions $(A_1, \dots, A_t)$ obtained by uniformly permuting the parts. This does not affect the distribution of t. Thus, often, $p(A)$ is invariant under permutations of the parts, but we do not require this. (Also, we do not assume that, as n varies, the sequence of partition distributions necessarily satisfies the marginalization property referred to as "consistency in distribution"; Pitman, 2006.)

For example, the partition distribution for the Dirichlet process is

$$p(A) = \frac{\vartheta^t}{\vartheta_{n\uparrow 1}\; t!} \prod_{i=1}^{t} (|A_i| - 1)! \tag{2}$$

for $A \in \mathcal{A}_t(n)$, where $\vartheta > 0$ and $x_{n\uparrow\delta} = x(x+\delta)(x+2\delta)\cdots(x+(n-1)\delta)$, with $x_{0\uparrow\delta} = 1$ by convention. The $t!$ in the denominator appears since we are working with ordered partitions. More generally, the partition distribution for the Pitman–Yor process is

$$p(A) = \frac{(\vartheta+\sigma)_{t-1\uparrow\sigma}}{(\vartheta+1)_{n-1\uparrow 1}\; t!} \prod_{i=1}^{t} (1-\sigma)_{|A_i|-1\uparrow 1} \tag{3}$$

for $A \in \mathcal{A}_t(n)$, where either $\sigma \in [0, 1)$ and $\vartheta \in (-\sigma, \infty)$, or $\sigma \in (-\infty, 0)$ and $\vartheta = N|\sigma|$ for some $N \in \{1, 2, \dots\}$. When $\sigma = 0$, this reduces to the partition distribution of the Dirichlet process. When $\sigma < 0$ and $\vartheta = N|\sigma|$, it is the partition distribution obtained by drawing $q = (q_1, \dots, q_N)$ from a symmetric N-dimensional Dirichlet with parameters $|\sigma|, \dots, |\sigma|$, sampling assignments $Z_1, \dots, Z_n$ i.i.d. from q, and removing any empty parts (Gnedin and Pitman, 2006). Thus, in this latter case, t is always in $\{1, \dots, N\}$.

Stick-breaking processes truncated at N components are sometimes used to approximate nonparametric models (Ishwaran and James, 2001). This approach gives rise to a partition distribution as follows: let $V_i \sim \text{Beta}(a_i, b_i)$ independently for $i = 1, \dots, N-1$, and $V_N = 1$, set $q_i = V_i \prod_{j<i}(1 - V_j)$ for $i = 1, \dots, N$, sample assignments $Z_1, \dots, Z_n$ i.i.d. from q, and remove any empty parts. In general, it seems that this partition distribution takes a slightly complicated form, however, in the very special case when $N = 2$ and $a_1 = b_1$, it is simply a Pitman–Yor process with $\sigma = -a_1 = -b_1$ and $\vartheta = 2|\sigma|$.

2.2 Partition-based Mixture Model

Consider the hierarchical model

$$\begin{aligned} p(A, t) &= p(A) \\ p(\theta_{1:t} \mid A, t) &= \prod_{i=1}^{t} \pi(\theta_i) \\ p(x_{1:n} \mid \theta_{1:t}, A, t) &= \prod_{i=1}^{t} \prod_{j \in A_i} p_{\theta_i}(x_j) \end{aligned} \tag{4}$$

where π is a prior density on component parameters $\theta \in \Theta \subset \mathbb{R}^k$ for some k, and $\{p_\theta : \theta \in \Theta\}$ is a parameterized family of densities on $x \in \mathcal{X} \subset \mathbb{R}^d$ for some d. Here, $x_{1:n} = (x_1, \dots, x_n)$ with $x_i \in \mathcal{X}$, $\theta_{1:t} = (\theta_1, \dots, \theta_t)$ with $\theta_i \in \Theta$, and $A \in \mathcal{A}_t(n)$. Assume that π is a density with respect to Lebesgue measure, and that $\{p_\theta : \theta \in \Theta\}$ are densities with respect to some sigma-finite Borel measure λ on $\mathcal{X}$, such that $(\theta, x) \mapsto p_\theta(x)$ is measurable. (The distribution of x under $p_\theta(x)$ may be discrete, continuous, or neither, depending on λ.)

For $x_1, \dots, x_n \in \mathcal{X}$ and $J \subset \{1, \dots, n\}$, define the *single-cluster marginal*,

$$m(x_J) = \int_\Theta \Big(\prod_{j \in J} p_\theta(x_j) \Big) \, \pi(\theta) \, d\theta, \tag{5}$$

where $x_J = (x_j : j \in J)$, and assume $m(x_J) < \infty$. By convention, $m(x_J) = 1$ when $J = \varnothing$. Note that $m(x_J)$ is a density with respect to the product measure λ^ℓ on $\mathcal{X}^\ell$, where $\ell = |J|$, and that $m(x_J)$ can (and often will) be positive outside the support of λ^ℓ.

Definition 2 *We refer to such a hierarchical model as a* partition-based mixture model.

In particular, it is a *Dirichlet process mixture model* when $p(A)$ is as in Equation 2, or more generally, a *Pitman–Yor process mixture model* when $p(A)$ is as in Equation 3.

The prior on the number of clusters under such a model is $p(T_n = t) = \sum_{A \in \mathcal{A}_t(n)} p(A)$. We use T_n, rather than T, to denote the random variable representing the number of clusters, as a reminder that its distribution depends on n.

Since we are concerned with the posterior $p(T_n = t \mid x_{1:n})$ on the number of clusters, we will be especially interested in the marginal density of $(x_{1:n}, t)$, given by $p(x_{1:n}, T_n = t) = \sum_{A \in \mathcal{A}_t(n)} p(x_{1:n}, A)$. Observe that

$$p(x_{1:n}, A) = p(A) \prod_{i=1}^{t} \int \Big(\prod_{j \in A_i} p_{\theta_i}(x_j) \Big) \pi(\theta_i)\, d\theta_i = p(A) \prod_{i=1}^{t} m(x_{A_i}). \tag{6}$$

For definiteness, we employ the usual version of the posterior of T_n,

$$p(T_n = t \mid x_{1:n}) = \frac{p(x_{1:n}, T_n = t)}{p(x_{1:n})} = \frac{p(x_{1:n}, T_n = t)}{\sum_{t'=1}^{\infty} p(x_{1:n}, T_n = t')}$$

whenever the denominator is nonzero, and $p(T_n = t \mid x_{1:n}) = 0$ otherwise (for notational convenience).

3. General Theorem

The essential ingredients in the main theorem are Conditions 3 and 4 below. For $A \in \mathcal{A}_t(n)$, define $R_A = \bigcup_{i:|A_i| \geq 2} A_i$, and for $j \in R_A$, define $B(A, j)$ to be the element B of $\mathcal{A}_{t+1}(n)$ such that $B_i = A_i \setminus j$ for $i = 1, \ldots, t$, and $B_{t+1} = \{j\}$ (that is, remove j from whatever part it belongs to, and make $\{j\}$ the $(t+1)^{\text{th}}$ part). Let $\mathcal{Z}_A = \{B(A, j) : j \in R_A\}$. For $n > t \geq 1$, define

$$c_n(t) = \frac{1}{n} \max_{A \in \mathcal{A}_t(n)} \max_{B \in \mathcal{Z}_A} \frac{p(A)}{p(B)}, \tag{7}$$

with the convention that $0/0 = 0$ and $y/0 = \infty$ for $y > 0$.

Condition 3 *Assume* $\limsup_{n \to \infty} c_n(t) < \infty$*, given some particular* $t \in \{1, 2, \ldots\}$.

For Pitman–Yor processes, Condition 3 holds for all relevant values of t; see Proposition 5 below. Now, given $n \geq t \geq 1$, $x_1, \ldots, x_n \in \mathcal{X}$, and $c \in [0, \infty)$, define

$$\varphi_t(x_{1:n}, c) = \min_{A \in \mathcal{A}_t(n)} \frac{1}{n} |S_A(x_{1:n}, c)|$$

where $S_A(x_{1:n}, c)$ is the set of indices $j \in \{1, \ldots, n\}$ such that the part A_ℓ containing j satisfies $m(x_{A_\ell}) \leq c\, m(x_{A_\ell \setminus j}) m(x_j)$.

Condition 4 *Given a sequence of random variables $X_1, X_2, \ldots \in \mathcal{X}$, and $t \in \{1, 2, \ldots\}$, assume*

$$\sup_{c \in [0,\infty)} \liminf_{n \to \infty} \varphi_t(X_{1:n}, c) > 0 \text{ with probability } 1.$$

Note that Condition 3 involves only the partition distributions, while Condition 4 involves only the data distribution and the single-cluster marginals.

Proposition 5 *Consider a Pitman–Yor process. If $\sigma \in [0, 1)$ and $\vartheta \in (-\sigma, \infty)$ then Condition 3 holds for any $t \in \{1, 2, \ldots\}$. If $\sigma \in (-\infty, 0)$ and $\vartheta = N|\sigma|$, then it holds for any $t \in \{1, 2, \ldots\}$ except N.*

Proof See Section 7. ■

Theorem 6 *Let $X_1, X_2, \ldots \in \mathcal{X}$ be a sequence of random variables (not necessarily i.i.d.). Consider a partition-based mixture model. For any $t \in \{1, 2, \ldots\}$, if Conditions 3 and 4 hold, then*

$$\limsup_{n \to \infty} p(T_n = t \mid X_{1:n}) < 1 \text{ with probability } 1.$$

If, further, the sequence $X_1, X_2, \ldots$ is i.i.d. from a mixture with t components, then with probability 1, the posterior of T_n (under the model) is not consistent for t.

Proof See Section 7. ■

4. Application to Discrete or Bounded Cases

By Theorem 6, the following result implies inconsistency in a large class of PYM models, including essentially all discrete cases (or more generally anything with at least one point mass) and a number of continuous cases as well.

Theorem 7 *Let $X_1, X_2, \ldots \in \mathcal{X}$ be a sequence of random variables (not necessarily i.i.d.). If there exists $U \subset \mathcal{X}$ such that*

(1) $\liminf_{n \to \infty} \frac{1}{n} \sum_{j=1}^{n} I(X_j \in U) > 0$ *with probability 1, and*

(2) $\sup \left\{ \frac{p_\theta(x)}{m(x)} : x \in U, \theta \in \Theta \right\} < \infty$ *(where $0/0 = 0$, $y/0 = \infty$ for $y > 0$),*

then Condition 4 holds for all $t \in \{1, 2, \ldots\}$.

Proof See Section 7. ■

The preceding theorem covers a fairly wide range of cases; here are some examples.

(i) **Finite sample space.** Suppose $\mathcal{X}$ is a finite set, λ is counting measure, and $m(x) > 0$ for all $x \in \mathcal{X}$. Then choosing $U = \mathcal{X}$, Conditions 7(1) and 7(2) of Theorem 7 are trivially satisfied, regardless of the distribution of $X_1, X_2, \ldots$. (Note that when λ is counting measure, $p_\theta(x)$ and $m(x)$ are p.m.f.s on $\mathcal{X}$.) It is often easy to check that $m(x) > 0$ by using the fact that this is true whenever $\{\theta \in \Theta : p_\theta(x) > 0\}$ has nonzero probability under π. This case covers, for instance, Multinomials (including Binomials), and the population genetics model from Section 1.

We should mention a subtle point here: when $\mathcal{X}$ is finite, mixture identifiability might only hold up to a certain maximum number of components (e.g., Teicher, 1963, Proposition 4, showed this for Binomials), making consistency impossible in general—however, consistency might still be possible within that identifiable range. Regardless, our result shows that PYMs are not consistent anyway.

Now, suppose P is a probability measure on $\mathcal{X}$, and $X_1, X_2, \ldots \overset{\text{iid}}{\sim} P$. Let us abuse notation and write $P(x) = P(\{x\})$ and $\lambda(x) = \lambda(\{x\})$ for $x \in \mathcal{X}$.

(ii) **One or more point masses in common.** If there exists $x_0 \in \mathcal{X}$ such that $P(x_0) > 0$, $\lambda(x_0) > 0$, and $m(x_0) > 0$, then it is easy to verify that Conditions 7(1) and 7(2) are satisfied with $U = \{x_0\}$. (Note that $\lambda(x_0) > 0$ implies $p_\theta(x_0) \leq 1/\lambda(x_0)$ for any $\theta \in \Theta$.)

(iii) **Discrete families.** Case (ii) essentially covers all discrete families—e.g., Poisson, Geometric, Negative Binomial, or any power-series distribution (see Sapatinas, 1995, for mixture identifiability of these)—provided that the data is i.i.d.. For, suppose $\mathcal{X}$ is a countable set and λ is counting measure. By case (ii), the theorem applies if there is any $x_0 \in \mathcal{X}$ such that $m(x_0) > 0$ and $P(x_0) > 0$. If this is not so, the model is extremely misspecified, since then the model distribution and the data distribution are mutually singular.

(iv) **Continuous densities bounded on some non-null compact set.** Suppose there exists $c \in (0, \infty)$ and $U \subset \mathcal{X}$ compact such that

(a) $P(U) > 0$,

(b) $x \mapsto p_\theta(x)$ is continuous on U for all $\theta \in \Theta$, and

(c) $p_\theta(x) \in (0, c]$ for all $x \in U$, $\theta \in \Theta$.

Then Condition 7(1) is satisfied due to item (a), and Condition 7(2) follows easily from (b) and (c) since $m(x)$ is continuous (by the dominated convergence theorem) and positive on the compact set U, so $\inf_{x \in U} m(x) > 0$. This case covers, for example, the following families (with any P):

(a) Exponential(θ), $\mathcal{X} = (0, \infty)$,

(b) Gamma(a, b), $\mathcal{X} = (0, \infty)$, with variance a/b^2 bounded away from zero,

(c) Normal(μ, Σ), $\mathcal{X} = \mathbb{R}^d$, (multivariate Gaussian) with $\det(\Sigma)$ bounded away from zero, and

(d) many location–scale families with scale bounded away from zero (for instance, Laplace(μ, σ) or Cauchy(μ, σ), with $\sigma \geq \varepsilon > 0$).

The examples listed in item (iv) are indicative of a deficiency in Theorem 7: Condition 7(2) is not satisfied in some important cases, such as multivariate Gaussians with unrestricted covariance. Nonetheless, it turns out that Condition 4 still holds for many exponential families; we turn to this next.

5. Exponential Families and Conjugate Priors

In this section, we state the usual definitions for exponential families and list the regularity conditions to be assumed.

5.1 Exponential Families

Consider an exponential family of the following form. Fix a sigma-finite Borel measure λ on $\mathcal{X} \subset \mathbb{R}^d$ such that $\lambda(\mathcal{X}) \neq 0$, let $s : \mathcal{X} \to \mathbb{R}^k$ be Borel measurable, and for $\theta \in \Theta \subset \mathbb{R}^k$, define a density p_θ with respect to λ by setting

$$p_\theta(x) = \exp(\theta^{\mathsf{T}} s(x) - \kappa(\theta))$$

where

$$\kappa(\theta) = \log \int_{\mathcal{X}} \exp(\theta^{\mathsf{T}} s(x))\, d\lambda(x).$$

Let P_θ be the probability measure on $\mathcal{X}$ corresponding to p_θ, that is, $P_\theta(E) = \int_E p_\theta(x)\, d\lambda(x)$ for $E \subset \mathcal{X}$ measurable. Any exponential family on $\mathbb{R}^d$ can be written in the form above by reparameterizing if necessary, and choosing λ appropriately. We will assume the following (very mild) regularity conditions.

Condition 8 *Assume the family* $\{P_\theta : \theta \in \Theta\}$ *is:*

(1) *full, that is,* $\Theta = \{\theta \in \mathbb{R}^k : \kappa(\theta) < \infty\}$,

(2) *nonempty, that is,* $\Theta \neq \varnothing$,

(3) *regular, that is,* Θ *is an open subset of* $\mathbb{R}^k$, *and*

(4) *identifiable, that is, if* $\theta \neq \theta'$ *then* $P_\theta \neq P_{\theta'}$.

Most commonly-used exponential families satisfy Condition 8, including multivariate Gaussian, Exponential, Gamma, Poisson, Geometric, and others. (A notable exception is the Inverse Gaussian, for which Θ is not open.) Let $\mathcal{M}$ denote the *moment space*, that is,

$$\mathcal{M} = \{\mathbb{E}_\theta s(X) : \theta \in \Theta\}$$

where $\mathbb{E}_\theta$ denotes expectation under P_θ. Finiteness of these expectations is guaranteed, thus $\mathcal{M} \subset \mathbb{R}^k$; see Appendix A for this and other well-known properties that we will use.

5.2 Conjugate Priors

Given an exponential family $\{P_\theta\}$ as above, let

$$\Xi = \Big\{(\xi, \nu) : \xi \in \mathbb{R}^k,\, \nu > 0 \text{ s.t. } \xi/\nu \in \mathcal{M}\Big\},$$

and consider the family $\{\pi_{\xi,\nu} : (\xi, \nu) \in \Xi\}$ where

$$\pi_{\xi,\nu}(\theta) = \exp\big(\xi^{\mathsf{T}}\theta - \nu\kappa(\theta) - \psi(\xi, \nu)\big)\, I(\theta \in \Theta)$$

is a density with respect to Lebesgue measure on $\mathbb{R}^k$. Here,

$$\psi(\xi, \nu) = \log \int_\Theta \exp\big(\xi^{\mathsf{T}}\theta - \nu\kappa(\theta)\big)\, d\theta.$$

In Appendix A, we note a few basic properties of this family—in particular, it is a conjugate prior for $\{P_\theta\}$.

Definition 9 *We will say that an exponential family with conjugate prior is* well-behaved *if it takes the form above, satisfies Condition 8, and has* $(\xi, \nu) \in \Xi$.

6. Application to Exponential Families

In this section, we apply Theorem 6 to prove that in many cases, a PYM model using a well-behaved exponential family with conjugate prior will exhibit inconsistency for the number of components.

Condition 10 *Consider an exponential family with sufficient statistics function* $s : \mathcal{X} \to \mathbb{R}^k$ *and moment space* $\mathcal{M}$. *Given a probability measure* P *on* $\mathcal{X}$, *let* $X \sim P$ *and assume:*

(1) $\mathbb{E}|s(X)| < \infty$,

(2) $\mathbb{P}(s(X) \in \overline{\mathcal{M}}) = 1$, *and*

(3) $\mathbb{P}(s(X) \in L) = 0$ *for any hyperplane* L *that does not intersect* $\mathcal{M}$.

Throughout, we use $|\cdot|$ to denote the Euclidean norm. Here, a *hyperplane* refers to a set $L = \{x \in \mathbb{R}^k : x^{\mathsf{T}}y = b\}$ for some $y \in \mathbb{R}^k \setminus \{0\}$, $b \in \mathbb{R}$. In Theorem 11 below, it is assumed that the data comes from a distribution P satisfying Condition 10. In Proposition 12, we give some simple conditions under which, if P is a finite mixture from the exponential family under consideration, then Condition 10 holds.

Theorem 11 *Consider a well-behaved exponential family with conjugate prior (as in Definition 9), along with the resulting collection of single-cluster marginals* $m(\cdot)$. *Let* P *be a probability measure on* $\mathcal{X}$ *satisfying Condition 10 (for the* s *and* $\mathcal{M}$ *from the exponential family under consideration), and let* $X_1, X_2, \ldots \overset{\text{iid}}{\sim} P$. *Then Condition 4 holds for any* $t \in \{1, 2, \ldots\}$.

Proof See Section 7. ■

This theorem implies inconsistency in several important cases. In particular, it can be verified that each of the following is well-behaved (when put in canonical form and given the conjugate prior in Section 5.2) and, using Proposition 12 below, that if P is a finite mixture from the same family then P satisfies Condition 10:

(a) Normal(μ, Σ) (multivariate Gaussian),

(b) Exponential(θ),

(c) Gamma(a, b),

(d) Log-Normal(μ, σ^2), and

(e) Weibull(a, b) with fixed shape $a > 0$.

Combined with the cases covered by Theorem 7, these results are fairly comprehensive.

Proposition 12 *Consider an exponential family* $\{P_\theta : \theta \in \Theta\}$ *satisfying Condition 8. If* $X \sim P = \sum_{i=1}^t \pi_i P_{\theta(i)}$ *for some* $\theta(1), \ldots, \theta(t) \in \Theta$ *and some* $\pi_1, \ldots, \pi_t \geq 0$ *such that* $\sum_{i=1}^t \pi_i = 1$*, then*

(1) $\mathbb{E}|s(X)| < \infty$*, and*

(2) $\mathbb{P}(s(X) \in \overline{\mathcal{M}}) = 1$.

If, further, the underlying measure λ *is absolutely continuous with respect to Lebesgue measure on* $\mathcal{X}$*,* $\mathcal{X} \subset \mathbb{R}^d$ *is open and connected, and the sufficient statistics function* $s : \mathcal{X} \to \mathbb{R}^k$ *is real analytic (that is, each coordinate function* $s_1, \ldots, s_k$ *is real analytic), then*

(3) $\mathbb{P}(s(X) \in L) = 0$ *for* any *hyperplane* $L \subset \mathbb{R}^k$.

Proof See Appendix A. ■

Sometimes, Condition 10(3) will be satisfied even when Proposition 12 is not applicable. In any particular case, it may be a simple matter to check this condition by using the characterization of $\mathcal{M}$ as the interior of the closed convex hull of support(λs^{-1}) (see Proposition 19(8) in the Appendix).

7. Proof of the General Theorem

The remainder of the paper consists of proofs of the results described in the preceding sections. In this section, we prove Theorem 6, as well as Proposition 5 and the application to discrete or bounded cases in Theorem 7; these proofs do not depend on anything in Section 8 or the appendices.

Proof of Proposition 5 There are two cases: (I) $\sigma \in [0, 1)$ and $\vartheta > -\sigma$, or (II) $\sigma < 0$ and $\vartheta = N|\sigma|$. In either case, we have $1 - \sigma > 0$ and $\vartheta + 1 > 0$; further, $\vartheta + t\sigma > 0$ for

(case I) $t \in \{1, 2, \dots\}$, (case II) $t \in \{1, \dots, N-1\}$. Let (case I) $t \in \{1, 2, \dots\}$, (case II) $t \in \{1, \dots, N-1\}$. Let $n > t$, $A \in \mathcal{A}_t(n)$, and $B \in \mathcal{Z}_A$, and suppose $B = B(A, j)$, $j \in A_\ell$. Note that $|A_\ell| \geq 2$.

By the preceding observations, all the factors in the expressions for $p(A)$ and $p(B)$ (Equation 3) are strictly positive, hence

$$\frac{1}{n}\frac{p(A)}{p(B)} = \frac{1}{n}\frac{t+1}{\vartheta + t\sigma}\,(1 - \sigma + |A_\ell| - 2) \leq \frac{t+1}{\vartheta + t\sigma}\,\frac{1 - \sigma + n - 2}{n},$$

which is bounded above for $n \in \{1, 2, \dots\}$. If $t > N$ in case II, then $p(A)/p(B) = 0/0 = 0$ by convention. (If $t = N$ in case II, then $p(A)/p(B) = \infty$.) Therefore, $\limsup_n c_n(t) < \infty$ in either case, for any $t \in \{1, 2, \dots\}$ except $t = N$ in case II. ■

Proof of Theorem 6 The central part of the argument is Lemma 13 below, from which the result follows easily. Let $t \in \{1, 2, \dots\}$, and assume Conditions 3 and 4 hold. Let $x_1, x_2, \dots \in \mathcal{X}$, and suppose $\sup_{c \in [0,\infty)} \liminf_n \varphi_t(x_{1:n}, c) > 0$ (which occurs with probability 1). We show that this implies $\limsup_n p(T_n = t \mid x_{1:n}) < 1$, proving the theorem.

Let $\alpha \in (0, \infty)$ such that $\limsup_n c_n(t) < \alpha$. Choose $c \in [0, \infty)$ and $\varepsilon \in (0, 1)$ such that $\liminf_n \varphi_t(x_{1:n}, c) > \varepsilon$. Choose $N > 2t/\varepsilon$ large enough that for any $n > N$ we have $c_n(t) < \alpha$ and $\varphi_t(x_{1:n}, c) > \varepsilon$. Then by Lemma 13, for any $n > N$,

$$p(T_n = t \mid x_{1:n}) \leq \frac{C_t(x_{1:n}, c)}{1 + C_t(x_{1:n}, c)} \leq \frac{2tc\alpha/\varepsilon}{1 + 2tc\alpha/\varepsilon},$$

since $\varphi_t(x_{1:n}, c) - t/n > \varepsilon - \varepsilon/2 = \varepsilon/2$ (and $y \mapsto y/(1+y)$ is monotone increasing on $[0, \infty)$). Since this upper bound does not depend on n (and is less than 1), $\limsup_n p(T_n = t \mid x_{1:n}) < 1$. ■

Lemma 13 *Consider a partition-based mixture model. Let $n > t \geq 1$, $x_1, \dots, x_n \in \mathcal{X}$, and $c \in [0, \infty)$. If $c_n(t) < \infty$ and $\varphi_t(x_{1:n}, c) > t/n$, then*

$$p(T_n = t \mid x_{1:n}) \leq \frac{C_t(x_{1:n}, c)}{1 + C_t(x_{1:n}, c)},$$

where $C_t(x_{1:n}, c) = t\,c\,c_n(t)/(\varphi_t(x_{1:n}, c) - t/n)$.

Proof To simplify notation, let us denote $\varphi = \varphi_t(x_{1:n}, c)$, $C = C_t(x_{1:n}, c)$, and $S_A = S_A(x_{1:n}, c)$. Recall the definitions of R_A and $B(A, j)$ from the beginning of Section 3. For $A \in \mathcal{A}_t(n)$, note that

$$|R_A \cap S_A| \geq |S_A| - t \geq n\varphi - t > 0. \tag{8}$$

Further, for any $j \in R_A \cap S_A$, we have $p(A) \leq n\,c_n(t)\,p(B(A, j))$ (by the definition of $c_n(t)$, in Equation 7), and $m(x_{A_\ell}) \leq c\,m(x_{A_\ell \setminus j})m(x_j)$ where A_ℓ is the part containing j (by the

definition of $S_A = S_A(x_{1:n}, c)$, in Section 3). Thus, for any $A \in \mathcal{A}_t(n)$, $j \in R_A \cap S_A$, we have (by Equation 6)

$$\begin{aligned} p(x_{1:n}, A) &= p(A) \prod_{i=1}^{t} m(x_{A_i}) \\ &\leq n\, c_n(t)\, p(B(A,j))\, c \prod_{i=1}^{t+1} m(x_{B_i(A,j)}) = c\, n\, c_n(t)\, p(x_{1:n}, B(A,j)), \end{aligned}$$

and hence, combining this with Equation 8,

$$\begin{aligned} p(x_{1:n}, A) &\leq \frac{c\, n\, c_n(t)}{|R_A \cap S_A|} \sum_{j \in R_A \cap S_A} p(x_{1:n}, B(A,j)) \\ &\leq \frac{c\, c_n(t)}{\varphi - t/n} \sum_{B \in \mathcal{A}_{t+1}(n)} p(x_{1:n}, B)\, I(B \in \mathcal{Y}_A), \end{aligned} \tag{9}$$

where $\mathcal{Y}_A = \big\{B(A,j) : j \in R_A \cap S_A\big\}$. For any $B \in \mathcal{A}_{t+1}(n)$,

$$\#\big\{A \in \mathcal{A}_t(n) : B \in \mathcal{Y}_A\big\} \leq t, \tag{10}$$

since there are only t parts that B_{t+1} could have come from. Therefore,

$$\begin{aligned} p(x_{1:n}, T_n = t) &= \sum_{A \in \mathcal{A}_t(n)} p(x_{1:n}, A) \\ &\overset{\text{(a)}}{\leq} \frac{c\, c_n(t)}{\varphi - t/n} \sum_{A \in \mathcal{A}_t(n)} \sum_{B \in \mathcal{A}_{t+1}(n)} p(x_{1:n}, B)\, I(B \in \mathcal{Y}_A) \\ &= \frac{c\, c_n(t)}{\varphi - t/n} \sum_{B \in \mathcal{A}_{t+1}(n)} p(x_{1:n}, B)\, \#\big\{A \in \mathcal{A}_t(n) : B \in \mathcal{Y}_A\big\} \\ &\overset{\text{(b)}}{\leq} \frac{t\, c\, c_n(t)}{\varphi - t/n} \sum_{B \in \mathcal{A}_{t+1}(n)} p(x_{1:n}, B) \\ &= C\, p(x_{1:n}, T_n = t+1), \end{aligned}$$

where (a) is by Equation 9, and (b) is by Equation 10.

If $p(T_n = t \mid x_{1:n}) = 0$, then trivially $p(T_n = t \mid x_{1:n}) \leq C/(C+1)$. On the other hand, if $p(T_n = t \mid x_{1:n}) > 0$, then $p(x_{1:n}, T_n = t) > 0$, and therefore

$$\begin{aligned} p(T_n = t \mid x_{1:n}) &= \frac{p(x_{1:n}, T_n = t)}{\sum_{t'=1}^{\infty} p(x_{1:n}, T_n = t')} \\ &\leq \frac{p(x_{1:n}, T_n = t)}{p(x_{1:n}, T_n = t) + p(x_{1:n}, T_n = t+1)} \leq \frac{C}{C+1}. \end{aligned}$$

■

Proof of Theorem 7 Suppose $U \subset \mathcal{X}$ satisfies (1) and (2), and let $t \in \{1, 2, \dots\}$. Define $c = \sup\left\{\frac{p_\theta(x)}{m(x)} : x \in U, \theta \in \Theta\right\}$. Let $n > t$ and $x_1, \dots, x_n \in \mathcal{X}$. Now, for any $x \in U$ and $\theta \in \Theta$, we have $p_\theta(x) \leq c\, m(x)$. Hence, for any $J \subset \{1, \dots, n\}$, if $j \in J$ and $x_j \in U$ then

$$m(x_J) = \int_\Theta p_\theta(x_j)\Big[\prod_{i \in J \smallsetminus j} p_\theta(x_i)\Big]\pi(\theta)\, d\theta \leq c\, m(x_j) m(x_{J \smallsetminus j}). \tag{11}$$

Thus, letting $R(x_{1:n}) = \big\{j \in \{1, \dots, n\} : x_j \in U\big\}$, we have $R(x_{1:n}) \subset S_A(x_{1:n}, c)$ for any $A \in \mathcal{A}_t(n)$, and hence, $\varphi_t(x_{1:n}, c) \geq \frac{1}{n}|R(x_{1:n})|$.

Therefore, by (1), with probability 1,

$$\liminf_{n\to\infty} \varphi_t(X_{1:n}, c) \geq \liminf_{n\to\infty} \frac{1}{n}|R(X_{1:n})| > 0.$$

■

8. Proof of the Application to Exponential Families

In this section, we prove Theorem 11. First, we need a few supporting results. Given $y_1, \dots, y_n \in \mathbb{R}^\ell$ (for some $\ell > 0$), $\beta \in (0, 1]$, and $U \subset \mathbb{R}^\ell$, define

$$\mathcal{I}_\beta(y_{1:n}, U) = \prod_{\substack{A \subset \{1,\dots,n\}:\\ |A| \geq \beta n}} I\Big(\frac{1}{|A|}\sum_{j \in A} y_j \in U\Big), \tag{12}$$

where as usual, $I(E)$ is 1 if E is true, and 0 otherwise.

Lemma 14 (Capture lemma) *Let $V \subset \mathbb{R}^k$ be open and convex. Let Q be a probability measure on $\mathbb{R}^k$ such that:*

(1) $\mathbb{E}|Y| < \infty$ *when* $Y \sim Q$,

(2) $Q(\bar{V}) = 1$, *and*

(3) $Q(L) = 0$ *for any hyperplane L that does not intersect V.*

If $Y_1, Y_2, \dots \overset{\text{iid}}{\sim} Q$, then for any $\beta \in (0, 1]$ there exists $U \subset V$ compact such that $\mathcal{I}_\beta(Y_{1:n}, U) \xrightarrow{\text{a.s.}} 1$ as $n \to \infty$.

Proof The proof is rather long, but not terribly difficult. See Appendix D. ■

Proposition 15 *Let $Z_1, Z_2, \dots \in \mathbb{R}^k$ be i.i.d.. If $\beta \in (0, 1]$ and $U \subset \mathbb{R}^k$ such that $\mathbb{P}(Z_j \notin U) < \beta/2$, then $\mathcal{I}_\beta(Y_{1:n}, [\frac{1}{2}, 1]) \xrightarrow{\text{a.s.}} 1$ as $n \to \infty$, where $Y_j = I(Z_j \in U)$.*

Proof By the law of large numbers, $\frac{1}{n}\sum_{j=1}^{n} I(Z_j \notin U) \xrightarrow{\text{a.s.}} \mathbb{P}(Z_j \notin U) < \beta/2$. Hence, with probability 1, for all n sufficiently large, $\frac{1}{n}\sum_{j=1}^{n} I(Z_j \notin U) \leq \beta/2$ holds. When it holds, we have that for any $A \subset \{1,\dots,n\}$ such that $|A| \geq \beta n$,

$$\frac{1}{|A|}\sum_{j\in A} I(Z_j \in U) = 1 - \frac{1}{|A|}\sum_{j\in A} I(Z_j \notin U) \geq 1 - \frac{1}{\beta n}\sum_{j=1}^{n} I(Z_j \notin U) \geq 1/2,$$

i.e., when it holds, we have $\mathcal{I}_\beta(Y_{1:n}, [\frac{1}{2}, 1]) = 1$. Hence, $\mathcal{I}_\beta(Y_{1:n}, [\frac{1}{2}, 1]) \xrightarrow{\text{a.s.}} 1$. ■

Given a well-behaved exponential family with conjugate prior, define

$$\mu_{x_A} = \frac{\xi + \sum_{j\in A} s(x_j)}{\nu + |A|} \tag{13}$$

(cf. Equation 14), where $x_A = (x_j : j \in A)$, $x_j \in \mathcal{X}$. In particular, $\mu_x = (\xi + s(x))/(\nu + 1)$ for $x \in \mathcal{X}$.

Proposition 16 *Consider a well-behaved exponential family with conjugate prior. Let P be a probability measure on $\mathcal{X}$ such that $\mathbb{P}(s(X) \in \overline{\mathcal{M}}) = 1$ when $X \sim P$. Let $X_1, X_2, \dots \overset{\text{iid}}{\sim} P$. Then for any $\beta \in (0, 1]$ there exists $U \subset \mathcal{M}$ compact such that $\mathcal{I}_\beta(Y_{1:n}, [\frac{1}{2}, 1]) \xrightarrow{\text{a.s.}} 1$ as $n \to \infty$, where $Y_j = I(\mu_{X_j} \in U)$.*

Proof Since $\mathcal{M}$ is open and convex, then for any $y \in \overline{\mathcal{M}}$, $z \in \mathcal{M}$, and $\rho \in (0, 1)$, we have $\rho y + (1-\rho)z \in \mathcal{M}$ (by e.g., Rockafellar, 1970, 6.1). Taking $z = \xi/\nu$ and $\rho = 1/(\nu+1)$, this implies that the set $U_0 = \{(\xi + y)/(\nu + 1) : y \in \overline{\mathcal{M}}\}$ is contained in $\mathcal{M}$. Note that U_0 is closed and $\mathbb{P}(\mu_X \in U_0) = \mathbb{P}(s(X) \in \overline{\mathcal{M}}) = 1$. Let $\beta \in (0, 1]$, and choose $r \in (0, \infty)$ such that $\mathbb{P}(|\mu_X| > r) < \beta/2$. Letting $U = \{y \in U_0 : |y| \leq r\}$, we have that $U \subset \mathcal{M}$, and U is compact. Further, $\mathbb{P}(\mu_X \notin U) < \beta/2$, so by applying Proposition 15 with $Z_j = \mu_{X_j}$, we have $\mathcal{I}_\beta(Y_{1:n}, [\frac{1}{2}, 1]) \xrightarrow{\text{a.s.}} 1$. ■

Proposition 17 (Splitting inequality) *Consider a well-behaved exponential family with conjugate prior. For any $U \subset \mathcal{M}$ compact there exists $C \in (0, \infty)$ such that we have the following:*

For any $n \in \{1, 2, \dots\}$, if $A \subset \{1,\dots,n\}$ and $B = \{1,\dots,n\} \setminus A$ are nonempty, and $x_1, \dots, x_n \in \mathcal{X}$ satisfy $\frac{1}{|A|}\sum_{j\in A} s(x_j) \in U$ and $\mu_{x_B} \in U$, then

$$\frac{m(x_{1:n})}{m(x_A)m(x_B)} \leq C\left(\frac{ab}{\nu + n}\right)^{k/2}$$

where $a = \nu + |A|$ and $b = \nu + |B|$. (Recall that k is the dimension of $s : \mathcal{X} \to \mathbb{R}^k$.)

Proof See Appendix B. ■

Lemma 18 *Consider a well-behaved exponential family with conjugate prior, and the resulting collection of single-cluster marginals $m(\cdot)$. Let P be a probability measure on $\mathcal{X}$ satisfying Condition 10 (for the s and $\mathcal{M}$ from the exponential family under consideration), and let $X_1, X_2, \ldots \overset{\text{iid}}{\sim} P$. Then for any $\beta \in (0,1]$ there exists $c \in (0,\infty)$ such that with probability 1, for all n sufficiently large, the following event holds: for every subset $J \subset \{1,\ldots,n\}$ such that $|J| \geq \beta n$, there exists $K \subset J$ such that $|K| \geq \frac{1}{2}|J|$ and for any $j \in K$,*

$$m(X_J) \leq c\, m(X_{J \smallsetminus j})\, m(X_j).$$

Proof Let $\beta \in (0,1]$. Since $\mathcal{M}$ is open and convex, and Condition 10 holds by assumption, then by Lemma 14 (with $V = \mathcal{M}$) there exists $U_1 \subset \mathcal{M}$ compact such that $\mathcal{I}_{\beta/2}(s(X_{1:n}), U_1) \xrightarrow{\text{a.s.}} 1$ as $n \to \infty$, where $s(X_{1:n}) = (s(X_1), \ldots, s(X_n))$. By Proposition 16 above, there exists $U_2 \subset \mathcal{M}$ compact such that $\mathcal{I}_\beta(Y_{1:n}, [\frac{1}{2}, 1]) \xrightarrow{\text{a.s.}} 1$ as $n \to \infty$, where $Y_j = I(\mu_{X_j} \in U_2)$. Hence,

$$\mathcal{I}_{\beta/2}(s(X_{1:n}), U_1)\, \mathcal{I}_\beta(Y_{1:n}, [\tfrac{1}{2}, 1]) \xrightarrow[n\to\infty]{\text{a.s.}} 1.$$

Choose $C \in (0,\infty)$ according to Proposition 17 applied to $U := U_1 \cup U_2$. We will prove the result with $c = (\nu+1)^{k/2} C$. (Recall that k is the dimension of $s : \mathcal{X} \to \mathbb{R}^k$.)

Let n large enough that $\beta n \geq 2$, and suppose that $\mathcal{I}_{\beta/2}(s(X_{1:n}), U_1) = 1$ and $\mathcal{I}_\beta(Y_{1:n}, [\frac{1}{2}, 1]) = 1$. Let $J \subset \{1,\ldots,n\}$ such that $|J| \geq \beta n$. Then for any $j \in J$,

$$\frac{1}{|J \smallsetminus j|} \sum_{i \in J \smallsetminus j} s(X_i) \in U_1 \subset U$$

since $\mathcal{I}_{\beta/2}(s(X_{1:n}), U_1) = 1$ and $|J \smallsetminus j| \geq |J|/2 \geq (\beta/2)n$. Hence, for any $j \in K$, where $K = \{j \in J : \mu_{X_j} \in U\}$, we have

$$\frac{m(X_J)}{m(X_{J \smallsetminus j})\, m(X_j)} \leq C \left(\frac{(\nu + |J| - 1)(\nu+1)}{\nu + |J|} \right)^{k/2} \leq C\,(\nu+1)^{k/2} = c$$

by our choice of C above, and

$$\frac{|K|}{|J|} \geq \frac{1}{|J|} \sum_{j \in J} I(\mu_{X_j} \in U_2) = \frac{1}{|J|} \sum_{j \in J} Y_j \geq 1/2$$

since $\mathcal{I}_\beta(Y_{1:n}, [\frac{1}{2}, 1]) = 1$ and $|J| \geq \beta n$. ∎

Proof of Theorem 11 Let $t \in \{1, 2, \ldots\}$ and choose c according to Lemma 18 with $\beta = 1/t$. We will show that for any $n > t$, if the event of Lemma 18 holds, then $\varphi_t(X_{1:n}, c) \geq 1/(2t)$. Since with probability 1, this event holds for all n sufficiently large, it will follow that with probability 1, $\liminf_n \varphi_t(X_{1:n}, c) \geq 1/(2t) > 0$.

So, let $n > t$ and $x_1, \ldots, x_n \in \mathcal{X}$, and assume the event of Lemma 18 holds. Let $A \in \mathcal{A}_t(n)$. There is at least one part A_ℓ such that $|A_\ell| \geq n/t = \beta n$. Then, by assumption there exists $K_A \subset A_\ell$ such that $|K_A| \geq \frac{1}{2}|A_\ell|$ and for any $j \in K_A$, $m(x_{A_\ell}) \leq c\, m(x_{A_\ell \smallsetminus j})\, m(x_j)$. Thus, $K_A \subset S_A(x_{1:n}, c)$, hence $|S_A(x_{1:n}, c)| \geq |K_A| \geq \frac{1}{2}|A_\ell| \geq n/(2t)$. Since $A \in \mathcal{A}_t(n)$ was

arbitrary, $\varphi_t(x_{1:n}, c) \geq 1/(2t)$. ■

Acknowledgments

We would like to thank Stu Geman for raising this question. Special thanks to the action editor, Yee Whye Teh, and the two anonymous referees, for several helpful comments and suggestions that substantially improved the organization and content of the paper. Thanks also to Steve MacEachern, Erik Sudderth, Mike Hughes, Tamara Broderick, Annalisa Cerquetti, Dan Klein, and Dahlia Nadkarni for helpful conversations. This work was supported in part by NSF grant DMS-1007593 and DARPA contract FA8650-11-1-715.

Appendix A. Exponential Family Properties

We note some well-known properties of exponential families satisfying Condition 8. For a general reference on this material, see Hoffmann-Jørgensen (1994). Let $S_\lambda(s) = \text{support}(\lambda s^{-1})$, that is,

$$S_\lambda(s) = \big\{z \in \mathbb{R}^k : \lambda(s^{-1}(U)) \neq 0 \text{ for every neighborhood } U \text{ of } z\big\}.$$

Let $C_\lambda(s)$ be the closed convex hull of $S_\lambda(s)$ (that is, the intersection of all closed convex sets containing it). Given $U \subset \mathbb{R}^k$, let U° denote its interior. Given a (sufficiently smooth) function $f : \mathbb{R}^k \to \mathbb{R}$, we use f' to denote its gradient, that is, $f'(x)_i = \frac{\partial f}{\partial x_i}(x)$, and $f''(x)$ to denote its Hessian matrix, that is, $f''(x)_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}(x)$.

Proposition 19 *If Condition 8 is satisfied, then:*

(1) κ *is* C^∞ *smooth and strictly convex on* Θ,

(2) $\kappa'(\theta) = \mathbb{E}s(X)$ *and* $\kappa''(\theta) = \operatorname{Cov} s(X)$ *when* $\theta \in \Theta$ *and* $X \sim P_\theta$,

(3) $\kappa''(\theta)$ *is symmetric positive definite for all* $\theta \in \Theta$,

(4) $\kappa' : \Theta \to \mathcal{M}$ *is a* C^∞ *smooth bijection,*

(5) $\kappa'^{-1} : \mathcal{M} \to \Theta$ *is* C^∞ *smooth,*

(6) Θ *is open and convex,*

(7) $\mathcal{M}$ *is open and convex,*

(8) $\mathcal{M} = C_\lambda(s)^\circ$ *and* $\overline{\mathcal{M}} = C_\lambda(s)$, *and*

(9) $\kappa'^{-1}(\mu) = \operatorname{argmax}_{\theta \in \Theta}(\theta^\mathsf{T}\mu - \kappa(\theta))$ *for all* $\mu \in \mathcal{M}$. *The maximizing* $\theta \in \Theta$ *always exists and is unique.*

Proof These properties are all well-known. Let us abbreviate Hoffmann-Jørgensen (1994) as HJ. For (1), see HJ 8.36(1) and HJ 12.7.5. For (6),(2),(3), and (4), see HJ 8.36, 8.36.2-3, 12.7(2), and 12.7.11, respectively. Item (5) and openness in (7) follow, using the inverse function theorem (Knapp, 2005, 3.21). Item (8) and convexity in (7) follow, using HJ 8.36.15 and Rockafellar (1970) 6.2-3. Item (9) follows from HJ 8.36.15 and item (4). ■

Given an exponential family with conjugate prior as in Section 5.2, the joint density of $x_1, \ldots, x_n \in \mathcal{X}$ and $\theta \in \mathbb{R}^k$ is

$$\begin{aligned} &p_\theta(x_1) \cdots p_\theta(x_n) \pi_{\xi,\nu}(\theta) \\ &= \exp\Big((\nu+n)\big(\theta^{\mathsf{T}} \mu_{x_{1:n}} - \kappa(\theta)\big) \Big) \exp(-\psi(\xi,\nu))\, I(\theta \in \Theta) \end{aligned} \tag{14}$$

where $\mu_{x_{1:n}} = (\xi + \sum_{j=1}^n s(x_j))/(\nu+n)$. The marginal density, defined as in Equation 5, is

$$m(x_{1:n}) = \exp\Big(\psi\big(\xi + \textstyle\sum s(x_j),\, \nu+n\big) - \psi(\xi,\nu) \Big) \tag{15}$$

when this quantity is well-defined.

Proposition 20 *If Condition 8 is satisfied, then:*

(1) *$\psi(\xi,\nu)$ is finite and C^∞ smooth on Ξ,*

(2) *if $s(x_1), \ldots, s(x_n) \in S_\lambda(s)$ and $(\xi,\nu) \in \Xi$, then $(\xi + \sum s(x_j),\, \nu+n) \in \Xi$,*

(3) *$\{\pi_{\xi,\nu} : (\xi,\nu) \in \Xi\}$ is a conjugate family for $\{p_\theta : \theta \in \Theta\}$, and*

(4) *if $s : \mathcal{X} \to \mathbb{R}^k$ is continuous, $(\xi,\nu) \in \Xi$, and $\lambda(U) \neq 0$ for any nonempty $U \subset \mathcal{X}$ that is open in $\mathcal{X}$, then $m(x_{1:n}) < \infty$ for any $x_1, \ldots, x_n \in \mathcal{X}$.*

Proof (1) For finiteness, see Diaconis and Ylvisaker (1979), Theorem 1. Smoothness holds for the same reason that κ is smooth; see Hoffmann-Jørgensen (1994, 8.36(1)). (Note that Ξ is open in $\mathbb{R}^{k+1}$, since $\mathcal{M}$ is open in $\mathbb{R}^k$.)

(2) Since $C_\lambda(s)$ is convex, $\frac{1}{n}\sum s(x_j) \in C_\lambda(s)$. Since $C_\lambda(s) = \overline{\mathcal{M}}$ and $\mathcal{M}$ is open and convex by 19(7) and (8), then $(\xi + \sum s(x_j))/(\nu+n) \in \mathcal{M}$, as a (strict) convex combination of $\frac{1}{n}\sum s(x_j) \in \overline{\mathcal{M}}$ and $\xi/\nu \in \mathcal{M}$ (Rockafellar, 1970, 6.1).

(3) Let $(\xi,\nu) \in \Xi$, $\theta \in \Theta$. If $X_1, \ldots, X_n \overset{\text{iid}}{\sim} P_\theta$ then $s(X_1), \ldots, s(X_n) \in S_\lambda(s)$ almost surely, and thus $(\xi + \sum s(X_j),\, \nu+n) \in \Xi$ (a.s.) by (2). By Equations 14 and 15, the posterior is $\pi_{\xi+\sum s(X_j),\,\nu+n}$.

(4) The assumptions imply $\{s(x) : x \in \mathcal{X}\} \subset S_\lambda(s)$, and therefore, for any $x_1, \ldots, x_n \in \mathcal{X}$, we have $(\xi + \sum s(x_j),\, \nu+n) \in \Xi$ by (2). Thus, by (1) and Equation 15, $m(x_{1:n}) < \infty$. ■

It is worth mentioning that while $\Xi \subset \big\{(\xi,\nu) \in \mathbb{R}^{k+1} : \psi(\xi,\nu) < \infty\big\}$, it may be a strict subset—often, Ξ is not quite the full set of parameters on which $\pi_{\xi,\nu}$ can be defined.

Proof of Proposition 12 (1) For any $\theta \in \Theta$ and any $j \in \{1, \ldots, k\}$,

$$\int_{\mathcal{X}} s_j(x)^2 p_\theta(x)\, d\lambda(x) = \exp(-\kappa(\theta)) \frac{\partial^2}{\partial \theta_j^2} \int_{\mathcal{X}} \exp(\theta^{\mathsf{T}} s(x))\, d\lambda(x) < \infty$$

(Hoffmann-Jørgensen, 1994, 8.36.1). Since P has density $f = \sum \pi_i p_{\theta(i)}$ with respect to λ, then

$$\mathbb{E} s_j(X)^2 = \int_{\mathcal{X}} s_j(x)^2 f(x)\, d\lambda(x) = \sum_{i=1}^{t} \pi_i \int_{\mathcal{X}} s_j(x)^2 p_{\theta(i)}(x)\, d\lambda(x) < \infty,$$

and hence

$$(\mathbb{E}|s(X)|)^2 \leq \mathbb{E}|s(X)|^2 = \mathbb{E} s_1(X)^2 + \cdots + \mathbb{E} s_k(X)^2 < \infty.$$

(2) Note that $S_P(s) \subset S_\lambda(s)$ (in fact, they are equal since P_θ and λ are mutually absolutely continuous for any $\theta \in \Theta$), and therefore

$$S_P(s) \subset S_\lambda(s) \subset C_\lambda(s) = \overline{\mathcal{M}}$$

by Proposition 19(8). Hence,

$$\mathbb{P}(s(X) \in \overline{\mathcal{M}}) \geq \mathbb{P}(s(X) \in S_P(s)) = P s^{-1}(\text{support}(P s^{-1})) = 1.$$

(3) Suppose λ is absolutely continuous with respect to Lebesgue measure, $\mathcal{X}$ is open and connected, and s is real analytic. Let $L \subset \mathbb{R}^k$ be a hyperplane, and write $L = \{z \in \mathbb{R}^k : z^{\mathsf{T}} y = b\}$ where $y \in \mathbb{R}^k \setminus \{0\}$, $b \in \mathbb{R}$. Define $g : \mathcal{X} \to \mathbb{R}$ by $g(x) = s(x)^{\mathsf{T}} y - b$. Then g is real analytic on $\mathcal{X}$, since a finite sum of real analytic functions is real analytic. Since $\mathcal{X}$ is connected, it follows that either g is identically zero, or the set $V = \{x \in \mathcal{X} : g(x) = 0\}$ has Lebesgue measure zero (Krantz, 1992). Now, g cannot be identically zero, since for any $\theta \in \Theta$, letting $Z \sim P_\theta$, we have

$$0 < y^{\mathsf{T}} \kappa''(\theta) y = y^{\mathsf{T}} (\text{Cov}\, s(Z)) y = \text{Var}(y^{\mathsf{T}} s(Z)) = \text{Var}\, g(Z)$$

by Proposition 19(2) and (3). Consequently, V must have Lebesgue measure zero. Hence, $P(V) = 0$, since P is absolutely continuous with respect to λ, and thus, with respect to Lebesgue measure. Therefore,

$$\mathbb{P}(s(X) \in L) = \mathbb{P}(g(X) = 0) = P(V) = 0.$$

■

Appendix B. Marginal Inequalities

In this section, we prove Proposition 17, which was used in the key lemma for the exponential family case (Lemma 18).

Consider a well-behaved exponential family with conjugate prior (as in Definition 9). The proof uses some simple bounds on the Laplace approximation (see Appendix C) to obtain inequalities involving the marginal density $m(x_{1:n})$ (cf. Equations 5 and 15) of

$x_{1:n} = (x_1, \ldots, x_n)$, where $x_j \in \mathcal{X}$. Of course, it is commonplace to apply the Laplace approximation to $m(X_{1:n})$ when $X_1, \ldots, X_n$ are i.i.d. random variables and n is sufficiently large. In contrast, our application of it is considerably more subtle. For our purposes, it is necessary to show that for every n, even without assuming i.i.d. data, the approximation is good enough as long as the sufficient statistics are not too extreme.

We make extensive use of the exponential family properties in Appendix A, often without mention. We use f' to denote the gradient and f'' to denote the Hessian of a (sufficiently smooth) function $f : \mathbb{R}^k \to \mathbb{R}$. For $\mu \in \mathcal{M}$, define

$$\begin{aligned} f_\mu(\theta) &= \theta^{\mathsf{T}}\mu - \kappa(\theta), \\ \mathcal{L}(\mu) &= \sup_{\theta \in \Theta} \big(\theta^{\mathsf{T}}\mu - \kappa(\theta)\big), \\ \theta_\mu &= \operatorname*{argmax}_{\theta \in \Theta} \big(\theta^{\mathsf{T}}\mu - \kappa(\theta)\big), \end{aligned}$$

and note that $\theta_\mu = \kappa'^{-1}(\mu)$ (Proposition 19). $\mathcal{L}$ is known as the Legendre transform of κ. Note that $\mathcal{L}(\mu) = f_\mu(\theta_\mu)$, and $\mathcal{L}$ is C^∞ smooth on $\mathcal{M}$ (since $\mathcal{L}(\mu) = \theta_\mu^{\mathsf{T}}\mu - \kappa(\theta_\mu)$, $\theta_\mu = \kappa'^{-1}(\mu)$, and both κ and κ'^{-1} are C^∞ smooth). As in Equation 13, define

$$\mu_{x_{1:n}} = \frac{\xi + \sum_{j=1}^n s(x_j)}{\nu + n} \tag{16}$$

and given $x_{1:n}$ such that $\mu_{x_{1:n}} \in \mathcal{M}$, define

$$\widetilde{m}(x_{1:n}) = (\nu + n)^{-k/2} \exp\big((\nu + n)\, \mathcal{L}(\mu_{x_{1:n}})\big),$$

where k is the dimension of the sufficient statistics function $s : \mathcal{X} \to \mathbb{R}^k$. Proposition 21 below provides uniform bounds on $m(x_{1:n})/\widetilde{m}(x_{1:n})$. Here, $\widetilde{m}(x_{1:n})$ is only intended to approximate $m(x_{1:n})$ up to a multiplicative constant—a better approximation could always be obtained via the usual asymptotic form of the Laplace approximation.

Proposition 21 *Consider a well-behaved exponential family with conjugate prior. For any $U \subset \mathcal{M}$ compact, there exist $C_1, C_2 \in (0, \infty)$ such that for any $n \in \{1, 2, \ldots\}$ and any $x_1, \ldots, x_n \in \mathcal{X}$ satisfying $\mu_{x_{1:n}} \in U$, we have*

$$C_1 \leq \frac{m(x_{1:n})}{\widetilde{m}(x_{1:n})} \leq C_2.$$

Proof Assume $U \neq \varnothing$, since otherwise the result is trivial. Let

$$V = \kappa'^{-1}(U) = \{\theta_\mu : \mu \in U\}.$$

It is straightforward to show that there exists $\varepsilon \in (0, 1)$ such that $V_\varepsilon \subset \Theta$ where

$$V_\varepsilon = \{\theta \in \mathbb{R}^k : d(\theta, V) \leq \varepsilon\}.$$

(Here, $d(\theta, V) = \inf_{\theta' \in V} |\theta - \theta'|$.) Note that V_ε is compact, since κ'^{-1} is continuous. Given a symmetric matrix A, define $\lambda_*(A)$ and $\lambda^*(A)$ to be the minimal and maximal eigenvalues, respectively, and recall that λ_*, λ^* are continuous functions of the entries of A. Letting

$$\alpha = \min_{\theta \in V_\varepsilon} \lambda_*(\kappa''(\theta)) \quad \text{and} \quad \beta = \max_{\theta \in V_\varepsilon} \lambda^*(\kappa''(\theta)),$$

we have $0 < \alpha \leq \beta < \infty$ since V_ε is compact and $\lambda_*(\kappa''(\cdot))$, $\lambda^*(\kappa''(\cdot))$ are continuous and positive on Θ. Letting

$$\gamma = \sup_{\mu \in U} e^{-f_\mu(\theta_\mu)} \int_\Theta \exp(f_\mu(\theta)) d\theta = \sup_{\mu \in U} e^{-\mathcal{L}(\mu)} e^{\psi(\mu,1)}$$

we have $0 < \gamma < \infty$ since U is compact, and both $\mathcal{L}$ (as noted above) and $\psi(\mu, 1)$ (by Proposition 20) are continuous on $\mathcal{M}$. Define

$$h(\mu, \theta) = f_\mu(\theta_\mu) - f_\mu(\theta) = \mathcal{L}(\mu) - \theta^{\mathsf{T}}\mu + \kappa(\theta)$$

for $\mu \in \mathcal{M}$, $\theta \in \Theta$. For any $\mu \in \mathcal{M}$, we have that $h(\mu, \theta) > 0$ whenever $\theta \in \Theta \setminus \{\theta_\mu\}$, and that $h(\mu, \theta)$ is strictly convex in θ. Letting $B_\varepsilon(\theta_\mu) = \{\theta \in \mathbb{R}^k : |\theta - \theta_\mu| \leq \varepsilon\}$, it follows that

$$\delta := \inf_{\mu \in U} \inf_{\theta \in \Theta \setminus B_\varepsilon(\theta_\mu)} h(\mu, \theta) = \inf_{\mu \in U} \inf_{u \in \mathbb{R}^k : |u| = 1} h(\mu, \theta_\mu + \varepsilon u)$$

is positive, as the minimum of a positive continuous function on a compact set.

Now, applying the Laplace approximation bounds in Corollary 24 with $\alpha, \beta, \gamma, \delta, \varepsilon$ as just defined, we obtain $c_1, c_2 \in (0, \infty)$ such that for any $\mu \in U$ we have (taking $E = \Theta$, $f = -f_\mu$, $x_0 = \theta_\mu$, $A = \alpha I_{k \times k}$, $B = \beta I_{k \times k}$)

$$c_1 \leq \frac{\int_\Theta \exp(t f_\mu(\theta)) d\theta}{t^{-k/2} \exp(t f_\mu(\theta_\mu))} \leq c_2$$

for any $t \geq 1$. We prove the result with $C_i = c_i \, e^{-\psi(\xi,\nu)}$ for $i = 1, 2$.

Let $n \in \{1, 2, \dots\}$ and $x_1, \dots, x_n \in \mathcal{X}$ such that $\mu_{x_{1:n}} \in U$. Choose $t = \nu + n$. By integrating Equation 14, we have

$$m(x_{1:n}) = e^{-\psi(\xi,\nu)} \int_\Theta \exp\left(t f_{\mu_{x_{1:n}}}(\theta)\right) d\theta,$$

and meanwhile,

$$\widetilde{m}(x_{1:n}) = t^{-k/2} \exp\left(t f_{\mu_{x_{1:n}}}(\theta_{\mu_{x_{1:n}}})\right).$$

Thus, combining the preceding three displayed equations,

$$0 < C_1 = c_1 e^{-\psi(\xi,\nu)} \leq \frac{m(x_{1:n})}{\widetilde{m}(x_{1:n})} \leq c_2 e^{-\psi(\xi,\nu)} = C_2 < \infty.$$

■

Proof of Proposition 17 Let U' be the convex hull of $U \cup \{\xi/\nu\}$. Then U' is compact (as the convex hull of a compact set in $\mathbb{R}^k$) and $U' \subset \mathcal{M}$ (since $U \cup \{\xi/\nu\} \subset \mathcal{M}$ and $\mathcal{M}$ is convex). We show that the result holds with $C = C_2 \exp(C_0)/C_1^2$, where $C_1, C_2 \in (0, \infty)$ are obtained by applying Proposition 21 to U', and

$$C_0 = \nu \sup_{y \in U'} |(\xi/\nu - y)^{\mathsf{T}} \mathcal{L}'(y)| + \nu \sup_{y \in U'} |\mathcal{L}(y)| < \infty. \tag{17}$$

Since $\mathcal{L}$ is convex (being a Legendre transform) and smooth, then for any $y, z \in \mathcal{M}$ we have

$$\inf_{\rho \in (0,1)} \frac{1}{\rho}\big(\mathcal{L}(y + \rho(z-y)) - \mathcal{L}(y)\big) = (z-y)^{\mathsf{T}}\mathcal{L}'(y)$$

(by e.g., Rockafellar, 1970, 23.1) and therefore for any $\rho \in (0,1)$,

$$\mathcal{L}(y) \le \mathcal{L}((1-\rho)y + \rho z) - \rho(z-y)^{\mathsf{T}}\mathcal{L}'(y). \tag{18}$$

Choosing $y = \mu_{x_{1:n}}$, $z = \xi/\nu$, and $\rho = \nu/(n+2\nu)$, we have

$$(1-\rho)y + \rho z = \frac{2\xi + \sum_{j=1}^{n} s(x_j)}{2\nu + n} = \frac{a\mu_{x_A} + b\mu_{x_B}}{a+b}. \tag{19}$$

Note that $\mu_{x_A}, \mu_{x_B}, \mu_{x_{1:n}} \in U'$, by taking various convex combinations of ξ/ν, $\frac{1}{|A|}\sum_{j\in A} s(x_j)$, $\mu_{x_B} \in U'$. Thus,

$$\begin{aligned}
(\nu+n)\mathcal{L}(\mu_{x_{1:n}}) &= (a+b)\mathcal{L}(y) - \nu\mathcal{L}(y) \\
&\overset{(a)}{\le} (a+b)\mathcal{L}((1-\rho)y + \rho z) - (a+b)\rho(z-y)^{\mathsf{T}}\mathcal{L}'(y) - \nu\mathcal{L}(y) \\
&\overset{(b)}{\le} (a+b)\mathcal{L}\Big(\frac{a\mu_{x_A} + b\mu_{x_B}}{a+b}\Big) + C_0 \\
&\overset{(c)}{\le} a\mathcal{L}(\mu_{x_A}) + b\mathcal{L}(\mu_{x_B}) + C_0,
\end{aligned}$$

where (a) is by Equation 18, (b) is by Equations 17 and 19, and (c) is by the convexity of $\mathcal{L}$. Hence, $(\nu+n)^{k/2}\widetilde{m}(x_{1:n}) \le (ab)^{k/2}\widetilde{m}(x_A)\widetilde{m}(x_B)\exp(C_0)$, so by our choice of C_1 and C_2,

$$\frac{m(x_{1:n})}{m(x_A)m(x_B)} \le \frac{C_2\widetilde{m}(x_{1:n})}{C_1^2\widetilde{m}(x_A)\widetilde{m}(x_B)} \le \frac{C_2\exp(C_0)}{C_1^2}\Big(\frac{ab}{n+\nu}\Big)^{k/2}.$$

■

Appendix C. Bounds on the Laplace Approximation

Our proof uses the following simple bounds on the Laplace approximation. These bounds are not fundamentally new, but the precise formulation we require does not seem to appear in the literature, so we have included it for the reader's convenience. Lemma 22 is simply a multivariate version of the bounds given by De Bruijn (1970), and Corollary 24 is a straightforward consequence, putting the lemma in a form most convenient for our purposes.

Given symmetric matrices A and B, let us write $A \trianglelefteq B$ to mean that $B - A$ is positive semidefinite. Given $A \in \mathbb{R}^{k\times k}$ symmetric positive definite and $\varepsilon, t \in (0,\infty)$, define

$$C(t,\varepsilon,A) = \mathbb{P}(|A^{-1/2}Z| \le \varepsilon\sqrt{t})$$

where $Z \sim \text{Normal}(0, I_{k\times k})$. Note that $C(t,\varepsilon,A) \to 1$ as $t \to \infty$. Let $B_\varepsilon(x_0) = \{x \in \mathbb{R}^k : |x - x_0| \le \varepsilon\}$ denote the closed ball of radius $\varepsilon > 0$ at $x_0 \in \mathbb{R}^k$.

Lemma 22 *Let $E \subset \mathbb{R}^k$ be open. Let $f : E \to \mathbb{R}$ be C^2 smooth with $f'(x_0) = 0$ for some $x_0 \in E$. Define*

$$g(t) = \int_E \exp(-tf(x))\,dx$$

for $t \in (0,\infty)$. Suppose $\varepsilon \in (0,\infty)$ such that $B_\varepsilon(x_0) \subset E$, $0 < \delta \le \inf\{f(x) - f(x_0) : x \in E \setminus B_\varepsilon(x_0)\}$, and A, B are symmetric positive definite matrices such that $A \trianglelefteq f''(x) \trianglelefteq B$ for all $x \in B_\varepsilon(x_0)$. Then for any $0 < s \le t$ we have

$$\frac{C(t,\varepsilon,B)}{|B|^{1/2}} \le \frac{g(t)}{(2\pi/t)^{k/2}e^{-tf(x_0)}} \le \frac{C(t,\varepsilon,A)}{|A|^{1/2}} + \Big(\frac{t}{2\pi}\Big)^{k/2} e^{-(t-s)\delta} e^{sf(x_0)} g(s)$$

where $|A| = |\det A|$.

Remark 23 *In particular, these assumptions imply f is strictly convex on $B_\varepsilon(x_0)$ with unique global minimum at x_0. Note that the upper bound is trivial unless $g(s) < \infty$.*

Proof By Taylor's theorem, for any $x \in B_\varepsilon(x_0)$ there exists z_x on the line between x_0 and x such that, letting $y = x - x_0$,

$$f(x) = f(x_0) + y^{\mathsf{T}} f'(x_0) + \tfrac{1}{2} y^{\mathsf{T}} f''(z_x) y = f(x_0) + \tfrac{1}{2} y^{\mathsf{T}} f''(z_x) y.$$

Since $z_x \in B_\varepsilon(x_0)$, and thus $A \trianglelefteq f''(z_x) \trianglelefteq B$,

$$\tfrac{1}{2} y^{\mathsf{T}} A y \le f(x) - f(x_0) \le \tfrac{1}{2} y^{\mathsf{T}} B y.$$

Hence,

$$\begin{aligned} e^{tf(x_0)} \int_{B_\varepsilon(x_0)} \exp(-tf(x))\,dx &\le \int_{B_\varepsilon(x_0)} \exp(-\tfrac{1}{2}(x-x_0)^{\mathsf{T}}(tA)(x-x_0))\,dx \\ &= (2\pi)^{k/2} |(tA)^{-1}|^{1/2}\ \mathbb{P}\left(|(tA)^{-1/2} Z| \le \varepsilon\right). \end{aligned}$$

Along with a similar argument for the lower bound, this implies

$$\Big(\frac{2\pi}{t}\Big)^{k/2} \frac{C(t,\varepsilon,B)}{|B|^{1/2}} \le e^{tf(x_0)} \int_{B_\varepsilon(x_0)} \exp(-tf(x))\,dx \le \Big(\frac{2\pi}{t}\Big)^{k/2} \frac{C(t,\varepsilon,A)}{|A|^{1/2}}.$$

Considering the rest of the integral, outside of $B_\varepsilon(x_0)$, we have

$$0 \le \int_{E \setminus B_\varepsilon(x_0)} \exp(-tf(x))\,dx \le \exp\big(-(t-s)(f(x_0)+\delta)\big)\, g(s).$$

Combining the preceding four inequalities yields the result. ∎

The following corollary tailors the lemma to our purposes. Given a symmetric positive definite matrix $A \in \mathbb{R}^{k\times k}$, let $\lambda_*(A)$ and $\lambda^*(A)$ be the minimal and maximal eigenvalues, respectively. By diagonalizing A, it is easy to check that $\lambda_*(A) I_{k\times k} \trianglelefteq A \trianglelefteq \lambda^*(A) I_{k\times k}$ and $\lambda_*(A)^k \le |A| \le \lambda^*(A)^k$.

Corollary 24 *For any $\alpha, \beta, \gamma, \delta, \varepsilon \in (0, \infty)$ there exist $c_1 = c_1(\beta, \varepsilon) \in (0, \infty)$ and $c_2 = c_2(\alpha, \gamma, \delta) \in (0, \infty)$ such that if E, f, x_0, A, B satisfy all the conditions of Lemma 22 (for this choice of δ, ε) and additionally, $\alpha \leq \lambda_*(A)$, $\beta \geq \lambda^*(B)$, and $\gamma \geq e^{f(x_0)} g(1)$, then*

$$c_1 \leq \frac{\int_E \exp(-tf(x))\, dx}{t^{-k/2} \exp(-tf(x_0))} \leq c_2$$

for all $t \geq 1$.

Proof The first term in the upper bound of the lemma is $C(t, \varepsilon, A)/|A|^{1/2} \leq 1/\alpha^{k/2}$, and with $s = 1$ the second term is less or equal to $(t/2\pi)^{k/2} e^{-(t-1)\delta} \gamma$, which is bounded above for $t \in [1, \infty)$. For the lower bound, a straightforward calculation (using $z^{\mathsf{T}} B z \leq \lambda^*(B) z^{\mathsf{T}} z \leq \beta z^{\mathsf{T}} z$ in the exponent inside the integral) shows that $C(t, \varepsilon, B)/|B|^{1/2} \geq \mathbb{P}(|Z| \leq \varepsilon\sqrt{\beta})/\beta^{k/2}$ for $t \geq 1$. ■

Although we do not need it (and thus, we omit the proof), the following corollary gives the well-known asymptotic form of the Laplace approximation. (As usual, $g(t) \sim h(t)$ as $t \to \infty$ means that $g(t)/h(t) \to 1$.)

Corollary 25 *Let $E \subset \mathbb{R}^k$ be open. Let $f : E \to \mathbb{R}$ be C^2 smooth such that for some $x_0 \in E$ we have that $f'(x_0) = 0$, $f''(x_0)$ is positive definite, and $f(x) > f(x_0)$ for all $x \in E \setminus \{x_0\}$. Suppose there exists $\varepsilon > 0$ such that $B_\varepsilon(x_0) \subset E$ and $\inf\{f(x) - f(x_0) : x \in E \setminus B_\varepsilon(x_0)\}$ is positive, and suppose there is some $s > 0$ such that $\int_E e^{-sf(x)}\, dx < \infty$. Then*

$$\int_E \exp(-tf(x))\, dx \;\sim\; \Big(\frac{2\pi}{t}\Big)^{k/2} \frac{\exp(-tf(x_0))}{|f''(x_0)|^{1/2}}$$

as $t \to \infty$.

Appendix D. Capture Lemma

In this section, we prove Lemma 14, which is restated here for the reader's convenience.

The following definitions are standard. Let $\mathcal{S}$ denote the unit sphere in $\mathbb{R}^k$, that is, $\mathcal{S} = \{x \in \mathbb{R}^k : |x| = 1\}$. We say that $H \subset \mathbb{R}^k$ is a *halfspace* if $H = \{x \in \mathbb{R}^k : x^{\mathsf{T}} u \prec b\}$, where $\prec$ is either $<$ or $\leq$, for some $u \in \mathcal{S}$, $b \in \mathbb{R}$. We say that $L \subset \mathbb{R}^k$ is a *hyperplane* if $L = \{x \in \mathbb{R}^k : x^{\mathsf{T}} u = b\}$ for some $u \in \mathcal{S}$, $b \in \mathbb{R}$. Given $U \subset \mathbb{R}^k$, let ∂U denote the *boundary of U*, that is, $\partial U = \bar{U} \setminus U^\circ$. So, for example, if H is a halfspace, then ∂H is a hyperplane. The following notation is also useful: given $x \in \mathbb{R}^k$, we call the set $R_x = \{ax : a > 0\}$ the *ray through x*.

We give the central part of the proof first, postponing some plausible intermediate results for the moment. Recall the definition of $\mathcal{I}_\beta(x_{1:n}, U)$ from Equation 12.

Lemma 26 (Capture lemma) *Let $V \subset \mathbb{R}^k$ be open and convex. Let P be a probability measure on $\mathbb{R}^k$ such that:*

(1) $\mathbb{E}|X| < \infty$ *when* $X \sim P$,

(2) $P(\bar{V}) = 1$, *and*

(3) $P(L) = 0$ *for any hyperplane* L *that does not intersect* V*.*

If $X_1, X_2, \ldots \overset{\text{iid}}{\sim} P$*, then for any* $\beta \in (0,1]$ *there exists* $U \subset V$ *compact such that* $\mathcal{I}_\beta(X_{1:n}, U) \xrightarrow{\text{a.s.}} 1$ *as* $n \to \infty$*.*

Proof Without loss of generality, we may assume $0 \in V$ (since otherwise we can translate to make it so, obtain U, and translate back). Let $\beta \in (0,1]$. By Proposition 28 below, for each $u \in \mathcal{S}$ there is a closed halfspace H_u such that $0 \in H_u^\circ$, R_u intersects $V \cap \partial H_u$, and $\mathcal{I}_\beta(X_{1:n}, H_u) \xrightarrow{\text{a.s.}} 1$ as $n \to \infty$. By Proposition 30 below, there exist $u_1, \ldots, u_r \in \mathcal{S}$ (for some $r > 0$) such that the set $U = \bigcap_{i=1}^r H_{u_i}$ is compact and $U \subset V$. Finally,

$$\mathcal{I}_\beta(X_{1:n}, U) = \prod_{i=1}^r \mathcal{I}_\beta(X_{1:n}, H_{u_i}) \xrightarrow[n\to\infty]{\text{a.s.}} 1.$$

■

The main idea of the lemma is exhibited in the following simpler case, which we will use to prove Proposition 28.

Proposition 27 *Let* $V = (-\infty, c)$*, where* $-\infty < c \leq \infty$*. Let* P *be a probability measure on* $\mathbb{R}$ *such that:*

(1) $\mathbb{E}|X| < \infty$ *when* $X \sim P$*, and*

(2) $P(V) = 1$*.*

If $X_1, X_2, \ldots \overset{\text{iid}}{\sim} P$*, then for any* $\beta \in (0,1]$ *there exists* $b < c$ *such that* $\mathcal{I}_\beta(X_{1:n}, (-\infty, b]) \xrightarrow{\text{a.s.}} 1$ *as* $n \to \infty$*.*

Proof Let $\beta \in (0,1]$. By continuity from above, there exists $a < c$ such that $\mathbb{P}(X > a) < \beta$. If $\mathbb{P}(X > a) = 0$ then the result is trivial, taking $b = a$. Suppose $\mathbb{P}(X > a) > 0$. Let b such that $\mathbb{E}(X \mid X > a) < b < c$, which is always possible, by a straightforward argument (using $\mathbb{E}|X| < \infty$ in the $c = \infty$ case). Let $B_n = B_n(X_1, \ldots, X_n) = \{i \in \{1, \ldots, n\} : X_i > a\}$. Then

$$\begin{aligned}\frac{1}{|B_n|} \sum_{i \in B_n} X_i &= \frac{1}{\frac{1}{n}|B_n|} \frac{1}{n} \sum_{i=1}^n X_i \, I(X_i > a) \\ &\xrightarrow[n\to\infty]{\text{a.s.}} \frac{\mathbb{E}(X \, I(X > a))}{\mathbb{P}(X > a)} = \mathbb{E}(X \mid X > a) < b.\end{aligned}$$

Now, fix $n \in \{1, 2, \ldots\}$, and suppose $0 < |B_n| < \beta n$ and $\frac{1}{|B_n|} \sum_{i \in B_n} X_i < b$, noting that with probability 1, this happens for all n sufficiently large. We show that this implies $\mathcal{I}_\beta(X_{1:n}, (-\infty, b]) = 1$. This will prove the result.

Let $A \subset \{1, \ldots, n\}$ such that $|A| \geq \beta n$. Let $M = \{\pi_1, \ldots, \pi_{|A|}\}$ where π is a permutation of $\{1, \ldots, n\}$ such that $X_{\pi_1} \geq \cdots \geq X_{\pi_n}$ (that is, $M \subset \{1, \ldots, n\}$ consists of the indices of

$|A|$ of the largest entries of $(X_1, \ldots, X_n)$). Then $|M| = |A| \geq \beta n \geq |B_n|$, and it follows that $B_n \subset M$. Therefore,

$$\frac{1}{|A|} \sum_{i \in A} X_i \leq \frac{1}{|M|} \sum_{i \in M} X_i \leq \frac{1}{|B_n|} \sum_{i \in B_n} X_i \leq b,$$

as desired. ■

The first of the two propositions used in Lemma 26 is the following.

Proposition 28 *Let V and P satisfy the conditions of Lemma 26, and also assume $0 \in V$. If $X_1, X_2, \ldots \overset{\text{iid}}{\sim} P$ then for any $\beta \in (0, 1]$ and any $u \in \mathcal{S}$ there is a closed halfspace $H \subset \mathbb{R}^k$ such that*

(1) *$0 \in H^\circ$,*

(2) *R_u intersects $V \cap \partial H$, and*

(3) *$\mathcal{I}_\beta(X_{1:n}, H) \xrightarrow{\text{a.s.}} 1$ as $n \to \infty$.*

Proof Let $\beta \in (0, 1]$ and $u \in \mathcal{S}$. Either (a) $R_u \subset V$, or (b) R_u intersects ∂V.

Case (a): Suppose $R_u \subset V$. Let $Y_i = X_i^\mathsf{T} u$ for $i = 1, 2, \ldots$. Then $\mathbb{E}|Y_i| \leq \mathbb{E}|X_i||u| = \mathbb{E}|X_i| < \infty$, and thus, by Proposition 27 (with $c = \infty$) there exists $b \in \mathbb{R}$ such that $\mathcal{I}_\beta(Y_{1:n}, (-\infty, b]) \xrightarrow{\text{a.s.}} 1$. Let us choose this b to be positive, which is always possible since $\mathcal{I}_\beta(Y_{1:n}, (-\infty, b])$ is nondecreasing as a function of b. Let $H = \{x \in \mathbb{R}^k : x^\mathsf{T} u \leq b\}$. Then $0 \in H^\circ$, since $b > 0$, and R_u intersects $V \cap \partial H$ at bu, since $R_u \subset V$ and $bu^\mathsf{T} u = b$. And since $\frac{1}{|A|} \sum_{i \in A} Y_i \leq b$ if and only if $\frac{1}{|A|} \sum_{i \in A} X_i \in H$, we have $\mathcal{I}_\beta(X_{1:n}, H) \xrightarrow{\text{a.s.}} 1$.

Case (b): Suppose R_u intersects ∂V at some point $z \in \mathbb{R}^k$. Note that $z \neq 0$ since $0 \notin R_u$. Since $\bar{V}$ is convex, it has a supporting hyperplane at z, and thus, there exist $v \in \mathcal{S}$ and $c \in \mathbb{R}$ such that $G = \{x \in \mathbb{R}^k : x^\mathsf{T} v \leq c\}$ satisfies $\bar{V} \subset G$ and $z \in \partial G$ (Rockafellar, 1970, 11.2). Note that $c > 0$ and $V \cap \partial G = \varnothing$ since $0 \in V$ and V is open. Letting $Y_i = X_i^\mathsf{T} v$ for $i = 1, 2, \ldots$, we have

$$\mathbb{P}(Y_i \leq c) = \mathbb{P}(X_i^\mathsf{T} v \leq c) = \mathbb{P}(X_i \in G) \geq \mathbb{P}(X_i \in \bar{V}) = P(\bar{V}) = 1,$$

and hence,

$$\mathbb{P}(Y_i \geq c) = \mathbb{P}(Y_i = c) = \mathbb{P}(X_i^\mathsf{T} v = c) = \mathbb{P}(X_i \in \partial G) = P(\partial G) = 0,$$

by our assumptions on P, since ∂G is a hyperplane that does not intersect V. Consequently, $\mathbb{P}(Y_i < c) = 1$. Also, as before, $\mathbb{E}|Y_i| < \infty$. Thus, by Proposition 27, there exists $b < c$ such that $\mathcal{I}_\beta(Y_{1:n}, (-\infty, b]) \xrightarrow{\text{a.s.}} 1$. Since $c > 0$, we may choose this b to be positive (as before). Letting $H = \{x \in \mathbb{R}^k : x^\mathsf{T} v \leq b\}$, we have $\mathcal{I}_\beta(X_{1:n}, H) \xrightarrow{\text{a.s.}} 1$. Also, $0 \in H^\circ$ since $b > 0$.

Now, we must show that R_u intersects $V \cap \partial H$. First, since $z \in R_u$ means $z = au$ for some $a > 0$, and since $z \in \partial G$ means $z^\mathsf{T} v = c > 0$, we find that $u^\mathsf{T} v > 0$ and $z = cu/u^\mathsf{T} v$. Therefore, letting $y = bu/u^\mathsf{T} v$, we have $y \in R_u \cap V \cap \partial H$, since

(i) $b/u^\mathsf{T} v > 0$, and thus $y \in R_u$,

(ii) $y^{\mathsf{T}}v = b$, and thus $y \in \partial H$,

(iii) $0 < b/u^{\mathsf{T}}v < c/u^{\mathsf{T}}v$, and thus y is a (strict) convex combination of $0 \in V$ and $z \in \bar{V}$, hence $y \in V$ (Rockafellar, 1970, 6.1).

■

To prove Proposition 30, we need the following geometrically intuitive facts.

Proposition 29 *Let $V \subset \mathbb{R}^k$ be open and convex, with $0 \in V$. Let H be a closed halfspace such that $0 \in H^\circ$. Let $T = \{x/|x| : x \in V \cap \partial H\}$. Then*

(1) *T is open in $\mathcal{S}$,*

(2) *$T = \{u \in \mathcal{S} : R_u$ intersects $V \cap \partial H\}$, and*

(3) *if $x \in H$, $x \neq 0$, and $x/|x| \in T$, then $x \in V$.*

Proof Write $H = \{x \in \mathbb{R}^k : x^{\mathsf{T}}v \leq b\}$, with $v \in \mathcal{S}$, $b > 0$. Let $\mathcal{S}_+ = \{u \in \mathcal{S} : u^{\mathsf{T}}v > 0\}$. (1) Define $f : \partial H \to \mathcal{S}_+$ by $f(x) = x/|x|$, noting that $0 \notin \partial H$. It is easy to see that f is a homeomorphism. Since V is open in $\mathbb{R}^k$, then $V \cap \partial H$ is open in ∂H. Hence, $T = f(V \cap \partial H)$ is open in $\mathcal{S}_+$, and since $\mathcal{S}_+$ is open in $\mathcal{S}$, then T is also open in $\mathcal{S}$. Items (2) and (3) are easily checked. ■

Proposition 30 *Let $V \subset \mathbb{R}^k$ be open and convex, with $0 \in V$. If $(H_u : u \in \mathcal{S})$ is a collection of closed halfspaces such that for all $u \in \mathcal{S}$,*

(1) *$0 \in H_u^\circ$ and*

(2) *R_u intersects $V \cap \partial H_u$,*

then there exist $u_1, \ldots, u_r \in \mathcal{S}$ (for some $r > 0$) such that the set $U = \bigcap_{i=1}^r H_{u_i}$ is compact and $U \subset V$.

Proof For $u \in \mathcal{S}$, define $T_u = \{x/|x| : x \in V \cap \partial H_u\}$. By part (1) of Proposition 29, T_u is open in $\mathcal{S}$, and by part (2), $u \in T_u$, since R_u intersects $V \cap \partial H_u$. Thus, $(T_u : u \in \mathcal{S})$ is an open cover of $\mathcal{S}$. Since $\mathcal{S}$ is compact, there is a finite subcover: there exist $u_1, \ldots, u_r \in \mathcal{S}$ (for some $r > 0$) such that $\bigcup_{i=1}^r T_{u_i} \supset \mathcal{S}$, and in fact, $\bigcup_{i=1}^r T_{u_i} = \mathcal{S}$. Let $U = \bigcap_{i=1}^r H_{u_i}$. Then U is closed and convex (as an intersection of closed, convex sets). Further, $U \subset V$ since for any $x \in U$, if $x = 0$ then $x \in V$ by assumption, while if $x \neq 0$ then $x/|x| \in T_{u_i}$ for some $i \in \{1, \ldots, r\}$ and $x \in U \subset H_{u_i}$, so $x \in V$ by Proposition 29(3).

In order to show that U is compact, we just need to show it is bounded, since we already know it is closed. Suppose not, and let $x_1, x_2, \ldots \in U \setminus \{0\}$ such that $|x_n| \to \infty$ as $n \to \infty$. Let $v_n = x_n/|x_n|$. Since $\mathcal{S}$ is compact, then (v_n) has a convergent subsequence such that $v_{n_i} \to u$ for some $u \in \mathcal{S}$. Then for any $a > 0$, we have $av_{n_i} \in U$ for all i sufficiently large (since av_{n_i} is a convex combination of $0 \in U$ and $|x_{n_i}|v_{n_i} = x_{n_i} \in U$ whenever $|x_{n_i}| \geq a$). Since $av_{n_i} \to au$, and U is closed, then $au \in U$. Thus, $au \in U$ for all $a > 0$, i.e., $R_u \subset U$.

But $u \in T_{u_j}$ for some $j \in \{1, \ldots, r\}$, so R_u intersects ∂H_{u_j} (by Proposition 29(2)), and thus $au \notin H_{u_j} \supset U$ for all $a > 0$ sufficiently large. This is a contradiction. Therefore, U is bounded. ■

References

C. E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, November 1974.

R. Argiento, A. Guglielmi, and A. Pievatolo. A comparison of nonparametric priors in hierarchical mixture modelling for AFT regression. *Journal of Statistical Planning and Inference*, 139(12):3989–4005, 2009.

D. A. Berry and R. Christensen. Empirical Bayes estimation of a binomial parameter via mixtures of Dirichlet processes. *The Annals of Statistics*, pages 558–568, 1979.

A. Bhattacharya and D. B. Dunson. Nonparametric Bayesian density estimation on manifolds with applications to planar shapes. *Biometrika*, 97(4):851–865, 2010.

D. Blackwell and J. B. MacQueen. Ferguson distributions via Pólya urn schemes. *The Annals of Statistics*, pages 353–355, 1973.

C. A. Bush and S. N. MacEachern. A semiparametric Bayesian model for randomised block designs. *Biometrika*, 83(2):275–285, 1996.

H. Chen, P. L. Morrell, V. E. T. M. Ashworth, M. de la Cruz, and M. T. Clegg. Tracing the geographic origins of major avocado cultivars. *Journal of Heredity*, 100(1):56–65, 2009.

D. B. Dahl. Model-based clustering for expression data via a Dirichlet process mixture model. *Bayesian Inference for Gene Expression and Proteomics*, pages 201–218, 2006.

N. G. De Bruijn. *Asymptotic Methods in Analysis*. North-Holland Publishing Co., Amsterdam, 1970.

P. Diaconis and D. Ylvisaker. Conjugate priors for exponential families. *The Annals of Statistics*, 7(2):269–281, 1979.

M. D. Escobar. *Estimating the Means of Several Normal Populations by Nonparametric Estimation of the Distribution of the Means*. PhD thesis, Yale University, 1988.

M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588, 1995.

M. D. Escobar and M. West. Computing nonparametric hierarchical models. In D. Dey, P. Müller, and D. Sinha, editors, *Practical Nonparametric and Semiparametric Bayesian Statistics*, pages 1–22. Springer-Verlag, New York, 1998.

P. Fearnhead. Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 14(1):11–21, 2004.

T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, pages 209–230, 1973.

T. S. Ferguson. Bayesian density estimation by mixtures of normal distributions. In M. H. Rizvi, J. Rustagi, and D. Siegmund, editors, *Recent Advances in Statistics*, pages 287–302. Academic Press, 1983.

E. B. Fox, E. B. Sudderth, and A. S. Willsky. Hierarchical Dirichlet processes for tracking maneuvering targets. In *10th International Conference on Information Fusion, 2007*, pages 1–8. IEEE, 2007.

S. Ghosal. The Dirichlet process, related priors and posterior asymptotics. In N. L. Hjort, C. Holmes, P. Müller, and S. G. Walker, editors, *Bayesian Nonparametrics*, pages 36–83. Cambridge University Press, 2010.

S. Ghosal and A. Van der Vaart. Posterior convergence rates of Dirichlet mixtures at smooth densities. *The Annals of Statistics*, 35(2):697–723, 2007.

S. Ghosal and A. W. Van der Vaart. Entropies and rates of convergence for maximum likelihood and Bayes estimation for mixtures of normal densities. *The Annals of Statistics*, pages 1233–1263, 2001.

S. Ghosal, J. K. Ghosh, and R. V. Ramamoorthi. Posterior consistency of Dirichlet mixtures in density estimation. *The Annals of Statistics*, 27(1):143–158, 1999.

S. K. Ghosh and S. Ghosal. Semiparametric accelerated failure time models for censored data. *Bayesian Statistics and its Applications*, pages 213–229, 2006.

A. Gnedin and J. Pitman. Exchangeable Gibbs partitions and Stirling triangles. *Journal of Mathematical Sciences*, 138(3):5674–5685, 2006.

E. G. Gonzalez and R. Zardoya. Relative role of life-history traits and historical factors in shaping genetic population structure of sardines (Sardina pilchardus). *BMC Evolutionary Biology*, 7(1):197, 2007.

P. J. Green and S. Richardson. Modeling heterogeneity with and without the Dirichlet process. *Scandinavian Journal of Statistics*, 28(2):355–375, June 2001.

B. Hansen and J. Pitman. Prediction rules for exchangeable sequences related to species sampling. *Statistics & Probability Letters*, 46(3):251–256, 2000.

J. Henna. On estimating of the number of constituents of a finite mixture of continuous distributions. *Annals of the Institute of Statistical Mathematics*, 37(1):235–240, 1985.

J. Henna. Estimation of the number of components of finite mixtures of multivariate distributions. *Annals of the Institute of Statistical Mathematics*, 57(4):655–664, 2005.

J. Hoffmann-Jørgensen. *Probability With a View Towards Statistics*, volume 2. Chapman & Hall, 1994.

J. P. Huelsenbeck and P. Andolfatto. Inference of population structure under a Dirichlet process model. *Genetics*, 175(4):1787–1802, 2007.

H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453), 2001.

H. Ishwaran and L. F. James. Generalized weighted Chinese restaurant processes for species sampling mixture models. *Statistica Sinica*, 13(4):1211–1236, 2003.

H. Ishwaran, L. F. James, and J. Sun. Bayesian model selection in finite mixtures by marginal density decompositions. *Journal of the American Statistical Association*, 96 (456), 2001.

L. F. James. Large sample asymptotics for the two-parameter Poisson–Dirichlet process. In *Pushing the Limits of Contemporary Statistics: Contributions in Honor of Jayanta K. Ghosh*, pages 187–199. Institute of Mathematical Statistics, 2008.

L. F. James, C. E. Priebe, and D. J. Marchette. Consistent estimation of mixture complexity. *The Annals of Statistics*, pages 1281–1296, 2001.

G. H. Jang, J. Lee, and S. Lee. Posterior consistency of species sampling priors. *Statistica Sinica*, 20(2):581, 2010.

Y. Ji, T. Lin, and H. Zha. CDP mixture models for data clustering. In *20th International Conference on Pattern Recognition (ICPR)*, pages 637–640. IEEE, 2010.

C. Keribin. Consistent estimation of the order of mixture models. *Sankhya Ser. A*, 62(1): 49–66, 2000.

S. Khazaei, J. Rousseau, and F. Balabdaoui. Nonparametric Bayesian estimation of densities under monotonicity constraint. *(Preprint)*, 2012.

A. W. Knapp. *Basic Real Analysis*. Birkhäuser, 2005.

S. G. Krantz. *Function Theory of Several Complex Variables*. AMS Chelsea Publishing, Providence, 1992.

W. Kruijer. *Convergence Rates in Nonparametric Bayesian Density Estimation*. PhD thesis, Department of Mathematics, Vrije Universiteit Amsterdam, 2008.

W. Kruijer, J. Rousseau, and A. Van der Vaart. Adaptive Bayesian density estimation with location-scale mixtures. *Electronic Journal of Statistics*, 4:1225–1257, 2010.

N. Lartillot and H. Philippe. A Bayesian mixture model for across-site heterogeneities in the amino-acid replacement process. *Molecular Biology and Evolution*, 21(6):1095–1109, 2004.

A. D. Leaché and M. K. Fujita. Bayesian species delimitation in West African forest geckos (Hemidactylus fasciatus). *Proceedings of the Royal Society B: Biological Sciences*, 277 (1697):3071–3077, 2010.

B. G. Leroux. Consistent estimation of a mixing distribution. *The Annals of Statistics*, 20 (3):1350–1360, 1992.

A. Lijoi, I. Prünster, and S. G. Walker. On consistency of nonparametric normal mixtures for Bayesian density estimation. *Journal of the American Statistical Association*, 100 (472):1292–1296, 2005.

A. Lijoi, R. H. Mena, and I. Prünster. Bayesian nonparametric estimation of the probability of discovering new species. *Biometrika*, 94(4):769–786, 2007.

A. Lijoi, I. Prünster, and S. G. Walker. Bayesian nonparametric estimators derived from conditional Gibbs structures. *The Annals of Applied Probability*, 18(4):1519–1547, 2008.

J. S. Liu. The collapsed Gibbs sampler in Bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89(427): 958–966, 1994.

A. Y. Lo. On a class of Bayesian nonparametric estimates: I. Density estimates. *The Annals of Statistics*, 12(1):351–357, 1984.

E. D. Lorenzen, P. Arctander, and H. R. Siegismund. Regional genetic structuring and evolutionary history of the impala Aepyceros melampus. *Journal of Heredity*, 97(2): 119–132, 2006.

S. N. MacEachern. Estimating normal means with a conjugate style Dirichlet process prior. *Communications in Statistics-Simulation and Computation*, 23(3):727–741, 1994.

S. N. MacEachern. Computational methods for mixture of Dirichlet process models. In *Practical Nonparametric and Semiparametric Bayesian Statistics*, pages 23–43. Springer, 1998.

S. N. MacEachern. Dependent nonparametric processes. In *ASA Proceedings of the Section on Bayesian Statistical Science*, pages 50–55, 1999.

S. N. MacEachern and P. Müller. Estimating mixture of Dirichlet process models. *Journal of Computational and Graphical Statistics*, 7(2):223–238, 1998.

M. Medvedovic and S. Sivaganesan. Bayesian infinite mixture model based clustering of gene expression profiles. *Bioinformatics*, 18(9):1194–1206, 2002.

J. W. Miller and M. T. Harrison. A simple example of Dirichlet process mixture inconsistency for the number of components. In *Advances in Neural Information Processing Systems, Vol. 26*, 2013.

R. M. Neal. Bayesian mixture modeling. In *Maximum Entropy and Bayesian Methods*, pages 197–211. Springer, 1992.

R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

X. L. Nguyen. Convergence of latent mixing measures in finite and infinite mixture models. *The Annals of Statistics*, 41(1):370–400, 2013.

A. Nobile. *Bayesian Analysis of Finite Mixture Distributions*. PhD thesis, Department of Statistics, Carnegie Mellon University, Pittsburgh, PA, 1994.

A. Nobile and A. T. Fearnside. Bayesian finite mixtures with an unknown number of components: The allocation sampler. *Statistics and Computing*, 17(2):147–162, 2007.

A. Onogi, M. Nurimoto, and M. Morita. Characterization of a Bayesian genetic clustering algorithm based on a Dirichlet process prior and comparison among Bayesian clustering methods. *BMC Bioinformatics*, 12(1):263, 2011.

E. Otranto and G. M. Gallo. A nonparametric Bayesian approach to detect the number of regimes in Markov switching models. *Econometric Reviews*, 21(4):477–496, 2002.

D. Pati, D. B. Dunson, and S. T. Tokdar. Posterior consistency in conditional distribution estimation. *Journal of Multivariate Analysis*, 116:456–472, 2013.

J. Pella and M. Masuda. The Gibbs and split-merge sampler for population mixture analysis from genetic data with incomplete baselines. *Canadian Journal of Fisheries and Aquatic Sciences*, 63(3):576–596, 2006.

M. Perman, J. Pitman, and M. Yor. Size-biased sampling of Poisson point processes and excursions. *Probability Theory and Related Fields*, 92(1):21–39, 1992.

D. B. Phillips and A. F. M. Smith. Bayesian model comparison via jump diffusions. In *Markov Chain Monte Carlo in Practice*, pages 215–239. Springer, 1996.

J. Pitman. Some developments of the Blackwell-MacQueen urn scheme. *Lecture Notes-Monograph Series*, pages 245–267, 1996.

J. Pitman. *Combinatorial Stochastic Processes*. Springer–Verlag, Berlin, 2006.

J. Pitman and M. Yor. The two-parameter Poisson–Dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, 25(2):855–900, 1997.

J. K. Pritchard, M. Stephens, and P. Donnelly. Inference of population structure using multilocus genotype data. *Genetics*, 155(2):945–959, 2000.

C. M. Richards, G. M. Volk, A. A. Reilley, A. D. Henk, D. R. Lockwood, P. A. Reeves, and P. L. Forsline. Genetic diversity and population structure in Malus sieversii, a wild progenitor species of domesticated apple. *Tree Genetics & Genomes*, 5(2):339–347, 2009.

S. Richardson and P. J. Green. On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society, Series B*, 59(4):731–792, 1997.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

J. Rousseau and K. Mengersen. Asymptotic behaviour of the posterior distribution in overfitted mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(5):689–710, 2011.

T. Sapatinas. Identifiability of mixtures of power-series distributions and related characterizations. *Annals of the Institute of Statistical Mathematics*, 47(3):447–459, 1995.

C. Scricciolo. Adaptive Bayesian density estimation using Pitman–Yor or normalized inverse-Gaussian process kernel mixtures. *arXiv:1210.8094*, 2012.

M. Stephens. Bayesian analysis of mixture models with an unknown number of components—An alternative to reversible jump methods. *The Annals of Statistics*, 28 (1):40–74, 2000.

Y. Tang and S. Ghosal. Posterior consistency of Dirichlet mixtures for estimating a transition density. *Journal of Statistical Planning and Inference*, 137(6):1711–1726, June 2007.

H. Teicher. Identifiability of finite mixtures. *The Annals of Mathematical Statistics*, 34(4): 1265–1269, 1963.

S. T. Tokdar. Posterior consistency of Dirichlet location-scale mixture of normals in density estimation and regression. *Sankhyā: The Indian Journal of Statistics*, pages 90–110, 2006.

S. G. Walker, A. Lijoi, and I. Prünster. On rates of convergence for posterior distributions in infinite-dimensional models. *The Annals of Statistics*, 35(2):738–746, 2007.

M. West. Hyperparameter estimation in Dirichlet process mixture models. *ISDS Discussion Paper #92-A03, Duke University*, 1992.

M. West, P. Müller, and M. D. Escobar. Hierarchical priors and mixture models, with application in regression and density estimation. In P. Freeman and A. F. Smith, editors, *Aspects of Uncertainty: A Tribute to D.V. Lindley*, pages 363–386. Wiley, 1994.

M.-J. Woo and T. N. Sriram. Robust estimation of mixture complexity. *Journal of the American Statistical Association*, 101(476), 2006.

M.-J. Woo and T. N. Sriram. Robust estimation of mixture complexity for count data. *Computational Statistics and Data Analysis*, 51(9):4379–4392, 2007.

Y. Wu and S. Ghosal. The L_1-consistency of Dirichlet mixtures in multivariate Bayesian density estimation. *Journal of Multivariate Analysis*, 101(10):2411–2419, November 2010.

E. P. Xing, K. A. Sohn, M. I. Jordan, and Y. W. Teh. Bayesian multi-population haplotype inference via a hierarchical Dirichlet process mixture. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 1049–1056, 2006.

Journal of Machine Learning Research 15 (2014) 3371-3399 Submitted 12/13; Revised 6/14; Published 10/14

Active Contextual Policy Search

Alexander Fabisch AFABISCH@INFORMATIK.UNI-BREMEN.DE
Jan Hendrik Metzen JHM@INFORMATIK.UNI-BREMEN.DE
Robotics Research Group, University Bremen
Robert-Hooke-Str. 1, D-28359 Bremen, Germany

Editor: Laurent Orseau

Abstract

We consider the problem of learning skills that are versatilely applicable. One popular approach for learning such skills is contextual policy search in which the individual tasks are represented as context vectors. We are interested in settings in which the agent is able to *actively* select the tasks that it examines during the learning process. We argue that there is a better way than selecting each task equally often because some tasks might be easier to learn at the beginning and the knowledge that the agent can extract from these tasks can be transferred to similar but more difficult tasks. The methods that we propose for addressing the task-selection problem model the learning process as a non-stationary multi-armed bandit problem with custom intrinsic reward heuristics so that the estimated learning progress will be maximized. This approach does neither make any assumptions about the underlying contextual policy search algorithm nor about the policy representation. We present empirical results on an artificial benchmark problem and a ball throwing problem with a simulated Mitsubishi PA-10 robot arm which show that active context selection can improve the learning of skills considerably.

Keywords: reinforcement learning, policy search, movement primitives, active learning, multi-task learning

1. Introduction

Artificial agents like robots are deployed in increasingly complex environments where they have to fulfill a range of different tasks. Hard-coding the full set of behaviors required by an agent in such an environment before deployment becomes increasingly difficult. An alternative approach is to provide the agent with means for learning of novel behavior. By this, the agent might acquire additional behaviors when required. However, learning every behavior from scratch is cumbersome and impractical for real robots, where it is important to learn novel behavior within a small amount of trials. Learning a set of versatilely applicable skills is one way to make an agent *competent* (White, 1959) in an environment, which means that the agent can efficiently learn behaviors for a multitude of tasks imposed on him by building on top of the set of skills.

One means for acquiring reusable skills is reinforcement learning (RL). In RL, a skill is represented by a policy π, which selects actions based on the current state.[1] One popular means for learning such a policy in a robotic setting is *policy search* (Deisenroth et al., 2013).

1. A policy can be stochastic, that is, define a conditional probability distribution over the actions given the state, or be deterministic, that is, a function that maps from states to actions.

In policy search, policies are parameterized by a parameter vector $\boldsymbol{\theta}$ and the objective is to find $\boldsymbol{\theta}$ such that the expected return of the policy $\pi_{\boldsymbol{\theta}}$ is maximized. A problem with this formulation is that it is not easily extended to settings in which different tasks are imposed onto the agent, which result in different returns for the same policy $\pi_{\boldsymbol{\theta}}$. For instance, when an agent tries to throw a ball to different target positions (the tasks), the same policy might obtain very different returns depending on whether the respective target object is hit or not.

To address such multi-task settings, we consider the problem of learning policies for contextual RL problems. That is, we assume that similar tasks are distinguished by context vectors $\boldsymbol{s} \in S \subseteq \mathbb{R}^{n_s}$; that is, contexts are described by n_s-dimensional real vectors. We use the terms task and context mostly interchangeable in this paper. Instead of searching directly for $\boldsymbol{\theta}$ in the space of control policies $\pi_{\boldsymbol{\theta}}$, one can introduce an *upper-level policy* $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$, which is parameterized by $\boldsymbol{\omega}$ and defines a probability distribution over the parameters $\boldsymbol{\theta}$ of the actual control policy. The expected return of an upper-level policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ in a parameterized RL problem is defined as

$$J(\boldsymbol{\omega}) = \int_{\boldsymbol{s}} p(\boldsymbol{s}) \int_{\boldsymbol{\theta}} \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s}) \mathcal{R}_{\boldsymbol{s},\boldsymbol{\theta}} \, \mathrm{d}\boldsymbol{\theta} \, \mathrm{d}\boldsymbol{s},$$

where $\mathcal{R}_{\boldsymbol{s},\boldsymbol{\theta}}$ is the expected return of policy $\pi_{\boldsymbol{\theta}}$ in context $\boldsymbol{s}$ and $p(\boldsymbol{s})$ is the probability density function of the context distribution. Searching for parameters $\boldsymbol{\omega}$ that maximize $J(\boldsymbol{\omega})$ is denoted as *contextual policy search* (Deisenroth et al., 2013). We do not make any further restricting assumptions about the upper-level policy, the control policy or the policy search algorithm in this paper.

Different methods have been proposed for contextual policy search: one class of methods learns first policies $\pi_{\boldsymbol{\theta}}$ for a set of tasks separately and uses thereupon regression algorithms to infer a deterministic function which generalizes the learned policy parameters over the entire context space (da Silva et al., 2012; Metzen et al., 2013). A second class of methods learns an upper-level policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ directly without separating learning in an RL and a regression part (Peters and Schaal, 2007; Kober et al., 2012; Kupcsik et al., 2013).

A restriction of both class of methods is that they typically assume that $p(\boldsymbol{s})$ cannot be controlled by the agent. This is appropriate when the environment or a user imposes tasks onto the agent externally. However, an agent might set its goals by itself or select tasks in which it would like to increase its performance autonomously. For instance, when learning target-oriented throwing (Wirkus et al., 2012), the agent might self-select certain target positions which it cannot hit yet reliably and where it would like to improve its performance. Moreover, there might be certain problem domains, for example grasping a cup in environments with obstacles and many different but similar task configurations, where some configurations might be harder to learn than others. For example, grasping a cup is more difficult when it is surrounded by other cups or when the handle is on the far side. One way to approach such situations is to start learning with easy tasks and progress to more and more complex tasks over time, where knowledge acquired in prior tasks is transferred and reused. Thus, by autonomously selecting learning tasks (contexts), for example based on intrinsic motivation (Barto et al., 2004), an agent might increase its competence in an environment in a self-controlled manner.

In this paper, we consider how an agent can actively select contexts to make the best progress in learning $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$, that is in increasing its competence. We propose considering this *active task-selection problem* as a non-stationary bandit problem and using algorithms like D-UCB (Kocsis and Szepesvári, 2006), which are suited for this setting, as task-selection heuristic. Moreover, we examine different intrinsic motivation heuristics for rewarding the agent for selecting contexts during the learning process which are considered to increase the learning progress, that is, to reduce the number of trials that are required to master a given contextual problem. This is important on real robots where the number of trials required for learning a behavior needs to be as small as possible to reduce wear and tear of the robot.

The paper is structured as follows: in Section 2, we present related work in the fields of skill learning, active learning, and multi-armed bandits. Thereupon, we formalize the problem and present the proposed method and the intrinsic reward heuristics for active contextual policy search in Section 3 and demonstrate the underlying model of the contextual learning process in Section 4. We compare the heuristics empirically on two benchmarks, the first being a toy problem in which advantages and disadvantages of different heuristics are systematically studied (see Section 5.1) and the second being a robotic control problem based on a simulated Mitsubishi PA-10 robot. In this setting, an agent learns to throw balls at targets on the ground with two different reward functions which results in the "grid problem" and the "dartboard problem" (see Section 5.2). We conclude and provide an outlook in Section 6.

2. Related Work

Our work is closely connected to two different fields of machine learning: learning general and broadly applicable skills with reinforcement learning or imitation learning and active learning for task selection. We summarize the related work of both fields in this section. In addition, we will briefly introduce the non-stationary multi-armed bandit problem which is related to our proposed method.

2.1 Skill Learning and Skill Transfer

In our setup, low-level policies are typically represented by dynamical movement primitives (DMPs; Ijspeert et al., 2013), although the proposed approaches are in no way restricted to this setting. DMPs encode arbitrarily shapeable, goal-directed trajectories. There are different variants of DMPs, which have in common that they use an internal time variable z (called phase) which replaces explicit timing and allows arbitrary temporal scaling of the movement. Furthermore, the *transformation system* of a DMP is a spring-damper system that generates a goal-directed movement which is controlled by the phase variable z and modified by a *forcing term* f. The shape can be adjusted by learning the weights w_i of the forcing term through imitation or reinforcement learning. We use a variant of DMPs that has been developed by Mülling et al. (2011, 2013), which allows additionally specifying a target velocity at the end of the movement. In summary, we use a low-level policy

$$\boldsymbol{x}_{t+1} = \pi_{\boldsymbol{v},\boldsymbol{w}}(\boldsymbol{x}_t, t), \quad \boldsymbol{v} = (\boldsymbol{x}_0, \boldsymbol{g}, \dot{\boldsymbol{g}}, \tau)\,,$$

where $\boldsymbol{x}_t$ is the state (position, velocity, and acceleration) at time t, $\boldsymbol{w}$ are the weights of the forcing term and $\boldsymbol{v}$ are the following meta-parameters: $\boldsymbol{x}_0$ is the initial state, $\boldsymbol{g}$ is the final state, $\dot{\boldsymbol{g}}$ the desired final velocity, and τ is the duration of the movement. Note that this kind of policy defines a state trajectory and thus requires a low-level controller that generates the appropriate actions such that the state transition from $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t+1}$ is performed, see Peters et al. (2012) for a discussion of this in a robotic setting.

The problem of learning broadly applicable skills has been approached from different perspectives. DMPs themselves have been designed to generalize over some meta-parameters such as the duration of the movement, the start position, and the final position. Additionally, some DMP variants are able to generalize over velocities at the end of the movement (Mülling et al., 2011, 2013). However, designing skills that generalize over more complex task parameterization is not straightforward. This is why machine learning techniques from the fields of imitation learning and reinforcement learning have been used to automatically infer upper-level policies.

Ude et al. (2010) used a form of self-imitation to generalize the task of throwing a ball at a desired target position. First, a number of throws with different policy parameterizations are generated and the final ball position is measured. That is, DMPs with different values of τ, $\boldsymbol{g}$, $\boldsymbol{w}$, and release times of the ball have been executed. Then, a mixture of locally weighted regression (Cleveland and Devlin, 1988) and Gaussian process regression (Rasmussen and Williams, 2005) has been used to find mappings from the position that has been hit on the ground to the corresponding values of the meta-parameters that generated the throw. Thus, the problem has been reduced to a regression problem. The same approach has been used for reaching tasks and drumming. Kronander et al. (2011) proposed a similar method to play mini-golf. As underlying policy representation, stable estimators of dynamical systems (Khansari-Zadeh and Billard, 2011) have been used and another set of meta-parameters is learned with Gaussian process regression and Gaussian mixture regression. Both methods can be regarded as generalizing imitation learning algorithms.

The idea of da Silva et al. (2012) was to use the reinforcement learning algorithm *policy learning by weighting exploration with the returns* (PoWER; Kober and Peters, 2011) to generate a training set for the regression algorithm, which consists of nearly optimal parameters $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots\}$ for the low-level policies in different contexts $\{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots\}$. Thereupon, a regression algorithm is used to infer a deterministic policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$, the so-called *parameterized skill*, which generalizes over the entire context space. The authors considered the problem of dart throwing and observed that this domain has the additional challenge of discontinuities in the mapping from task parameters to meta-parameters of the policy. Such discontinuities are less likely when the examples have been generated by a human operator but appear especially if there are multiple solutions for a task with similar quality and if there are no constraints for the meta-parameters during reinforcement learning. To address this problem, the authors use Isomap (Tenenbaum et al., 2000) to extract manifolds in the meta-parameter space and learn different support vector regression models for these manifolds. An extension of parameterized skill denoted as *skill templates* has been proposed by Metzen et al. (2013), which learns not only the upper-level policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ from the experience but also an estimate of the uncertainty in this generalization, which can be useful for subsequent adaptation of the policy.

An advantage of these approaches is that the upper level policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$, which maps from contexts to parameters $\boldsymbol{\theta}$ of the low-level policy can be an arbitrary function and hence, sophisticated regression methods can be used for generalization, which can deal for example with discontinuities. On the other hand, a functional relationship between context and control-policy parameters $\boldsymbol{\theta}$ as in a deterministic policy might not always be adequate; for instance, there might be multiple optima in the space of $\boldsymbol{\theta}$ for a single context (for example, think of fore- and backhand strokes). In addition, the approach requires to learn close-to-optimal parameters for the control policy in a context to generate just one training example for the regression algorithms. Consequently, all other experience collected during learning these close-to-optimal parameters is not being used for generalizing over the context space.

Several reinforcement-learning algorithms have been proposed that learn the upper-level policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ without separating learning in an RL and a regression part (Peters and Schaal, 2007; Kober et al., 2012; Kupcsik et al., 2013). These approaches allow transferring experience between different contexts even if the behavior in these contexts is suboptimal. Thus, these approaches can make use of all collected experience and are thus typically more sample-efficient, that is, they can learn a close-to-optimal upper-level policy within less trials. Furthermore, the upper-level policy $\pi_{\boldsymbol{\omega}}$ can be stochastic, which means it can be implemented by a conditional probability distribution. This has the advantage that the agent's explorative behavior is explicitly modeled and can be adapted by the learning algorithm on the upper level. Furthermore, multiple optima in the space of the low-level policies can be represented by a multi-modal probability distribution on the upper level (Daniel et al., 2012).

Peters and Schaal (2007) applied reinforcement learning to learn contextual policies directly with reward weighted regression. Here, a stochastic upper-level policy similar to the mapping from task parameters to policy meta-parameters in the regression setting is learned directly. Kober et al. (2012) extended this to non-parametric policies in an approach denoted as cost-regularized kernel regression (CrKR). CrKR has been used to learn throwing movements as well as table tennis. Another reinforcement learning algorithm that can be used to learn the upper-level policy is contextual relative entropy policy search (C-REPS, Kupcsik et al., 2013). REPS is an information-theoretic approach to policy search, which aims in each iteration at maximizing the expected return of the new policy while bounding the Kullback-Leibler divergence between the old and new policy. Bounding this divergence enforces that the new policy is not too different from the old policy as large changes of the policy could for instance be dangerous in a robotic setting. C-REPS is an extension of this approach to the contextual policy search setting. We refer to Appendix A for more details on C-REPS.

The representation of the upper-level policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ in these approaches is typically restricted: in CrKR, the upper level policy is modeled as a Gaussian process with the outputs being assumed to be independent (Deisenroth et al., 2013). C-REPS allows any policy which can be learned by weighted maximum likelihood. A frequently used variant is $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s}) = \mathcal{N}\left(\boldsymbol{\theta}|\boldsymbol{W}^T\varphi(\boldsymbol{s}), \boldsymbol{\Sigma}\right)$, where $\varphi(\boldsymbol{s})$ contains the linear and quadratic terms of the context vector $\boldsymbol{s}$, $\boldsymbol{W}$ is a weight matrix, and $\boldsymbol{\Sigma}$ the covariance of the stochastic policy.

A common assumption in contextual policy search is that the selection of the task in which the next trial will be performed is either not under the control of the agent or that

it does not matter in which order and in which frequency tasks are selected. In this paper, we study strategies for actively selecting the next task by means of active learning.

2.2 Active Learning and Artificial Curiosity

In machine learning it is typically desirable to require as few training examples as possible because it is often costly to acquire examples. For instance, in supervised learning it can be expensive to label data. Similarly, in reinforcement learning domains such as robotics, performing a trial is typically very costly. Hence, in general one would like to perform only those trials that maximize the learning progress. A research field that deals with selecting data or tasks from which one can learn the most is *active learning* (Settles, 2010). The goal of active learning is to label only data or examine only tasks that promise the greatest learning progress, that is: the most informative instances. Different query strategies to find the most informative instance are discussed by Settles (2010).

The idea to actively select tasks that accelerate the learning progress is related to *curriculum learning*: Bengio et al. (2009) and Gulcehre and Bengio (2013) assume that learning simple concepts first helps to learn more complex concepts that build upon those previously learned simple concepts in the context of supervised learning. We also try to exploit this hypothesis in our empirical evaluation. Another related work has been published by Ruvolo and Eaton (2013) in the context of lifelong multi-task learning, which compares several heuristics for actively selecting the task that will be learned. Among these heuristic are the following: select the task that maximizes the expected information gain (information maximization heuristic) and select the task on which the current model performs worst (diversity heuristic). Once a task is selected, all labels of data instances of this task are revealed to the agent.

It is not straightforward to transfer this approach to the reinforcement learning setting. A problem that occurs in reinforcement learning is that one does not get the correct solution, that is the optimal policy, of a queried task directly. Instead, one receives only an immediate reward and needs to optimize the solution by trial-and-error in order to maximize the long-term reward, which requires to select the same training tasks multiple times consecutively until a close-to-optimal policy is learned. This approach is followed by da Silva et al. (2014), where the parameterized skill approach discussed in Section 2.1 is extended to an active learning setting. For this, the authors propose a novel criterion for skill selection. In this criterion, the skill performance is modeled using Gaussian process regression with a spatiotemporal kernel which addresses the inherent non-stationarity of tracking the skill performance. Based on this estimate of the skill performance, the next task is chosen such that the maximum expected improvement in skill performance would be obtained if the outcome of learning this task is assumed to be an optimistic upper bound. A drawback of this approach is that it may not be the optimal (most sample-efficient) task-selection strategy to stick for many trials to the same task until a close-to-optimal policy for this task is learned. In this paper, we address how active task selection can be performed if a novel task is chosen after each trial which involves that typically no close-to-optimal policy has been learned in the last task.

A field related to active learning with applications in reinforcement learning is artificial curiosity (Oudeyer and Kaplan, 2004; Gottlieb et al., 2013). In particular, an architecture

called *self-adaptive goal generation - robust intelligent adaptive curiosity* (SAGG-RIAC) has been derived from the ideas of artificial curiosity by Baranes and Oudeyer (2013) and has been applied to learning of contextual problems. The goal self-generation and self-selection of SAGG-RIAC divides the context space into rectangular regions. These are split once the number of contexts that have been explored in these regions exceeds a threshold. For each region R_i, one can compute an interest value based on the derivative of the competence in that region

$$\text{interest}_i = \frac{1}{\zeta} \left| \left(\sum_{j=|R_i|-\zeta}^{|R_i|-\frac{\zeta}{2}} r(\boldsymbol{s}_j, \boldsymbol{\theta}_j) \right) - \left(\sum_{j=|R_i|-\frac{\zeta}{2}}^{|R_i|} r(\boldsymbol{s}_j, \boldsymbol{\theta}_j) \right) \right|,$$

where $|R_i|$ is the number of contexts that have been explored in region R_i, ζ is the size of a sliding window, and $r(\boldsymbol{s}_j, \boldsymbol{\theta}_j)$ is the return of a low-level policy with parameters $\boldsymbol{\theta}_j$ in context $\boldsymbol{s}_j$ (not to be confused with the expected return $\mathcal{R}_{\boldsymbol{s},\boldsymbol{\theta}} = \mathbb{E}\{r(\boldsymbol{s}, \boldsymbol{\theta})\}$). Essentially, this means regions with greater differences between recent and previous returns are considered to be more interesting. Hence, on the one hand SAGG-RIAC focuses on regions where the reward increases considerably over time. On the other hand, it also favors regions where the reward decreases. The intuition for this is that such a decrease might be due to a change in the environment in this region and, hence, it would make sense to explore this region more strongly. SAGG-RIAC selects goals either randomly from the whole context space or from a random region, where the probability of each region corresponds to its interest. For a selected region, the goal is either drawn from a uniform random distribution or from the vicinity of the context with the lowest previous return. Each of the three cases is again selected randomly with fixed probabilities.

One advantage of SAGG-RIAC is that it naturally copes with continuous context spaces. The method will most of the time concentrate on regions where it expects a high learning progress but does not converge to a single region of the context space. As an alternative to the SAGG-RIAC heuristic, we present an approach that is based on heuristic estimates of the learning progress and multi-armed bandit algorithms, which more naturally trades off exploration and exploitation of the noisy estimate of the learning progress. The corresponding algorithms will be discussed in the next section.

2.3 Non-Stationary Multi-Armed Bandit Problems

Multi-armed bandit problems (MABP: Robbins, 1952; Bubeck and Cesa-Bianchi, 2012) are situated between supervised and reinforcement learning. A MABP can be regarded as a one-step or one-state Markov decision process, in which an agent has to select one out of K possible actions and obtains a reward afterwards that is typically assumed to be drawn independent and identically distributed for each action. The agent tries to minimize the regret, which is defined as the expected difference between its accumulated reward and the reward that would have been accumulated with the optimal but unknown stationary policy. One popular algorithm in this setting is upper confidence bound (UCB: Agrawal, 1995b). UCB learns a deterministic policy that always selects the action with the maximum upper bound on the confidence interval of the expected reward. This upper bound is constructed from the past rewards for the action and is based on their empirical mean and a padding function. The padding function summarizes the uncertainty in the estimate of the expected

reward. Since UCB always selects the action with the maximum upper bound, this results in choosing actions where the reward is either very uncertain (large padding) or expected to be high (large empirical mean). By this, UCB inherently trades off exploration and exploitation.

One crucial assumption of most algorithms like UCB is that the reward distributions do not change over time. This assumption will not be satisfied in our settings. There exist bandit algorithms that are designed to deal with non-stationary MABPs, in which the reward distributions might change over time, either abruptly, leading to sliding window UCB (Garivier and Moulines, 2011), or slowly but continuously, leading to discounted UCB (Kocsis and Szepesvári, 2006). In the next section, we show how we can use these kind of algorithms for active task selection by modeling the task-selection problem as a non-stationary multi-armed bandit problem.

3. Proposed Methods

In this paper, we consider contextual policy search problems. Thus, we try to learn $\boldsymbol{\omega}$ which maximizes $J(\boldsymbol{\omega}) = \int_{\boldsymbol{s}} p(\boldsymbol{s}) \int_{\boldsymbol{\theta}} \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s}) \mathcal{R}_{\boldsymbol{s},\boldsymbol{\theta}} \, \mathrm{d}\boldsymbol{\theta} \, \mathrm{d}\boldsymbol{s}$ by performing rollouts of the control policy with parameters $\boldsymbol{\theta}$ in contexts $\boldsymbol{s}$. However, in contrast to prior work, we consider the context distribution during learning not to be given by the environment but to be under the agent's control. While this assumption might not apply to all contextual policy search problems, there are sufficiently many settings, like for instance ball throwing with self-selecting goal positions, to make studying this setting worthwhile. Note that the context distribution $p(\boldsymbol{s})$ within $J(\boldsymbol{\omega})$ remains fixed during evaluation and cannot be modified by the agent.

We introduce the problem of active context selection and its objective, namely to maximize learning progress, formally in Section 3.1. We propose different intrinsic reward functions which can be considered as proxies for the learning progress objective (see Section 3.2). In Section 3.3, we discuss how a context selection policy can be learned using multi-armed bandit algorithms.

3.1 Active Context Selection

Formally, we introduce a context selection policy π_β, which selects the context in which the next trial will be performed. π_β aims at optimizing the expected *learning progress* of a contextual policy search method like C-REPS and, hence, minimizing the required number of episodes to reach a desired level of performance. In contrast, the control policy $\pi_{\boldsymbol{\theta}}$ and the upper-level policy $\pi_{\boldsymbol{\omega}}$ are learned to maximize $J(\boldsymbol{\omega})$ directly. An illustration of the components of contextual policy search is given in Figure 1.

We define the learning progress at time t when performing a trial in context $\boldsymbol{s}$, with the control policy parameters $\boldsymbol{\theta}$ selected according to $\pi_{\boldsymbol{\omega}}$, as $\Delta_{\boldsymbol{s}}(t) = J(\boldsymbol{\omega}_{t+k}) - J(\boldsymbol{\omega}_t)$, where $\boldsymbol{\omega}_t$ are the parameters learned by contextual policy search at time t. Note that contextual policy search methods like C-REPS typically do not update $\boldsymbol{\omega}$ after every rollout but only after a certain number of rollouts k. Hence, it is more appropriate to define the learning progress over the window k than between successive values of $J(\boldsymbol{\omega}_t)$.

The learning task on the task-selection level can now be framed as finding π_β such that π_β selects contexts that maximize the expected learning progress $\mathbb{E}_{\pi_{\boldsymbol{\omega}}}\{\Delta_{\boldsymbol{s}}(t)\}$. The learning progress $\Delta_{\boldsymbol{s}}(t)$ is stochastic because the rollouts and the upper-level policy $\pi_{\boldsymbol{\omega}}$ are

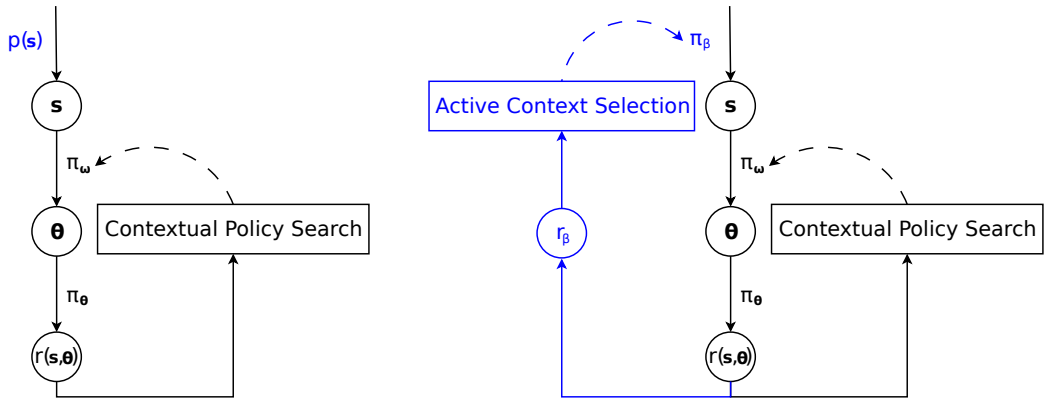


Figure 1: Contextual policy search (left): A context vector $\boldsymbol{s}$ is given by the environment according to some fixed distribution $p(\boldsymbol{s})$, the upper-level policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ generates the parameters of the control policy for $\boldsymbol{s}$, the control policy is executed in the environment and a reward $r(\boldsymbol{s},\boldsymbol{\theta})$ is obtained. A contextual policy search component updates the upper-level policy based on the reward with the goal to maximize $J(\boldsymbol{\omega})$. Active contextual policy search (right): The context vector $\boldsymbol{s}$ is selected by the context selection policy π_β which will be adjusted by an active context selection component based on the intrinsic reward r_β. r_β is an estimate of the learning progress based on $r(\boldsymbol{s},\boldsymbol{\theta})$. Differing parts are marked in blue.

stochastic, hence the expectation. Furthermore, the learning progress is also non-stationary since it depends on the quality of $\boldsymbol{\omega}_t$: if $J(\boldsymbol{\omega}_t)$ is already close to the optimum, the typical learning progress is smaller than the learning progress at the beginning of learning when $\boldsymbol{\omega}_t$ is usually far from optimal.

We restrict ourselves to cases in which π_β can only select among a finite number K of predetermined contexts $\{\boldsymbol{s}_1,\ldots,\boldsymbol{s}_K\}$ during learning. These K predetermined contexts are, however, elements of a continuous context space S over which $\pi_{\boldsymbol{\omega}}$ shall generalize and where $J(\boldsymbol{\omega})$ is evaluated. The restriction to a discrete set of training contexts allows to apply well-established multi-armed bandit algorithms. Choosing the discrete set of training contexts can be based either on domain knowledge or on simple heuristics: for instance, for low-dimensional context spaces, an equally spaced grid over the context space can be used for training. In high-dimensional context spaces this would become infeasible due to the course of dimensionality. Hence, in this situation sampling the set of training contexts from a uniform random distribution over the continuous context space is more viable.

3.2 Intrinsic Reward Functions

Since $\mathbb{E}_{\pi_{\boldsymbol{\omega}}}\{\Delta_{\boldsymbol{s}}(t)\}$ is unknown to the agent and difficult to estimate because of its non-stationarity, we propose several heuristic but easily computable reward functions r_β which reward the agent for certain proxy criteria. These proxy reward functions can be considered as means for intrinsic motivation of an agent (Barto et al., 2004), which drives it to engage in activity that increases its competence in task solving on a larger time scale. In the empirical

experiments, we evaluate which r_β is a good proxy for the actual expected learning progress. Note that the heuristics that we propose are not exactly comparable to any query strategy from supervised learning since the learning process in reinforcement learning in a specific task is iterative unlike in typical supervised learning settings.

In a generalized form, we can write the proposed proxies of the learning progress as

$$r_\beta = f(r(\boldsymbol{s}_t, \boldsymbol{\theta}_t) - \hat{b}_{\boldsymbol{s}_t}),$$

where $r(\boldsymbol{s}_t, \boldsymbol{\theta}_t)$ is the return obtained in episode t with policy $\pi_{\boldsymbol{\theta}_t}$ in context $\boldsymbol{s}_t$, $\hat{b}_{\boldsymbol{s}_t}$ is a baseline term, and f is an operator. Note that the bandit algorithm addresses the stochasticity in $r(\boldsymbol{s}_t, \boldsymbol{\theta}_t)$ and, hence, the heuristics usually need not account for this stochasticity themselves.

3.2.1 Best-Reward Heuristic

This heuristic directly uses the reward obtained by the control policy $\pi_{\boldsymbol{\theta}}$ in task $\boldsymbol{s}$ in rollout t as the proxy for the learning progress, that is $r_\beta = r(\boldsymbol{s}, \boldsymbol{\theta}_t)$. Thus $\hat{b}_{\boldsymbol{s}_t} = 0$ and the operator f is simply the identity. This heuristic corresponds to assuming that the agent makes the most progress when it focuses on improving its performance in tasks in which it is already performing well. The idea behind this heuristic is that we should first learn easy tasks perfectly because this can help us to learn similar but more difficult tasks later on.

A potential problem of this heuristic are settings in which high reward does not correspond to easy tasks, for instance when the maximum achievable reward differs in different context. Additionally, the best-reward heuristic requires that knowledge can be transferred very well between contexts. Otherwise it converges to the context with the highest rewards.

3.2.2 Diversity Heuristic

Quite opposite to the best-reward heuristic, this heuristic encourages to select tasks in which the agent receives the worst reward. For this, the negative actual reward $r_\beta = -r(\boldsymbol{s}, \boldsymbol{\theta}_t)$ is used. Thus $\hat{b}_{\boldsymbol{s}_t} = 0$ and the operator f is simply $f(x) = -x$. The intuition for this heuristic is that one should focus on the hardest tasks in which the current performance is worst since in these tasks the potential for large improvements is high. This heuristic bears similarities to the heuristic proposed by Ruvolo and Eaton (2013) for supervised learning and is hence called "diversity" heuristic.

Similar to the best-reward heuristic, this heuristic might have problems in settings in which the maximum achievable reward differs in different context and thus, the obtained rewards might not be comparable. Another disadvantage appears in cases with unlearnable contexts. The diversity heuristic would then focus on these unlearnable contexts.

3.2.3 1-step Progress Heuristic

This heuristic uses the difference of the last two rewards obtained in the respective context as proxy for the actual learning progress, that is, $r_\beta = r(\boldsymbol{s}, \boldsymbol{\theta}_t) - r(\boldsymbol{s}, \boldsymbol{\theta}_{\bar{t}})$, where $\bar{t}$ is the index of the previous rollout in context $\boldsymbol{s}$. Thus the baseline is simply the last obtained reward $\hat{b}_{\boldsymbol{s}_t} = r(\boldsymbol{s}, \boldsymbol{\theta}_{\bar{t}})$ and the operator f is the identity. The heuristic can be seen as the most straight-forward proxy for the learning progress as it replaces effectively the integration over

Algorithm 1 Discounted Upper-Confidence Bound (D-UCB) (Kocsis and Szepesvári, 2006)

Require: K: number of tasks; γ: discounting factor; $\xi > 0$: some parameter controlling the strength of padding; t: number of rollouts; $i_1, \ldots, i_t$: previously selected tasks; $r_1, \ldots, r_t$: previous rewards

if $t < K$ **then**
  return t # Sample each task at least once
else
  for $i \in \{1, \ldots, K\}$ **do**
    $n_i \leftarrow \sum_{t_j=1}^{t} \gamma^{t-t_j} \mathbb{1}_{\{i_{t_j}=i\}}$ # Discounted number of rollouts in task i
  end for
  $n \leftarrow \sum_{i=1}^{K} n_i$ # Discounted total number of rollouts
  for $i \in \{1, \ldots, K\}$ **do**
    $\overline{r}_i \leftarrow \frac{1}{n_i} \sum_{t_j=1}^{t} \gamma^{t-t_j} r_{t_j} \mathbb{1}_{\{i_{t_j}=i\}}$ # Discounted mean reward in task i
    $c_i \leftarrow 2B\sqrt{\frac{\xi \log n}{n_i}}$ # Padding function for task i
  end for
  return $\arg\max_{i \in \{1,\ldots,K\}} \overline{r}_i + c_i$ # Select task in which discounted UCB is maximal
end if

$\boldsymbol{s}$ and $\boldsymbol{\theta}$ in $J(\omega)$ by single samples and uses the reward sample $r(\boldsymbol{s}, \boldsymbol{\theta}_t)$ as estimate for $\mathcal{R}_{\boldsymbol{s},\boldsymbol{\theta}_t}$. As it is based solely on differences of rewards rather than absolute values, it should be able to cope better than the best-reward and diversity heuristic with situations, in which the maximum achievable reward differs in different contexts.

A drawback of the 1-step progress heuristic is that it generates reward signals r_β with high variance. Variance in r_β is inevitable because of the stochasticity of both the environment and the upper-level policy $\pi_{\boldsymbol{\omega}}$. However, since the baseline of this heuristic is the last obtained reward, this baseline has also a high variance, which need not be the case.

3.2.4 Monotonic Progress Heuristic

Although bandit algorithms account for the stochasticity in the 1-step progress heuristic, it might be useful to reduce already the variance of the intrinsic reward. We can use more stable baselines such as the maximum reward of all previous rollouts in the context $\boldsymbol{s}$, that is: $\hat{b}_{\boldsymbol{s}} = \max_{t:\boldsymbol{s}_t=\boldsymbol{s}} r(\boldsymbol{s}, \boldsymbol{\theta}_t)$. Furthermore, since the learning progress is typically monotonically increasing, using the operator $f(x) = \max(0, x)$ to avoid negative r_β appears to be reasonable. The resulting heuristic is denoted as "monotonic progress heuristic" and has the form $r_\beta = \max(0, r(\boldsymbol{s}, \boldsymbol{\theta}_t) - \max_{t:\boldsymbol{s}_t=\boldsymbol{s}} r(\boldsymbol{s}, \boldsymbol{\theta}_t))$.

The heuristic considers the learning progress to be monotonic and strictly positive. In comparison to the 1-step progress heuristic, the intrinsic reward r_β will be more often zero and its baseline has less variance since unsuccessful explorative rollouts have no effect on its value.

3.3 Learning Context Selection Policies

Since the reward r_β is stochastic, we propose to use a learning algorithm for determining π_β. As we consider only a finite number of predetermined contexts, the problem of learning π_β can be framed as a MABP, where the contexts correspond to the "arms" and r_β to the bandit's reward. Due to the non-stationarity of r_β, standard UCB-like algorithms are not sufficient as they expect the rewards to be drawn from a non-changing probability distribution. In contrast, we propose to use D-UCB (see Algorithm 1; Kocsis and Szepesvári, 2006) for active context selection since it explicitly addresses changing reward distributions. For this, D-UCB estimates the instantaneous expected reward by a weighted average of past rewards where higher weight is given to recent rewards. More specifically, a discount factor $\gamma \leq 1$ is introduced and the reward that has been obtained at time step t_j is weighted with the factor γ^{t-t_j} at time t. The central idea of D-UCB is that the discounting can compensate for continuously but slowly changing reward distributions.

The upper confidence bound of the D-UCB algorithm for arm i takes the form $\overline{r}_i + c_i$ where $\overline{r}_i = \frac{1}{n_i} \sum_{t_j=1}^{t} \gamma^{t-t_j} r_{t_j} \mathbb{1}_{\{i_{t_j}=i\}}$ is the discounted mean and $c_i = 2B\sqrt{\frac{\xi \log n}{n_i}}$ is the padding function which controls the width of the confidence intervals. In this formulas, n_i corresponds to the discounted number of draws of arm i and n to the total discounted number of draws (see Algorithm 1). Furthermore, r_{t_j} is the reward obtained in the t_j-th rollout and i_{t_j} is the arm played in this rollout. $\mathbb{1}_{\{i_{t_j}=i\}}$ is the Kronecker delta which is one if the equality holds and zero otherwise. B and ξ are parameters of the algorithm which control the width of the confidence intervals, where B is an upper bound on the rewards and ξ needs to be chosen appropriately.

4. Model of the Contextual Learning Problem

In this section, we show how contextual policy search algorithms can benefit from active context selection by means of a simple artificial model of the contextual learning problem. The model abstracts away the contextual policy search which is possible because our approach treats it as a black box (see Figure 1).

We assume that the context space $S = \{0, 1, \ldots, 9\}$ is discrete and associated to each context $\boldsymbol{s}$ is a hidden value $l_{\boldsymbol{s}} \in [0, 100]$ that indicates the agent's competence in $\boldsymbol{s}$, that is, how well $\boldsymbol{s}$ has been learned. Large values of $l_{\boldsymbol{s}}$ simulate that the current policy parameters in context $\boldsymbol{s}$ are close to the optimal policy parameters $\boldsymbol{\theta}^*(\boldsymbol{s})$. The true reward in context $\boldsymbol{s}$, which is given by

$$r(\boldsymbol{s}) = (1 + \exp(-0.1 l_{\boldsymbol{s}} + 4))^{-1} + b_{\boldsymbol{s}},$$

depends directly on $l_{\boldsymbol{s}}$. $r(\boldsymbol{s})$ corresponds to a scaled and shifted logistic function so that $r(\boldsymbol{s}) \approx b_{\boldsymbol{s}}$ if $l_{\boldsymbol{s}} \approx 0$ and $r(\boldsymbol{s}) \approx 1 + b_{\boldsymbol{s}}$ if $l_{\boldsymbol{s}} \geq 100$. In real learning problems, different tasks often have a different maximum reward. To simulate this, we artificially create a true reward baseline $b_{\boldsymbol{s}}$ for each context. The baseline is randomly sampled from a normal distribution with zero mean and standard deviation σ_b. The true reward is not observed directly. Instead, we add Gaussian noise with zero mean and standard deviation σ_r to simulate trial and error of a learning agent.

We assume that each context has an intrinsic complexity which controls how much the agent learns in a single trial in this context. This complexity can change abruptly between

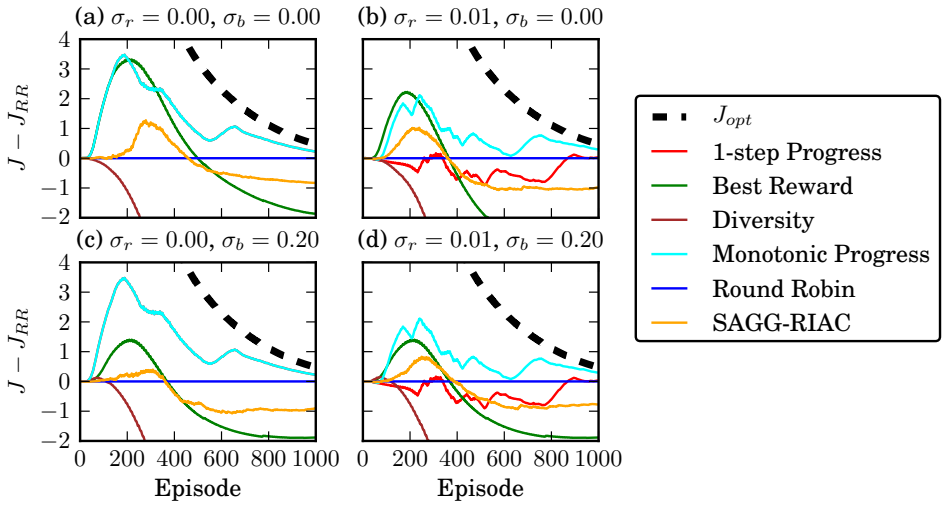


Figure 2: Relative learning curves of several active context selection methods. The performance of the round robin context selection baseline (J_{RR}) is subtracted from all learning curves. The values of the standard deviation for reward measurements σ_r and of the standard deviation for the baseline σ_b have been varied. After 1000 episodes the monotonic progress heuristic has approximately reached the upper bound of the performance J_{opt}.

neighboring contexts even in continuous domains. This model is motivated for instance by reaching tasks, in which it might happen that a slight modification of the goal position requires that the agent needs to avoid an obstacle that blocks the direct path. As a result, the slightly different context corresponding to the blocked goal would have a significantly worse expected learning progress than its neighbors and its solution cannot be transferred well. We model this behavior of learning progress and skill transfer by assigning a different learning progress factor $w_{\boldsymbol{s}} \in \{0.1^2, 0.2^2, \ldots, 1^2\}$ to each context randomly, where large $w_{\boldsymbol{s}}$ corresponds to a larger improvement of competence after one additional trial in $\boldsymbol{s}$.

Each time, the reward of a context $\boldsymbol{s}_t$ will be queried, the values $l_{\boldsymbol{s}}$ of each contexts $\boldsymbol{s}$ will be updated to simulate the learning progress according to the update rule

$$l_{\boldsymbol{s}}^{(t)} = l_{\boldsymbol{s}}^{(t-1)} + w_{\boldsymbol{s}_t} \cdot 0.5^{|\boldsymbol{s}-\boldsymbol{s}_t|},$$

which models that experience obtained in one context generalizes to other contexts based on their similarity (measured here using the euclidean distance). Contexts which are easier learnable (large $w_{\boldsymbol{s}}$) lead to higher overall learning progress at the beginning. However, it does not make sense to focus at the context with maximum w_s indefinitely because $r(s)$ saturates once $l_s \geq 100$. Hence, the estimate of the learning progress has to be adaptive. Since we only have a discrete set of contexts, we can exactly compute $J = \sum_{\boldsymbol{s}} r(\boldsymbol{s})$ and the upper bound of J is $J_{opt} = 10 + \sum_{\boldsymbol{s}} b_{\boldsymbol{s}}$.

In addition to our proposed methods, we examine the context generation and selection method from SAGG-RIAC and context selection in a fixed order (round robin). In order to

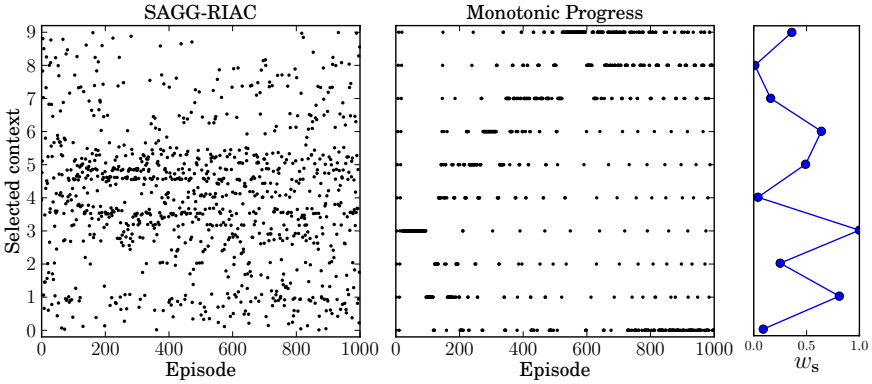


Figure 3: Selected contexts per episode with SAGG-RIAC and monotonic progress heuristic and D-UCB (with $\sigma_r = 0$). The learning progress factor $w_{\boldsymbol{s}}$ of each context is displayed on the right side.

show different properties of the active context selection methods, we display the number of episodes versus the relative J in comparison to round robin selection in Figure 2. We have used the parameters $\gamma = 0.95$ and $\xi = 10^{-8}$ in all heuristics that are based on D-UCB. For the diversity and the best-reward heuristics we have used $B = 1$ and for all others $B = 0.25$. For SAGG-RIAC, we set the maximum size of samples per region before we split the region to 8 and the window size which is used to compute the interest value is 10.

We can see that despite different learning progress factors, round robin context selection is a good baseline. The best-reward heuristic that focuses on the best learnable context is a good heuristic at the beginning. However, when the best context approaches the optimum reward, the learning progress decreases and approaches zero. At this time it would be better to switch to a context in which greater learning progress can be achieved. This will be even more significant for larger context spaces because the transferability of knowledge decreases with the size of the context space and the complexity of the different contexts. In addition, an artificial baseline for each reward (see Figure 2 (c) and (d)) leads to severe degradation because the context with maximum $r(\boldsymbol{s})$ is not necessarily the context with maximum learning progress. The diversity heuristic does not work well either. This is because the differences of the expected learning progress are too large between contexts and it will select the worst learnable context.

The 1-step progress heuristic and monotonic progress heuristic essentially focus on the same context as the best-reward heuristic at the beginning. But they switch to other contexts with greater learning progress when the learning progress in the best learnable context decreases. Moreover, the 1-step progress and monotonic progress heuristics are invariant under different baselines. The heuristics behave identically when the reward is noise-free, that is $\sigma_r = 0$. If there is noise (which simulates the exploration of the agent), the monotonic progress heuristic is usually better (see Figure 2 (b) and (d)).

A context selection method that differs from all others is the context generation and selection method from SAGG-RIAC. As it has been described in Section 2.2, it is designed for continuous context spaces. However, it has a crucial disadvantage in our model of the

learning progress: we assume that the expected learning progress of neighboring contexts can change abruptly. This is a a problem for SAGG-RIAC because it focuses on *regions* of the context space that have a high reward derivative. Among these are not only regions with a high learning progress but also regions with abruptly changing learning progress. In Figure 3 we can see which contexts have been selected by SAGG-RIAC during the simulated learning process: it focuses most of the time on the region around the contexts 3, 4 and 5 because of the greatly varying learning progress factor $w_{\boldsymbol{s}}$ in this region, which results in a high competence derivative. Therefore, a significant part of the explored contexts are not informative because the context 4 has a very low learning progress.

The monotonic progress heuristic, in contrast, selects most of the time tasks with a high learning progress as we can see in Figure 3. At the beginning, it focuses on the context 3 which has the highest $w_{\boldsymbol{s}}$. After some time, when the learning progress in this context saturates, it concentrates on other contexts. At the end it concentrates on the contexts that have not been learned perfectly yet even though they have a low intrinsic learning progress factor.

5. Results

We provide an empirical evaluation of the proposed methods on an artificial contextual benchmark problem in Section 5.1 and in two ball throwing tasks with a simulated Mitsubishi PA-10 in Section 5.2.

5.1 Contextual Function Optimization

In this section, we evaluate the proposed approach on an artificial test problem, compare it to reasonable baseline methods, and analyze the effect of different intrinsic reward heuristics. The test problem is chosen such that some contexts are harder in the sense that the parameters $\boldsymbol{\theta}$ need to be chosen more precisely to reach the same level of return. By focusing on learning primarily the parameters $\boldsymbol{\theta}$ for these contexts, active context selection should be able to outperform a uniform random context selection.

5.1.1 Problem Domain

The context is denoted by $\boldsymbol{s} \in S = [-1, 1]^{n_s}$. We use the objective function

$$f(\boldsymbol{\theta}, \boldsymbol{s}) = -||A\boldsymbol{\theta} - \boldsymbol{s}||_2 \cdot ||s||_2^2 + \sum_{i=0}^{n_s-1} \boldsymbol{s}_i,$$

where $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ denotes the low-level parameters and the matrix $A \in [0, 1]^{n_\theta \times n_s}$ is chosen uniform randomly such that it has rank n_s ($n_\theta > n_s$). The objective function consists of three terms: the *parameter error* $-||A\boldsymbol{\theta} - \boldsymbol{s}||_2$ which can be influenced by the agent's choice of $\boldsymbol{\theta}$, the *context complexity* $||s||_2^2$, which controls how strongly the agent's parameter error deteriorates the task performance, and the *baseline* $\sum_{i=0}^{n_s-1} \boldsymbol{s}_i$, which controls the maximum value in a context. Since A has rank n_s, $\boldsymbol{\theta}$ can always be chosen such that the parameter error becomes 0 and thus the optimal value $f^*(\boldsymbol{s})$ is equal to the baseline $\sum_{i=0}^{n_s-1} \boldsymbol{s}_i$. However, if the agent chooses $\boldsymbol{\theta}$ suboptimally, the same parameter error has different effects on the value of f in different contexts: in contexts with high context complexity, the value of f will

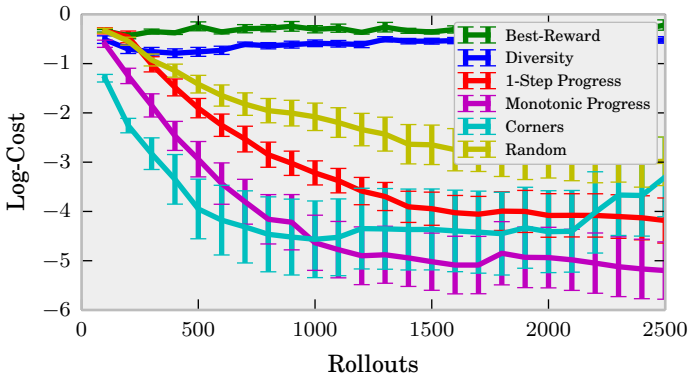


Figure 4: Learning Curves of Different Task Selection Heuristics. A contextual upper-level policy has been learned using C-REPS for different active task-selection strategies. The logarithm of the cost $|f(\boldsymbol{\theta}, \boldsymbol{s}) - f^*(\boldsymbol{s})|$ averaged over 100 test contexts is used as performance measure. Shown are mean and standard error of the mean for 20 runs of 2500 rollouts.

be considerably smaller than f^*, while the difference will be less pronounced in contexts with lower context complexity. Most extremely, for $\boldsymbol{s} = \mathbf{0}$, the choice of $\boldsymbol{\theta}$ is arbitrary since f will always be equal to f^*. Thus, an agent should focus on learning $\boldsymbol{\theta}$ in the contexts with high complexity if its objective is to minimize $|f(\boldsymbol{\theta}, \boldsymbol{s}) - f^*(\boldsymbol{s})|$.

5.1.2 Comparison of Task Selection Heuristics

In a first experiment, we compare task selection with D-UCB for different intrinsic reward heuristics to two baseline methods. In this experiment, training takes place on 25 contexts placed on an equidistant grid over a context space with $n_s = 2$ dimensions; that is, the set of contexts that will be used for training is $S_{\text{train}} = [-1, -\frac{1}{2}, 0, \frac{1}{2}, 1]^2$. The objective of learning, however, is to generalize $\pi_{\boldsymbol{\omega}}$ over the entire context space S, that is, to choose $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ such that $f(\boldsymbol{\theta}, \boldsymbol{s})$ is maximized. As baseline, we use a "Random" task-selection heuristic, which selects uniform randomly among the training contexts. Moreover, we use a "Corner" task-selection heuristic, which selects the four contexts, where the context complexity is maximal, that is $\boldsymbol{s} = (\pm 1, \pm 1)$, in a round-robin fashion.

For the D-UCB, we have used $B = 1.0$, $\gamma = 0.99$, and $\xi = 10^{-8}$. The external reward $r(\boldsymbol{s}, \boldsymbol{\theta})$, based on which the intrinsic reward r_β is computed, is set to $r(\boldsymbol{s}, \boldsymbol{\theta}) = f(\boldsymbol{\theta}, \boldsymbol{s})$ where $\boldsymbol{\theta} = \pi_{\boldsymbol{\omega}}(\boldsymbol{s})$ is sampled from the upper-level policy for the given context $\boldsymbol{s}$. Note that the rewards in $\boldsymbol{s}$ have high variance because of the agent's explorative behavior and are also non-stationary since they depend on the current upper level policy $\pi_{\boldsymbol{\omega}}$. Contextual policy search was conducted with C-REPS with $\epsilon = 2.0$, $N = 50$, and performing an update every 25 rollouts. The evaluation criterion is the expected value of $|f(\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s}), \boldsymbol{s}) - f^*(\boldsymbol{s})|$ of the learned contextual policy $\pi_{\boldsymbol{\omega}}$ over the context space S, where exploration of $\pi_{\boldsymbol{\omega}}$ is disabled. We approximate this quantity by computing the average return of $\pi_{\boldsymbol{\omega}}$ on 100 test contexts sampled uniform randomly from S.

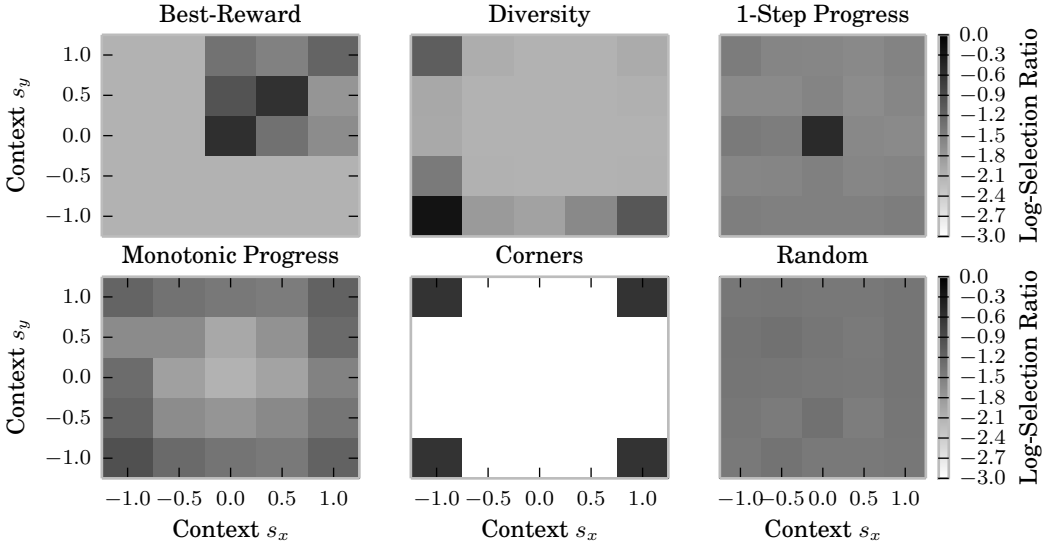


Figure 5: Task preferences of different task-selection strategies during the first 250 rollouts of training. Shown is the logarithm of the mean selection ratio.

Figure 4 shows the learning curves for different task-selection strategies. Figure 5 shows which contexts (tasks) are selected by the different strategies during the first 250 rollouts of training. D-UCB with the "Best-Reward" and the "Diversity" intrinsic reward performs significantly worse than a uniform random task selection. The reason for this is that "Best-Reward" focuses mostly on tasks where the baseline term is large (upper-right area in Figure 5) or where the task complexity is small (central area). Conversely, "Diversity" focuses on areas where the baseline term is small. Both strategies are too imbalanced if the baseline term's contribution is not negligible.

D-UCB with the "1-step Progress" intrinsic reward performs equally bad during the first 250 rollouts. The reason for the low initial progress is that the "1-step Progress" intrinsic reward not only rewards progress but also penalizes regression. However, regression is inevitable during the initial explorative phase. Because of this, this intrinsic reward heuristic focuses initially on contexts with small context complexity where the parameter error and thus the explorative behavior have only a small effect on the actual reward. After the initial explorative phase, this intrinsic reward gets more informative and the corresponding active task selection outperforms uniform random task selection in the long run.

D-UCB with the "Monotonic Progress" intrinsic reward performs considerably better than both D-UCB with the other heuristics and uniform random task selection. The reason for this is that it initially favors complex contexts (the outer areas in Figure 5 with $||s||_2 \gg 0$), where the potential reward improvement is large, without any preference for tasks with small of large baseline value. This task-selection strategy works well and results in a large and stable learning progress. Based on this, we have tested a second baseline denoted as "Corners", which selects the four contexts with maximum context complexity in a round-robin fashion. While this resulted in a very rapid learning progress initially, it is slightly

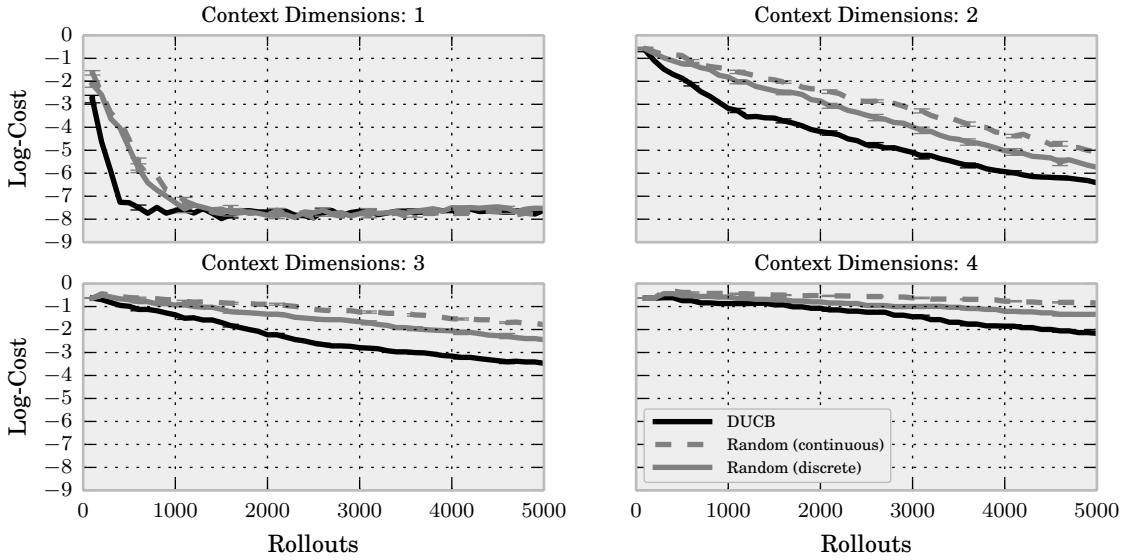


Figure 6: Learning Curves for Different Context Dimensionality. Shown are mean and standard error of the mean for 20 runs of 5000 rollouts.

suboptimal and unstable in the long run since only four of the tasks are ever sampled. Conversely, the "Monotonic Progress" intrinsic reward leads to a more balanced selection of tasks when converging and is thus favorable in the long run. In summary, D-UCB with the "Monotonic Progress" intrinsic reward selects tasks in a way which increased C-REPS' learning progress considerably and proved to be stable at the same time.

5.1.3 Dimensionality of the Context Space

In a second experiment, we compare the performance of D-UCB to uniform random task selection for different dimensionality of the context space. A discrete set S_{train} of 25 contexts for training has been generated by selecting the k-the context $\boldsymbol{s}_k$ uniform random from $\mathbb{R}^{n_s}$ under the constraint $||\boldsymbol{s}_k||_2 = k/25$. The set of test contexts has been generated in the same way but with an other random seed. D-UCB has been combined with the "Monotonic Progress" heuristic. The D-UCB parameters have been set to $B = 1.0$, $\gamma = 0.99$, and $\xi = 10^{-8}$, and the C-REPS parameters to $\epsilon = 1.0$ and $N = 25n_s^2$, and an update was performed every $13n_s^2$ rollouts. As baseline, two different random context selection strategies have been tested: "Random (discrete)" chooses tasks uniform randomly from S_{train} while "Random (continuous)"chooses tasks uniform randomly from $\mathbb{R}^{n_s}$ under the constraint $||\boldsymbol{s}_k||_2 \leq 1$.

Figure 6 shows the learning curves for $n_s \in \{1, 2, 3, 4\}$. For any value of n_s, D-UCB outperforms random task selection in terms of the initial learning progress (not in terms of the final performance). One can further see that selecting a finite, discrete set of training contexts from the continuous context space does not necessarily impair performance; conversely "Random (discrete)" typically performs slightly better than "Random (continuous)". While the general learning progress decreases considerably for higher dimensionality

n_s, this is not directly an issue of the task-selection strategy (since D-UCB still outperforms random task selection) but rather of the underlying contextual policy search method. Thus, the results show that active context selection with D-UCB works satisfyingly while higher dimensional context spaces remain a general challenge for contextual policy search.

5.2 Generalize Throwing Movements

In this section, we consider the problem of learning to throw a ball at a given target. A similar setting has been investigated by Wirkus et al. (2012), where the objective was to learn throwing an object at a specified target based on a forward model of the system. In our experiments, the target can be located at different positions in a predetermined area on the ground and we do not provide any model of the system, making it a contextual model-free policy search problem with the target position being the context. We consider two different reward functions for this experiment. One reward function is continuous (grid problem) while the other has discontinuities and flat regions without a "reward gradient" (dartboard problem), resulting in a setting with easy and more difficult tasks. We provide an empirical evaluation on the grid problem and apply the gained insight on the dartboard problem. In our experiments, we use a simulated Mitsubishi PA-10 robot arm with seven joints for throwing (see Figure 7).

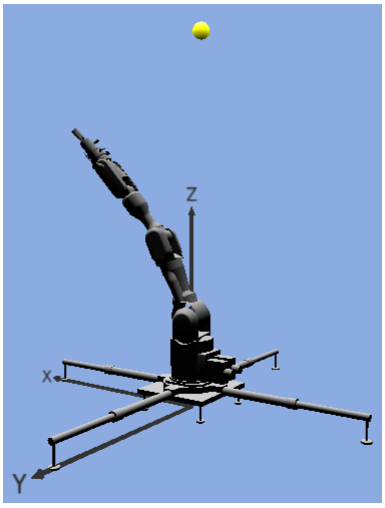


Figure 7: Visualization of the simulated Mitsubishi PA-10 throwing a ball.

5.2.1 Policy Representation

A throwing behavior in this experiment consists of a sequence of two movement primitives (DMPs), where the first corresponds to the strike out and the second one to the actual throwing movement. Both primitives have a duration of $\tau_1 = \tau_2 = 0.5\,\mathrm{s}$. The movement primitives define joint trajectories directly for the seven joints of the Mitsubishi PA-10. The parameters of the two DMPs include the weights of the forcing terms and the respective meta-parameters (see Section 2.1). These parameters have been initialized such that the resulting throw hits the ground position $(-3.55, -3.55)$, where $(0, 0)$ is the ground position at which the PA-10 is mounted and the unit of the coordinate system is in meter. Some of the meta-parameters of the initial policy are to be adapted later on such that other ground positions are hit. These include the final state of the first movement primitive $\boldsymbol{g}_1$ and the velocities at the end of the two movement primitives $\dot{\boldsymbol{g}}_1, \dot{\boldsymbol{g}}_2$. Instead of modifying $\boldsymbol{\theta} = (\boldsymbol{g}_1, \dot{\boldsymbol{g}}_1, \dot{\boldsymbol{g}}_2)$ directly, we use the values from the initial policy $\boldsymbol{\theta}_0$ as the base and we modify the offset $\boldsymbol{\epsilon}_t$ so that $\boldsymbol{\theta}_t = \boldsymbol{\theta}_0 + \boldsymbol{\epsilon}_t$. The weights $\boldsymbol{w}$ of the forcing terms of the two primitives and the other movement primitives remain fixed during this adaptation. Thus,

the actual contextual learning task consists of finding a mapping for $3 \cdot 7 = 21$ parameters such that different target positions on the ground are hit.

5.2.2 Methods

We use C-REPS for contextual policy search with the context $\boldsymbol{s}$ which contains the Cartesian coordinates of the target position for the throw. The mapping φ projects the context to polar coordinates and generates all quadratic terms of the polar coordinates. This mapping results from the constraint of C-REPS to match feature averages instead of the actual context distribution, which would result in an infinite number of constraints (Deisenroth et al., 2013). A policy update is performed after every 50 trials and a memory of at most 300 trials is used for the update. This memory consists of the best $300/K$ trials for each of the K tasks, which results in an update that is similar to importance sampling in PoWER (Kober and Peters, 2011). We restricted the maximum Kullback-Leibler divergence of the old and new policy distributions to $\epsilon = 0.5$, the initial weight matrix was set to $\boldsymbol{W} = \boldsymbol{0}$, and the initial covariance was set to $\boldsymbol{\Sigma} = \sigma_0^2 \boldsymbol{I}$ with $\sigma_0^2 = 0.02$. For D-UCB, we use $\gamma = 0.99$, $B = 10,000$ and $\xi = 10^{-9}$ in all cases.

5.2.3 Grid Problem

We define two contextual learning problems in this setting whose main difference is the structure of the reward function. The first reward function provides the squared distance of the position hit by the throw to the target position as reward: $r(\boldsymbol{s}, \boldsymbol{\theta}) = -||\boldsymbol{s} - \boldsymbol{b}_{\boldsymbol{\theta}}||_2^2$, where $\boldsymbol{s}$ is the target and $\boldsymbol{b}_{\boldsymbol{\theta}}$ are the Cartesian coordinates of the ball when it hits the ground after executing policy $\pi_{\boldsymbol{\theta}}$. In this problem, we generate an equidistant grid of 25 targets for training over the area $[-3, -5] \times [-3, -5]$. Another set of 16 targets is used to test the generalization. These targets form an equidistant grid in the area $[-3.25, -4.75] \times [-3.25, -4.75]$. While different contexts share the same reward function, learning them might differ in complexity. Some of the relative positions of the target to the PA-10 arm are harder to hit because of the kinematic structure of the arm. In addition, the distance to the initial policy makes some contexts easier to solve by exploration than others.

We compared our methods to three baselines: continuous random sampling of contexts from $[-3, -5] \times [-3, -5]$ ("Random (cont.)"), selection in a fixed order ("Round Robin") and the best policy that we have found in all experiments ("Best Policy"). On the left side of Figure 8 the learning curves of several intrinsic reward heuristics are shown and on the right side the best intrinsic reward heuristic is compared to the baselines.

The "Diversity" heuristic, which prefers selecting hard tasks in which a low reward is obtained, performs worse than round robin or random selection. Thus, selecting the more difficult tasks first is not beneficial to speed up learning in this setting. This effect might be even more pronounced in cases where the most difficult tasks are unsolvable. The "Best-Reward" heuristic performs worst here. We observed that it quickly converged to nearly always selecting the same task and hence failed to learn a generalizable upper-level policy. The reason for this behavior is that it encourages D-UCB to focus on the simplest task first. Because it improves quickly in this task, the reward will be considerably greater than the reward of the other tasks. Even though it periodically samples other tasks, the reward in these tasks will typically be smaller and thus, D-UCB sticks to the same task. Selecting

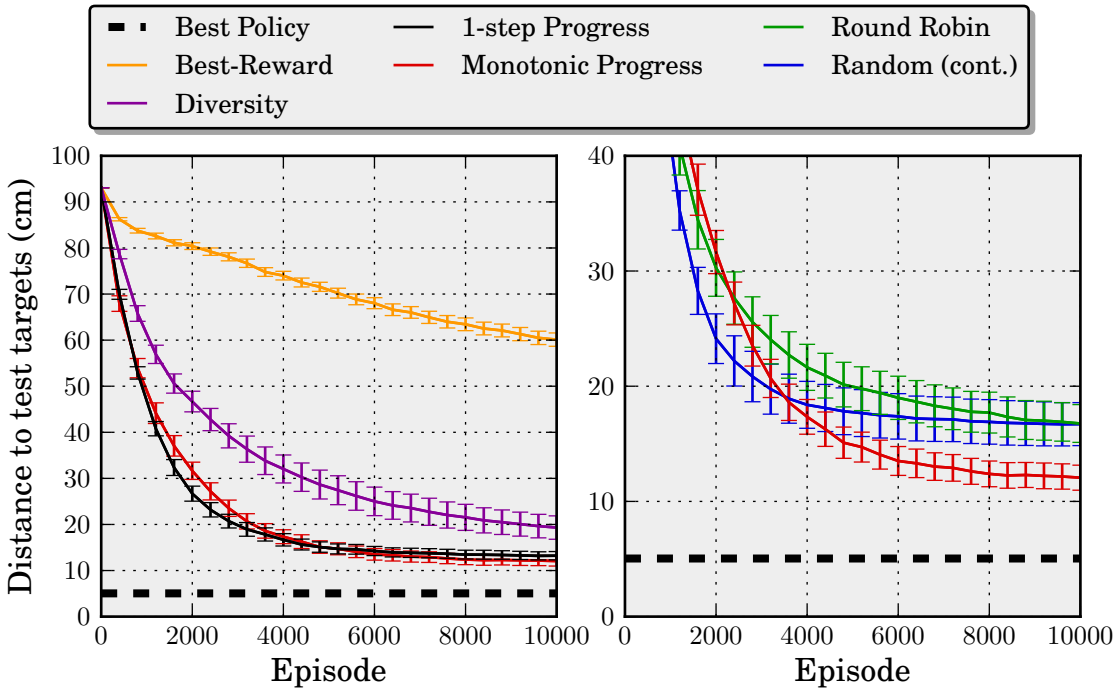


Figure 8: Left: Learning curves of several intrinsic reward heuristics for D-UCB in the grid problem. We measured the average distance to the test targets. The curves and the error bars show the mean and standard error of mean over 20 runs. "Best Policy" indicates the performance of the best policy that we have generated in all experiments. Right: Comparison of D-UCB with intrinsic reward based on the monotonic progress with the baselines.

the tasks in which one can make the greatest estimated learning progress gives a significant advantage in this problem. In contrast to the results in Section 5.1, the "1-step Progress" heuristic is on a par with the "Monotonic Progress" heuristic. A possible reason for this is that the inherent context complexities in this task do not vary as strongly as in the problem in Section 5.1.

In comparison to the baselines, D-UCB under the monotonic progress intrinsic reward is on a par with round robin selection and slightly worse than continuous random selection. Continuous random sampling learns more quickly in the beginning because the comparison of different methods is done on a set of test contexts that are maximally dissimilar from the discrete training context set. Methods that use only the discrete set of training contexts during the training phase have thus a disadvantage compared to continuous random sampling which typically samples closer to the test contexts.

In the long-term, however, D-UCB performs better than both baselines and its average performance gets close to the best policy's performance. This shows that active task selection can accelerate learning considerably and improve the reliability of the result of contextual policy search. A potential reason why continuous random sampling performs worse in the long term is that it cannot use the strategy otherwise employed in C-REPS,

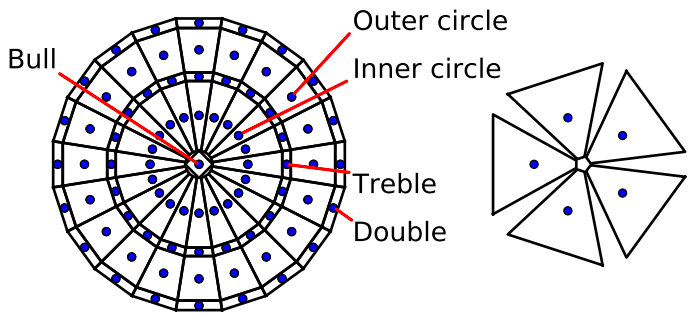


Figure 9: Left: The dartboard is divided into 81 quadrangular fields. Bull and bull's eye are regarded as one field, the rest belongs to one of the four circles (inner circle, outer circle, doubles, and trebles). The initial policy given by $\boldsymbol{\theta}_0$ would throw at the center of the dartboard, that is the bull. Right: The target regions of the bull, inner circle, and outer circle are greater than the corresponding fields on the dartboard and overlap neighboring fields. Shown here are 5 of the 20 target regions of the inner circle. These target regions are only used to compute the reward. Larger target regions can be assumed to make the corresponding learning tasks easier.

namely to keep a separate history of samples per context of identical size ($300/K$ rollouts per context), because it does not encounter any context twice.

5.2.4 Dartboard Problem

As second scenario with the PA-10, we consider a problem which poses tasks of more significantly varying complexity onto the agent. The targets are designed to correspond to the fields on a dartboard. This virtual dartboard is placed on the ground in front of the PA-10. Placing the dartboard on the ground instead of a wall was mainly done to simplify implementation. The diameter is set to 1.4 m (real dartboards have a diameter of 0.451 m). Each field is approximated by quadrangles (see Figure 9) and the center of this field is the context of the task corresponding to the field. For each target, we assign a corresponding *target area* which is usually the quadrangular field. For some tasks, we enlarge each side of the quadrangular field by the factor 3.5 to build the corresponding target region (see Figure 9 for details). This will make these tasks considerably easier than others. The reward function gives a constant negative reward outside of this target area and only provides a reward gradient inside of the target area. The reward function is defined as

$$r(\boldsymbol{s}, \boldsymbol{\theta}) = \begin{cases} -\frac{10{,}000}{d_{\boldsymbol{s}}} ||\boldsymbol{s} - \boldsymbol{b}_{\boldsymbol{\theta}}||_2 & \text{if } \boldsymbol{b}_{\boldsymbol{\theta}} \text{ is within the target area of } \boldsymbol{s} \\ -10,000 & \text{otherwise} \end{cases},$$

where $d_{\boldsymbol{s}}$ is the respective maximum distance to the center within the target area of context $\boldsymbol{s}$. Providing a constant reward outside of the target area complicates the problem compared

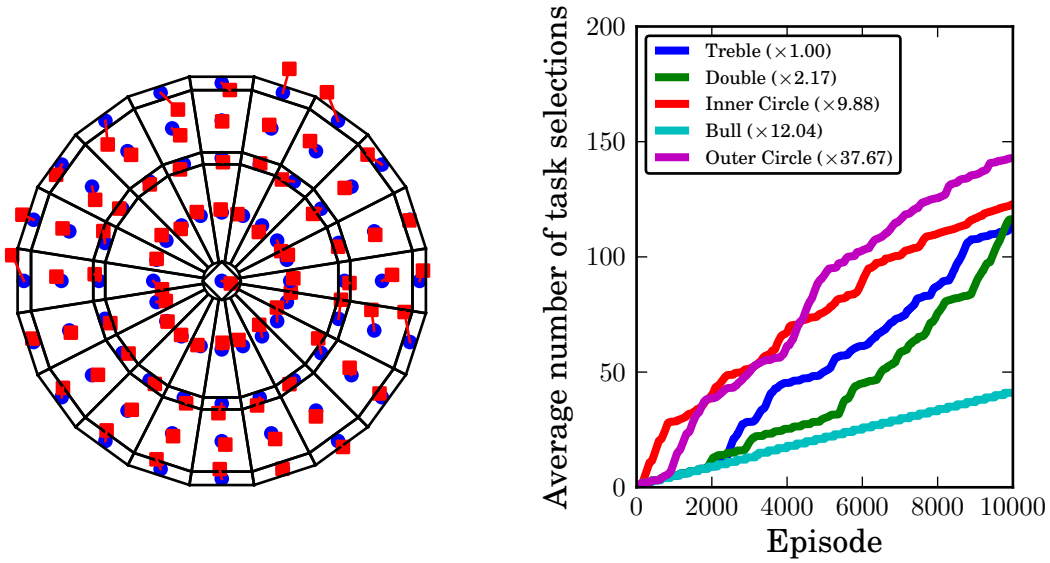


Figure 10: Left: Final upper-level policy. The blue circles mark the center of each field, the red squares that are connected to the corresponding centers with a red line show the position at which the robot arm has actually thrown the ball. Right: Accumulated average number of selections for different task categories. Larger target areas and target areas that are closer to the initial policy are selected more often at the beginning. The relative size in comparison to the smallest target region is given in brackets in the legend.

to the grid setting as no reward gradient helps the agent in improving its policy if it does not hit the target area, within which a reward gradient guides the agent to the center of the field. Thus, the tasks corresponding to larger fields can be considered to be easier since it is more likely that the agent finds a reward gradient during the initial exploration. The agent should thus focus first on these tasks and select the more difficult tasks not until it has improved its contextual policy so far that it is able to hit the corresponding target areas.

We trained with all 81 targets for $50,000$ trials and use the monotonic progress intrinsic reward. The result is displayed in Figure 10: the learned policy hits the "bull", 18 of 20 fields in the inner circle, 15 of 20 trebles, 20 of 20 fields in the outer circle, and 9 of 20 doubles. The overall mean error is 4.62 cm, the median error is 3.77 cm, and the maximum error is 15.96 cm.

Doubles are the most difficult tasks to learn for the agent because they are far away from the initial policy, the target region is small and we cannot transfer much knowledge from neighboring tasks, because they are on the edge of the dartboard. For this reason, some doubles have not been learned very well. In contrast, the trebles can be learned easily, because the solution can be obtained approximately by transferring the solutions of neighboring fields from the inner circle and the outer circle.

In Figure 10 we display which kind of tasks have been selected in the initial learning phase. We can see that the inner circle and the outer circle, which have the greatest target

regions are selected most often in the beginning. The targets from the inner circle are selected even more often than the targets from the outer circle during the first 1,000 episodes even though they are considerably smaller. This is because they are more likely to be reached when exploring from the initial policy which throws at the center of the dartboard. For the same reason trebles are selected more often than doubles at the beginning. After this initial phase, the fields of the outer circle are selected more often because they are now much easier to learn. The bull is so rarely selected because the initial policy will already generate a very good result for this context, hence, improvement is hardly possible.

6. Conclusion and Outlook

We have considered the problem of active task selection in contextual policy search. The hypothesis investigated in this paper is that learning all tasks with the same priority in a round robin or random fashion is not always optimal, in particular if the tasks have different characteristics which make some of them more difficult than others. We have proposed an active task-selection heuristic based on the non-stationary bandit algorithm D-UCB which learns to select tasks that have a large intrinsic reward. These intrinsic rewards can be considered as proxies for the actual learning progress which encourage the agent to engage in those tasks in which its performance is increasing the most. The underlying model of the learning process assumes that each task has a different intrinsic expected learning progress which might change abruptly in the context space and that knowledge can be transferred between neighboring contexts.

Our empirical results have shown that active task selection can make a considerable difference for the learning speed of contextual policy search. In general, we found that a task-selection method should explore in the beginning, then focus on several easy tasks to acquire some initial procedural knowledge, thereupon transfer knowledge to similar but more difficult tasks, and concentrate in the end on those tasks that have not been learned yet. Some of the proposed intrinsic reward heuristics provide a considerable advantage over round robin or uniform random selection in our empirical evaluations on a contextual function maximization problem and a simulated robotic ball throwing experiment.

We restricted ourselves to a discrete set of context vectors for training. In the future it would be interesting to explore not only a discrete set of tasks but a continuous range of task parameters such as arbitrary rather than grid-based target positions for ball throwing as in Section 5.2.3. This could be modeled as a non-stationary continuum-armed bandit problem. However, the non-stationary version of the continuum-armed bandit problem (Agrawal, 1995a) has not been explored thoroughly yet.

Moreover, it would also be desirable to stop sampling of tasks for which the policy has reached an acceptable level of performance, that is, tasks that can be regarded as solved. Since the expected learning progress becomes very small in such tasks, an active context selection mechanism should typically learn this automatically. However, since D-UCB assumes non-stationary rewards, it will continue to sample such tasks as it assumes that the expected reward could increase again. Active context selection algorithms which take the property of the learning progress to converge to zero in the long run into account could thus be candidates for improving over D-UCB.

The goal of the proposed method is to reduce the wear and tear of (possibly robotic) agents as well as the human intervention during learning. Selecting the context actively typically requires some mechanism that sets the environment in the requested context. To limit the amount of human intervention, it is very desirable that the agent can set the desired context on its own. This is typically trivial in simulation. In real-world problems, it can often be achieved by means of previously learned or hard-coded skills. In case that a human supervisor needs to produce the requested context, the feasibility of active context selection depends on the amount of work imposed on the human supervisor and thus, on the specific task.

Among the kind of tasks for which we consider active contextual policy search to be promising are: various kind of goal-directed reaching, hitting and throwing problems like for example ball throwing, darts and hockey. Moreover, scenarios in which an agent can select from a small set of predefined contexts, for example grasping one object from a set of objects with varying size and shape, are promising. Another scenario could be that a robot has to learn how to walk on a restricted set of different terrain surfaces, where the context consists of the properties of the surfaces.

In the future, within the research we will conduct in the context of the project "Behaviors for Mobile Manipulation" (BesMan),[2] we plan to evaluate the proposed approach on different robotic target platforms, among others the humanoid robot AILA. One of the challenges for this is to bridge the simulation-reality gap: multiple executions of the same policy often have significantly different outcomes on actual robots like AILA due to different initial states and other unobserved properties. Thus, the learning approach needs to be able to deal with noise in the executions.

Acknowledgments

This work was supported through two grants of the German Federal Ministry of Economics and Technology (BMWi, FKZ 50 RA 1216 and FKZ 50 RA 1217). In particular, we would like to thank Yohannes Kassahun for helpful comments.

Appendix A. Implementation of Contextual REPS

Contextual relative entropy policy search has been explained in detail by Deisenroth et al. (2013) and Kupcsik et al. (2013). We summarize C-REPS in Algorithm 2. A stochastic policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ is used to generate low-level controllers $\pi_{\boldsymbol{\theta}}$ for a context $\boldsymbol{s}$. In this paper, we use a Gaussian policy with linear mean (Deisenroth et al., 2013) so that $\boldsymbol{\omega} = (\boldsymbol{W}, \boldsymbol{\Sigma})$. After collecting N experience tuples $(\boldsymbol{s}_i, \boldsymbol{\theta}_i, r(\boldsymbol{s}_i, \boldsymbol{\theta}_i))$ (lines 2-6), C-REPS determines a weight d_i for each experience based on the context-dependent baseline $V(\boldsymbol{s}) = \boldsymbol{v}^T\varphi(\boldsymbol{s})$ (lines 7-10). With the obtained weights, C-REPS updates the policy through weighted maximum likelihood (lines 11-17).

Implementing C-REPS is not straightforward. In our experiments and for our reward functions it was sometimes numerically unstable. To improve the reproducibility of the

2. See `http://robotik.dfki-bremen.de/en/research/projects/besman-1.html` for more information.

Algorithm 2 Contextual relative entropy policy search (C-REPS)

Require: ϵ: maximum Kullback-Leibler divergence between two successive policy distributions; $\varphi(\boldsymbol{s})$: extracts features from the context $\boldsymbol{s}$; N: number of samples per update; $\boldsymbol{W}$: initial weights of upper level policy; $\boldsymbol{\Sigma}$: initial covariance of upper level policy distribution

1: **while** not converged **do**
2: **for** $i \in \{1, \ldots, N\}$ **do**
3: Select context $\boldsymbol{s}_i$ # For example, based on *some* specified task-selection method
4: $\boldsymbol{\theta}_i \sim \mathcal{N}(\boldsymbol{W}^T \varphi(\boldsymbol{s}_i), \boldsymbol{\Sigma})$ # Draw policy parameters
5: Obtain $r(\boldsymbol{s}_i, \boldsymbol{\theta}_i)$ # Return of policy $\pi_{\boldsymbol{\theta}_i}$ in environment with context $\boldsymbol{s}_i$
6: **end for**
7: Solve constrained optimization problem $[\eta, \boldsymbol{v}] = \arg\min_{\eta', \boldsymbol{v}'} g(\eta', \boldsymbol{v}')$, s.t. $\eta > 0$

$$g(\eta, \boldsymbol{v}) = \eta\epsilon + \boldsymbol{v}^T \hat{\varphi} + \eta \log \left(\sum_{i=1}^{N} \frac{1}{N} \exp \left(\frac{r(\boldsymbol{s}_i, \boldsymbol{\theta}_i) - \boldsymbol{v}^T \varphi(\boldsymbol{s}_i)}{\eta} \right) \right)$$

8: **for** $i \in \{1, \ldots, N\}$ **do**
9: $d_i \leftarrow \exp\left(\frac{r(\boldsymbol{s}_i, \boldsymbol{\theta}_i) - \boldsymbol{v}^T \varphi(\boldsymbol{s}_i)}{\eta} \right)$ # Determine weight for each experience tuple
10: **end for**
11: **for** $i \in \{1, \ldots, N\}$ **do** # Prepare policy update
12: $\boldsymbol{\Phi}_i = \varphi(\boldsymbol{s}_i)^T$
13: $\boldsymbol{\Theta}_i = \boldsymbol{\theta}_i^T$
14: **end for**
15: $\boldsymbol{D} = \begin{pmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_N \end{pmatrix}$
16: $\boldsymbol{W}_{\text{new}} \leftarrow (\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Theta}$ # Update policy mean
17: $\boldsymbol{\Sigma}_{\text{new}} \leftarrow \frac{\sum_i d_i}{\left(\sum_i d_i\right)^2 - \sum_i d_i^2} (\boldsymbol{\Theta} - \boldsymbol{\Phi}\boldsymbol{W})^T \boldsymbol{D} (\boldsymbol{\Theta} - \boldsymbol{\Phi}\boldsymbol{W})$ # Update exploration covariance
18: **end while**

results, we explain some of the modifications that we had to make in addition to the log-sum-exp trick (in line 7) that is already mentioned by Deisenroth et al. (2013).

The weighted least squares problem is sometimes ill-conditioned. Hence, we added the regularization term $\lambda \boldsymbol{I}$ so that

$$\boldsymbol{W}_{\text{new}} \leftarrow (\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Theta}.$$

In our experiments, we use $\lambda = 10^{-4}$.

At the end of the learning process, η sometimes becomes very small which causes numerical problems. It is helpful to use an other lower bound $\eta_{\min}$, so that $\eta > \eta_{\min}$, for example $\eta_{\min} \in (10^{-6}, 10^{-4})$.

For very large negative rewards the weights d_i are sometimes so small that they cannot be represented properly with 64-bit floating-point numbers. But we can replace d_i by the

softmax-like term

$$\overline{d}_i = \frac{d_i}{\sum_j d_j} = \frac{\exp\left(\frac{r(\boldsymbol{s}_i,\boldsymbol{\theta}_i)-\boldsymbol{v}^T\varphi(\boldsymbol{s}_i)}{\eta}\right)}{\sum_j \exp\left(\frac{r(\boldsymbol{s}_j,\boldsymbol{\theta}_j)-\boldsymbol{v}^T\varphi(s_j)}{\eta}\right)},$$

which does neither change the weights of the linear policy $\boldsymbol{W}_{\text{new}}$, nor the covariance $\boldsymbol{\Sigma}_{\text{new}}$. Softmax can be implemented numerically stable:

$$\overline{d}_i = \frac{\exp\left(\frac{r(\boldsymbol{s}_i,\boldsymbol{\theta}_i)-\boldsymbol{v}^T\varphi(\boldsymbol{s}_i)}{\eta}\right)}{\sum_j \exp\left(\frac{r(\boldsymbol{s}_j,\boldsymbol{\theta}_j)-\boldsymbol{v}^T\varphi(s_j)}{\eta}\right)} = \frac{\exp\left(\frac{r(\boldsymbol{s}_i,\boldsymbol{\theta}_i)-\boldsymbol{v}^T\varphi(\boldsymbol{s}_i)}{\eta} - m\right)}{\sum_j \exp\left(\frac{r(\boldsymbol{s}_j,\boldsymbol{\theta}_j)-\boldsymbol{v}^T\varphi(s_j)}{\eta} - m\right)},$$

where $m = \max_i \frac{r(\boldsymbol{s}_i,\boldsymbol{\theta}_i)-\boldsymbol{v}^T\varphi(\boldsymbol{s}_i)}{\eta}$.

In addition, for the experiments in Section 5.2 we use a kind of importance sampling to reuse experience tuples from previous rollouts: for every task we maintain a memory of the best N/K rollouts, where N is the maximum number of samples for a policy update and K is the number of tasks. Note that this might have a negative effect if there are multiple local optima that are separated by regions with low return in some context $\boldsymbol{s}$.

References

Rajeev Agrawal. The continuum-armed bandit problem. *SIAM Journal on Control and Optimization*, 33(6):1926–1951, 1995a.

Rajeev Agrawal. Sample mean based index policies with O(log n) regret for the multi-armed bandit problem. *Advances in Applied Probability*, 27(4):1054–1078, 1995b.

Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1): 49–73, 2013.

Andrew G. Barto, Satinder Singh, and Nuttapong Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference of Developmental Learning*, pages 112–119, LaJolla, CA, USA, 2004.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning*, pages 41–48, 2009.

Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

William S. Cleveland and Susan J. Devlin. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83 (403):596–610, 1988.

Bruno Castro da Silva, George Konidaris, and Andrew G. Barto. Learning parameterized skills. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Bruno Castro da Silva, George Konidaris, and Andrew Barto. Active learning of parameterized skills. In *Proceedings of the 31st International Conference on Machine Learning*, Beijing, China, 2014.

Christian Daniel, Gerhard Neumann, and Jan Peters. Hierarchical relative entropy policy search. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics, AISTATS 2012*, pages 273–281, 2012.

Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1–2):328–373, 2013.

Aurelélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *Proceedings of the 22nd International Conference on Algorithmic Learning Theory*, pages 174–188, 2011.

Jacqueline Gottlieb, Pierre-Yves Oudeyer, Manuel Lopes, and Adrien Baranes. Information-seeking, curiosity, and attention: computational and neural mechanisms. *Trends in Cognitive Sciences*, 17(11):585–93, 2013.

Caglar Gulcehre and Yoshua Bengio. Knowledge matters: Importance of prior information for optimization. In *International Conference on Learning Representations*, 2013.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, 2013.

S. Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84(1–2):171–203, 2011.

Jens Kober, Andreas Wilhelm, Erhan Oztop, and Jan Peters. Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots*, 33(4):361–379, 2012.

Levente Kocsis and Csaba Szepesvári. Discounted UCB. In *2nd PASCAL Challenges Workshop*, 2006.

Klas Kronander, Mohammad S. M. Khansari-Zadeh, and Aude Billard. Learning to control planar hitting motions in a minigolf-like task. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 710–717, 2011.

Andras G. Kupcsik, Marc Peter Deisenroth, Jan Peters, and Gerhard Neumann. Data-efficient generalization of robot skills with contextual policy search. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2013.

Jan Hendrik Metzen, Alexander Fabisch, Lisa Senger, José de Gea Fernández, and Elsa Andrea Kirchner. Towards learning of generic skills for robotic manipulation. *Künstliche Intelligenz*, 2013. doi: 10.1007/s13218-013-0280-1.

Katharina Mülling, Jens Kober, and Jan Peters. A biomimetic approach to robot table tennis. *Adaptive Behavior*, 19(5):359–376, 2011.

Katharina Mülling, Jens Kober, Oliver Krömer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *International Journal of Robotics Research*, 32(3), 2013.

Pierre-Yves Oudeyer and Frederic Kaplan. Intelligent adaptive curiosity: a source of self-development. In Luc Berthouze, Hideki Kozima, Christopher G. Prince, Giulio Sandini, Georgi Stojanov, G. Metta, and C. Balkenius, editors, *Proceedings of the 4th International Workshop on Epigenetic Robotics*, pages 127–130. Lund University Cognitive Studies, 2004.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

Jan Peters, Katharina Mülling, Jens Kober, Duy Nguyen-Tuong, and Oliver Krömer. Robot skill learning. In *Proceedings of the 20th European Conference on Artificial Intelligence*, pages 40–45, 2012.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

Paul Ruvolo and Eric Eaton. Scalable lifelong learning with active task selection. In *Proceedings of the AAAI 2013 Spring Symposium on Lifelong Machine Learning*, 2013.

Burr Settles. Active Learning Literature Survey. Technical Report 1648, University of Wisconsin–Madison, 2010.

Joshua B. Tenenbaum, Vin De Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

Aleš Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, 26 (5):800–815, 2010.

Robert W. White. Motivation reconsidered: the concept of competence. *Psychological review*, 66:297–333, 1959.

Malte Wirkus, José de Gea Fernández, and Yohannes Kassahun. Realizing target-directed throwing with a real robot using machine learning techniques. In *Proceedings of the 5th International Workshop on Evolutionary and Reinforcement Learning for Autonomous Robot Systems*, pages 37–43, 2012.

Journal of Machine Learning Research 15 (2014) 3401-3423 Submitted 1/13; Revised 6/14; Published 10/14

Matrix Completion with the Trace Norm: Learning, Bounding, and Transducing

Ohad Shamir OHAD.SHAMIR@WEIZMANN.AC.IL
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science
Rehovot 7610001, Israel

Shai Shalev-Shwartz SHAIS@CS.HUJI.AC.IL
School of Computer Science and Engineering
The Hebrew University
Givat Ram, Jerusalem 9190401, Israel

Editor: Tommi Jaakkola

Abstract

Trace-norm regularization is a widely-used and successful approach for collaborative filtering and matrix completion. However, previous learning guarantees require strong assumptions, such as a uniform distribution over the matrix entries. In this paper, we bridge this gap by providing such guarantees, under much milder assumptions which correspond to matrix completion as performed in practice. In fact, we claim that previous difficulties partially stemmed from a mismatch between the standard learning-theoretic modeling of matrix completion, and its practical application. Our results also shed some light on the issue of matrix completion with bounded models, which enforce predictions to lie within a certain range. In particular, we provide experimental and theoretical evidence that such models lead to a modest yet significant improvement.

Keywords: collaborative filtering, matrix completion, trace-norm regularization, transductive learning, sample complexity

1. Introduction

We consider the problem of matrix completion, where the goal is to predict entries of an unknown matrix based on a subset of its observed entries. A popular approach to achieve this is via trace-norm regularization, where one seeks a matrix that agrees well with the observed entries, while constraining its complexity in terms of the trace-norm. The trace-norm is well-known to be a convex surrogate to the matrix rank, and has repeatedly shown good performance in practice (Srebro et al., 2004; Salakhutdinov and Mnih, 2007; Bach, 2008; Candès and Tao, 2009).

However, in terms of distribution-free guarantees, previous results on trace-norm regularization have been surprisingly weak. Most non-trivial guarantees (e.g., Srebro and Shraibman, 2005; Candès and Tao, 2009; Candès and Recht, 2009) assume that the observed entries are sampled uniformly at random. In most matrix completion tasks, this is an extremely unrealistic assumption. For example, in the Netflix challenge data set, where the matrix contains the ratings of users (rows) for movies (columns), the number and distri-

bution of ratings differ drastically between users. Modeling such data as a uniform sample is not a reasonable assumption. Another paper (Negahban and Wainwright, 2010) studied the problem of matrix completion under a non-uniform distribution. However, the analysis is still not distribution-free, and requires strong assumptions on the underlying matrix. Moreover, the results do not apply to standard trace-norm regularization, but rather to a carefully re-weighted version of trace-norm regularization.

In practice, we know that standard trace-norm regularization works quite well even for data which is very non-uniform. Moreover, we know that in other learning problems, one is able to derive distribution-free guarantees, and there is no a-priori reason why this should not be possible here. Nevertheless, obtaining a non-trivial guarantee for trace-norm regularization has remained elusive. This partially motivated work on alternative complexity measures for matrix completion, such as the max-norm and weighted variants of the trace-norm (see further discussion below).

In this paper, we bridge this gap between our theoretical understanding and practical performance of trace-norm regularization. We show that by adding very mild assumptions, which correspond to matrix completion as performed *in practice*, it is possible to learn in a distribution-free manner by observing $\mathcal{O}(n^{3/2})$ entries from an $m \times n$ matrix (where $m \leq n$, and for a reasonable trace-norm regime). Moreover, this bound is tight. When $m = \Theta(n)$, this corresponds to viewing a vanishingly small portion of the entries, hence we get a non-trivial learning guarantee. In fact, we claim that the difficulties in providing such guarantees partially stemmed from a mismatch between the standard theoretical modeling of matrix completion, and its practical application. We emphasize that our bounds are weaker than previous bounds in the literature, which required observing as few as $\tilde{\mathcal{O}}(n)$ entries (up to log factors). However, these bounds hold only under restrictive distributional assumptions, whereas our bounds hold under any distribution, and are provably tight in such a distribution-free setting.

First, we show that one can obtain such guarantees, if one takes into account that the values to be predicted are bounded. For example, in predicting movie ratings, it is known in advance that the ratings are on a scale of (say) 1 to 5, and practitioners usually clip their predictions to be inside this range. While this seems like an almost trivial operation, we show that taking it into account has far-reaching implications in terms of the theoretical guarantees. The proof relies on a decomposition technique which might also be useful for regularizers other than the trace-norm.

Second, we argue that the standard *inductive model* of learning, where the training data is assumed to be sampled i.i.d. from some distribution, may not be the best way to analyze matrix completion. Instead, we look at the *transductive model*, where sampling of the data is done without replacement. In the context of matrix completion, we show this makes a large difference in terms of the attainable guarantees.

Our results show that a transductive model, and boundedness assumptions, play an important role in obtaining distribution-free guarantees. This relates to a line of recent works, which suggest to incorporate prior knowledge on the range of predicted values into the learning process, by explicitly bounding the predictions. We provide an empirical study, which indicates that this indeed provides a modest, yet significant, improvement in performance, and corroborates our theoretical findings. Finally, we discuss how recent work,

which appeared since the preliminary version of this paper was published, relate to and strengthen our observations.

The paper is structured as follows. We begin by describing the setting and the notation we use in Section 2, and introduce the sample complexity issues of matrix completion with the trace norm in Section 3. In Section 4, we show how we can non-trivially learn with the trace-norm in an inductive i.i.d. setting, under boundedness assumptions. In Section 5, we show how similar performance can be ensured if we switch from an inductive setting to a transductive setting, where each entry appears only once in the data. We provide matching lower bounds in Section 6. In Section 7, we experimentally investigate how boundedness assumptions affect practical performance. Section 8 contains a discussion of how some recent works relate to our paper, and Section 9 contains full proofs of our results. We end with a discussion and some open issues in Section 10.

2. Setting

Our goal is to predict entries of an unknown $m \times n$ matrix X, based on a random subset of observed entries of X. A common way to achieve this, following standard learning approaches, is to find an $m \times n$ matrix W from a constrained class of matrices $\mathcal{W}$, which minimizes the discrepancy from X on the observed entries. More precisely, if we let $S = \{i_\alpha, j_\alpha\}$ denote the set of (row,column) observed entries, and ℓ is a loss function measuring the discrepancy between the predicted and actual value, then we solve the optimization problem

$$\min_{W \in \mathcal{W}} \frac{1}{|S|} \sum_{\alpha=1}^{|S|} \ell(W_{i_\alpha,j_\alpha}, X_{i_\alpha,j_\alpha}), \tag{1}$$

An important and widely used class of matrices $\mathcal{W}$ are those with bounded *trace-norm* (sometimes also denoted as the nuclear norm or the Ky-Fan n norm). Given a matrix W, its trace-norm $\|W\|_{tr}$ is defined as the sum of the singular values. The class of matrices with bounded trace-norm has several useful properties, such as it being a convex approximation of class of rank-bounded matrices (e.g., Srebro and Shraibman, 2005). Thus, we can often optimize Equation (1) in a computationally tractable manner, learning predictors which are competitive with low-rank matrices. The trace-norm of any $m \times n$ matrix W is at least $\|W\|_F$ and at most $\mathrm{Rank}(W)\|W\|_F$, where $\|W\|_F$ is the Frobenius norm (Horn and Johnson, 1985), and therefore the trace-norm of constant-rank $m \times n$ matrices with bounded entries is $\Theta(\sqrt{mn})$. Therefore, we wish to attain learning guarantees which are non-trivial when the trace norm is at least on the order of $t = \Theta(\sqrt{mn})$. However, our theorems will hold for any t.

For now, we will consider the inductive model of learning, which parallels the standard agnostic-PAC learnability framework. The model is defined as follows: We assume there exists an unknown distribution $\mathcal{D}$ over $\{1, \ldots, m\} \times \{1, \ldots, n\}$. Each instantiation (i, j) provides the value $X_{i,j}$ of an entry at a randomly picked row i and column j. An i.i.d. sample $S = \{i_\alpha, j_\alpha\}$ of indices is chosen, and the corresponding entries $\{X_{i_\alpha,j_\alpha}\}$ are revealed. Our goal is to find a matrix $W \in \mathcal{W}$ such that its risk (or generalization error), $\mathbb{E}_{(i,j)\sim\mathcal{D}}[\ell(W_{i,j}, X_{i,j})]$, is as close as possible to the smallest possible risk over all $W \in \mathcal{W}$. It is well-known that this can be achieved by solving the optimization problem in Equa-

tion (1), if we can provide a non-trivial uniform sample complexity bound, namely a bound on

$$\sup_{W\in\mathcal{W}}\left(\mathbb{E}_{i,j}\left[\ell(W_{i,j},X_{i,j})\right]-\frac{1}{|S|}\sum_{\alpha=1}^{|S|}\ell(W_{i_\alpha,j_\alpha},X_{i_\alpha,j_\alpha})\right). \tag{2}$$

A major focus of this paper is studying the difficulties and possibilities of obtaining such bounds.

3. Sample Complexity Bounds for the Trace-Norm

Consider the class of trace-norm constrained matrices, $\mathcal{W}=\{W:\|W\|_{tr}\leq t\}$. Although learning with respect to this class is widely used in matrix completion, understanding its generalization and sample-complexity properties has proven quite elusive. Sample complexity bounds of the form $\mathcal{O}(\sqrt{(m+n)/|S|})$ (when $t=\Theta(\sqrt{mn})$, and ignoring logarithmic factors) were obtained under the strong assumption of a uniform distribution over the matrix entries (Srebro and Shraibman, 2005). However, this assumption does not correspond to real-world matrix completion data sets, where the distribution of the revealed entries appears to be highly non-uniform. Other works, which focused on exact matrix completion (e.g., Candès and Tao, 2009; Candès and Recht, 2009), also assume a uniform sampling distribution.

The bounds in Srebro and Shraibman (2005) are based on the Rademacher complexity of the class $\mathcal{W}$, and will be utilized in our analysis as well. Formally, we define the (empirical) Rademacher complexity of a hypothesis class $\mathcal{W}$ combined with a loss function ℓ, with respect to a sample S, as

$$R_S(\ell\circ\mathcal{W})=\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{W\in\mathcal{W}}\frac{1}{|S|}\sum_{\alpha=1}^{|S|}\sigma_\alpha\ell(W_{i_\alpha,j_\alpha},X_{i_\alpha,j_\alpha})\right], \tag{3}$$

where $\sigma_1,\ldots,\sigma_{|S|}$ are i.i.d. random variables taking the values -1 and $+1$ with equal probability.

Rademacher complexities play a key role in obtaining sample complexity bounds, either in expectation or in high probability. The following is a typical example (based on Boucheron and Lugosi, 2005, Theorem 3.2):

Theorem 1 *The expected value of Equation (2) is at most $2R_S(\ell\circ\mathcal{W})$. Moreover, if there is a constant b_ℓ such that $\sup_{i,j,W\in\mathcal{W}}|\ell(W_{i,j},X_{i,j})|\leq b_\ell$, then for any $\delta\in(0,1)$, Equation (2) is bounded with probability at least $1-\delta$ by $2R_S(\ell\circ\mathcal{W})+b_\ell\sqrt{2\log(2/\delta)/|S|}$.*

In general, the dominant term in the bound above is the Rademacher complexity $R_S(\ell\circ\mathcal{W})$. Thus, if we can upper-bound the Rademacher complexity by a quantity much smaller than 1, we get a non-trivial upper bound on Equation (2). Such a bound implies that the empirical risk (or average loss over the training set) is close to the true risk uniformly for all $W\in\mathcal{W}$, and therefore that solving Equation (1) will lead to a predictor with near-optimal risk.

Unfortunately, for the class $\mathcal{W}=\{W:\|W\|_{tr}\leq t\}$ and general distributions over the matrix entries, the Rademacher complexity can be large, leading to vacuous bounds. To see why, suppose that the loss function ℓ is 1-Lipschitz in its first argument. Then the standard

way to analyze Equation (3) (see Bartlett and Mendelson, 2003) is to use the contraction principle to upper bound it by

$$\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{W\in\mathcal{W}}\frac{1}{|S|}\sum_{\alpha=1}^{|S|}\sigma_\alpha W_{i_\alpha,j_\alpha}\right],$$

and then using Hölder's inequality to upper bound it by

$$\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{W\in\mathcal{W}}\frac{1}{|S|}\|\Gamma\|_{sp}\|W\|_{tr}\right] \;=\; t\frac{1}{|S|}\mathbb{E}[\|\Gamma\|_{sp}],$$

where Γ is a matrix whose (i,j)-th entry is defined as $\sum_{\alpha:i_\alpha=i,j_\alpha=j}\sigma_\alpha$, and $\|\cdot\|_{sp}$ is the spectral norm (i.e., the largest singular value), which is well-known to be dual to the trace-norm (Fazel et al., 2001). However, if for instance all σ_α are on the same entry i,j, then $\mathbb{E}[\|\Gamma\|_{sp}]$ equals $\mathbb{E}[|\sum_\alpha\sigma_\alpha|] = \Theta(\sqrt{|S|})$, leading to a bound of the form $\mathcal{O}(t/\sqrt{|S|})$. As discussed earlier, t is typically at least on the order of $\sqrt{mn}$, in which case we get a bound on the Rademacher complexity which is $\mathcal{O}(\sqrt{mn/|S|})$ — smaller than 1 only when the sample size $|S|$ is larger than the total number mn of matrix entries. It is a trivial bound, since the entire goal of matrix completion is prediction based on observing just a small subset of the matrix entries.

Unfortunately, this bound appears impossible to improve in general (see section 6.2.2 in Srebro, 2004). Srebro and Shraibman (2005) circumvent this by imposing a strong uniform distribution assumption, under which a tighter bound is attainable. The main drive of our paper is that by modifying the setting in some very simple ways, which often correspond to matrix completion as done in practice, one can obtain non-trivial learning guarantees without any distributional assumptions.

4. Results for the Inductive Model

In this section, we show that by introducing *boundedness* conditions into the learning problem, one can obtain non-trivial bounds on the Rademacher complexity, and hence on the sample complexity of learning with trace-norm constraints.

We will start with the case where we actually learn with respect to the hypothesis class of trace-norm-constrained matrices, $\mathcal{W}=\{W:\|W\|_{tr}\leq t\}$, and the only boundedness is in terms of the loss function:

Theorem 2 *Consider the hypothesis class $\mathcal{W}=\{W:\|W\|_{tr}\leq t\}$. Suppose that for all i,j the loss function $\ell(\cdot,X_{i,j})$ is both b_ℓ-bounded and l_ℓ-Lipschitz in its first argument: Namely, that $\ell(W_{i,j},X_{i,j})\leq b_\ell$ for any W,i,j, and that $\frac{|\ell(W_{i,j},X_{i,j})-\ell(W'_{i,j},X_{i,j})|}{|W_{i,j}-W'_{i,j}|}\leq l_\ell$ for any W,W',i,j. Then*

$$R_S(\ell\circ\mathcal{W})\leq\sqrt{9Cl_\ell b_\ell\frac{t(\sqrt{m}+\sqrt{n})}{|S|}},$$

where C is the universal constant appearing in Theorem 8.

When $t = \Theta(\sqrt{mn})$, the theorem implies that a sample of size $\mathcal{O}(n\sqrt{m} + m\sqrt{n})$ is sufficient to obtain good generalization performance. We note that the boundedness assumption is non-trivial, since the trace-norm constraint does not imply entries of constant magnitude (the entries can be as large as t for a matrix whose trace norm is t). On the other hand, as discussed earlier, the obtainable bound on the Rademacher complexity without a boundedness assumption is no better than $\mathcal{O}((m+n)/\sqrt{|S|})$, which leads to a trivial required sample size of $\mathcal{O}((m+n)^2)$. Moreover, we emphasize that the result makes no assumptions on the underlying distribution from which the data was sampled. The proof is presented in Subsection 9.1. We note that it relies on a decomposition technique which might also be useful for regularizers other than the trace-norm.

An alternative way to introduce boundedness, and get a non-trivial guarantee, is by composing the entries of a matrix W with a bounded transfer function. In particular, rather than just learning a matrix W with bounded trace-norm, we can learn a model $\phi \circ W$, where W has bounded trace-norm, and $\phi : \mathbb{R} \mapsto I$ is a fixed mapping of each entry of W into some bounded interval $I \subseteq \mathbb{R}$. This model is used in practice, and is useful in the common situation where the entries of X are known to be in a certain bounded interval. In Section 7, we return to this model in greater depth. In terms of the theoretical guarantee, one can provide a result similar to Theorem 2, without assuming boundedness of the loss function.

Theorem 3 *Consider the hypothesis class $\mathcal{W} = \{\phi \circ W : \|W\|_{tr} \leq t\}$. Let $\phi : \mathbb{R} \mapsto [-b_\phi, b_\phi]$ be a bounded l_ϕ-Lipschitz function, and suppose that for all i, j, $\ell(\cdot, X_{i,j})$ is l_ℓ-Lipschitz on the domain $[-b_\phi, b_\phi]$. Then*

$$R_S(\ell \circ \mathcal{W}) \leq l_\ell \sqrt{9 C l_\phi b_\phi \frac{t(\sqrt{m} + \sqrt{n})}{|S|}},$$

where C is the universal constant appearing in Theorem 8.

The bound in this theorem scales similarly to Theorem 2, in terms of its dependence on m, n. Another possible variant is directly learning a matrix W with both a constraint on the trace-norm, as well as an ∞-norm constraint (i.e., $\max_{i,j} |W_{i,j}| \leq c$ for some constant c) which forces the matrix entries to be constant. This model has some potential benefits which shall be further discussed in Section 10.

Theorem 4 *Consider the hypothesis class $\mathcal{W} = \{W : \|W\|_{tr} \leq t, \|W\|_\infty \leq b\}$, where $\|W\|_\infty = \max_{i,j} |W_{i,j}|$. Suppose that for all i, j, $\ell(\cdot, X_{i,j})$ is l_ℓ-Lipschitz on the domain $[-b, b]$. Then*

$$R_S(\ell \circ \mathcal{W}) \leq l_\ell \sqrt{9 C b \frac{t(\sqrt{n} + \sqrt{m})}{|S|}},$$

where C is the universal constant appearing in Theorem 8.

Assuming b is a constant (which is the reasonable assumption here), we get a similar bound as before.

So, we see that by inserting mild boundedness assumptions on the loss function or the matrix entries, it is possible to derive non-trivial guarantees for learning with trace norm

constraints. These were all obtained under the standard inductive model, where we assume that our data is an i.i.d. sample from an underlying distribution. In the next section, we will discuss a different learning model, which we argue to more closely resemble matrix completion as done in practice, and leads to better bounds on the Rademacher complexity, without making boundedness assumptions.

5. Improved Results for the Transductive Model

In the inductive model we have considered so far, the goal is to predict well with respect to an unknown distribution over matrix entries, given an i.i.d. sample from that distribution. The *transductive* learning model (see for instance Vapnik, 1998) is different, in that our goal is to predict well with respect to a *specific* subset of entries, whose location is known in advance. More formally, we fix an arbitrary subset of S entries, and then split it uniformly at random into two subsets $S_{train} \cup S_{test}$. We are then given the values of the entries in S_{train}, and our goal is to predict the values of the entries in S_{test}. For simplicity, we will assume that $|S_{train}| = |S_{test}| = |S|/2$, but our results can be easily generalized to more general partitions.

We note that this procedure is *exactly* the one often performed in experiments reported in the literature: Given a data set of entries, one randomly splits it into a training set and a test set, learns a matrix on the training set, and measures its performance on the test set (e.g., Toh and Yun, 2009; Jaggi and Sulovský, 2010). Even for other train-test split methods, such as holding out a certain portion of entries from each row, the transductive model seems closer to reality than the inductive model. Moreover, the transductive model captures another important feature of real-world matrix completion: the fact that no entry is repeated in the training set. In contrast, in the inductive model the training set is collected i.i.d., so the same entry might be sampled several time over. In fact, this is virtually certain to happen whenever the sample size is at least on the order of $\sqrt{mn}$, due to the birthday paradox. This does not appear to be a mere technicality, since the proofs of our theorems in the inductive model have to rely on a careful separation of the entries according to the number of times they were sampled. However, in reality each entry appears in the data set only once, matching the transductive learning setting.

To analyze the transductive model, we require analogues of the tools we have for the inductive model, such as the Rademacher complexity. Fortunately, such analogues were already obtained in the literature (El-Yaniv and Pechyoni, 2009), and we will rely on their results. In particular, based on Theorem 1 in that paper, we can use our notion of Rademacher complexity, as defined in Equation (3), to provide sample complexity bounds in the transductive model:[1]

Theorem 5 *Fix a hypothesis class $\mathcal{W}$, and suppose that $\sup_{i,j,W\in\mathcal{W}} |\ell(W_{i,j}, X_{i,j})| \leq b_\ell$. Let a set S of ≥ 2 distinct indices be fixed, and suppose it is uniformly and randomly split to two equal subsets S_{train}, S_{test}. Then with probability at least $1-\delta$ over the random split, it*

1. In El-Yaniv and Pechyoni (2009), a more general notion of transductive Rademacher complexity was defined, where the σ_α random variables could also take 0 values. However, when $|S_{train}| = |S_{test}|$, that complexity can always be upper bounded by the standard definition of Rademacher complexity — see Lemma 1 in their paper.

holds for any $W \in \mathcal{W}$ *that*

$$\frac{1}{|S_{test}|}\sum_{(i,j)\in S_{test}} \ell(W_{i,j}, X_{i,j}) - \frac{1}{|S_{train}|}\sum_{(i,j)\in S_{train}} \ell(W_{i,j}, X_{i,j})$$
$$\leq 4R_S(\ell \circ \mathcal{W}) + \frac{b_\ell\left(11 + 4\sqrt{\log(1/\delta)}\right)}{\sqrt{|S_{train}|}}.$$

This theorem implies that if $R_S(\ell \circ \mathcal{W})$ is effectively bounded, then the average loss over S_{train} is close to the average loss over S_{test}, uniformly for any W, and therefore minimizing the average loss over S_{train} will result in a predictor with near-optimal average loss over S_{test}.

We now present our main result for the transductive model, which implies non-trivial bounds on the Rademacher complexity of matrices with constrained trace-norm. Unlike the inductive model, here we make no additional boundedness assumptions, yet the bound is superior. The proof appears in Subsection 9.4.

Theorem 6 *Consider the hypothesis class* $\mathcal{W} = \{W : \|W\|_{tr} \leq t\}$*. Then in the transductive model, it holds that*

$$R_S(\ell \circ \mathcal{W}) \leq C l_\ell \frac{3t\left(\sqrt{m} + \sqrt{n}\right)}{2|S|},$$

where C *is the universal constant appearing in Theorem 8. Alternatively, letting* $N = \max_i |\{j : (i,j) \in S\}|$ *and* $M = \max_j |\{i : (i,j) \in S\}|$*, then*

$$R_S(\ell \circ \mathcal{W}) \leq C l_\ell \frac{t\ \max\left\{\sqrt{M}, \sqrt{N}\right\}}{|S|}\sqrt[4]{\log(\min\{m,n\})},$$

where C *is the universal constant appearing in Theorem 9.*

We note that the second bound, while containing an additional logarithmic term, depends on the distribution of the entries, and can be considerably tighter than the worst-case. To see this, suppose (for simplicity) a rectangular matrix, so that $m = n$, and that $t = \Theta(\sqrt{mn}) = \Theta(n)$. Then in the worst-case, the bound becomes meaningful when $|S| = \Omega(n^{3/2})$. However, if the entries in S are (approximately) uniformly distributed throughout the matrix, then the maximal number of entries in each row and column is $\mathcal{O}(|S|/n)$. In that case, plugging $|S|/n$ instead of M and N, as well as $t = \Theta(n)$, we obtain the bound

$$R_S(\ell \circ \mathcal{W}) \leq \tilde{O}\left(\sqrt{\frac{n}{|S|}}\right)$$

(ignoring logarithmic factors), which is already meaningful when $|S| = \tilde{\Omega}(n)$. Interestingly, this bound is similar (up to logarithmic factors) to previous bounds in the inductive setting (e.g., Srebro and Shraibman, 2005)), which relied on a uniform distribution assumption. However, our Rademacher complexity bound in Theorem 6 also applies to non-uniform distributions, and is meaningful for any distribution.

Compared to the results in Section 4, the result here is also superior in that the Rademacher complexity does not depend on the loss magnitude bound b_ℓ. Although this

factor does appear in a different term in the cited overall sample complexity bound (Theorem 5), we conjecture that its true effect is modest at best. This is in light of recent work, which imply that using particular online matrix completion algorithms, one can learn comparatively well in a transductive setting, without explicit boundedness assumptions (see Section 8).

Another interesting feature of Theorem 6 is that the Rademacher complexity falls off at the rate of $\mathcal{O}(1/|S|)$ rather than $\mathcal{O}(1/\sqrt{|S|})$. While such a "fast rate" is unusual in the inductive setting, here it is a natural outcome of the different modeling of the training data. This does not lead to a $\mathcal{O}(1/|S|)$ sample complexity bound, because the bound in Theorem 5 contains an additional low rate term $\mathcal{O}(1/\sqrt{|S|})$. However, it still leads to a better bound because the low rate term is not explicitly multiplied by functions of m, n or t.

6. Lower Bounds

The previous results showed that for $m \times n$ matrices (where $m \leq n$), $\mathcal{O}(t\sqrt{n})$ samples are sufficient for learning. In this section, we show that such a sample size is also necessary, in both the inductive and transductive settings, hence establishing the tightness of our bounds. We remark that this lower bound applies in the distribution-free case (where any distribution over the matrix entries is allowed), and hence does not contradict tighter upper-bounds, which hold under distributional assumptions, such as in Negahban and Wainwright (2010); Candès and Tao (2009); Srebro and Shraibman (2005). Also, this lower bound result is not really new, and a different version of it appears in Hazan et al. (2012) for the inductive setting. However, we reproduce it here due to its relevance, and since it resolved an open problem posed in a preliminary version of our paper (Shamir and Shalev-Shwartz, 2011). For simplicity, we will consider $n \times n$ matrices.

The lower bound is based on the following theorem:

Theorem 7 *Fix a parameter $t \in \left[n, n^{3/2}\right]$, and consider the class of $n \times n$ matrices $\mathcal{W} = \{W : \|W\|_{tr} \leq t\}$. Let $\mathcal{W}'$ be the set of all matrices whose entries are $\{-1, +1\}$ on the first $\lfloor t\sqrt{n} \rfloor$ rows, and 0 everywhere else. Then $\mathcal{W}' \subset \mathcal{W}$.*

Proof We need to show that any matrix $W \in \mathcal{W}'$ has trace-norm at most t. To see why, note that W is non-zero on only $\lfloor t/\sqrt{n} \rfloor$, hence its rank is at most $\lfloor t/\sqrt{n} \rfloor$. Letting $\|\cdot\|_F$ denote the Frobenius norm and using the inequality $\|A\|_{tr} \leq \sqrt{\mathrm{rank}(A)}\|A\|_F$, we have

$$\|W\|_{tr} \;\leq\; \sqrt{\mathrm{rank}(W)}\|W\|_F \;\leq\; \sqrt{\frac{t}{\sqrt{n}}}\sqrt{n * \frac{t}{\sqrt{n}}} \;=\; t.$$

■

We now argue, based on this theorem, that learning is impossible unless the sample size $|S|$ is at least $\Omega(t\sqrt{n})$, matching our previous upper bounds (which were smaller than 1 only when $|S| > \Omega(t\sqrt{n})$). To see why, assume w.l.o.g. that S lies in the first $\left\lfloor \frac{t}{2\sqrt{n}} \right\rfloor$ rows, and let us consider first the inductive setting. Suppose we are asked to predict the values of a

matrix X, with respect to a uniform distribution over its entries in the first $\lfloor t/\sqrt{n} \rfloor$ rows, and where the value of each of these entries was independently chosen from $\{-1,+1\}$. If we are given a sample of size $|S| \leq \lfloor t\sqrt{n}/2 \rfloor$ from this matrix, it means that most of the relevant binary entries remain unobserved. Moreover, they were chosen uniformly at random, hence we have no way to predict their value. For any reasonable loss function, this would imply an expected error which is at least constant. In contrast, by the theorem above, there exists some $W \in \mathcal{W}$ which predicts perfectly all of these entries, and its expected error would be zero. In other words, for any algorithm returning a (possibly randomized) predicted matrix W,

$$\mathbb{E}_W \left[\mathbb{E}_{i,j}[\ell(W_{i,j}, X_{i,j})] - \inf_{W \in \mathcal{W}} \mathbb{E}_{i,j}[\ell(W_{i,j}, X_{i,j})] \right] \geq c,$$

for some constant $c > 0$, and hence we are unable to learn with such a sample size. A similar result holds in the transductive setting: If S is supported on those first $\lfloor t/\sqrt{n} \rfloor$ rows, and is randomly split to S_{train} and S_{test}, we have no way to predict the entries of S_{test} given S_{test}, and would achieve constant expected error. Moreover, using standard VC dimension techniques, even for larger sample sizes $|S|$ the attainable error cannot be better than $\Omega(\sqrt{t\sqrt{n}/|S|})$.

7. Should Boundedness be Enforced?

As mentioned earlier in the paper, we often know the range of entries to be predicted (e.g., 1 to 5 for movie rating prediction). The results of Section 4 suggest that in the inductive model, some sort of boundedness seems essential to get non-trivial results. In the transductive model, boundedness also plays a smaller role, by appearing in the final sample-complexity bound (Theorem 5), although not in the Rademacher complexity bound (Theorem 6). These results suggest the natural idea of incorporating into the learning algorithm the prior knowledge we have on the range of entries. Indeed, several recent papers have considered the possibility of directly learning a model $\phi \circ W$, where ϕ is usually a sigmoid function (Salakhutdinov and Mnih, 2007; Ma et al., 2008; Piotte and Chabbert, 2009; Kozma et al., 2009). Another common practice (not just with trace-norm regularization) is to clip the learned matrix entries to the known range. Our theoretical results are not sufficiently refined to understand the precise effect of boundedness, so it is of interest to understand experimentally how much clipping or enforcing boundedness helps the learning process. We note that while bounded models have been tested experimentally, we could not find in prior literature a clear empirical study of their effect, in the context of trace-norm regularization.

We conducted experiments on two standard matrix completion data sets,[2] movielens100K and movielens1M. movielens100K contains 10^5 ratings of 943 users for 1770 movies, while movielens1M contains 10^6 ratings of 6040 users for 3706 movies. All ratings are in the range $[1,5]$. For each data set, we performed a random $80\% - 20\%$ of the data to obtain a training set and a test set. We considered two hypothesis classes: trace-norm constrained matrices $\{W : \|W\|_{tr} \leq t\}$, and bounded trace-norm constrained matrices $\{\phi \circ W : \|W\|_{tr} \leq t\}$, where ϕ is a sigmoid function interpolating between 1 and 5. For each hypothesis class, we

2. These data sets are taken from `www.grouplens.org/node/73`

trained a trace-norm regularized algorithm using the squared loss. Specifically, we used the common approach of stochastic gradient descent on a factorized representation $W = U^\top V$: First, we note that for any t, minimizing $\sum_{(i,j)\in S}(X_{i,j} - W_{i,j})^2$ over all $W : \|W\|_{tr} \leq t$ is equivalent to minimizing

$$\sum_{(i,j)\in S}(X_{i,j} - W_{i,j})^2 + \lambda\|W\|_{tr} \tag{4}$$

over all matrices W, where λ is some suitable soft-regularization parameter. Second, we use the fact that the trace norm can also be written as $\|W\|_{tr} = \min_{W=U^\top V}\frac{1}{2}\left(\|U\|_F^2 + \|V\|_F^2\right)$, so minimizing *Equation* (4) over W is equivalent to minimizing

$$\sum_{(i,j)\in S}\left(X_{i,j} - U_i^\top V_j\right)^2 + \frac{\lambda}{2}\left(\|U\|_F^2 + \|V\|_F^2\right) \tag{5}$$

over U, V. Similarly, for learning bounded models, we can find U, V which minimize

$$\sum_{(i,j)\in S}\left(X_{i,j} - \phi(U_i^\top V_j)\right)^2 + \frac{\lambda}{2}\left(\|U\|_F^2 + \|V\|_F^2\right). \tag{6}$$

We note that both problems are non-convex, although for the formulation in Equation (5), it is possibly to show there are any local minimum is also a global one.

Tuning of λ was performed with a validation set. Note that in practice, for computational reasons, one often constrains U and V to have a bounded number of rows. However, this forces W to have low rank, which is an additional complexity control. Since our goal is to study the performance of trace-norm constrained matrices, and not matrices which are also low-rank, we did not constrain U, V in this manner. The downside of this is that we were unable to perform experiments on very large-scale data sets, such as Netflix, and that is why we focused on the more modest-sized movielens100K and movielens1M data sets.

To estimate the performance of the learned matrix W on the test set, we used two measures which are standard in the literature: the root-mean-squared-error (RMSE),

$$\sqrt{\sum_{(i,j)\in S_{test}}\frac{(W_{i,j} - X_{i,j})^2}{|S_{test}|}},$$

and the normalized-mean-absolute-error (NMAE),

$$\sum_{i,j\in S_{test}}\frac{|W_{i,j} - X_{i,j}|}{r|S_{test}|},$$

where r is the range of possible values in X ($5 - 1 = 4$ for our data sets).

The experiments were repeated 5 times over random train-test splits of the data, and the results are summarized in Table 1. From the table, we see that in almost all cases, clipping and bounding lead to a statistically significant improvement. However, note that in absolute terms, the improvement is rather modest, especially with the NMAE measure which is less sensitive to large mispredictions. This accords with our theoretical results: boundedness seems to be an important and useful property, but in the transductive model (corresponding to our experiments) it plays only a modest role.

	100K (NMAE)	100K (RMSE)	1M (NMAE)	1M (RMSE)
unclipped	0.1882 ± 0.0005	0.9543 ± 0.0019	0.1709 ± 0.0003	0.8670 ± 0.0016
clipped	0.1874 ± 0.0005	0.9486 ± 0.0018	0.1706 ± 0.0002	0.8666 ± 0.0016
bounded	0.1871 ± 0.0004	0.9434 ± 0.0023	0.1698 ± 0.0002	0.8618 ± 0.0017
Δ Clipping ($*10^{-3}$)	0.77 ± 0.07	5.7 ± 0.6	0.33 ± 0.01	0.48 ± 0.04
Δ Bounding ($*10^{-3}$)	0.3 ± 0.4	5.2 ± 1.5	0.79 ± 0.02	4.8 ± 0.1

Table 1: Error on test set (mean and standard deviation over 5 repeats of the experiment). The columns refer to the data set (movielens100K or movielens1M) and the performance measure used (NMAE or RMSE). The first two rows refer to the results using the 'unbounded' model as in Equation (5), with the output used as-is or clipped to the range $[1-5]$. The third row refers to the results using the 'bounded' model as in Equation (6). The fourth row is the improvement in test error by clipping the predictions after learning (i.e., the difference between the first and second row). The fifth row is the additional improvement achieved by using a bounded model (i.e., the difference between the second and third row).

Empirically, one would have expected the use of bounded models to help a lot (in absolute terms), if learning just trace-norm constrained matrices (without clipping/bounding) leads to many predictions being outside the interval $[1, 5]$, in which we know the ratings lie. But indeed, this does not seem to be the case. Table 2 shows the prediction with largest magnitude, over all entries in the test set, as well as the percentage of predictions which fall outside the $[1, 5]$ interval. It is clearly evident that such out-of-interval predictions are relatively rare, and this explains why the bounding and clipping only leads to modest improvements.

	100K	1M
largest value	5.95 ± 0.35	6.13 ± 0.16
% outside interval	0.69 ± 0.05	0.79 ± 0.01

Table 2: Out-of-Interval Values

We emphasize that our results should only be interpreted in the context of pure trace-norm regularization. There are many other approaches to matrix completion, and it is quite possible that using bounded models has more or less impact in the context of other approaches or for other application domains.

8. Follow-Up Work

Since the preliminary version of this paper appeared (Shamir and Shalev-Shwartz, 2011), several related works have been published. In this section, we survey these results, and discuss how they relate to the current work and the insights it provides.

While this work focuses on a stochastic setting, a closely related problem has been matrix completion in an *online* setting. In online learning (Cesa-Bianchi and Lugosi, 2006; Shalev-Shwartz, 2012), rather than having examples sampled from a stochastic process, the examples arrive in an online fashion and are arbitrary, possibly provided by an all-powerful adversary. The goal in this setting is to minimize *regret*, namely the difference between the learner's loss and that of the best single hypothesis from some hypothesis class. In the context of matrix completion, this can be modeled as a sequential game where at each round a matrix entry is arbitrarily chosen, and the learner needs to predict its value. The actual entry value is then revealed, and the learner suffers some loss (such as the absolute difference between the prediction and actual value). In our case, the regret can be measured with respect to the class of matrices with bounded trace-norm. Note that this setting is generally harder than our stochastic setting, since the entry are chosen arbitrarily rather than in a stochastic manner, and it is known that in general, any online learning algorithm can be converted to a learning algorithm in a stochastic setting, with similar guarantees. Despite the difference between the settings, regret guarantees in the online learning setting are often strikingly similar to sample complexity guarantees in the stochastic learning setting.

The problem of online matrix completion with trace-norm bounded matrix has been dealt with in several recent works. Interestingly, the same insights provided in our work — the importance of entry boundedness or a transductive model — were crucial for attaining online learning algorithms. Considering $n \times m$ matrices with bounded trace-norm t as well as bounded entries, Hazan et al. (2012) showed that one can efficiently obtain vanishing regret after $\mathcal{O}(t\sqrt{n})$ rounds (assuming $m \leq n$). Note that this parallels our sample complexity guarantees in a stochastic setting (assuming bounded entries), which imply learnability for sample size $\mathcal{O}(t\sqrt{n})$. Alternatively, if one considers a transductive online setting (where each entry can be chosen only once), Cesa-Bianchi and Shamir (2011) showed that one can also efficiently obtain vanishing regret after $\mathcal{O}(t\sqrt{n})$ rounds. In Rakhlin et al. (2012) this was shown to be possible for Lipschitz-continuous losses, even if the entries are not explicitly bounded — the transductive setting alone suffices to achieve results of this order.

Another recent related work is Shalev-Shwartz et al. (2011), which deals with a supposedly different problem: Approximately solving convex optimization problems over the (non-convex) domain of low-rank matrices. However, one of their results provides an alternative justification of our $\mathcal{O}(t\sqrt{n})$ sample complexity guarantee, for the case of bounded trace-norm matrices whose entries are clipped to a bounded range (Theorem 3). To sketch the argument, Shalev-Shwartz et al. (2011, Section 4) show that if we have $|S|$ observed entries in our matrix, then for every matrix W with bounded trace-norm $\|W\|_{tr}$, there exists a low-rank matrix $\bar{W}$, with rank $\mathcal{O}(\|W\|_{tr}^2/|S|)$, which approximates W arbitrarily well in terms of average loss over the observed entries. Since a matrix of rank r is parameterized by $\mathcal{O}(rn)$ parameters, it follows that the generalization error of clipped r-rank matrices is arbitrarily small when $|S| \geq \tilde{\Omega}(rn)$. This indirectly provides a generalization error bound for our original matrix W. Plugging in $r = \|W\|_{tr}^2/|S|$, and noting that in our case $\|W\|_{tr} = n$, we get that learnability is possible for a sample of size $|S| \geq \tilde{\Omega}(n^{3/2}) = \tilde{\Omega}(t\sqrt{n})$.

Finally, we note that several recent works explored the possibility of replacing the standard trace-norm constraint by other matrix norms. These include the max-norm (Srebro et al., 2004; Lee et al., 2010); weighted trace-norm (Salakhutdinov and Srebro, 2010) and smoothed/empirical variants (Foygel et al., 2011); and 'local' max-norms (Foygel et al.,

2012). An important motivation of these works is that they allow us to learn non-trivial classes of matrices, with a sample complexity of $\mathcal{O}(n)$ — smaller than earlier trivial results for the trace-norm and the $\mathcal{O}(n^{3/2})$ results we obtain here (when the trace-norm is $\Theta(n)$). Essentially, this is achieved by using classes of matrices which are less rich than trace-norm-bounded one, hence are statistically easier to learn.

9. Proofs of Upper Bounds

In our proofs, we use $\|\cdot\|_{sp}$ to denote the spectral norm of matrices, which is well-known to be the dual of the trace-norm (see for instance Fazel et al., 2001).

Our proofs utilize the following two theorems, which bounds the expected spectral norm $\|\cdot\|_{sp}$ of random matrices.

Theorem 8 (Latała, 2005) *Let Z be a matrix composed of independent zero-mean entries. Then for some fixed constant C, $\mathbb{E}[\|Z\|_{sp}]$ is at most*

$$C\left(\max_i \sqrt{\sum_j \mathbb{E}[Z_{i,j}^2]} + \max_j \sqrt{\sum_i \mathbb{E}[Z_{i,j}^2]} + \sqrt[4]{\sum_{i,j} \mathbb{E}[Z_{i,j}^4]}\right).$$

Theorem 9 (Seginer, 2000) *Let A be an arbitrary $m \times n$ matrix, such that $m, n > 1$. Let Z denote a matrix composed of independent zero-mean entries, such that $Z_{i,j} = A_{i,j}$ with probability $1/2$ and $Z_{i,j} = -A_{i,j}$ with probability $1/2$. Then for some fixed constant C, $\mathbb{E}[\|A\|_{sp}]$ is at most*

$$C\sqrt[4]{\log(\min\{m,n\})}\max\left\{\max_i \sqrt{\sum_j A_{i,j}^2}, \max_j \sqrt{\sum_i A_{i,j}^2}\right\}$$

9.1 Proof of Theorem 2

We write $R_S(\ell \circ \mathcal{W})$ as

$$\frac{1}{|S|}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{W\in\mathcal{W}} \sum_{i,j} \Gamma_{i,j}\ell(W_{i,j}, X_{i,j})\right], \tag{7}$$

where Γ is a matrix whose (i,j)-th entry is defined as $\sum_{\alpha: i_\alpha = i, j_\alpha = j} \sigma_\alpha$. As discussed in Section 3, a standard analysis will proceed to reduce this to

$$\frac{1}{|S|}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{W\in\mathcal{W}} \sum_{i,j} \Gamma_{i,j} W_{i,j}\right], \tag{8}$$

but this leads to a trivial bound. However, examining the analysis in Section 3, we see that the problem is when a single entry is "hit" many times in the sample. This will cause the magnitude of that entry to be very large (as much as $\Theta(\sqrt{|S|})$), and as a result make Equation (8) as large as $\Theta(t/\sqrt{|S|})$. However, recall that our original goal is to bound Equation (7), not Equation (8), and in Equation (7) we have the loss operator, which is

bounded by a constant b_ℓ. Therefore, even if some $\Gamma_{i,j}$ has a large value, it can only be multiplied by a factor as large as b_ℓ, and *not* the trace-norm bound t. This observation is the key for our analysis.

Intuitively, instead of going directly from Equation (7) to Equation (8), we first decompose Γ into two matrices Y and Z, where Y contains the "heavily-hit" entries, and Z the "lightly-hit" entries, where the two types of entries are differentiated according to some threshold p. We perform a different type analysis for each matrix, and then tune p appropriately to get the desired result.

More formally, given i, j, let $h_{i,j}$ be the number of times the sample S hits entry i, j, or more precisely $h_{i,j} = |\{\alpha : i_\alpha = i, j_\alpha = j\}|$. Let $p > 0$ be an arbitrary parameter to be specified later, and define

$$Y_{i,j} = \begin{cases} \Gamma_{i,j} & h_{i,j} > p \\ 0 & h_{i,j} \leq p \end{cases} \qquad Z_{i,j} = \begin{cases} 0 & h_{i,j} > p \\ \Gamma_{i,j} & h_{i,j} \leq p. \end{cases} \tag{9}$$

Clearly, we have $\Gamma = Y + Z$. Thus, we can upper bound the Rademacher complexity by

$$\frac{1}{|S|}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{W\in\mathcal{W}}\sum_{i,j} Y_{i,j}\ell(W_{i,j}, X_{i,j})\right] + \frac{1}{|S|}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{W\in\mathcal{W}}\sum_{i,j} Z_{i,j}\ell(W_{i,j}, X_{i,j})\right]. \tag{10}$$

Since $|\ell(W_{i,j}, X_{i,j})| \leq b_\ell$, the first term can be upper bounded by

$$\frac{1}{|S|}\mathbb{E}_{\boldsymbol{\sigma}}\left[b_\ell \sum_{i,j} |Y_{i,j}|\right] = \frac{b_\ell}{|S|}\mathbb{E}_{\boldsymbol{\sigma}}[\|Y\|_1]. \tag{11}$$

Using the Rademacher contraction principle,[3] the second term in Equation (10) can be upper bounded by

$$\frac{l_\ell}{|S|}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{W\in\mathcal{W}}\sum_{i,j} Z_{i,j}W_{i,j}\right].$$

Applying Hölder's inequality, and using the fact that the spectral norm $\|\cdot\|_{sp}$ is the dual to the trace norm $\|\cdot\|_{tr}$, we can upper bound the above by

$$\frac{l_\ell}{|S|}\mathbb{E}_{\boldsymbol{\sigma}} \sup_{W\in\mathcal{W}} [\|Z\|_{sp}\|W\|_{tr}] = \frac{l_\ell t}{|S|}\mathbb{E}_{\boldsymbol{\sigma}} [\|Z\|_{sp}]. \tag{12}$$

Combining this with Equation (11) and substituting into Equation (10), we get an upper bound of the form

$$\frac{b_\ell}{|S|}\mathbb{E}_{\boldsymbol{\sigma}} [\|Y\|_1] + \frac{l_\ell t}{|S|}\mathbb{E}_{\boldsymbol{\sigma}} [\|Z\|_{sp}].$$

Using Lemma 10 and Lemma 11, which are given below, we can upper bound this by

$$\frac{b_\ell}{\sqrt{p}} + \frac{2.2Cl_\ell t\sqrt{p}(\sqrt{m}+\sqrt{n})}{|S|},$$

3. Strictly speaking, we use a slight generalization of it, where the loss function is allowed to differ w.r.t. every $W_{i,j}$ — see Meir and Zhang (2003, Lemma 5).

where p is the parameter used to define Y and Z in Equation (9). Choosing $p = \frac{|S| b_\ell}{2.2 C l_\ell t(\sqrt{m}+\sqrt{n})}$, we get the bound in the theorem.

Lemma 10 *Let Y be a random matrix defined as in Equation (9). Then*

$$\mathbb{E}[\|Y\|_1] \leq \mathbb{E}\left[\sum_{i,j:h_{i,j}>p} \sqrt{h_{i,j}}\right] \leq \frac{|S|}{\sqrt{p}}$$

Proof $\mathbb{E}[\|Y\|_1]$ equals

$$\mathbb{E}\left[\sum_{i,j:h_{i,j}>p} |\Gamma_{i,j}|\right] = \mathbb{E}\left[\mathbb{E}\left[\sum_{i,j:h_{i,j}>p}\left(\left|\sum_{\alpha:(i_\alpha,j_\alpha)=(i,j)} \sigma_\alpha\right|\right) \;\middle|\; \{h_{i,j}\}\right]\right]$$

The expression inside the absolute value is the sum of $h_{i,j}$ i.i.d. random variables, and it is easily seen that its expected absolute value is at most $\sqrt{h_{i,j}}$. Therefore, we can upper bound the above by $\mathbb{E}[\sum_{i,j:h_{i,j}>p} \sqrt{h_{i,j}}]$. We can further upper bound it, in a manner which does not depend on the values of $h_{i,j}$, by

$$\max_{c\in\{1,\ldots,mn\}} \max_{h_1,\ldots,h_c\in\mathbb{R}:\forall i\ h_i>p, \sum_{i=1}^c h_i=|S|} \sum_{i=1}^{c} \sqrt{h_i}.$$

Note that the constraints imply that

$$|S| = \sum_{i=1}^{c} h_i \geq \sqrt{p} \sum_{i=1}^{c} \sqrt{h_i},$$

so $\sum_{i=1}^c \sqrt{h_i}$ can be at most $|S|/\sqrt{p}$ as required. ■

Lemma 11 *Let Z be a random matrix defined as in Equation (9). Then the expected spectral norm $\mathbb{E}_{\boldsymbol{\sigma}}[\|Z\|_{sp}]$ is at most*

$$C\left(\max_i \sqrt{\sum_{j:h_{i,j}\leq p} h_{i,j}} + \max_j \sqrt{\sum_{i:h_{i,j}\leq p} h_{i,j}} + \sqrt[4]{3\sum_{i,j:h_{i,j}\leq p} h_{i,j}^2}\right),$$

where C is the universal constant which appears in the main theorem of Latała (2005). Moreover, this quantity can be upper bounded by $2.2C\sqrt{p}\,(\sqrt{m}+\sqrt{n})$

Proof With $h_{i,j}$ held fixed, Z is a random matrix composed of independent entries. By using Theorem 8, we only need to analyze $\mathbb{E}[Z_{i,j}^2]$ and $\mathbb{E}[Z_{i,j}^4]$. For any i,j, if $h_{i,j} \leq p$ then $Z_{i,j}$ is a sum of $h_{i,j}$ i.i.d. variables taking values in $\{-1,+1\}$. Therefore, $\mathbb{E}[Z_{i,j}^2] = h_{i,j}$ and $\mathbb{E}[Z_{i,j}^4] \leq 3h_{i,j}^2$. Plugging into Theorem 8 yields the first part of the lemma. To get the second part, we can upper bound the right-hand side of the first part by

$$\begin{aligned} C\sqrt{p}\left(\sqrt{m}+\sqrt{n}+\sqrt[4]{3mn}\right) &\leq C\sqrt{p}\left(\sqrt{m}+\sqrt{n}+\sqrt[4]{3/2}(\sqrt{m}+\sqrt{n})\right) \\ &\leq 2.2C\sqrt{p}\left(\sqrt{m}+\sqrt{n}\right). \end{aligned}$$

■

9.2 Proof of Theorem 3

We can rewrite the definition of $R_S(\ell \circ \mathcal{W})$ (see Equation 3) as

$$\frac{1}{|S|} E_{\boldsymbol{\sigma}} \left[\sup_{W \in \mathcal{W}} \sum_{i,j} \Gamma_{i,j} \ell(W_{i,j}, X_{i,j}) \right],$$

where Γ is a matrix defined as $\Gamma_{i,j} = \sum_{\alpha: i_\alpha = i, j_\alpha = j} \sigma_\alpha$. Using the Rademacher contraction principle (as in Meir and Zhang, 2003, Lemma 5), this is at most

$$\frac{l_\ell}{|S|} E_{\boldsymbol{\sigma}} \left[\sup_{W \in \mathcal{W}} \sum_{i,j} \Gamma_{i,j} W_{i,j} \right]. \tag{13}$$

Decomposing $\Gamma = Y + Z$ as in Equation (9) according to a parameter p, we can upper bound the Rademacher complexity by

$$\frac{l_\ell}{|S|} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{W \in \mathcal{W}} \sum_{i,j} Y_{i,j} W_{i,j} \right] + \frac{l_\ell}{|S|} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{W \in \mathcal{W}} \sum_{i,j} Z_{i,j} W_{i,j} \right]. \tag{14}$$

By definition of $\mathcal{W}$, $|W_{i,j}| \leq b_\phi$, so the first term can be upper bounded by

$$\frac{l_\ell}{|S|} \mathbb{E}_{\boldsymbol{\sigma}} \left[b_\phi \sum_{i,j} |Y_{i,j}| \right] = \frac{l_\ell b_\phi}{|S|} \mathbb{E}_{\boldsymbol{\sigma}} [\|Y\|_1]. \tag{15}$$

The second term in Equation (14) equals

$$\frac{l_\ell}{|S|} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{W: \|W\|_{tr} \leq t} \sum_{i,j} Z_{i,j} \phi(W_{i,j}) \right] \leq \frac{l_\ell l_\phi}{|S|} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{W: \|W\|_{tr} \leq t} \sum_{i,j} Z_{i,j} W_{i,j} \right],$$

again by the Rademacher contraction principle. Applying Hölder's inequality, and using the fact that the spectral norm $\|\cdot\|_{sp}$ is the dual to the trace norm $\|\cdot\|_{tr}$, we can upper bound the above by

$$\frac{l_\ell l_\phi}{|S|} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{W: \|W\|_{tr} \leq t} \|Z\|_{sp} \|W\|_{tr} \right] = \frac{l_\ell l_\phi t}{|S|} \mathbb{E}_{\boldsymbol{\sigma}} \left[\|Z\|_{sp} \right].$$

Combining this with Equation (15) and substituting into Equation (14), we get an upper bound of the form

$$\frac{l_\ell b_\phi}{|S|} \mathbb{E}_{\boldsymbol{\sigma}} \left[\|Y\|_1 \right] + \frac{l_\ell l_\phi t}{|S|} \mathbb{E}_{\boldsymbol{\sigma}} \left[\|Z\|_{sp} \right].$$

Using Lemma 10 and Lemma 11, we can upper bound this by

$$\frac{l_\ell b_\phi}{\sqrt{p}} + \frac{2.2 C l_\ell l_\phi t \sqrt{p} (\sqrt{m} + \sqrt{n})}{|S|},$$

where p is the parameter used to define Y and Z in Equation (9). Choosing $p = \frac{|S| b_\phi}{2.2 C l_\phi t (\sqrt{m} + \sqrt{n})}$, we get the bound in the theorem.

9.3 Proof of Theorem 4

Before we begin, we will need the following technical result:

Lemma 12 *The dual of the norm* $\|W\| = \max\{\|W\|_{tr}/t, \|W\|_\infty/b\}$ *equals*

$$\|\Gamma\|_* = \min_{Y+Z=\Gamma} b\|Y\|_1 + t\|Z\|_{sp},$$

where $\|Y\|_1 = \sum_{i,j} |Y_{i,j}|$ *and* $\|Z\|_{sp}$ *is the spectral norm of* Z*.*

It is possible to prove the lemma directly using duality of infimal convolution. However, for the sake of completeness we give below a self-contained proof.

Proof By definition of a dual norm, we have

$$\|\Gamma\|_* = \sup_{W:\|W\|\le 1} \langle \Gamma, W\rangle,$$

and our goal is to show that

$$\sup_{W:\|W\|\le 1} \langle \Gamma, W\rangle = \min_{Y+Z=\Gamma} b\|Y\|_1 + t\|Z\|_{sp}.$$

First, we recall that the dual norm of $\|W\|_{tr}$ is the spectral norm $\|W\|_{sp}$, and the dual of $\|W\|_\infty$ is the 1-norm $\|W\|_1 = \sum_{i,j} |W_{i,j}|$. Now, for any Y, Z such that $Y + Z = \Gamma$, we have by Hölder's inequality that

$$\begin{aligned}
\sup_{W:\|W\|\le 1} \langle \Gamma, W\rangle &= \sup_{W:\|W\|\le 1} \langle Y, W\rangle + \langle Z, W\rangle \\
&\le \sup_{W:\|W\|\le 1} \|Y\|_1\|W\|_\infty + \|Z\|_{sp}\|W\|_{tr} \\
&\le b\|Y\|_1 + t\|Z\|_{sp}.
\end{aligned}$$

This holds for any Y, Z, and in particular

$$\sup_{W:\|W\|\le 1} \langle \Gamma, W\rangle \le \min_{Y+Z=\Gamma} b\|Y\|_1 + t\|Z\|_{sp}. \tag{16}$$

It remains to show the opposite direction, namely

$$\sup_{W:\|W\|\le 1} \langle \Gamma, W\rangle \ge \min_{Y+Z=\Gamma} b\|Y\|_1 + t\|Z\|_{sp}.$$

To show this, let W^* be the matrix which maximizes the inner product above. We know that $\|W^*\| \le 1$, which means that either $\|W^*\|_\infty \le b$, or $\|W^*\|_{tr} \le t$. If $\|W^*\|_\infty \le b$, it follows that

$$\sup_{W:\|W\|\le 1} \langle \Gamma, W\rangle = \sup_{W:\|W\|_\infty \le b} \langle \Gamma, W\rangle = b\|\Gamma\|_1 \ge \min_{Y+Z=\Gamma} b\|Y\|_1 + t\|Z\|_{sp}.$$

In the other case, if $\|W^*\|_{tr} \le t$, it follows that

$$\sup_{W:\|W\|\le 1} \langle \Gamma, W\rangle = \sup_{W:\|W\|_{tr} \le t} \langle \Gamma, W\rangle = t\|\Gamma\|_{sp} \ge \min_{Y+Z=\Gamma} b\|Y\|_1 + t\|Z\|_{sp}.$$

So in either case,

$$\sup_{W:\|W\|\leq 1} \langle \Gamma, W \rangle \geq \min_{Y+Z=\Gamma} b\|Y\|_1 + t\|Z\|_{sp}.$$

Combining this with Equation (16), the result follows. ■

We now turn to the proof of Theorem 4 itself. Since $\ell(W_{i,j}, X_{i,j})$ is assumed to be l_ℓ-Lipschitz, we can use the Rademacher contraction principle to upper bound $R_S(\ell \circ \mathcal{W})$ by

$$l_\ell \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{W \in \mathcal{W}} \frac{1}{|S|} \sum_{\alpha=1}^{|S|} \sigma_\alpha W_{i_\alpha, j_\alpha} \right] = \frac{l_\ell}{|S|} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{W \in \mathcal{W}} \sum_{i,j} \Gamma_{i,j} W_{i,j} \right],$$

where Γ is a matrix defined as $\Gamma_{i,j} = \sum_{\alpha: i_\alpha = i, j_\alpha = j} \sigma_\alpha$.

Thinking of Γ, W as vectors, the equation above is the expected supremum of an inner product between Γ and W. By Hölder's inequality, we can upper bound this by

$$\frac{l_\ell}{|S|} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{W \in \mathcal{W}} \|\Gamma\|_* \|W\| \right] \tag{17}$$

for any norm $\|\cdot\|$ and its dual norm $\|\cdot\|_*$. In particular, we will choose the norm $\|W\| = \max\{\|W\|_{tr}/t, \|W\|_\infty/b\}$. Note that by definition of W, $\sup_{W \in \mathcal{W}} \|W\| \leq 1$. Also, by Lemma 12,

$$\|\Gamma\|_* = \min_{Y+Z=\Gamma} b\|Y\|_1 + t\|Z\|_{sp},$$

where $\|Y\|_1 = \sum_{i,j} |Y_{i,j}|$, and $\|Z\|_{sp}$ is the spectral norm of Z . Thus, we can upper bound Equation (17) by

$$\frac{l_\ell}{|S|} \mathbb{E}_\Gamma \left[\min_{Y+Z=\Gamma} b\|Y\|_1 + t\|Z\|_{sp} \right]. \tag{18}$$

Recall that Γ is random matrix, where each entry is the sum of Rademacher variables. Let $h_{i,j}$ denote the number of variables 'hitting' entry (i,j) — formally, $h_{i,j} = |\{\alpha : (i_\alpha = i, j_\alpha = j\}|$. We can upper bound Equation (18) by replacing the optimal decomposition of Γ into Y, Z by any fixed decomposition rule. In particular, for an arbitrary parameter p, we can decompose Γ into Y, Z as in Equation (9), and get an upper bound on Equation (18) of the form

$$\frac{l_\ell}{|S|} \left(b\mathbb{E}_\Gamma[\|Y\|_1] + t\mathbb{E}_\Gamma[\|Z\|_{sp}] \right). \tag{19}$$

Bounds for the two expectations are provided in Lemma 10 and Lemma 11. Plugging them in, we get

$$\frac{bl_\ell}{\sqrt{p}} + \frac{2.2 l_\ell C t \sqrt{p} \left(\sqrt{m} + \sqrt{n}\right)}{|S|}.$$

Choosing $p = \frac{b|S|}{2.2Ct(\sqrt{m}+\sqrt{n})}$ and simplifying, we get the bound in the theorem.

9.4 Proof of Theorem 6

We write $R_S(\ell \circ \mathcal{W})$ as

$$\frac{1}{|S|}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{W\in\mathcal{W}}\sum_{i,j}\Gamma_{i,j}\ell(W_{i,j},X_{i,j})\right],$$

where Γ is a matrix with $\sigma_{i,j}$ in its (i,j)-th entry, if $(i,j)\in S$, and 0 otherwise. By the Rademacher contraction property,[4] we can upper bound this by

$$\frac{l_\ell}{|S|}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{W\in\mathcal{W}}\sum_{i,j}\Gamma_{i,j}W_{i,j}\right].$$

By Hölder's inequality, this is at most

$$\frac{l_\ell}{|S|}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{W\in\mathcal{W}}\|\Gamma\|_{sp}\|W\|_{tr}\right]=\frac{l_\ell t}{|S|}\mathbb{E}_{\boldsymbol{\sigma}}\left[\|\Gamma\|_{sp}\right]. \tag{20}$$

The setting so far is rather similar to the one we had in the inductive setting (see the proof of any of the theorems in Section 4). But now, we need to bound just the expected spectral norm of Γ, which is guaranteed to have only a single Rademacher variable in each entry. By applying Theorem 8 on Equation (20), we get

$$R_S(\ell\circ\mathcal{W})\le Cl_\ell\frac{t\left(\sqrt{M}+\sqrt{N}+\sqrt[4]{|S|}\right)}{|S|}.$$

Since S can contain at most m and n indices for any single row and column respectively, and $\sqrt[4]{|S|}\le\sqrt[4]{mn}\le\frac{1}{2}\left(\sqrt{m}+\sqrt{n}\right)$, we can upper bound the above by $3Cl_\ell t\left(\sqrt{m}+\sqrt{n}\right)/(2|S|)$.

To get the other bound in the theorem, we apply Theorem 9 instead of Theorem 8 on Equation (20).

10. Discussion

In this paper, we analyzed the sample complexity of matrix completion with trace-norm regularization, obtaining the first non-trivial, distribution-free guarantees. Our results were based on either mild boundedness assumptions, or a switch from the standard inductive learning model to the transductive learning model. Moreover, we argue that such a transductive model may be a better way to model matrix completion as performed in practice, as it seems more natural and leads to a substantial difference in terms of obtainable results. We also discussed the issue of learning with bounded models, and provided an empirical study which indicates that these lead to a modest improvement in performance, in line with our theoretical findings. We also show that our results are essentially tight, and discuss some recent work which relates to the results and insights provided here.

One interesting open question arises from our experiments in Section 7. In all our experiments, minimizing the squared loss over the training data (with trace-norm regularization)

4. As in the inductive case, we use in fact a slight generalization where the loss function is allowed to differ w.r.t. every $W_{i,j}$, as in Meir and Zhang (2003, Lemma 5).

resulted in matrices whose entries have reasonably small values, even when boundedness was not enforced. This is probably an important factor in explaining why explicitly enforcing boundedness resulted in only a modest performance improvement. However, if boundedness is not enforced, there is no a-priori reason why the resulting matrix shouldn't have some very large values (up to the trace-norm constraint) in some of the test set entries. Thus, we may raise the following conjecture: If we minimize training loss over data, consisting of bounded entries, over trace-norm constrained matrices, then the resulting matrix will have bounded entries as well. If this conjecture holds, it means that enforcing boundedness will always lead to only a modest performance improvement.

Acknowledgements

We thank Nati Srebro and Ruslan Salakhutdinov for helpful discussions.

References

F. Bach. Consistency of trace-norm minimization. *Journal of Machine Learning Research*, 9:1019–1048, 2008.

P. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2003.

O. Bousquet Boucheron, S. and G. Lugosi. Theory of classification: A survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323 – 375, 2005.

E. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9, 2009.

E. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Trans. Inform. Theory*, 56(5):2053–2080, 2009.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

N. Cesa-Bianchi and O. Shamir. Efficient online learning via randomized rounding. In *NIPS*, 2011.

Ran El-Yaniv and Dmitry Pechyoni. Transductive rademacher complexity and its applications. *Journal of AI Research*, 35:193–234, 2009.

M. Fazel, H. Hindi, and S. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *American Control Conference, 2001. Proceedings of the 2001*, volume 6, pages 4734–4739. IEEE, 2001.

R. Foygel, R. Salakhutdinov, O. Shamir, and N. Srebro. Learning with the weighted trace-norm under arbitrary sampling distributions. In *NIPS*, 2011.

R. Foygel, N. Srebro, and R. Salakhutdinov. Matrix reconstruction with the local max norm. In *NIPS*, 2012.

E. Hazan, S. Kale, and S. Shalev-Shwartz. Near-optimal algorithms for online matrix prediction. In *COLT*, 2012.

R. Horn and C. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

M. Jaggi and M. Sulovský. A simple algorithm for nuclear norm regularized problems. In *ICML*, 2010.

L. Kozma, A. Ilin, and T. Raiko. Binary principal component analysis in the netflix collaborative filtering task. In *IEEE MLSP Workshop*, 2009.

R. Latała. Some estimates of norms of random matrices. *Proceedings of the AMS*, 133(5): 1273–1282, 2005.

J. Lee, B. Recht, R. Salakhutdinov, N. Srebro, and J. Tropp. Practical large-scale optimization for max-norm regularization. In *NIPS*, 2010.

H. Ma, H. Yang, M. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM*, 2008.

R. Meir and T. Zhang. Generalization error bounds for bayesian mixture algorithms. *Journal of Machine Learning Research*, 4:839–860, 2003.

S. Negahban and M. Wainwright. Restricted strong convexity and weighted matrix completion: Optimal bounds with noise. arXiv:1009.2118, 2010.

M. Piotte and M. Chabbert. The pragmatic theory solution to the netflix grand prize. Available at `http://www.netflixprize.com/assets/GrandPrize2009_BPC_PragmaticTheory.pdf`, 2009.

A. Rakhlin, O. Shamir, and K. Sridharan. Relax and randomize : From value to algorithms. In *NIPS*, 2012.

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, 2007.

R. Salakhutdinov and N. Srebro. Collaborative filtering in a non-uniform world: Learning with the weighted trace norm. In *NIPS*, 2010.

Y. Seginer. The expected norm of random matrices. *Combinatorics, Probability & Computing*, 9(2):149–166, 2000.

S. Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012.

S. Shalev-Shwartz, A. Gonen, and O. Shamir. Large-scale convex minimization with a low-rank constraint. In *ICML*, 2011.

O. Shamir and S. Shalev-Shwartz. Collaborative filtering with the trace norm: Learning, bounding, and transducing. In *COLT*, 2011.

N. Srebro. *Learning with Matrix Factorizations*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, August 2004.

N. Srebro and A. Shraibman. Rank, trace-norm and max-norm. In *COLT*, 2005.

N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In *NIPS*, 2004.

K. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Optimization Online*, 2009.

V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

Journal of Machine Learning Research 15 (2014) 3425-3446 Submitted 5/13; Revised 6/14; Published 10/14

Statistical Analysis of Metric Graph Reconstruction

Fabrizio Lecci LECCI@CMU.EDU
Alessandro Rinaldo ARINALDO@CMU.EDU
Larry Wasserman LARRY@CMU.EDU
Department of Statistics
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Editor: Matthias Hein

Abstract

A metric graph is a 1-dimensional stratified metric space consisting of vertices and edges or loops glued together. Metric graphs can be naturally used to represent and model data that take the form of noisy filamentary structures, such as street maps, neurons, networks of rivers and galaxies. We consider the statistical problem of reconstructing the topology of a metric graph embedded in $\mathbb{R}^D$ from a random sample. We derive lower and upper bounds on the minimax risk for the noiseless case and tubular noise case. The upper bound is based on the reconstruction algorithm given in Aanjaneya et al. (2012).

Keywords: metric graph, filament, reconstruction, manifold learning, minimax estimation

1. Introduction

We are concerned with the problem of estimating the topology of filamentary data structure. Data sets consisting of points roughly aligned along intersecting or branching filamentary paths embedded in 2 or higher dimensional spaces have become an increasingly common type of data in a variety of scientific areas. For instance, road reconstruction based on GPS traces, localization of earthquakes faults, galaxy reconstruction are all instances of a more general problem of estimating basic topological features of an underlying filamentary structure. The recent paper by Aanjaneya et al. (2012), upon which our work is based, contains further applications, as well as numerous references. To provide a more concrete example, consider Figure 1. The left hand side displays raw data portraying a neuron from the hippocampus of a rat (Gulyás et al., 1999). The data were obtained from NeuroMorpho.Org (Ascoli et al., 2007). The right hand side of the figure shows the output of the metric graph reconstruction obtained using the algorithm analyzed in this paper, originally proposed by Aanjaneya et al. (2012). The reconstruction, which takes the form of a graph, captures perfectly all the topological features of the neuron, namely, the relationship between the edges and vertices, the number of branching points and the degree of each node.

Metric graphs provide the natural geometric framework for representing intersecting filamentary structures. A metric graph embedded in a D-dimensional Euclidean space ($D \geq 2$) is a 1-dimensional stratified metric space. It consists of a finite number of points (0-dimensional strata) and curves (1-dimensional strata) of finite length, where the boundary

of each curve is given by a pair (of not-necessarily distinct) vertices (see the next section for a formal definition of a metric graph).

In this paper we study the problem of reconstructing the topology of metric graphs from possibly noisy data, from a statistical point of view. Specifically, we assume that we have a sample of points from a distribution supported on a metric graph or in a small neighborhood and we are interested in recovering the topology of the corresponding metric graph. To this end, we use the metric graph reconstruction algorithm given in Aanjaneya et al. (2012). Furthermore, in our theoretical analysis we characterize explicitly the minimal sample size required for perfect topological reconstruction as a direct function of parameters defining the shape of the metric graph, introduced in Section 2. This leads to an upper bound on the risk of topological reconstruction. Finally, we obtain a lower bound on the risk of topological reconstruction, which, in the noiseless case, almost matches the derived upper bound, indicating that the algorithm of Aanjaneya et al. (2012) behaves nearly optimally.

Outline. In Section 2 we formally define metric graphs, the statistical models we will consider and the assumptions we will use throughout. We will also describe several geometric quantities that are central to our analysis. Section 3 contains detailed analysis of the performance of algorithm of Aanjaneya et al. (2012) for metric graph reconstruction, under modified settings and assumptions. In Section 4 we derive lower and upper bounds for the minimax risk of metric graph reconstruction problem. In Section 5 we conclude with some final comments.

Related Work. The work most closely related to ours is Aanjaneya et al. (2012) which was, in fact, the motivation for our work. From the theoretical side, we replace the key assumption in Aanjaneya et al. (2012) of the sample being a (ε, R)-approximation to the underlying metric graph, by the milder assumption of the sample being dense in a neighborhood of the metric graph. Approximation and reconstruction of metric graphs has also been considered in Chazal and Sun (2013) and Ge et al. (2011). Metric graph reconstruction is related to the problem of estimating stratified spaces (basically, intersecting manifolds). Stratified spaces have been studied by a number of authors such as Bendich et al. (2010, 2012). A spectral method for estimating intersecting structures is given in Arias-Castro et al. (2011). There are a variety of algorithms for specific problems, for example, see Ahmed and Wenk (2012); Chen et al. (2010) for the reconstruction of road networks. Finally, Chernov and Kurlin (2013) derived an alternative algorithm that uses ideas from homology.

2. Background and Assumptions

The assumptions in Aanjaneya et al. (2012) lead to a reconstruction process that is aimed at capturing the intrinsic structure of the data and is somewhat oblivious to its extrinsic embedding. The authors assume that the sample comes with a metric that is close to the intrinsic metric of the underlying graph, by imposing a limit on the Gromov-Hausdorff distance between the two metrics. By considering data embedded in the Euclidean space and focusing on the topological aspect, we show that the notion of *dense sample* is sufficient to guarantee a correct reconstruction.

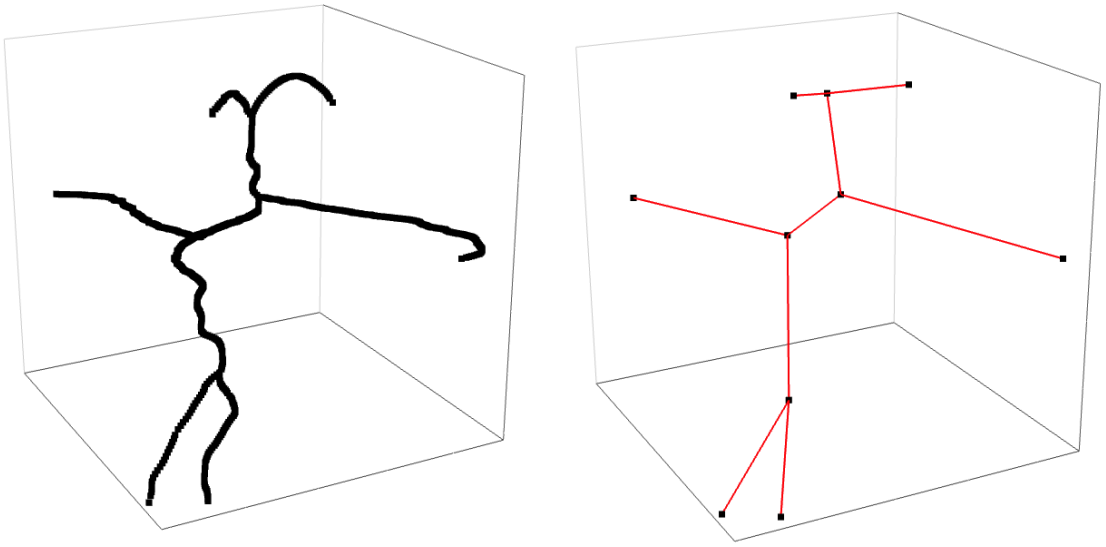


Figure 1: Left: Neuron cr22e from the hippocampus of a rat; NeuroMorpho.Org (Ascoli et al., 2007). Right: A metric graph reconstruction of the neuron.

In this section we provide background on metric graph spaces and describe the assumptions and the geometric parameters that we will be using throughout.
Informally, a metric graph is a collection of vertices and edges glued together in some fashion. Here we state the formal definitions of path metric space and metric graph. For more details see Aanjaneya et al. (2012) and Kuchment (2004).

Definition 1 *A metric space (G, d_G) is a path metric space if the distance between any pair of points is equal to the infimum of the lengths of the continuous curves joining them. A metric graph is a path metric space (G, d_G) that is homeomorphic to a 1-dimensional stratified space. A vertex of G is a 0-dimensional stratum of G and an edge of G is a 1-dimensional stratum of G.*

We will consider metric graphs embedded in $\mathbb{R}^D$. Note that, if one ignores the metric structure, namely the length of edges and loops, the shape or topology of a metric graph (G, d_G) is encoded by a graph, whose vertices and edges correspond to vertices and edges of G. Since we allow for two vertices to be connected by more than one edge we are actually dealing with pseudographs. We recall that an undirected pseudograph (V, E) is a set of vertices V, a multiset E of unordered pairs of (not necessarily distinct) vertices. To a given pseudograph we can associate a function $f : E \to V \times V$, which, when applied to an edge $e \in E$, simply extracts the vertices to which e is adjacent. Thus, if $e_1, e_2 \in E$ are such that $f(e_1) = f(e_2)$, then e_1 and e_2 are parallel edges. Similarly, if $e \in E$ is such that $f(e) = \{v, v\}$ for some $v \in V$, then e is a loop. For each pair $(u, v) \in V \times V$, let $\nu(u, v) = |f^{-1}(\{u, v\})|$ if $\{u, v\} \in E$ and 0 otherwise. In particular, $\nu(u, v)$ is the number of edges between u and v (or loops if $u = v$).

We say that a metric graph reconstruction algorithm perfectly recovers the topology of G if outputs a pseudograph isomorphic to the pseudograph representing the topology of G.

We now define some key quantities regarding the structure of a metric graph. We start with the definition of reach. Let M be a 1-dimensional manifold embedded in $\mathbb{R}^D$. Let $T_u M$ denote the 1-dimensional tangent space to M and let $T_u^{\perp}\mathrm{M}$ be the $(D-1)$-dimensional normal space.

Definition 2 *Define the fiber of size a at $u \in M$ to be $L_a(u, M) = T_u^{\perp} M \bigcap B(u, a)$, where $B(u, a)$ is the D-dimensional ball of radius a centered at u. If M has boundary $\{v_1, v_2\}$, the fiber of size a at v_i is defined as the limit of $L_a(u, M)$, as u approaches v_i in $M \backslash \{v_1, v_2\}$. The reach of M is the largest number τ such that the fibers $L_\tau(u, M)$ never intersect.*

The reach sets a limit on the curvature of a manifold. A manifold with large reach does not come too close to be self-intersecting. For example the reach of an arc of a circle is equal to its radius. The quantity $1/\tau$ is called the *condition number* in Niyogi et al. (2008). For more details see also Federer (1959); Chazal and Lieutier (2006); Genovese et al. (2012a). Each edge of a metric graph (G, d_G) can be seen as a 1-dimensional manifold with boundary. Let the *local reach* of metric graph G be the minimum reach associated to an edge of G.

When 2 edges intersect at a vertex v they create an angle, where the angle between two intersecting curves is formally defined as follows. Suppose that e_1 and e_2 intersect at x. Let $B(x, \epsilon)$ be the D-dimensional ball of radius ϵ centered at x. Let $\ell_1(\epsilon)$ be the line segment joining the two points x and $\partial B(x, \epsilon) \bigcap e_1$. Let $\ell_2(\epsilon)$ be the line segment joining the two points x and $\partial B(x, \epsilon) \bigcap e_2$. Let $\alpha_\epsilon(e_1, e_2)$ be the angle between $\ell_1(\epsilon)$ and $\ell_2(\epsilon)$. The angle between e_1 and e_2 is $\alpha(e_1, e_2) = \lim_{\epsilon \to 0} \alpha_\epsilon(e_1, e_2)$. We assume that, for each pair of intersecting edges e_1 and e_2, the angle $\alpha(e_1, e_2)$ is well-defined.

To control points far away in the graph distance, but close in the embedding space, we define

$$A_G = \{(x, x') \in G \times G : d_G(x, x') \geq \min(b, \tau\alpha)\},$$

where b is the shortest edge of G, τ is the local condition number and α is the smallest angle formed by two edges of G. We define the *global reach* as the infimum of the Euclidean distances among pairs of point in A_G, that is $\xi = \inf_{A_G} \|x - x'\|_2$.

Let (G, d_G) be a metric graph and, for a constant $\sigma \geq 0$, let $G_\sigma = \{y : \inf_{x \in G} ||x - y||_2 \leq \sigma\}$ be the σ-tube around G. If $\sigma = 0$, then, trivially, $G_\sigma = G$. Notice that G_σ is a set of dimension D if $\sigma > 0$.

We will use the assumption that the sample $\mathbb{Y}$ is sufficiently dense in G_σ with respect to the Euclidean metric, as formalized below.

Definition 3 *The sample $\mathbb{Y} = \{y_1, \ldots, y_n\} \subset G_\sigma \subset \mathbb{R}^D$ is $\frac{\delta}{2}$-dense in G_σ if for every $x \in G_\sigma$, there exists a $y \in \mathbb{Y}$ such that $\|x - y\|_2 < \frac{\delta}{2}$.*

The problem of metric graph reconstruction consists of reconstructing a metric graph G given a dense sample $\{y_1, \ldots, y_n\} = \mathbb{Y} \subset G_\sigma$ endowed with a distance $d_{\mathbb{Y}}$, which could be the D-dimensional Euclidean distance or some more complicate notion of distance. If $\sigma = 0$ we say that the sample $\mathbb{Y}$ is noiseless, while if $\sigma > 0$, we say that $\mathbb{Y}$ is a noisy sample.

Throughout our analysis we restrict the attention to metric graphs embedded in $\mathbb{R}^D$ that satisfy the following assumptions:

A1 The graphs have finite total length and are free of nodes of degree 2 (though they may contain vertices of degree 1 or 3 and higher).

A2 Each edge is a smooth embedded sub-manifold of dimension 1, of length at least $b > 0$ and with reach at least $\tau > 0$.

A3 Each pair of intersecting edges forms a well-defined angle of size at least $\alpha > 0$.

A4 The global reach is at least $\xi > 0$.

Assumptions A1 and A2 allow us to consider each edge of a metric graph as a single smooth curve. A3 and A4 are additional regularity conditions on the separation between different edges. Assumptions similar to A1-A4 are common in the literature. For different regularity conditions that allow for corners within an edge see, for example, Chazal et al. (2009) and Chen et al. (2010).

Let $\mathcal{G}$ be the set of metric graphs embedded in $\mathbb{R}^D$ that satisfy assumptions A1, A2, A3 and A4, involving the parameters b, α, τ, ξ. We consider two noise models:

Noiseless. We observe data $Y_1, \ldots, Y_n \sim P$, where $P \in \mathcal{P}$, a collection of probability distributions supported over metric graphs (G, d_G) in $\mathcal{G}$ having densities p with respect to the length of G bounded from below by a constant $a > 0$.

Tubular Noise. We observe data $Y_1, \ldots, Y_n \sim P_{G,\sigma}$ where $P_{G,\sigma}$ is uniform on the σ-tube G_σ. In this case we consider the collection $\mathcal{P} = \{P_{G,\sigma} : G \in \mathcal{G}\}$.

We are interested in bounding the minimax risk

$$R_n = \inf_{\widehat{G}} \sup_{P \in \mathcal{P}} P^n\Big(\widehat{G} \not\simeq G\Big), \tag{1}$$

where the infimum is over all estimators $\widehat{G}$ of the topology of (G, d_G), the supremum is over the class of distributions $\mathcal{P}$ for $\mathbb{Y}$ and $\widehat{G} \not\simeq G$ means that $\widehat{G}$ and G are not isomorphic.
In Section 4 we will find lower and upper bounds for R_n in the noiseless case and the tubular noise case.
We conclude this section by summarizing the many parameters and symbols involved in our analysis. See Table 1.

3. Performance Analysis for the Algorithm of Aanjaneya et al. (2012)

In this section we study the performance of the metric graph reconstruction algorithm of Aanjaneya et al. (2012), under assumptions A1-A4 and with a choice of parameters adapted to our setting. In Section 4 we will use these results to derive bounds on the minimax rate for topology reconstruction. The metric graph reconstruction algorithm is presented in Algorithm 1.

The algorithm takes a (possibly noisy) sample $\mathbb{Y}$ from a metric graph G and a distance $d_{\mathbb{Y}}$ defined on $\mathbb{Y}$ and returns a graph $\widehat{G}$ that approximates G. The key idea is the following:

Symbol	Meaning
(G, d_G)	metric graph
α	smallest angle
b	shortest edge
τ	local reach
ξ	global reach
$\mathcal{G}$	set of metric graphs embedded in $\mathbb{R}^D$, satisfying A1-A4
$\mathcal{P}$	set of distributions on G or G_σ
G_σ	σ tube around G
$\mathbb{Y}$	sample, subset of G_σ
δ	$\mathbb{Y}$ is a $\delta/2$-dense sample

Table 1: Summary of the symbols used in our analysis.

a shell of radius r is constructed around each point in the sample, which is labeled *edge point* if its shell contains 2 well separated clusters of sampled points and *vertex point* otherwise. Several steps of the algorithm require the construction of a Rips-Vietoris graph of parameter δ: $\mathcal{R}_\delta(S_y)$ is a graph whose vertices are all the points of S_y and there is an edge between two points if the Euclidean distance between them is not larger than δ. At Step 11 some of the edge points that are close to vertices are re-labeled as vertex points. This expansion guarantees a precise borderline between clusters of vertex points and clusters of edge points. At steps 15-17 each of these clusters is associated to a vertex or to an edge of the reconstructed graph $\widehat{G}$. We will analyze the algorithm considering the Euclidean

Algorithm 1 Metric Graph Reconstruction Algorithm

Input: sample $\mathbb{Y}, d_{\mathbb{Y}}, r, p_{11}$.
1: **Labeling points as edge or vertex**
2: for all $y \in \mathbb{Y}$ do
3: $S_y \leftarrow B(y, r+\delta) \backslash B(y, r)$
4: $\deg_r(y) \leftarrow$ Number of connected components of Rips-Vietoris graph $\mathcal{R}_\delta(S_y)$
5: if $\deg_r(y) = 2$ then
6: Label y as a edge point
7: else
8: Label y as a preliminary vertex point.
9: end if
10: end for.
11: Label all points within Euclidean distance p_{11} from a preliminary vertex point as vertices.
12: Let $\mathbb{E}$ be the point of $\mathbb{Y}$ labeled as edge points.
13: Let $\mathbb{V}$ be the point of $\mathbb{Y}$ labeled as vertices.
14: **Reconstructing the graph structure**
15: Compute the connected components of the Rips-Vietoris graphs $\mathcal{R}_\delta(\mathbb{E})$ and $\mathcal{R}_\delta(\mathbb{V})$.
16: Let the connected components of $\mathcal{R}_\delta(\mathbb{V})$ be the vertices of of the reconstructed graph $\widehat{G}$.
17. Let there be an edge between vertices of $\widehat{G}$ if their corresponding connected components in $\mathcal{R}_\delta(\mathbb{V})$ contain points at distance less than δ from the same connected component of $\mathcal{R}_\delta(\mathbb{E})$.
Output: $\widehat{G}$.

distance on the sample $\mathbb{Y}$, that is, $d_{\mathbb{Y}} = \|\cdot\|_2$. The inner radius of the shell at Step 3 and the width of the expansion at Step 11 are parameters the user has to specify.

Before finding how dense a sample has to be in order to guarantee a correct reconstruction of a metric graph, we show that it is sufficient to study a particular metric graph embedded in $\mathbb{R}^2$, which represents the worst case. In other words, if the metric graph algorithm can reconstruct this particular planar graph, then it can reconstruct any other metric graph that satisfies A1-A4.

3.1 The Worst Case: a Metric Graph in $\mathbb{R}^2$

The worst case is the one for which it is hard to distinguish two edges that intersect at a vertex because they are too close in the embedding space.
Figure 2 (top left) shows an edge e that intersects two edges e_1, e_2 with reach τ, forming an angle α at vertex x. In the plots, the embedding space is $\mathbb{R}^3$ ($D = 3$) and we show the projections of e, e_1 and e_2 on the (limit) plane formed by e_1 and e_2, passing through x.

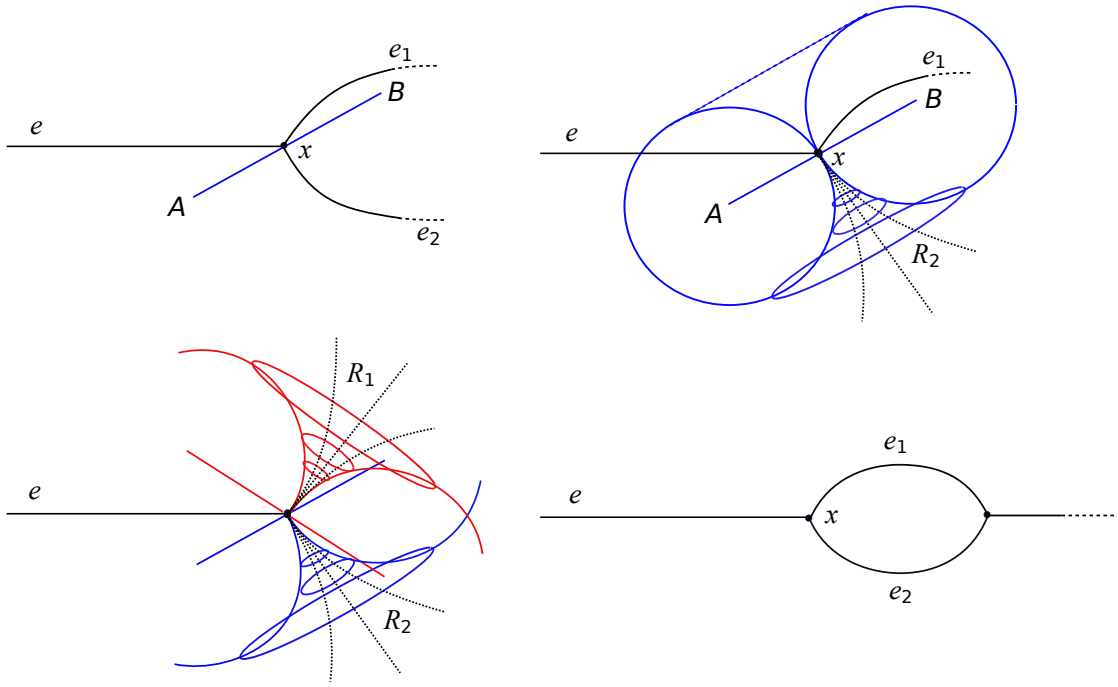


Figure 2: Even in the worst case, edges e_1 and e_2 must lie outside of the torii constructed on the fibers $L_\tau(x, e_1)$ and $L_\tau(x, e_2)$.

We focus on edge e_2. The fiber $L_\tau(x, e_2)$ of size τ around x is a $(D-1)$-dimensional ball centered at x and orthogonal to e_2. In $\mathbb{R}^3$, $L_\tau(x, e_2)$ is a disk of radius τ, whose projection on the plane is the segment $\overline{AB}$. By definition, for any $y \in e_2$, the fiber $L_\tau(y, e_2)$ can not intersect the fiber $L_\tau(x, e_2)$, otherwise the assumption involving the reach τ would be violated. We represent this condition by considering a D-dimensional ball of radius τ, centered at each point of the boundary of $L_\tau(x, e_2)$. Edge e_2 must lie outside of these balls,

in a feasible region that we denote by R_2, so that its fibers do not intersect $L_\tau(x, e_2)$. In $\mathbb{R}^3$, this procedure forms a horn torus (a torus with no hole) around vertex x. See the top right plot of Figure 2. The same reasoning applies to edge e_1, which must lie in the region R_1, outside of the balls of radius τ centered at each point of the boundary of $L_\tau(x, e_1)$. See the bottom left plot.

At each given distance from vertex x, two points of e_1 and e_2 are as close as possible when they lie on the boundaries of R_1 and R_2, on the same (limit) plane formed by e_1 and e_2, passing through x. When e_1 and e_2 lie on this plane, on the boundaries of the two feasible regions, they are as close as possible in the embedding space. This worst case is represented in the bottom right plot of Figure 2. Note that e_1 and e_2 are simply arcs of circles of radius τ.

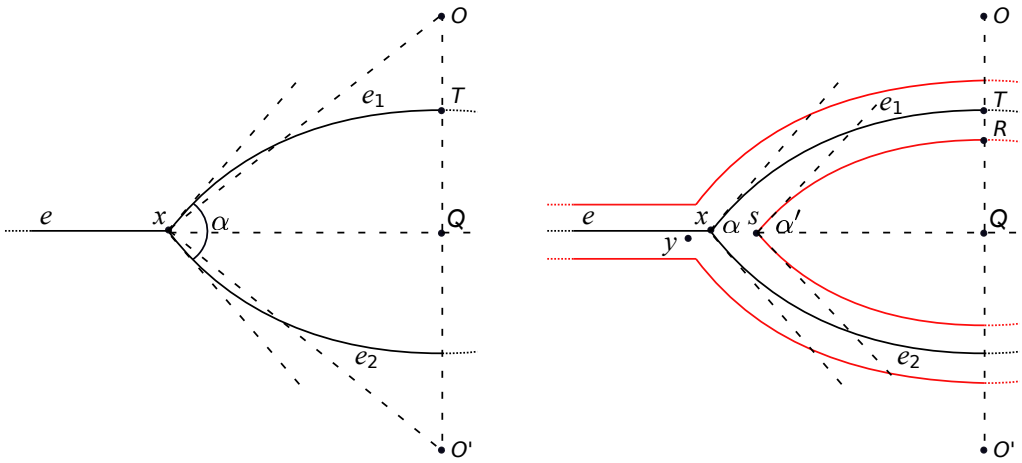


Figure 3: Left: edges e_1 and e_2 with minimum reach τ forming the smallest angles α at vertex x. Right: same metric graph with a tube of radius σ around it.

We will use basic trigonometric properties of the worst case. In Figure 3 (left), O and O' are the centers of the circles associated to edges e_1 and e_2. It is easy to see that angle $O\hat{x}O'$ has width $\pi - \alpha$. It can be shown that

$$x\widehat{O}O' = \alpha/2, \tag{2}$$

$$T\hat{x}Q = \alpha/4. \tag{3}$$

Let $\mathbb{Y}$ be a noisy sample of G. In other words $\mathbb{Y}$ is a subset of G_σ, the tube of radius $\sigma \geq 0$ around the metric graph G. See Figure 3 (right). Let Q be the midpoint of segment $\overline{OO'}$ and let T be the intersection point of $\overline{OO'}$ and edge e_1. For $0 \leq \sigma \leq \overline{QT} = \tau - \tau\cos(\alpha/2)$, the smallest angle formed by the inner faces of the tube around the metric graph is

$$\alpha' = \pi - \arccos \frac{2(\tau - \sigma)^2 - 4\tau^2 \cos^2(\alpha/2)}{2(\tau - \sigma)^2}, \tag{4}$$

where we applied the cosine law to the triangle OsO' and the fact that angle $O\hat{s}O'$ has width $\pi - \alpha'$. Note that if $\sigma = 0$ then $\alpha' = \alpha$. As in (3), it can be shown that

$$R\hat{s}Q = \alpha'/4. \tag{5}$$

The few basic trigonometric equations described above will be used to determine under which conditions on $b, \alpha, \tau, \xi, \sigma$ the metric graph reconstruction algorithm can reconstruct the worst case.

3.2 Analysis of Algorithm 1 with Euclidean Distance

In this section we analyze Algorithm 1. The Euclidean distance is used at every step of the algorithm, which requires the specification of r, the inner radius of the shell, and p_{11}, the parameter governing the expansion of Step 11. We set

$$r = \frac{\delta}{2} + \sigma + \tau \sin(\alpha/2) - (\tau - \sigma)\sin(\alpha'/2) + \frac{\delta}{2\sin(\alpha'/4)} \tag{6}$$

and

$$p_{11} = \frac{\delta}{2} + \tau \sin(\alpha/2) - (\tau - \sigma)\sin(\alpha'/2) + \frac{r + \delta}{\sin(\alpha'/2)} \tag{7}$$

This choice is justified in the proof of Proposition 4.
Define

$$f(b, \alpha, \tau, \xi, \sigma) := \frac{(\tau - \sigma)\sin\left(\frac{\min(b, \alpha\tau) - (\alpha - \alpha')\tau}{2\tau}\right) - [\tau \sin(\alpha/2) - (\tau - \sigma)\sin(\alpha'/2)]\left(1 + \frac{2}{\sin(\alpha'/2)}\right) - \frac{2\sigma}{\sin(\alpha'/2)}}{1 + 3[\sin(\alpha'/2)]^{-1} + [\sin(\alpha'/2)\sin(\alpha'/4)]^{-1}}, \tag{8}$$

where α' is given in 4. Note that $f(b, \alpha, \tau, \xi, \sigma)$ is a decreasing function of σ.

Proposition 4 *If $\mathbb{Y}$ is $\frac{\delta}{2}$-dense in G_σ and*

$$0 < r + \delta < \xi - 2\sigma, \tag{9}$$

$$0 < \delta < f(b, \alpha, \tau, \xi, \sigma), \tag{10}$$

then the graph $\widehat{G}$ provided by Algorithm 1 (input: $\mathbb{Y}, \|\cdot\|_2, r, p_{11}$) is isomorphic to G.

Proof Our objective is to use Algorithm 1 to reconstruct edges and vertices of a metric graph G embedded in R^D. Condition (9) guarantees that points of G which are far apart in the metric graph distance d_G, and close in the embedding space, do not interfere in the construction of the shells at Steps 3-4. Therefore we can restrict the attention to adjacent edges in a neighborhood of the vertex at which they intersect. In particular, since Algorithm 1 is based on the Euclidean distance between the edges, if we can distinguish two adjacent edges that are as close as possible in the embedding space, then we can distinguish any other pair of adjacent edges. As shown in Section 3.1, two adjacent edges e_1 and e_2, forming an angle of width α at vertex x, are as close as possible when they lie on the same plane, on the boundaries of the feasible regions determined by the condition on the reach (assumption A2). We will show that under conditions (9) and (10), Algorithm 1 can reconstruct this worst case. This will imply that the algorithm can reconstruct the topology of other vertices and edges in the D-dimensional space.

The rest of the proof involves condition (10). Since the sample is $\frac{\delta}{2}$-dense in the tube, there is at least a point $y \in \mathbb{Y}$ inside the ball of radius $\frac{\delta}{2}$ centered at any vertex $x \in G$. When using Algorithm 1 we want to be sure that y is labeled as a vertex, that is, the number of connected components of the shell around y is different than 2 (Steps 3-4). The worst case is depicted in Figure 4 (left), where x is the vertex of minimum angle α, formed by two edges, e_1 and e_2 of reach τ. First, we show that for the the value of r selected in (6), points close to an actual vertex are labeled as vertices at Steps 3-10 and points far from actual vertices are labeled as edges. The inner faces of the tube of radius σ around e_1 and

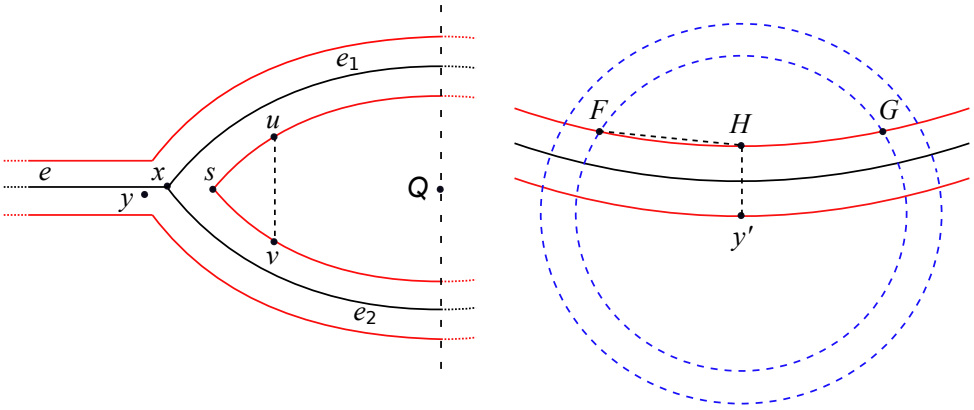


Figure 4: Left: edges e_1 and e_2 with minimum reach τ forming the smallest angles α at vertex x. Right: The distance $\|F - G\|_2$ between the two connected components of the shell around an edge point y' must be greater than δ.

e_2 form an angle of width α' at vertex s, as described in Section 3.1. Let u and v be the two points on the faces of the tube such that they are equidistant from x and $\|u - v\|_2 = \delta$. Since at Step 4 we construct a δ-graph to determine the number of connected components of the shell S_y and we want y to be a vertex, we choose r, the inner radius of the shell S_y, so that if $u, v \in \mathbb{Y}$ then $r \geq \max\{d_{\mathbb{Y}}(y, u), d_{\mathbb{Y}}(y, v)\}$. This guarantees that $\forall t_1, t_2 \in \mathbb{Y}$ with t_1 around edge e_1, t_2 around edge e_2 such that $\{t_1, t_2\} \subset S_y$, we have $d_{\mathbb{Y}}(t_1, t_2) \geq \delta$, that is t_1 and t_2 belong to different connected components of the shell around y at Step 4.
The distance between y and u is bounded by $\|y - x\|_2 + \|x - s\|_2 + \|s - u\|_2$, where, using (2),

$$\|x - s\|_2 = \|x - Q\|_2 - \|s - Q\|_2 = \tau \sin(\alpha/2) - (\tau - \sigma)\sin(\alpha'/2)$$

and using (5),

$$\|s - u\|_2 \leq \frac{\delta}{2\sin(\alpha'/4)}. \tag{11}$$

Therefore we require that r, the inner radius of the shell of Step 4 satisfies

$$\begin{aligned} r &\geq \frac{\delta}{2} + \|x - s\|_2 + \frac{\delta}{2\sin(\alpha'/4)} \\ &\geq \|y - x\|_2 + \|x - s\|_2 + \|s - u\|_2. \end{aligned} \tag{12}$$

Another condition on r arises when we label edge points far from actual vertices. See Figure 4 (right). If $y' \in \mathbb{Y}$, then it should be labeled as an edge point. That is, at Step 4, the Rips graph $\mathcal{R}_\delta(S_{y'})$ on the shell $S_{y'}$ should have 2 connected components. Therefore the distance $\|F - G\|_2$ between them must be greater than δ. We require that

$$r \geq 2\sigma + \delta/\sqrt{2} \tag{13}$$

which implies $\|F - G\|_2 > \delta$ when r is small enough, as implied by (10).
Note that the value $r = \frac{\delta}{2} + \sigma + \|x - s\|_2 + \frac{\delta}{2\sin(\alpha'/4)}$ satisfies both (12) and (13).

The outer radius of the shell at Steps 3-4 has length $r + \delta$. This guarantees that when the shell around an edge point intersects the tube around G there is at least a point $y \in \mathbb{Y}$ in each connected component of the shell, since $\mathbb{Y}$ is $\frac{\delta}{2}$-dense in G_σ.

In the last part of this proof we show that condition (10) is needed to guarantee that the sample is dense enough and the radius of the shells of Step 3 has the correct size, so that, even in the worst case, each vertex is associated to one set of sampled points at Steps 15-17 and these connected components are correctly linked by sets of sampled points labeled as edge points.
Let $z \in G_\sigma$ be the point around e_2 where the segment of length $r + \delta$, orthogonal to the face of the tube around edge e_1, intersects the face of the tube around edge e_2. See Figure 5. If this segment does not exist we simply consider the segment of length $r + \delta$ from s to a point z on e_2.

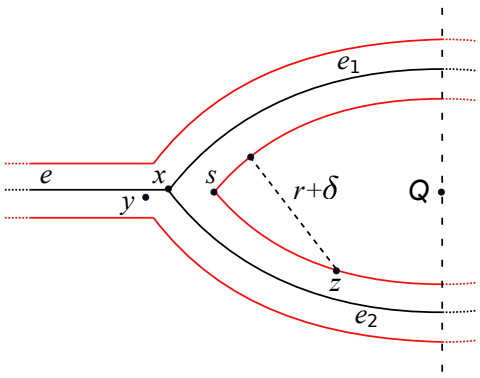


Figure 5: The shell around z is tangent to edge e_2.

Suppose $z \in \mathbb{Y}$. Among the points that might be labeled as vertices at Step 6 because of their closeness to vertex x, z is the furthest from x, since the shell around z is tangent to the tube around e_1. At Step 11, in order to control the labeling of the points in the tube between y and z we would like to label all the points in $\{y' \in \mathbb{Y} : \|y' - y\|_2 \leq \|y - z\|_2\}$ as vertices. To simplify the calculation we use the following bound

$$\|y - z\|_2 \leq \|y - x\|_2 + \|x - s\|_2 + \|s - z\|_2,$$

where, using (5),

$$\|s - z\|_2 \leq \frac{r + \delta}{\sin(\alpha'/2)}. \tag{14}$$

This justifies the choice of $p_{11} = \frac{\delta}{2} + \|x - s\|_2 + \frac{r+\delta}{\sin(\alpha'/2)} \geq \|y - z\|_2$. Thus, at Step 11 we label all the points in $\{y' \in \mathbb{Y} : \|y' - y\|_2 \leq p_{11}$ and y is labeled as vertex at Step 6 $\}$ as vertices. If z is actually labeled as a vertex at Step 6, then through the expansion of Step 11, all the points at distance not greater than p_{11} from z are labeled as vertices.
Finally we determine under which conditions there is at least a point in the tube around e_2 labeled as an edge point after Step 11. Consider the worst case in which e_1 and e_2 are forming an angle of size α at both their extremes x and x'. See Figure 6.

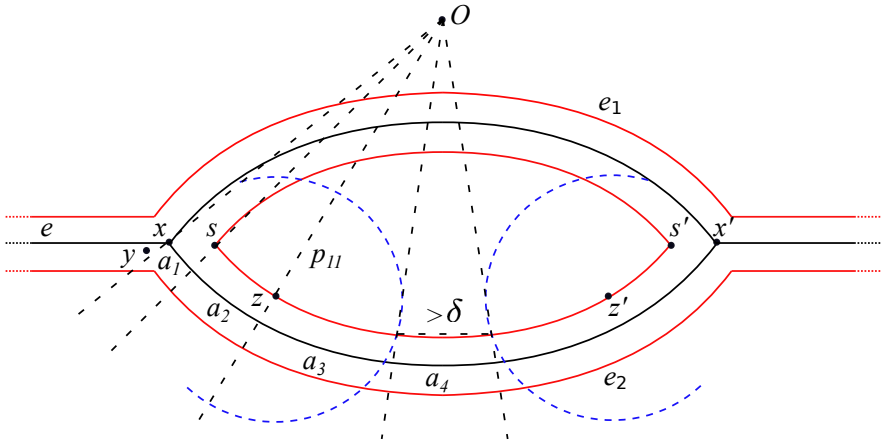


Figure 6: Edges e_1 and e_2, forming an angle of size α at both their extremes x and x'.

All the points $y' \in \mathbb{Y}$ such that $\|y' - z\|_2 \leq p_{11}$ or $\|y' - z'\|_2 \leq p_{11}$ might be labeled as vertices. When we construct $\mathcal{R}(\mathbb{E})_\delta$ and $\mathcal{R}(\mathbb{V})_\delta$ at Step 15 the two sets of vertices around x and x' must be disconnected and there must be at least an edge point between them. A sufficient condition is that the length of edge e_2 is greater than $2(a_1 + a_2 + a_3) + a_4$, where

- a_1 is the length of the arc of e_2 formed by the projections of lines $\overline{Ox}$ and $\overline{Os}$ on e_2,
- a_2 is the length of the arc of e_2 formed by the projection of the chord of length $\|s - z\|_2$,
- a_3 is the length of the arc of e_2 formed by the projection of the chord of length p_{11},
- a_4 is the length of the arc of e_2 formed by the projection of the chord of length δ.

Note that, in Figure 6, $e_2 = 2\tau \arcsin\left(\frac{\|x-x'\|_2}{2\tau}\right) = \alpha\tau$ but in general it might be shorter, so that e_1 and e_2 might not intersect in x'. However, by assumptions A2, e_2 must be longer than b. Thus we require

$$\min(b, \alpha\tau) > 2(a_1 + a_2 + a_3) + a_4. \tag{15}$$

By simple properties involving arcs and chords we have

$$a_1 = \left(\frac{\alpha - \alpha'}{2}\right)\tau, \qquad a_2 = 2\tau \arcsin\left(\frac{\|s - z\|_2}{2(\tau - \sigma)}\right),$$
$$a_3 = 2\tau \arcsin\left(\frac{p_{11}}{2(\tau - \sigma)}\right), \quad a_4 = 2\tau \arcsin\left(\frac{\delta}{2(\tau - \sigma)}\right).$$

Since the arcsin is superadditive in $[0,1]$ we require the stronger condition

$$\min(b, \alpha\tau) - (\alpha - \alpha')\tau > 2\tau \arcsin\left(\frac{2\|s-z\|_2 + 2p_{11} + \delta}{2(\tau - \sigma)}\right),$$

which holds if

$$\sin\left(\frac{\min(b, \alpha\tau) - (\alpha - \alpha')\tau}{2\tau}\right) > \frac{2\frac{r+\delta}{\sin(\alpha'/2)} + 2p_{11} + \delta}{2(\tau - \sigma)}.$$

The last condition is equivalent to (10). If this condition is satisfied then the graph is correctly reconstructed at Steps 15-17: every connected component of $\mathcal{R}_\delta(\mathbb{V})$ corresponds to a vertex of G and every connected component of $\mathcal{R}_\delta(\mathbb{E})$ corresponds to an edge of G. ■

Example 1 ***A Neuron in Three-Dimensions.*** *We return to the neuron example and we try to apply Propositions 4 to the 3D data of Figure 1, namely the neuron cr22e from the hippocampus of a rat (Gulyás et al., 1999). The data were obtained from NeuroMorpho.Org (Ascoli et al., 2007). The total length of the graph is* $1750.86\mu m$*. We assume the smallest edge has length* $100\mu m$*, the smallest angle* $\pi/3$*, the local reach* $30\mu m$ *and* $\xi = 50\mu m$*. The conditions of Proposition 4 are satisfied for* $\delta = 2.00\mu m$*. Algorithm 1 reconstructs the topology of the metric graph starting from a* $\delta/2$*-dense sample. Figure 1b shows the reconstructed graph.*

4. Minimax Analysis

In this section we derive lower and upper bound for the minimax risk

$$R_n = \inf_{\widehat{G}} \sup_{P \in \mathcal{P}} P^n\Big(\widehat{G} \not\simeq G\Big), \tag{16}$$

where, as described in Section 2, the infimum is over all estimators $\widehat{G}$ of the metric graph G, the supremum is over the class of distributions $\mathcal{P}$ for $\mathbb{Y}$ and $\widehat{G} \not\simeq G$ means that $\widehat{G}$ and G are not isomorphic.

4.1 Lower Bounds

To derive a lower bound on the minimax risk, we make repeated use of Le Cam's lemma. See, e.g., Yu (1997) and Chapter 2 of Tsybakov (2008). Recall that the total variation distance between two measures P and Q on the same probability space is defined by $\mathrm{TV}(P,Q) = \sup_A |P(A) - Q(A)|$ where the supremum is over all measurable sets. It can be shown that $\mathrm{TV}(P,Q) = P(H) - Q(H)$, where $H = \{y : p(y) \geq q(y)\}$ and p and q are the densities of P and Q with respect to any measure that dominates both P and Q.

Lemma 5 (Le Cam) *Let $\mathcal{Q}$ be a set of distributions. Let $\theta(Q)$ take values in a metric space with metric ρ. Let $Q_1, Q_2 \in \mathcal{Q}$ be any pair of distributions in $\mathcal{Q}$. Let $Y_1, \ldots, Y_n$ be drawn iid from some $Q \in \mathcal{Q}$ and denote the corresponding product measure by Q^n. Then*

$$\inf_{\widehat{\theta}} \sup_{Q \in \mathcal{Q}} \mathbb{E}_{Q^n}\Big[\rho(\widehat{\theta}, \theta(Q))\Big] \geq \frac{1}{8}\rho(\theta(Q_1), \theta(Q_2))(1 - TV(Q_1, Q_2))^{2n} \tag{17}$$

where the infimum is over all the estimators of $\theta(Q)$.

Below we apply Le Cam's lemma using several pairs of distributions. Any pair Q_1, Q_2 is associated with a pair of metric graphs $G', G'' \in \mathcal{G}$. We take $\theta(Q_1)$ and $\theta(Q_2)$ to be the classes of graphs that are isomorphic to G' and G''. We set $\rho(\theta(Q_1), \theta(Q_2)) = 0$ if G' and G'' are isomorphic and $\rho(\theta(Q_1), \theta(Q_2)) = 1$ otherwise. Figure 7 shows several pairs of metric graphs that are used to derive lower bounds in the noiseless case and in the tubular noise case. In the noiseless case we ignore the σ-tubes around the metric graphs.

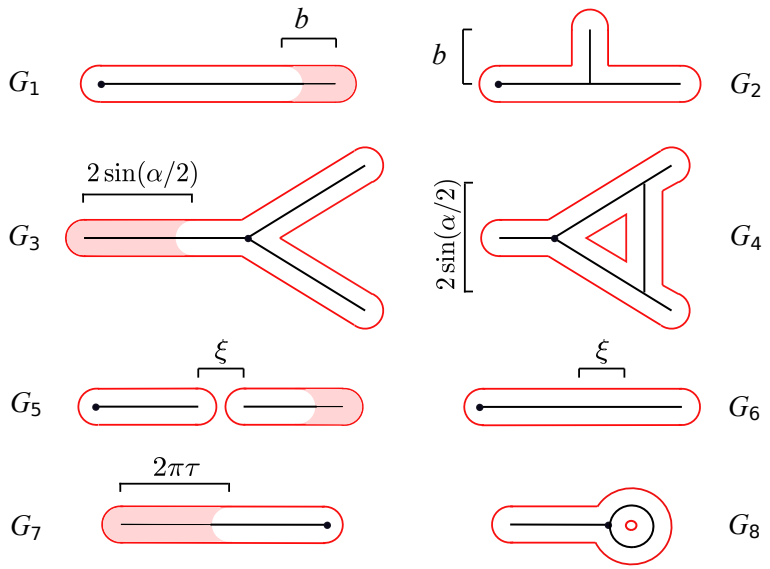


Figure 7: Pairs of metric graphs used in the derivation of lower bounds in the noiseless case and in the tubular noise case.

Recall that, in the noiseless case, we restrict the attention to probability distributions supported over metric graphs (G, d_G) in $\mathcal{G}$, having densities p with respect to the length of G bounded from below by a constant $a > 0$.

Theorem 6 *In the noiseless case ($\sigma = 0$), for $b \leq b_0(a)$, $\alpha \leq \alpha_0(a)$, $\xi \leq \xi_0(a)$, $\tau \leq \tau_0(a)$, where $b_0(a), \alpha_0(a), \xi_0(a)$ and $\tau_0(a)$ are constants which depend on a, a lower bound on the minimax risk for metric graph reconstruction is*

$$R_n \geq \exp\Big(-2a \min\{b, 2\sin(\alpha/2), \xi, 2\pi\tau\} n\Big). \tag{18}$$

Proof We consider the 4 parameters separately. See Figure 7, ignoring the red lines representing the tubular noise that is not considered in this theorem.

Shortest edge b. Consider the metric graph G_1 consisting of a single edge of length $1+b$ and metric graph G_2 with an edge of length 1 and an orthogonal edge of length b glued in the middle. The density on G_1 is constructed in the following way: on the set $G_1 \backslash G_2$ of length b we set $p_1(x) = a$ and the rest of the mass is evenly distributed over the remaining portion of G_1. Similarly, for G_2 we set $p_2(x) = a$ on $G_2 \backslash G_1$, which correspond to the orthogonal edge of length b. We evenly spread the remaining mass. The two densities differ

only on the sets $G_1 \backslash G_2$ and $G_2 \backslash G_1$. Therefore $\mathrm{TV}(p_1, p_2) \leq ab$ and, by Le Cam's lemma, $R_n \geq \frac{1}{8}(1-ab)^{2n} \geq \frac{1}{8}\mathrm{e}^{-2abn}$ for all $b \leq b_0(a)$, where $b_0(a)$ is a constant depending on a.

Smallest angle α. Now consider the metric graphs G_3 and G_4. G_3 consists of two edges of length 2 forming an angle α and a third edge of length $1+2\sin(\alpha/2)$ glued to the first two. G_4 is similar: an edge of length $2\sin(\alpha/2)$ is added to complete the triangle, while the edge on the left has length 1. As in the previous case we set $p_3(x) = a$ on $G_3 \backslash G_4$, $p_4(x) = a$ on $G_4 \backslash G_3$ and spread evenly the rest of the mass. The total variation distance is $\mathrm{TV}(p_3, p_4) \leq 2a \sin\left(\frac{\alpha}{2}\right)$ and, by Le Cam's lemma, $R_n \geq \frac{1}{8}(1 - 2a\sin(\alpha/2))^{2n} \geq \frac{1}{8}\mathrm{e}^{-4a\sin(\alpha/2)n}$ for all $\alpha \leq \alpha_0(a)$, where $\alpha_0(a)$ is a constant depending on a.

Global reach ξ. We defined the global reach as the shortest Euclidean distance between two points that are far apart in the graph distance. Figure 7 shows metric graph G_5 formed by a single edge of length 1 and metric graph G_6 consisting of two edges of length 0.5, ξ apart from each other. Again, we set $p_5(x) = a$ on $G_5 \backslash G_6$, $p_6(x) = a$ on $G_6 \backslash G_5$ and evenly spread the rest. We obtain $\mathrm{TV}(p_5, p_6) \leq a\xi$ and, by Le Cam's lemma, $R_n \geq \frac{1}{8}(1-a\xi)^{2n} \geq \frac{1}{8}\mathrm{e}^{-2a\xi n}$ for all $\xi \leq \xi_0(a)$, where $\xi_0(a)$ is a constant depending on a.

Local reach τ. The local reach τ is the smallest reach of the edges forming the metric graph. Consider metric graphs G_7 and G_8. G_7 consists of a loop of radius τ attached to an edge of length 1 and metric graph G_8 is a single edge of length $1 + 2\pi\tau$. As in the previous cases $p_7(x) = a$ on $G_7 \backslash G_8$ and $p_8(x) = a$ on $G_8 \backslash G_7$. It follows that $\mathrm{TV}(p_7, p_8) \leq 2a\pi\tau$ and, by Le Cam's lemma, $R_n \geq \frac{1}{8}(1 - 2a\pi\tau)^{2n} \geq \frac{1}{8}\mathrm{e}^{-4a\pi\tau n}$ for all $\tau \leq \tau_0(a)$, where $\tau_0(a)$ is a constant depending on a. ■

For the tubular noise case we assume that σ is small enough to guarantee that $R_n < 1$, that is, the problem is not hopeless. In particular, we require that σ satisfies conditions (9) and (10) of Proposition 4, which can be combined into the following condition

$$0 < \min\left\{ \frac{\xi - 3\sigma - \tau\sin(\alpha/2) + (\tau - \sigma)\sin(\alpha'/2)}{3/2 + [2\sin(\alpha'/4)]^{-1}},\ f(b, \alpha, \tau, \xi, \sigma) \right\}. \tag{19}$$

Theorem 7 *Assume that σ is positive and satisfies condition* (19)*. In the tubular noise case, for $b \leq b_0(D)$, $\alpha \leq \alpha_0(D)$, $\xi \leq \xi_0(D)$, $\tau \leq \tau_0(D)$, where $b_0(D), \alpha_0(D), \xi_0(D)$ and $\tau_0(D)$ are constants which depend on the ambient dimension D, a lower bound on the minimax risk for metric graph reconstruction is*

$$R_n \geq \frac{1}{8}\exp\Big(-2\min\{C_{D,1}b, C_{D,2}\sin(\alpha/2), C_{D,3}\xi, C_{D,4}\tau\}n\Big), \tag{20}$$

for some constants $C_{D,1}, C_{D,2}, C_{D,3}, C_{D,4}$.

Proof As in the proof oh Theorem 6 we consider the 4 parameters separately. We compare the pairs of graphs shown in Figure 7, including the tubular regions constructed around them, from which we get samples uniformly.

Shortest edge b. Consider the metric graph G_1 consisting of a single edge of length $1+b$ and metric graph G_2 with an edge of length 1 and an orthogonal edge of length b glued in the middle. Since $\mathrm{vol}(G_1) > \mathrm{vol}(G_2)$, the density q_1 at a point in the tube around G_1 is lower than the density q_2 at a point around G_2. From the definition of total variation

$TV = q_1(H) - q_2(H)$ where H is the set where $q_1 > q_2$, the shaded area in Figure 7. Note that $q_2(H) = 0$ and

$$TV(q_1, q_2) = q_1(H) = \frac{\text{vol}(H)}{\text{vol}(G_1)} \leq C_{D,1} \frac{b\sigma^{D-1}}{(1+b)\sigma^{D-1}} \leq C_{D,1} b.$$

By Le Cam's lemma, $R_n \geq \frac{1}{8}(1 - C_{D,1}b)^{2n} \geq \frac{1}{8}\mathrm{e}^{-2C_{D,1}bn}$ for all $b \leq b_0(D)$, where $b_0(D)$ is a constant depending on D.

Smallest angle α. Now consider the metric graphs G_3 and G_4. Since $\text{vol}(G_3) > \text{vol}(G_4)$, the density q_3 at a point in the tube around G_3 is lower than the density q_4 at a point around G_4. $TV = q_3(H) - q_4(H)$ where H is the set where $q_3 > q_4$, the shaded area in the tube around G_3. Note that $q_4(H) = 0$ and

$$TV(q_3, q_4) = q_3(H) = \frac{\text{vol}(H)}{\text{vol}(G_3)} \leq C_{D,2} \frac{\sin(\alpha/2)\sigma^{D-1}}{(1+\sin(\alpha/2))\sigma^{D-1}} \leq C_{D,2} \sin(\alpha/2).$$

By Le Cam's lemma, $R_n \geq \frac{1}{8}(1 - C_{D,2}\sin(\alpha/2))^{2n} \geq \frac{1}{8}\mathrm{e}^{-2C_{D,2}\sin(\alpha/2)n}$ for all $\alpha \leq \alpha_0(D)$, where $\alpha_0(D)$ is a constant depending on D.

Global reach ξ. Figure 7 shows metric graph G_5 formed by a single edge of length 1 and metric graph G_6 consisting of two edges of length 0.5, ξ apart from each other. Since $\text{vol}(G_5) > \text{vol}(G_6)$, the density q_5 at a point in the tube around G_5 is lower than the density q_6 at a point around G_6. $TV = q_5(H) - q_6(H)$ where H is the set where $q_5 > q_6$, the shaded area in the tube around G_5. Note that $q_6(H) = 0$ and

$$TV(q_5, q_6) = q_5(H) = \frac{\text{vol}(H)}{\text{vol}(G_5)} \leq C_{D,3} \frac{\xi\sigma^{D-1}}{\sigma^{D-1}} = C_{D,3}\xi.$$

By Le Cam's lemma, $R_n \geq \frac{1}{8}(1 - C_{D,3}\xi)^{2n} \geq \frac{1}{8}\mathrm{e}^{-2C_{D,3}\xi n}$ for all $\xi \leq \xi_0(D)$, where $\xi_0(D)$ is a constant depending on D.

Local reach τ. The local reach τ is the smallest reach of the edges forming the metric graph. Consider metric graphs G_7 and G_8 in Figure 7. Since $\text{vol}(G_7) > \text{vol}(G_8)$, the density q_7 at a point in the tube around G_7 is lower than the density q_8 at a point around G_8. $TV = q_7(H) - q_8(H)$ where H is the set where $q_7 > q_8$, the shaded area in the tube around G_7. Note that $q_8(H) = 0$ and

$$TV(q_7, q_8) = q_7(H) = \frac{\text{vol}(H)}{\text{vol}(G_7)} \leq C_{D,4} \frac{\tau\sigma^{D-1}}{(1+\tau)\sigma^{D-1}} \leq C_{D,4}\tau.$$

By Le Cam's lemma, $R_n \geq \frac{1}{8}(1 - C_{D,4}\tau)^{2n} \geq \frac{1}{8}\mathrm{e}^{-2C_{D,4}\tau n}$ for all $\tau \leq \tau_0(D)$, where $\xi_0(D)$ is a constant depending on D. ■

Note that, up to constants, the lower bound obtained in the tubular noise case is identical to the lower bound of Proposition 6 for the noiseless case.

4.2 Upper Bounds

In this section we use the analysis of the performance of Algorithm 1 to derive an upper bound on the minimax risk. We will use the strategy of Niyogi et al. (2008) to find the

sample size that guarantees a $\delta/2$-dense sample with high probability. We will use the following two lemmas.

Lemma 8 (5.1 in Niyogi et al. 2008) *Let $\{A_i\}$ for $i = 1, \ldots, l$ be a finite collection of measurable sets and let μ be a probability measure on $\bigcup_{i=1}^{l} A_i$ such that for all $1 \leq i \leq l$, we have $\mu(A_i) > \gamma$. Let $\bar{x} = \{x_1, \ldots, x_n\}$ be a set of n i.i.d. draws according to μ. Then if*

$$n \geq \frac{1}{\gamma}\left(\log l + \log\left(\frac{1}{\lambda}\right)\right)$$

we are guaranteed that with probability $> 1 - \lambda$, the following is true:

$$\forall i, \;\; \bar{x} \cap A_i \neq \emptyset.$$

Recall that the ϵ-covering number $C(\epsilon)$ of a set S is the smallest number of Euclidean balls of radius ϵ required to cover the set. The ϵ-packing number $P(\epsilon)$ is the maximum number of sets of the form $B(x, \epsilon) \cap S$, where $x \in S$, that may be packed into S without overlap.

Lemma 9 (5.2 in Niyogi et al. 2008) *For every $\epsilon > 0$, $P(2\epsilon) \leq C(2\epsilon) \leq P(\epsilon)$.*

Combining Lemma 8 and Proposition 4, we obtain an upper bound on R_n for the noiseless case.

Theorem 10 *In the noiseless case ($\sigma = 0$), an upper bound on the minimax risk R_n is given by*

$$R_n \leq \frac{8 \; \mathit{length}(G)}{\delta} \; \exp\left\{-\frac{a\,\delta\,n}{4 \; \mathit{length}(G)}\right\},$$

where

$$\delta = \frac{1}{2}\min\left\{\xi\frac{2\sin(\alpha/4)}{3\sin(\alpha/4)+1} \;,\; \frac{\tau\sin(\alpha/2)\sin(\alpha/4)}{\sin(\alpha/2)\sin(\alpha/4)+3\sin(\alpha/4)+1}\sin\left(\frac{\min\{b, \alpha\tau\}}{2\tau}\right)\right\}. \tag{21}$$

Proof In the noiseless case, Proposition 4 implies that the graph G can be reconstructed from a $\delta/2$-dense sample $\mathbb{Y}$ if

$$\delta < \min\left\{\xi\frac{2\sin(\alpha/4)}{3\sin(\alpha/4)+1} \;,\; f(b, \alpha, \tau, \xi, 0)\right\}. \tag{22}$$

The value of δ selected in (21) satisfies condition (22), which follows from conditions (9) and (10), with $\sigma = 0$. We look for the sample size n that guarantees a $\delta/2$-dense sample with high probability. Following the strategy in Niyogi et al. (2008), we consider a cover of the metric graph G by balls of radius $\delta/4$. Let $\{x_i : 1 \leq i \leq l\}$ be the centers of such balls that constitute a minimal cover. We can choose $A_i^{\delta/4} = B_{\delta/4}(x_i) \cap G$. Applying Lemma 8 we find that the sample size that guarantees a correct reconstruction with probability at least $1 - \lambda$ is

$$\frac{1}{\gamma}\left(\log l + \log\frac{1}{\lambda}\right), \tag{23}$$

where

$$\gamma \geq \min_i \frac{a \text{ length}(A_i^{\delta/4})}{\text{length}(G)} \geq \frac{a\delta}{4 \text{ length}(G)} \quad ,$$

and we bound the covering number l in terms of the packing number, using Lemma 9:

$$l \leq \frac{\text{length}(G)}{\min_i \text{length}(A_i^{\delta/8})} \leq \frac{8 \text{ length}(G)}{\delta}.$$

Therefore, from (23), if

$$n = \frac{4 \text{ length } (G)}{a\delta} \left[\log \left(\frac{8 \text{ length}(G)}{\delta} \right) + \log \frac{1}{\lambda} \right] \tag{24}$$

we have a $\delta/2$-dense sample with probability at least $1 - \lambda$ and, by Proposition 4, $\mathbb{P}(\widehat{G} \not\approx G) \leq \lambda$. Rearranging we have the result. ■

Note that, in the noiseless case, the upper and lower bounds are tight up to polynomial factors in the parameters τ, b, ξ. There is a small gap with respect to α; closing this gap is an open problem.
In the tubular noise case, we assume that σ is small enough, to guarantee that Algorithm 1 correctly reconstructs a metric graph starting from a $\delta/2$-dense sample.

Theorem 11 *Assume that σ satisfies condition* (19) *and $0 < \sigma < \min\{3\tau/16, \delta/8\}$, where*

$$\delta = C_0 \min \left\{ \frac{\xi - 3\sigma - \tau \sin(\alpha/2) - (\tau - \sigma)\sin(\alpha'/2)}{3/2 + [2\sin(\alpha'/4)]^{-1}},\ f(b, \alpha, \tau, \xi, \sigma) \right\}, \tag{25}$$

for some $0 < C_0 < 1$. Under the tubular noise model, an upper bound on the minimax risk R_n is given by

$$R_n \leq \frac{16 length(G)}{\delta} \exp\left(-\frac{C'_D \delta(\tau - 8\sigma)n}{\tau \, length(G)} \right),$$

where C'_D is a constant depending on the ambient dimension.

Proof Proposition 4 implies that the graph G can be reconstructed from a $\delta/2$-dense sample $\mathbb{Y}$ if

$$\delta < \min \left\{ \frac{\xi - 3\sigma - \tau \sin(\alpha/2) - (\tau - \sigma)\sin(\alpha'/2)}{3/2 + [2\sin(\alpha'/4)]^{-1}},\ f(b, \alpha, \tau, \xi, \sigma) \right\}, \tag{26}$$

which is satisfied by the value of δ selected in (25). We look for the sample size n that guarantees a $\delta/2$-dense sample in G_σ with high probability.
We consider a cover of the metric graph G by Euclidean balls of radius $\delta/8$. Let $\{x_i : 1 \leq i \leq l\}$ be the centers of such balls that constitute a minimal cover. Note that D-dimensional balls of radius $\delta/8 + \sigma \leq \delta/4$ centered at the same $x_i's$ constitute a cover of the tubular region G_σ. We define $A_i^{\delta/8+\sigma} = B_{\delta/8+\sigma}(x_i) \cap G_\sigma$. Applying Lemma 8 we find that the sample size

that guarantees a $\delta/2$-dense sample in G_σ (and a correct topological reconstruction of G) with probability at least $1-\lambda$ is

$$\frac{1}{\gamma}\left(\log l + \log\frac{1}{\lambda}\right), \tag{27}$$

where

$$\gamma = \min_i \frac{\mathrm{vol}(A_i^{\delta/8+\sigma})}{\mathrm{vol}(G_\sigma)}. \tag{28}$$

Define $\tilde{A}_i^\delta = B_\delta(x_i)\cap G$. The covering number l is bounded in terms of the packing number, using Lemma 9,

$$l \leq \frac{\mathrm{length}(G)}{\min_i \mathrm{length}(\tilde{A}_i^{\delta/16})} \leq \frac{16\ \mathrm{length}(G)}{\delta}.$$

We construct a lower bound on γ by deriving an upper bound on the denominator of (28) and a lower bound on the numerator.

Upper bound on vol(G_σ). Let N_σ be the σ-covering number of G and let $\mathcal{C}_\sigma$ be the set of centers of this cover. By Lemma 9, N_σ is bounded by the $\sigma/2$-packing number. A simple volume argument gives $N_\sigma \leq C\mathrm{length}(G)/\sigma$, for some constant C. Note that 2σ D-dimensional balls around each of the centers in $\mathcal{C}_\sigma$ cover G_σ. Thus $\mathrm{vol}(G_\sigma) \leq v_D N_\sigma (2\sigma)^D \leq C_D \mathrm{length}(G)\sigma^{D-1}$ for some constant C_D depending on the ambient dimension.

Lower bound on vol$(A_i^{\delta/8+\sigma})$, for all i. Let $P_A(\sigma)$ be the σ-packing number of $\tilde{A}_i^{\delta/8}$ and let $\mathcal{D}_A$ be the set of centers of this packing. Then $\mathrm{vol}(A_i^{\delta/8+\sigma}) \geq P_A(\sigma) v_D \sigma^D$, because the union of σ balls around $\mathcal{D}_A$ is contained in $A_i^{\delta/8+\sigma}$. Let $C_A(2\sigma)$ be the 2σ-covering number of $\tilde{A}_i^{\delta/8}$ and let $\mathcal{C}_A = \{z_1, \ldots, z_{C_A(2\sigma)}\}$ be the set of centers of this cover. By Lemma 9,

$$P_A(\sigma) \geq C_A(2\sigma) \geq \frac{\mathrm{length}(\tilde{A}_i^{\delta/8})}{\max_{z_j\in\mathcal{C}_A} \mathrm{length}(B_{2\sigma}(z_j)\cap \tilde{A}_i^{\delta/8})} \geq \frac{\delta/8}{\max_{z_j\in\mathcal{C}_A} \mathrm{length}(B_{2\sigma}(z_j)\cap \tilde{A}_i^{\delta/8})}$$

and, since $2\sigma < 3\tau/8$, by Corollary 1.3 in Chazal (2013),

$$\max_{z_j\in\mathcal{C}_A} \mathrm{length}(B_{2\sigma}(z_j)\cap \tilde{A}_i^{\delta/8}) \leq C_2\left(\frac{\tau}{\tau-8\sigma}\right)\sigma,$$

for some constant C_2. Thus

$$\gamma \geq \frac{P_A(\sigma) v_D \sigma^D}{C_D \mathrm{length}(G)\sigma^{D-1}} \geq C'_D \frac{\delta(\tau-8\sigma)}{\tau \mathrm{length}(G)} \quad ,$$

where C'_D is a constant depending on the ambient dimension.

Finally, from (27), if

$$n = \frac{\tau\ \mathrm{length}\ (G)}{C'_D\delta(\tau-8\sigma)}\left[\log\left(\frac{16\ \mathrm{length}(G)}{\delta}\right) + \log\frac{1}{\lambda}\right], \tag{29}$$

then the sample is $\delta/2$-dense with probability at least $1-\lambda$ and $\mathbb{P}(\widehat{G} \not\simeq G) \leq \lambda$. Rearranging we obtain

$$R_n \leq \exp\left(-\frac{C'_D \delta(\tau - 8\sigma)n}{\tau\, \text{length}(G)} + \log\left(\frac{16\text{length}(G)}{\delta}\right)\right).$$

■

5. Discussion

In this paper, we presented a statistical analysis of metric graph reconstruction. We derived sufficient conditions on random samples from a graph metric space that guarantee topological reconstruction and we derived lower and upper bounds on the minimax risk for this problem. Various improvements and theoretical extensions are possible. In Proposition 4 we have analyzed Algorithm 1 using the Euclidean distance at every step. It is possible to obtain a similar result using a different notion of distance, for example, the distance induced by a Rips-Vietoris graph constructed on the sample.

While in our analysis we mainly relied on the assumption of a dense sample, Aanjaneya et al. (2012) used the more refined but stronger assumption of the sample being an approximation of the metric graph, which we recall: given positive numbers ε and R, we say that $(\mathbb{Y}, d_{\mathbb{Y}})$ is an (ε, R)-*approximation* of the metric space (G, d_G) if there exists a correspondence $C \subset G \times \mathbb{Y}$ such that

$$(x,y),(x',y') \in C, \min(d_G(x,x'), d_{\mathbb{Y}}(y,y')) \leq R \implies \left|d_G(x,x') - d_{\mathbb{Y}}(y,y')\right| \leq \varepsilon. \tag{30}$$

As shown in Aanjaneya et al. (2012), the (ε, R)-approximation assumption is sufficient, for appropriate choice of the parameters ε and R, to recover not only the topology of a metric graph (G, d_G), but also its metric d_G with high accuracy. However, when compared to the dense sample assumption, it demands a larger sample complexity to achieve accurate topological reconstruction. A strategy similar to the one used in this paper could be used to determine the sample size that guarantees an (ε, R)-approximation of the underlying metric graph with high probability. This would guarantee a correct topological reconstruction, as well as an approximation of the metric d_G.

We are also investigating the idea of combining metric graph reconstruction with the subspace constrained mean-shift algorithm (Fukunaga and Hostetler, 1975; Comaniciu and Meer, 2002; Genovese et al., 2012b) to provide similar guarantees. Our preliminary results indicate that this mixed strategy works very well under more general noise assumptions and with relatively low sample size.

Acknowledgments

Research supported by NSF CAREER Grant DMS 114967, Air Force Grant FA95500910373, NSF Grant DMS-0806009. The authors thank the referees for helpful comments and suggestions.

References

Mridul Aanjaneya, Frédéric Chazal, Daniel Chen, Marc Glisse, Leonidas Guibas, and Dmitriy Morozov. Metric graph reconstruction from noisy data. *International Journal of Computational Geometry & Applications*, 22(04):305–325, 2012.

Mahmuda Ahmed and Carola Wenk. Probabilistic street-intersection reconstruction from gps trajectories: approaches and challenges. In *Proceedings of the Third ACM SIGSPATIAL International Workshop on Querying and Mining Uncertain Spatio-Temporal Data*, pages 34–37. ACM, 2012.

Ery Arias-Castro, Guangliang Chen, and Gilad Lerman. Spectral clustering based on local linear approximations. *Electronic Journal of Statistics*, 5:1537–1587, 2011.

Giorgio A Ascoli, Duncan E Donohue, and Maryam Halavi. Neuromorpho. org: a central resource for neuronal morphologies. *The Journal of Neuroscience*, 27(35):9247–9251, 2007.

Paul Bendich, Sayan Mukherjee, and Bei Wang. Towards stratification learning through homology inference. *arXiv preprint arXiv:1008.3572*, 2010.

Paul Bendich, Bei Wang, and Sayan Mukherjee. Local homology transfer and stratification learning. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1355–1370. SIAM, 2012.

Frédéric Chazal. An upper bound for the volume of geodesic balls in submanifolds of Euclidean spaces. Technical report, INRIA, January 2013.

Frédéric Chazal and André Lieutier. Topology guaranteeing manifold reconstruction using distance function to noisy data. In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, pages 112–118. ACM, 2006.

Frédéric Chazal and Jian Sun. Gromov-Hausdorff approximation of metric spaces with linear structure. *arXiv preprint arXiv:1305.1172*, 2013.

Frédéric Chazal, David Cohen-Steiner, and André Lieutier. A sampling theory for compact sets in Euclidean space. *Discrete & Computational Geometry*, 41(3):461–479, 2009.

Daniel Chen, Leonidas J Guibas, John Hershberger, and Jian Sun. Road network reconstruction for organizing paths. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1309–1320. SIAM, 2010.

Alexey Chernov and Vitaliy Kurlin. Reconstructing persistent graph structures from noisy images. *Electronic Journal Image-A*, 3(5):19–22, 2013.

Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

Herbert Federer. Curvature measures. *Transactions of the American Mathematical Society*, 93(3):418–491, 1959.

Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on*, 21(1):32–40, 1975.

Xiaoyin Ge, Issam I Safa, Mikhail Belkin, and Yusu Wang. Data skeletonization via Reeb graphs. In *Advances in Neural Information Processing Systems*, pages 837–845, 2011.

Christopher R Genovese, Marco Perone-Pacifico, Isabella Verdinelli, and Larry Wasserman. Minimax manifold estimation. *Journal of Machine Learning Research*, 13:1263–1291, 2012a.

Christopher R Genovese, Marco Perone-Pacifico, Isabella Verdinelli, and Larry Wasserman. Nonparametric ridge estimation. *arXiv preprint arXiv:1212.5156*, 2012b.

Attila I Gulyás, Manuel Megías, Zsuzsa Emri, and Tamás F Freund. Total number and ratio of excitatory and inhibitory synapses converging onto single interneurons of different types in the ca1 area of the rat hippocampus. *The Journal of Neuroscience*, 19(22):10082–10097, 1999.

Peter Kuchment. Quantum graphs: I. Some basic structures. *Waves in Random Media*, 14 (1):107–128, 2004.

Partha Niyogi, Stephen Smale, and Shmuel Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete & Computational Geometry*, 39(1-3):419–441, 2008.

Alexandre B Tsybakov. *Introduction to Nonparametric Estimation*. Springer, 2008.

Bin Yu. Assouad, Fano, and Le Cam. In *Festschrift for Lucien Le Cam*, pages 423–435. Springer, 1997.

Journal of Machine Learning Research 15 (2014) 3447-3481 Submitted 6/13; Revised 5/14; Published 10/14

Alternating Linearization for Structured Regularization Problems

Xiaodong Lin LIN@BUSINESS.RUTGERS.EDU
Department of Management Science and Information Systems
Rutgers University
Piscataway, NJ 08854

Minh Pham PTUANMINH@GMAIL.COM
Statistical and Applied Mathematical Sciences Institute (SAMSI)
Durham, NC 27707

Andrzej Ruszczyński RUSZ@BUSINESS.RUTGERS.EDU
Department of Management Science and Information Systems
Rutgers University
Piscataway, NJ 08854

Editor: S. V. N. Vishwanathan

Abstract

We adapt the alternating linearization method for proximal decomposition to structured regularization problems, in particular, to the generalized lasso problems. The method is related to two well-known operator splitting methods, the Douglas–Rachford and the Peaceman–Rachford method, but it has descent properties with respect to the objective function. This is achieved by employing a special update test, which decides whether it is beneficial to make a Peaceman–Rachford step, any of the two possible Douglas–Rachford steps, or none. The convergence mechanism of the method is related to that of bundle methods of nonsmooth optimization. We also discuss implementation for very large problems, with the use of specialized algorithms and sparse data structures. Finally, we present numerical results for several synthetic and real-world examples, including a three-dimensional fused lasso problem, which illustrate the scalability, efficacy, and accuracy of the method.

Keywords: lasso, fused lasso, nonsmooth optimization, operator splitting

1. Introduction

Regularization techniques that encourage sparsity in parameter estimation have gained increasing popularity recently. The most widely used example is *lasso* (Tibshirani, 1996), where the loss function $f(\cdot)$ is penalized by the ℓ_1-norm of the unknown coefficients $\beta \in \mathbb{R}^p$, to form a modified objective function,

$$\mathcal{L}(\beta) = f(\beta) + \lambda\|\beta\|_1, \quad \lambda > 0,$$

in order to shrink irrelevant coefficients to zero. Many efficient algorithms have been proposed to solve this problem; see Fu (1998); Daubechies et al. (2004); Efron et al. (2004) and Friedman et al. (2007). Some of them are capable of handling massive data sets with tens of thousands of variables and observations.

For many practical applications, physical constraints and domain knowledge may mandate additional structural constraints on the parameters. For example, in cancer research, it may be important to consider groups of interacting genes in each pathway rather than individual genes. In image analysis, it is natural to regulate the differences between neighboring pixels in order to achieve smoothness and reduce noise. In light of these popular demands, a variety of structured penalties have been proposed to incorporate prior information regarding model parameters. One of the most important structural penalties is the *fused lasso* proposed by Tibshirani et al. (2005). It utilizes the natural ordering of input variables to achieve parsimonious parameter estimation on neighboring coefficients. Chen et al. (2010) developed the *graph induced fused lasso* that penalizes differences between coefficients associated with nodes in a graph that are connected. Rudin et al. (1992) proposed the *total variation penalty* for image denoising and deblurring, in a similar fashion to the two-dimensional fused lasso. Similar penalty functions have been successfully applied to several neuroimaging studies (Michel et al., 2011; Grosenick et al., 2011, 2013). More recently, Zhang et al. (2012) applied a generalized version of fused lasso to reconstruct gene copy number variant regions. A general structural lasso framework was proposed by Tibshirani and Taylor (2011), with the following form:

$$\mathcal{L}(\beta) = f(\beta) + \lambda \|R\beta\|_1, \quad \lambda > 0, \tag{1}$$

where R is an $m \times p$ matrix that defines the structural constraints one wants to impose on the coefficients. Many regularization problems, including high dimensional fused lasso and graph induced fused lasso, can be cast in this framework.

When the structural matrix R is relatively simple, as in the original lasso case with $R = I$, traditional path algorithms and coordinate descent techniques can be used to solve the optimization problems efficiently (Friedman et al., 2007). For more complex structural regularization, these methods cannot be directly applied. One of the key difficulties is the non-separability of the nonsmooth penalty function. Coordinate descent methods fail to converge under this circumstances (Tseng, 2001). Generic solvers, such as interior point methods, can sometimes be used; unfortunately they become increasingly inefficient for large size problems, particularly when the design matrix is ill-conditioned (Chen et al., 2012).

In the past decade, many efforts have been devoted to developing efficient optimization techniques for solving regularization problems using structured penalties. Liu et al. (2010) and Ye and Xie (2011) developed a first-order and a split Bregman scheme, respectively, for solving similar class of problems. Chen et al. (2012) proposed a modified proximal technique for the general structurally penalized problems. It is based on a first order approximation of the nonsmooth penalty function, which can become unstable when dimension is high. Meanwhile, several path algorithms have also been proposed to compute the whole regularization path for the general fused lasso problem. Hoefling (2010) developed a path algorithm for solving (1) when the matrix $X^T X$ is nonsingular, where X is the design matrix. This technique is not applicable to cases with large dimension of β and small number of observations, such as gene expression and brain imaging analysis. Tibshirani and Taylor (2011) extended the path algorithm to include all design matrices X, by computing the regularization path of the dual problem. Although fairly general, this version of the path algorithm does not scale well with data dimension, as the knots of the piecewise linear solution path

become very dense. Many of the proposed approaches are versions of the *operator splitting methods* or their dual versions, *alternating direction methods* (see, e.g., Boyd et al., 2010, Combettes and Pesquet, 2011, and the references therein). Although fairly general and universal, they frequently suffer from slow tail convergence (see He and Yuan, 2011 and the references therein).

Thus, a need arises to develop a general approach that can solve large scale structured regularization problem efficiently. For such an approach to be successful in practice, it should guarantee to converge at a fast rate, be able to handle massive data sets, and should not rely on approximating the penalty function. In this paper, we propose a framework based on the alternating linearization algorithm of Kiwiel et al. (1999), that satisfies all these requirements.

We consider the following generalization of (1):

$$\mathcal{L}(\beta) = f(\beta) + \lambda \|R\beta\|_{\diamond}, \quad \lambda > 0, \tag{2}$$

where $\|\cdot\|_{\diamond}$ is a norm in $\mathbb{R}^m$. Our considerations and techniques will apply to several possible choices of this norm, in particular, to the ℓ_1 norm $\|\cdot\|_1$, and to the total variation norm $\|\cdot\|_{\mathrm{TV}}$ used in image processing.

Formally, we write the objective function as a sum of two convex functions,

$$\mathcal{L}(\beta) = f(\beta) + h(\beta), \tag{3}$$

where $f(\beta)$ is a loss function, which is assumed to be convex with respect to β, and $h(\cdot)$ is a convex penalty function. Any of the functions (or both) may be nonsmooth, but an essential requirement of our framework is that each of them can be easily minimized with respect to β, when augmented by a separable linear-quadratic term $\sum_{i=1}^{p} \left(s_i\beta_i + d_i\beta_i^2\right)$, with some vectors $s, d \in \mathbb{R}^p$, $d > 0$. Our method bears resemblance to operator splitting and alternating direction approaches, but differs from them in the fact that it is *monotonic* with respect to the values of (3). We discuss these relations and differences later in Section 2.2, but roughly speaking, a special test applied at every iteration of the method decides which of the operator splitting iterations is the most beneficial one.

In our applications, we focus on the quadratic loss function $f(\cdot)$ and the penalty function in the form of generalized lasso (2), as the most important case, where comparison with other approaches is available. This case satisfies the requirement specified above, and allows for substantial specialization and acceleration of the general framework of alternating linearization. In fact, it will be clear from our presentation that any convex loss function $f(\cdot)$ can be handled in exactly the same way.

An important feature of our approach is that problems with the identity design matrix are solved exactly in one iteration, even for very large dimension.

The remainder of the paper is organized as follows. In Section 2, we introduce the alternating linearization method and we discuss its relations to other approaches. Section 3 briefly discusses the application to lasso problems. In Section 4 we describe the application to general structured regularization problems. Section 5 presents simulation results and real data examples, which illustrate the efficacy, accuracy, and scalability of the alternating linearization method. Concluding remarks are presented in Section 6. The appendix contains details about the algorithms used to solve the subproblems of the alternating linearization method.

2. The Alternating Linearization Method

In this section, we describe the alternating linearization (ALIN) approach to minimize (3).

2.1 Outline of the Method

The ALIN is an iterative method, which generates a sequence of approximations $\{\hat{\beta}^k\}$ converging to a solution of the original problem (3), and two auxiliary sequences: $\{\tilde{\beta}_h^k\}$ and $\{\tilde{\beta}_f^k\}$, where k is the iteration number. Each iteration of the ALIN algorithm consists of solving two subproblems: the *h-subproblem* and the *f-subproblem*, and of an *update step*, applied after any of the subproblems, or after each of them.

At the beginning we set $\tilde{\beta}_f^0 = \hat{\beta}^0$, where $\hat{\beta}^0$ is the starting point of the method. In the description below, we suppress the superscript k denoting the iteration number, to simplify notation.

The h-subproblem

We linearize $f(\cdot)$ at $\tilde{\beta}_f$, and approximate it by the function

$$\tilde{f}(\beta) = f(\tilde{\beta}_f) + s_f^T(\beta - \tilde{\beta}_f).$$

If $f(\cdot)$ is differentiable, then $s_f = \nabla f(\tilde{\beta}_f)$; for a general convex $f(\cdot)$, we select a subgradient $s_f \in \partial f(\tilde{\beta}_f)$. In the first iteration, this may be an arbitrary subgradient; at later iterations special selection rules apply, as described in (2.1) below.

The approximation is used in the optimization problem

$$\min_{\beta} \ \tilde{f}(\beta) + h(\beta) + \tfrac{1}{2}\|\beta - \hat{\beta}\|_D^2, \tag{4}$$

in which the last term is defined as follows:

$$\|\beta - \hat{\beta}\|_D^2 = (\beta - \hat{\beta})^T D(\beta - \hat{\beta}),$$

with a diagonal matrix $D = \mathrm{diag}\{d_j,\, j = 1, \ldots, p\}$, $d_j > 0$, $j = 1, \ldots, p$. The solution of the h-subproblem (4) is denoted by $\tilde{\beta}_h$.

We complete this stage by calculating the subgradient of $h(\cdot)$ at $\tilde{\beta}_h$, which features in the optimality condition for the minimum in (4):

$$0 \in s_f + \partial h(\tilde{\beta}_h) + D(\tilde{\beta}_h - \hat{\beta}).$$

Elementary calculation yields the right subgradient $s_h \in \partial h(\tilde{\beta}_h)$:

$$s_h = -s_f - D(\tilde{\beta}_h - \hat{\beta}). \tag{5}$$

The f-subproblem

Using the subgradient s_h we construct a linear minorant of the penalty function $h(\cdot)$ as follows:

$$\tilde{h}(\beta) = h(\tilde{\beta}_h) + s_h^T(\beta - \tilde{\beta}_h).$$

This approximation is employed in the optimization problem

$$\min_{\beta} \; f(\beta) + \tilde{h}(\beta) + \tfrac{1}{2}\|\beta - \hat{\beta}\|_D^2. \tag{6}$$

The optimal solution of this problem is denoted by $\tilde{\beta}_f$. It will be used in the next iteration as the point at which the new linearization of $f(\cdot)$ will be constructed. The next subgradient of $f(\cdot)$ to be used in the h-subproblem will be

$$s_f = -s_h - D(\tilde{\beta}_f - \hat{\beta}).$$

The update step

The update step can be applied after any of the subproblems, or after both of them. It changes the current best approximation of the solution $\hat{\beta}$, if certain improvement conditions are satisfied. We describe it here for the case of applying the update step after the f-subproblem; analogous operations are carried out if the update step is applied after the h-subproblem.

At the beginning of the update step the stopping criterion is verified. If

$$f(\tilde{\beta}_f) + \tilde{h}(\tilde{\beta}_f) \geq f(\hat{\beta}) + h(\hat{\beta}) - \varepsilon, \tag{7}$$

the algorithm terminates. Here $\varepsilon > 0$ is the stopping test parameter.

If the the stopping test is not satisfied, we check the inequality

$$f(\tilde{\beta}_f) + h(\tilde{\beta}_f) \leq (1-\gamma)\big[f(\hat{\beta}) + h(\hat{\beta})\big] + \gamma\big[f(\tilde{\beta}_f) + \tilde{h}(\tilde{\beta}_f)\big]. \tag{8}$$

If it is satisfied, then we update $\hat{\beta} \leftarrow \tilde{\beta}_f$; otherwise $\hat{\beta}$ remains unchanged. Here the parameter $\gamma \in (0,1)$. In the implementation, we use $\gamma = 0.2$. The choice of this parameter does not influence the overall performance of the algorithm.

If the update step is applied after the h-subproblem, we use $\tilde{\beta}_h$ instead of $\tilde{\beta}_f$ in the inequalities (7) and (8).

The update step is a crucial component of the alternating linearization algorithm; it guarantees that the sequence $\{\mathcal{L}(\hat{\beta}^k)\}$ is monotonic (see Lemma 2 in Section 2.3), and it stabilizes the entire algorithm (see the remarks at the end of Section 5.2). It is a specialized form of the main distinction between *null* and *serious* steps in *bundle methods* for nonsmooth optimization. The Reader may consult the books of Bonnans et al. (2003); Hiriart-Urruty and Lemaréchal (1993); Kiwiel (1985); Ruszczyński (2006), and the references therein, for the theory of bundle methods and the significance of null and serious steps in these methods.

In Lemma 3 in Subsection 2.3 we show that under simple conditions the method is convergence linearly between successive serious steps.

2.2 Relation to Operator Splitting and Alternating Direction Methods

Our approach is intimately related to *operator splitting methods* and their dual versions, *alternating direction methods*, which are recently very popular in the area of signal processing (see, e.g., Boyd et al., 2010; Combettes and Pesquet, 2011; Fadili and Peyré, 2011). To

discuss these relations, it is convenient to present our method formally, and to introduce two running *proximal centers*:

$$z_f = \hat{\beta} - D^{-1}s_f,$$
$$z_h = \hat{\beta} - D^{-1}s_h.$$

After elementary manipulations we can absorb the linear terms into the quadratic terms and summarize the alternating linearization method as follows.

Algorithm 1 Alternating Linearization

1: **repeat**
2: $\tilde{\beta}_h \leftarrow \arg\min \left\{h(\beta) + \frac{1}{2}\|\beta - z_f\|_D^2\right\}$
3: **if** (*Update Test for* $\tilde{\beta}_h$) **then**
4: $\hat{\beta} \leftarrow \tilde{\beta}_h$
5: **end if**
6: $z_h \leftarrow \hat{\beta} + \tilde{\beta}_h - z_f$
7: $\tilde{\beta}_f \leftarrow \arg\min \left\{f(\beta) + \frac{1}{2}\|\beta - z_h\|_D^2\right\}$
8: **if** (*Update Test for* $\tilde{\beta}_f$) **then**
9: $\hat{\beta} \leftarrow \tilde{\beta}_f$
10: **end if**
11: $z_f \leftarrow \hat{\beta} + \tilde{\beta}_f - z_h$
12: **until** (*Stopping Test*)

The *Update Test* in lines 3 and 8 is the corresponding version of inequality (8). The *Stopping Test* is inequality (7).

If we assume that the update steps in lines 4 and 9 are carried out after *every* h-subproblem and *every* f-subproblem, without verifying the update test (8), then the method becomes equivalent to a scaled version of the Peaceman–Rachford algorithm (originally proposed by Peaceman and Rachford (1955) for PDEs and later generalized and analyzed by Lions and Mercier (1979); see also Combettes (2009) and the references therein). If $D = \rho I$ with $\rho > 0$, then we obtain an unscaled version of this algorithm.

If we assume that the update steps are carried out after *every* h-subproblem without verifying inequality (8), but *never* after f-subproblems, then the method becomes equivalent to a scaled version of the Douglas–Rachford algorithm (introduced by Douglas and Rachford (1956), and generalized and analyzed by Lions and Mercier (1979); see also Bauschke and Combettes (2011) and the references therein). As the roles of f and h can be switched, the method in which updates are carried always after f-subproblems, but never after h-subproblems, is also equivalent to a scaled Douglas–Rachford method.

Operator splitting methods are not monotonic with respect to the values of the objective function $\mathcal{L}(\beta)$. Their convergence is based on monotonicity with respect to the distance to the optimal solution of the problem (Lions and Mercier, 1979; Eckstein and Bertsekas, 1992).

In contrast, the convergence mechanism of our method is different; it draws from some ideas of bundle methods in nonsmooth optimization (Hiriart-Urruty and Lemaréchal, 1993; Kiwiel, 1985; Ruszczyński, 2006). Its key element is the update test employed in (8). At every iteration we decide whether it is beneficial to make a Peaceman–Rachford step, any

of the two possible Douglas–Rachford steps, or none. In the latter case, which we call the *null step*, $\hat{\beta}$ remains unchanged, but the trial points $\tilde{\beta}_h$ and $\tilde{\beta}_f$ are updated. These updates continue, until $\tilde{\beta}_h$ or $\tilde{\beta}_f$ become better than $\hat{\beta}$, or until optimality is detected (*cf.* the remarks at the end of section 5.2). In may be worth noticing that the recent application of the idea of alternating linearization by Goldfarb et al. (2013) removes the update test from the method of Kiwiel et al. (1999), thus effectively reducing it to an operator splitting method.

Alternating direction methods are dual versions of the operator splitting methods, applied to the following equivalent form of the problem of minimizing (3):

$$\min f(\beta_1) + h(\beta_2), \quad \text{subject to} \quad \beta_1 = \beta_2.$$

In regularized signal processing problems, when $f(\beta) = \varphi(X\beta)$ with some fixed matrix X, the convenient problem formulation is

$$\min \varphi(v) + h(\beta), \quad \text{subject to} \quad v = X\beta.$$

The dual functional,

$$L_D(\lambda) = \min_v \left\{\varphi(v) - \lambda^T v\right\} + \min_\beta \left\{h(\beta) + \lambda^T X\beta\right\},$$

has the form of a sum of two functions, and the operator splitting methods apply. The reader may consult the papers of Boyd et al. (2010) and Combettes and Pesquet (2011) for appropriate derivations. It is also worth mentioning that the alternating direction methods are sometimes called *split Bregman methods* in the signal processing literature (see, e.g., Goldstein and Osher, 2009; Ye and Xie, 2011, and the references therein). Recently, Qin and Goldfarb (2012) applied alternating direction methods to some structured regularization problems resulting from group lasso models.

However, to apply our alternating linearization method to the dual problem of maximizing $L_D(\lambda)$, we would have to be able to quickly compute the value of the dual functions, in order to verify the update condition (8), as discussed in detail in Kiwiel et al. (1999). The second dual function, $\min_\beta \left\{h(\beta) + \lambda^T X\beta\right\}$ is rather difficult to evaluate, and it makes the update test time consuming. Without this test, our method reduces to the alternating direction method, which does not have descent properties, and whose tail convergence may be slow. Our experiments reported at the end of Section 5.2 confirm these observations.

2.3 Convergence

Convergence properties of the alternating linearization method follow from the general theory developed by Kiwiel et al. (1999). Indeed, after the change of variables $\xi = D^{1/2}\beta$ we see that the method is identical to Algorithm 3.1 of Kiwiel et al. (1999), with $\rho_k = 1$. The following statement is a direct consequence of (Kiwiel et al., 1999, Theorem 4.8).

Theorem 1 *Suppose that the set of minima of the function* (3) *is nonempty. Then the sequence* $\{\hat{\beta}^k\}$ *generated by the algorithm is convergent to a minimum point* β^* *of the function* (3)*. Moreover, every accumulation point* (s_f^*, s_h^*) *of the sequence* $\{(s_f^k, s_h^k)\}$ *satisfies the relations:* $s_f^* \in \partial f(\beta^*)$*,* $s_h^* \in \partial h(\beta^*)$*, and* $s_f^* + s_h^* = 0$*.*

For structured regularization problems the assumption of the theorem is satisfied, because both the loss function $f(\cdot)$ and the regularizing function $h(\cdot)$ are bounded from below, and one of the purposes of the regularization term is to make the set of minima of the function $\mathcal{L}(\cdot)$ nonempty and bounded.

It may be of interest to look closer at the stopping test (7) employed in the update step.

Lemma 2 *Suppose β^* is the unique minimum point of $\mathcal{L}(\cdot) = f(\cdot) + h(\cdot)$ and let $\alpha > 0$ be such that $\mathcal{L}(\beta) - \mathcal{L}(\beta^*) \geq \alpha\|\beta - \beta^*\|_D^2$ for all β. Then the stopping criterion (7) implies that*

$$\mathcal{L}(\hat{\beta}) - \mathcal{L}(\beta^*) \leq \frac{\varepsilon}{\alpha}. \tag{9}$$

Proof As $\tilde{h}(\cdot) \leq h(\cdot)$, inequality (7) implies that

$$\begin{aligned} \min_\beta \Big\{ f(\beta) + h(\beta) + \frac{1}{2}\|\beta - \hat{\beta}\|_D^2 \Big\} &\geq \min_\beta \Big\{ f(\beta) + \tilde{h}(\beta) + \frac{1}{2}\|\beta - \hat{\beta}\|_D^2 \Big\} \\ &= f(\tilde{\beta}_f) + \tilde{h}(\tilde{\beta}_f) + \frac{1}{2}\|\tilde{\beta}_f - \hat{\beta}\|_D^2 \\ &\geq f(\hat{\beta}) + h(\hat{\beta}) - \varepsilon. \end{aligned} \tag{10}$$

The expression on the left hand side of this inequality is the Moreau–Yosida regularization of the function $\mathcal{L}(\cdot)$ evaluated at $\hat{\beta}$. By virtue of (Ruszczyński, 2006, Lemma 7.12), after setting $\tilde{x} = \beta^*$ and with the norm $\|\cdot\|_D$, the Moreau–Yosida regularization satisfies the following inequality:

$$\min_\beta \Big\{ \mathcal{L}(\beta) + \frac{1}{2}\|\beta - \hat{\beta}\|_D^2 \Big\} \leq \mathcal{L}(\hat{\beta}) - \frac{\big(\mathcal{L}(\hat{\beta}) - \mathcal{L}(\beta^*)\big)^2}{\|\hat{\beta} - \beta^*\|_D^2}.$$

Combining this inequality with (10) and simplifying, we conclude that

$$\frac{\big(\mathcal{L}(\hat{\beta}) - \mathcal{L}(\beta^*)\big)^2}{\|\hat{\beta} - \beta^*\|_D^2} \leq \varepsilon.$$

Substitution of the denominator by the upper estimate $\big(\mathcal{L}(\beta) - \mathcal{L}(\beta^*)\big)/\alpha$ yields (9). ■

If β^* is unique then $\mathcal{L}(\cdot)$ grows at least quadratically in the neighborhood of β^*. This implies that $\alpha > 0$ satisfying the assumptions of Lemma 2 exists. Our use of the norm $\|\cdot\|_D$ amounts to comparing the function $(\beta - \beta^*)^T X^T X(\beta - \beta^*)$ to its diagonal approximation $(\beta - \beta^*)^T D(\beta - \beta^*)$ for $D = \mathrm{diag}(X^T X)$. The reader may also consult (Ruszczyński, 1995, Lemma 1) for the accuracy of the diagonal approximation when the matrix X is sparse.

By employing the estimate of Lemma 2, we can now prove linear rate of convergence of the method between serious steps.

Lemma 3 *Suppose β^* is the unique minimum point of $\mathcal{L}(\cdot) = f(\cdot) + h(\cdot)$ and let $\alpha > 0$ be such that $\mathcal{L}(\beta) - \mathcal{L}(\beta^*) \geq \alpha\|\beta - \beta^*\|_D^2$ for all β. Then at every serious step, when the update test (8) is satisfied, we have the inequality*

$$\mathcal{L}(\tilde{\beta}_f) - \mathcal{L}(\beta^*) \leq (1 - \gamma\alpha)\big[\mathcal{L}(\hat{\beta}) - \mathcal{L}(\beta^*)\big]. \tag{11}$$

Proof If follows from inequality (8) that

$$\mathcal{L}(\tilde{\beta}_f) \le (1-\gamma)\mathcal{L}(\hat{\beta}) + \gamma\big[f(\tilde{\beta}_f) + \tilde{h}(\tilde{\beta}_f)\big].$$

Using Lemma 2, we obtain

$$\mathcal{L}(\hat{\beta}) - \mathcal{L}(\beta^*) \le \frac{1}{\alpha}\big[\mathcal{L}(\hat{\beta}) - f(\tilde{\beta}_f) - \tilde{h}(\tilde{\beta}_f)\big].$$

Combining these inequalities and simplifying, we conclude that

$$\begin{aligned}\mathcal{L}(\tilde{\beta}_f) &\le (1-\gamma)\mathcal{L}(\hat{\beta}) + \gamma\Big\{\alpha\mathcal{L}(\beta^*) - \alpha\mathcal{L}(\hat{\beta}) + \mathcal{L}(\hat{\beta})\Big\} \\ &= \mathcal{L}(\hat{\beta}) - \gamma\alpha\big[\mathcal{L}(\hat{\beta}) - \mathcal{L}(\beta^*)\big].\end{aligned}$$

Subtracting $\mathcal{L}(\beta^*)$ from both sides, we obtain the linear rate (11). ■

If the original problem is to minimize (3) subject to the constraint that $\beta \in B$ for some convex closed set B, we can formally add the indicator function of this set,

$$\delta(\beta) = \begin{cases} 0 & \text{if } \beta \in B, \\ +\infty & \text{if } \beta \notin B, \end{cases}$$

to $f(\cdot)$ or to $h(\cdot)$ (whichever is more convenient). This will result in including the constraint in one of the subproblems, and changing the subgradients accordingly. The theory of Kiwiel et al. (1999) covers this case as well, and Theorem 1 remains valid.

3. Application to Lasso Regression

First, we demonstrate the alternating linearization algorithm (ALIN) on the classical lasso regression problem. Due to the separable nature of the penalty function, very efficient coordinate descent methods are applicable to this problem as well (Tseng, 2001), but we wish to illustrate our approach on the simplest case first.

In the lasso regression problem we have

$$f(\beta) = \tfrac{1}{2}\|y - X\beta\|_2^2, \qquad h(\beta) = \lambda\|\beta\|_1,$$

where X is the $n \times p$ design matrix, $y \in \mathbb{R}^n$ is the vector of response variables, $\beta \in \mathbb{R}^p$ is the vector of regression coefficients, and $\lambda > 0$ is a parameter of the model.

We found it essential to use $D = \mathrm{diag}(X^T X)$, that is, $d_j = X_j^T X_j$, $j = 1, \ldots, p$. This choice is related to the *diagonal quadratic approximation* of the function $f(\beta) = \frac{1}{2}\|y - X\beta\|_2^2$, which was employed (for similar objectives in the context of augmented Lagrangian minimization) by Ruszczyński (1995). Indeed, in the h-subproblem in the formula (12) below, the quadratic regularization term is a quadratic form built on the diagonal of the Hessian of $f(\cdot)$.

The h-subproblem

The problem (4), after skipping constants, simplifies to the following form

$$\min_{\beta} \ s_f^T \beta + \lambda \|\beta\|_1 + \tfrac{1}{2}\|\beta - \hat{\beta}\|_D^2, \tag{12}$$

with $s_f = X^T(X\tilde{\beta}_f - y)$. Writing $\tau_j = \hat{\beta} - \tilde{s}_{fj}/d_j$, we obtain the following closed form solutions of (12), which can be calculated component-wise:

$$\tilde{\beta}_{hj} = \operatorname{sgn}(\tau_j) \max\Big(0, |\tau_j| - \frac{\lambda}{d_j}\Big), \quad j = 1, \ldots, p.$$

The subgradient s_h of $h(\cdot)$ at $\tilde{\beta}_h$ is calculated by (5).

The f-subproblem

The problem (6), after skipping constants, simplifies to the unconstrained quadratic programming problem

$$\min_{\beta} \ s_h^T \beta + \tfrac{1}{2}\|y - X\beta\|_2^2 + \tfrac{1}{2}\|\beta - \hat{\beta}\|_D^2.$$

Its solution can be obtained by solving the following symmetric linear system in $\delta = \beta - \hat{\beta}$:

$$(X^T X + D)\delta = X^T(y - X\hat{\beta}) - s_h. \tag{13}$$

This system can be efficiently solved by the preconditioned conjugate gradient method (see, e.g., Golub and Van Loan, 1996), with the diagonal preconditioner $D = \operatorname{diag}(X^T X)$. Its application does not require the explicit form of the matrix $X^T X$; only matrix-vector multiplications with X and X^T are employed, and they can be implemented with sparse data structures.

The numerical accuracy of this approach is due to the improved condition index of the resulting matrix, as explained in the following lemma.

Lemma 4 *The application of the preconditioned conjugate gradient method with preconditioner D to the system* (13) *is equivalent to the application of the conjugate gradient method to a system with a symmetric positive definite matrix $\bar{H}$ whose condition index is at most $p + 1$.*

Proof By construction, the preconditioned conjugate gradient method with a preconditioner D applied to a system with a matrix H is the standard conjugate gradient method applied to a system with the matrix $\bar{H} = D^{-\frac{1}{2}} H D^{-\frac{1}{2}}$. Substituting the matrix from (13), we obtain:

$$\bar{H} = D^{-\frac{1}{2}}(X^T X + D)D^{-\frac{1}{2}} = D^{-\frac{1}{2}} X^T X D^{-\frac{1}{2}} + I.$$

Define $\bar{X} = X D^{-\frac{1}{2}}$. By the construction of D, all columns $\bar{x}^j$, $j = 1, \ldots, p$, of $\bar{X}$ have Euclidean length 1.

The condition index of $\bar{H}$ is equal to

$$\operatorname{cond}(\bar{H}) = \frac{\lambda_{\max}(\bar{X}^T \bar{X}) + 1}{\lambda_{\min}(\bar{X}^T \bar{X}) + 1}.$$

As the matrix $\bar{X}^T\bar{X}$ is positive semidefinite, $\lambda_{\min}(\bar{X}^T\bar{X}) \geq 0$. To estimate $\lambda_{\max}(\bar{X}^T\bar{X})$ suppose v is the corresponding eigenvector of Euclidean length 1. We obtain the chain of relations:

$$\sqrt{\lambda_{\max}(\bar{X}^T\bar{X})} = \|\bar{X}v\|_2 = \Big\| \sum_{j=1}^{p} v_j \bar{x}^j \Big\|_2 \leq \sum_{j=1}^{p} |v_j| \|\bar{x}^j\|_2 = \|v\|_1 \leq \sqrt{p}\|v\|_2 = \sqrt{p}.$$

Therefore, the condition index of $\bar{H}$ is at most $p+1$. ■

While this universal estimate can be still very large, our experience is that the preconditioned conjugate gradient method solves the system (13) in a small number of iterations even for large p (see the discussion following Figure 4).

4. Application to General Structured Regularization Problems

In the following we apply the alternating linearization algorithm to solve more general structured regularization problems including the generalized Lasso (2). Here we assume the least square loss, as in the previous subsection. The objective function can be written as follows:

$$\mathcal{L}(\beta) = f(\beta) + h(\beta) = \tfrac{1}{2}\|y - X\beta\|_2^2 + \lambda\|R\beta\|_\diamond. \tag{14}$$

For example, for the one-dimensional fused lasso, R is the following $(p-1)\times p$ matrix:

$$R = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & -1 & 1 \end{bmatrix},$$

and the norm $\|\cdot\|_\diamond$ is the ℓ_1-norm $\|\cdot\|_1$, but our derivations are valid for any form of R, and any norm $\|\cdot\|_\diamond$.

The h-subproblem

The h-subproblem can be equivalently formulated as follows:

$$\min_{\beta,z} \; s_f^T\beta + \lambda\|z\|_\diamond + \tfrac{1}{2}\|\beta - \hat{\beta}\|_D^2 \quad \text{subject to} \quad R\beta = z. \tag{15}$$

Owing to the use of $D = \mathrm{diag}(X^TX)$, and with $s_f = X^T(X\hat{\beta} - y)$, it is a quite accurate approximation of the original problem, especially for sparse X (Ruszczyński, 1995).

The Lagrangian of problem (15) has the form

$$L(\beta, z, \mu) = s_f^T\beta + \lambda\|z\|_\diamond + \mu^T(R\beta - z) + \tfrac{1}{2}\|\beta - \hat{\beta}\|_D^2,$$

where μ is the dual variable. Consider the dual norm $\|\cdot\|_*$, defined as follows:

$$\|\mu\|_* = \max_{\|z\|_\diamond \leq 1} \mu^T z, \qquad \|z\|_\diamond = \max_{\|\mu\|_* \leq 1} \mu^T z.$$

We see that the minimum of the Lagrangian with respect to z is finite if and only if $\|\mu\|_* \leq \lambda$ (Ruszczyński, 2006, Example 2.94). Under this condition, the minimum value of the z-terms is zero and we can eliminate them from the Lagrangian. We arrive to its reduced form,

$$\hat{L}(\beta, \mu) = s_f^T \beta + \mu^T R\beta + \tfrac{1}{2}\|\beta - \hat{\beta}\|_D^2. \tag{16}$$

To calculate the dual function, we minimize $\hat{L}(\beta, \mu)$ over $\beta \in \mathbb{R}^p$. After elementary calculations, we obtain the solution

$$\tilde{\beta}_h = \hat{\beta} - D^{-1}(s_f + R^T \mu). \tag{17}$$

Substituting it back to (16), we arrive to the following dual problem:

$$\max_{\mu} \; -\tfrac{1}{2}\mu^T R D^{-1} R^T \mu + \mu^T R(\hat{\beta} - D^{-1} s_f) \quad \text{subject to} \quad \|\mu\|_* \leq \lambda. \tag{18}$$

This is a norm-constrained optimization problem. Its objective function is quadratic, and the specific form of the constraints depends on the norm $\|\cdot\|_\diamond$ used in the regularizing term of (2).

The case of the ℓ_1-norm

If the norm $\|\cdot\|_\diamond$ is the ℓ_1-norm $\|\cdot\|_1$, then the dual norm is the ℓ_∞-norm:

$$\|\mu\|_* = \|\mu\|_\infty = \max_{1 \leq j \leq m} |\mu_j|.$$

In this case (18) becomes a box-constrained quadratic programming problem, for which many efficient algorithms are available. One possibility is the active-set box-constrained preconditioned conjugate gradient algorithm with spectral projected gradients, as described by Birgin and Martínez (2002); Friedlander and Martínez (1994). It should be stressed that its application does not require the explicit form of the matrix $RD^{-1}R^T$; only matrix-vector multiplications with R and R^T are employed, and they can be implemented with sparse data structures.

An even better possibility, due to the separable form of the constraints, is coordinate-wise optimization (see, e.g., Ruszczyński, 2006, Sec. 5.8.2) in the dual problem (18). In our experiments, the dual coordinate-wise optimization method strictly outperforms the box-constrained algorithm, in terms of the solution time.

The solution $\tilde{\mu}$ of the dual problem can be substituted into (17) to obtain the primal solution.

The case of a sum of ℓ_2-norms

Another important case arises when the vector $z = R\beta$ is split into I subvectors $z^1, z^2, \ldots, z^I$, and

$$\|z\|_\diamond = \sum_{i=1}^{I} \|z^i\|_2. \tag{19}$$

This is the group lasso model, also referred to as the ℓ_1/L_2-norm regularization (see, e.g., Qin and Goldfarb, 2012). A special case of it is the *total variation norm* is discussed in Section 5.4.

We can directly verify that the dual norm has the following form:

$$\|\mu\|_* = \max_{1\le i\le I} \|\mu^i\|_2.$$

It follows that problem (18) is a block-quadratically constrained quadratic optimization problem:

$$\begin{aligned} &\max_{\mu} \ -\tfrac{1}{2}\mu^T R D^{-1} R^T \mu + \mu^T R(\hat{\beta} - D^{-1} s_f) \\ &\text{s. t. } \|\mu^i\|_2^2 \le \lambda^2, \quad i = 1, \ldots, I. \end{aligned}$$

This problem can be very efficiently solved by a cyclical block-wise optimization with respect to the subvectors $\mu^1, \mu^2, \ldots, \mu^I$. At each iteration of the method, optimization with respect to the corresponding subvector μ^j is performed, subject to one constraint $\|\mu^j\|_2^2 \le \lambda^2$. The other subvectors, μ^i, $i \ne j$ are kept fixed on their last values. After that, j is incremented (if $j < I$) or reset to 1 (if $j = I$), and the iteration continues. The method stops when no significant improvements over I steps can be observed. The dual block optimization method performs well in the applications we are interested in. General convergence theory can be found in Tseng (2001).

Again, the solution $\tilde{\mu}$ of the dual problem is substituted into (17) to obtain the primal solution.

The f-subproblem

We obtain the update $\tilde{\beta}_f$ by solving the linear equation system (13), exactly as in the lasso case.

The special case of $X = I$

If the design matrix $X = I$ in (14), then our method solves the problem in one iteration, when started from $\hat{\beta} = y$. Indeed, in this case we have $s_f = 0$, $D = I$, and the first h-subproblem becomes equivalent to the original problem (14):

$$\min_{\beta, z} \ \lambda\|z\|_\diamond + \tfrac{1}{2}\|\beta - y\|_2^2 \quad \text{subject to} \quad R\beta = z.$$

The dual problem (18) simplifies as follows:

$$\max_{\mu} \ -\tfrac{1}{2}\mu^T R R^T \mu + \mu^T R y \quad \text{subject to} \quad \|\mu\|_* \le \lambda.$$

It can be solved by the same block-wise optimization method, as in the general case. The optimal primal solution is then $\tilde{\beta}_h = y - R^T\mu$.

5. Numerical Experiments

In this section, we present results of a number of studies on simulated and real data, involving a variety of non-differentiable penalty functions. We compare the alternating linearization algorithm (ALIN) with competing approaches in terms of iteration counts, computation time, and estimation accuracy. All these studies were performed on an AMD 2.6 GHZ, 4 GB RAM computer using MATLAB.

5.1 ℓ_1 Regularization

In this section, we compare ALIN with some competing methods for solving the ℓ_1 regularization problem:

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1, \quad \lambda > 0.$$

The methods that we are comparing with are: SpaRSA, a type of iterative thresholding method (Wright et al., 2009; Xiao and Zhang, 2013); FISTA, a variation of the Nesterov method, considered to be state-of-the-art among the first order methods; and SPG, a spectral gradient method (den Berg and Friedlander, 2008). We follow the procedure described by Wright et al. (2009) to generate a data set for comparisons. The elements of the matrix X are generated independently using a Gaussian distribution with mean zero and variance 10^{-2}. The dimension of X is $n = 2^{10}$ by $p = 2^{12}$, $p = 2^{13}$, and $p = 2^{14}$. The true signal, β, is a vector with 160 randomly placed ± 1 spikes and zeros elsewhere. The dependent variables are $y = X\beta + \epsilon$, where ϵ is Gaussian noise with variance 10^{-4}.

		$p = 2^{12}$		$p = 2^{13}$		$p = 2^{14}$	
$\lambda = \tau$	*ALIN*	17.99	(10.68)	36.60	(15.08)	105.14	(40.88)
	FISTA	8.58	(4.00)	18.63	(9.35)	58.73	(42.01)
	SPARSA	8.18	(2.34)	8.18	(3.80)	34.23	(49.55)
	SPG	160.72	(27.48)	160.72	(47.82)	404.59	(79.62)
$\lambda = 10^{-1}\tau$	*ALIN*	9.35	(2.99)	34.01	(15.18)	74.06	(34.27)
	FISTA	16.91	(4.57)	36.43	(16.66)	131.91	(33.94)
	SPARSA	20.55	(10.08)	36.81	(19.29)	169.88	(73.01)
	SPG	136.26	(23.30)	186.94	(46.56)	460.71	(38.93)
$\lambda = 5 \times 10^{-2}\tau$	*ALIN*	6.30	(2.73)	21.83	(10.17)	65.79	(39.49)
	FISTA	18.88	(2.95)	48.37	(15.77)	158.73	(21.86)
	SPARSA	35.56	(11.32)	74.66	(24.49)	234.75	(64.67)
	SPG	140.00	(23.15)	190.43	(45.48)	473.78	(6.41)
$\lambda = 10^{-2}\tau$	*ALIN*	4.58	(2.05)	16.96	(10.99)	28.88	(16.03)
	FISTA	18.85	(3.04)	46.58	(14.91)	169.94	(19.76)
	SPARSA	33.52	(16.10)	76.69	(28.18)	214.85	(124.56)
	SPG	140.63	(22.43)	196.54	(43.96)	483.71	(4.51)
$\lambda = 10^{-3}\tau$	*ALIN*	3.68	(1.20)	6.67	(2.43)	20.16	(4.31)
	FISTA	18.88	(2.84)	45.40	(14.28)	162.85	(36.76)
	SPARSA	19.94	(12.10)	39.55	(27.45)	92.76	(102.53)
	SPG	138.73	(19.56)	201.74	(48.09)	467.91	(101.32)

Table 1: Average run time (in CPU seconds) and standard deviation (in parenthesis) comparison for combinations of dimension p and tuning parameter λ.

To make a fair comparison between the methods, we run FISTA on each instance of the problem. FISTA is set to run to "tol" $= 10^{-5}$ or $5,000$ iterations, whichever comes first. Then ALIN, SpaRSA, and SPG are set to run until the objective function values obtained are as good as that of FISTA. We set a parameter $\tau = 0.1\|X^T y\|_\infty$ and chose

values of $\lambda = \tau, 10^{-1}\tau$, $5 \times 10^{-2}\tau$, $10^{-2}\tau$, and $10^{-3}\tau$. We allow SpaRSA to run its *monotone* and *continuation* feature. *Continuation* is a special feature of SpaRSA for cases when the parameter λ is small. With this feature, SpaRSA computes the solutions for bigger values of λ and uses them to find solutions for smaller values of λ. We did not let SpaRSA use its special *de-bias* feature since it involves removing zero coefficients to reduce the size of the data set. This feature makes it unfair for the other competing methods. In Table 1, we report the average time elapsed (in seconds) and the standard deviation after 20 runs.

We can see that the performance of ALIN is comparable to the other methods. In terms of running time, ALIN does better than all competing methods for all but the largest λ considered. For the large value of λ, ALIN performs worse than FISTA and SPARSA. In this case, the solution is fairly close to the starting point 0, therefore the overhead cost of the update steps and the f-subproblem slow ALIN down. When the value of λ decreases, the benefits of these steps become more evident. ALIN outperforms other methods in terms of the running time, by factors of two to three.

		$p = 2^{12}$		$p = 2^{13}$		$p = 2^{14}$	
$\lambda = \tau$	*ALIN*	12.35	(0.05)	30.46	(0.05)	66.02	(0.05)
	FISTA	6.99	(0.05)	17.25	(0.05)	44.07	(0.05)
	SPARSA	3.16	(0.05)	6.14	(0.05)	12.26	(0.05)
	SPG	47.43	(0.05)	74.72	(0.05)	161.54	(0.05)
$\lambda = 10^{-1}\tau$	*ALIN*	5.45	(0.02)	19.17	(0.03)	62.13	(0.04)
	FISTA	12.08	(0.02)	25.40	(0.03)	79.48	(0.04)
	SPARSA	14.69	(0.02)	26.85	(0.03)	110.48	(0.04)
	SPG	53.54	(0.03)	93.18	(0.04)	175.77	(0.05)
$\lambda = 5 \times 10^{-2}\tau$	*ALIN*	4.97	(0.02)	18.03	(0.02)	57.55	(0.03)
	FISTA	15.58	(0.02)	32.26	(0.02)	94.91	(0.03)
	SPARSA	20.62	(0.02)	42.67	(0.02)	168.76	(0.03)
	SPG	55.82	(0.03)	94.13	(0.04)	177.17	(0.05)
$\lambda = 10^{-2}\tau$	*ALIN*	5.06	(0.01)	12.67	(0.02)	29.10	(0.04)
	FISTA	18.74	(0.01)	38.59	(0.02)	106.43	(0.04)
	SPARSA	38.76	(0.01)	79.33	(0.02)	191.51	(0.04)
	SPG	54.85	(0.03)	95.58	(0.04)	180.52	(0.05)
$\lambda = 10^{-3}\tau$	*ALIN*	3.78	(0.03)	4.98	(0.04)	9.59	(0.05)
	FISTA	21.61	(0.03)	38.50	(0.04)	106.53	(0.05)
	SPARSA	32.78	(0.03)	45.69	(0.04)	98.75	(0.05)
	SPG	56.90	(0.04)	96.95	(0.05)	183.67	(0.05)

Table 2: Average run time (in CPU seconds) and MSE (in parenthesis) comparison for combinations of dimension p and tuning parameter λ.

We further compare the efficiency of these methods by a validation experiment. All the features of the previous experiment remain the same, except that the noise variance is increased from 10^{-4} to 10^{-1} (which should favor higher regularization). We generate $n = 2^{11}$ samples, where half of them are used as the training set and the other half for

testing. For each of the five λ values, we apply the four algorithms to the training set. The fitted models are then validated on the testing set to obtain out-of-sample estimation errors (measured by mean square error). The results are reported in Table 2. All the methods obtain similar MSE on the testing data and the smallest MSE is achieved at $\lambda = 10^{-2}\tau$ and $\lambda = 5 \times 10^{-2}\tau$. In both cases, ALIN clearly performs the best among the four competing methods. As shown in Figure 1, ALIN is also the fastest in terms of the overall computation cost.

It's worth noting that the implementation of FISTA was in the C programming language, while ALIN was implemented as a MATLAB script. Moreover, SpaRSA is a very efficient method specially designed for separable regularization. From our numerical studies, for medium and small values of λ, FISTA makes very small improvement over $5,000$ iterations and SPARSA has to go through many previous values of λ to reach the desired level of objective function values.

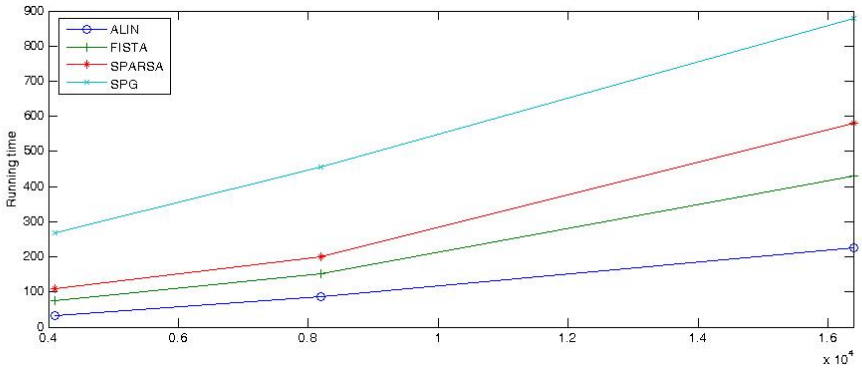


Figure 1: Run time (in CPU seconds) comparisons on the validation experiment for ALIN, FISTA, SPARSA and SPG. The sample size $n = 2^{11}$.

5.2 Fused Lasso Regularization

In this experiment, we compare the ALIN algorithm with four different approaches using data sets generated from a linear regression model $y = \sum_{j=1}^{p} x_j \beta_j + \epsilon$, with pre-specified coefficients β_j, and varying dimension p. The values of x_j are drawn from the multivariate normal distribution with zero mean and pairwise correlation of $\sigma = 0.3$. The noise ϵ is generated from the normal distribution with zero mean and variance equal to 0.01. Among the coefficients β_j, 10% equal 1, 20% equal 2, and the rest are zero. For instance, with $p = 100$, we may have

$$\beta_j = \begin{cases} 1 & \text{for } j = 11, 12, \ldots, 20, \\ 2 & \text{for } j = 21, \ldots, 40, \\ 0 & \text{otherwise.} \end{cases}$$

The regularization problem we attempt to solve is

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda \sum_{j=2}^{p} |\beta_j - \beta_{j-1}|, \quad \lambda > 0.$$

Table 3 reports the run times of ALIN and the four competing algorithms: the generic quadratic programming solver (SQOPT), an implementation of Nesterov's method, SLEP, of Liu et al. (2011); Nesterov (2007), the Split Bregman method of Ye and Xie (2011) (BREGMAN), and alternating direction method of multipliers (ADMM) (Eckstein, 2012). We fix the sample size at n=1000 and vary the dimension p of the problem from 5000 to 50000. To keep the comparison fair, we first run ADMM using the stopping criteria of Boyd et al. (2010) with $tol = 1e-6$ or 30 iterations since each iteration of ADMM can be expensive. Then, the other methods are run until they obtain the same objective function value, or their stopping criteria are satisfied. Each method is repeated on 10 different randomly generated data sets for different values of the tuning parameter λ, and the average running time is reported.

Judging from these results, ALIN clearly outperforms the other methods in terms of speed for most cases. The relative improvements on run time can be as much as 8-fold, depending on the experimental setting, and become more significant, when the data dimension grows. This is particularly significant in view of the fact that ALIN was implemented as a MATLAB code, as opposed to the executables in the other cases. Figure 2 presents the solutions obtained by ALIN, ADMM, SLEP and BREGMAN. The first three methods achieve results that are very close to the original β, and their final objective function values are similar. From our experience, BREGMAN performs well when the dimension p is not significantly larger than the number of observations n. When $p \gg n$, although BREGMAN has a very good running time, it tends to terminate early and does not provide accurate results. In Figure 2, we can see that the solution obtained by BREGMAN is not as good as those of the other three methods.

We further investigated how ALIN approaches the optimal objective function value compared to the other methods. Using the above simulated data set with $n = 1000$, $p = 5000$, and $\lambda = 0.5$, we run ADMM, ALIN, and SLEP until convergence. At each iteration, we calculated the difference between the optimal value $\mathcal{L}^*$ (obtained by SQOPT) and the current function value for each method. Figure 3 displays (in a logarithmic scale) the progress of these methods. It is clear that ALIN achieves the same accuracy as SLEP and ADMM in a much smaller number of iterations. Furthermore, the convergence of ALIN is monotonic, whereas that of SLEP is not. ADMM makes good improvements at the beginning but then it takes thousands of iterations to get to the desired objective function value.

In Figure 4 we show how the ADMM, ALIN and SLEP scale with the dimension of the problem. From this plot, the computation cost of ALIN is consistently lower than those of ADMM and SLEP. The efficiency of the method is due mainly to its good convergence properties, but also to the efficiency of the preconditioned conjugate gradient method for solving the subproblems. It employs sparse data structures and converges rapidly. Usually, between 10 and 20 iterations of the conjugate gradient method are sufficient to find the solution of a subproblem.

		$p = 5000$	p=10,000	p=20,000	p=50,000
$\lambda = 10^{-4}$	*ADMM*	47.45	324.93	590.22	973.48
	SQOPT	1076.00	NA	NA	NA
	SLEP	58.93	569.40	1041.91	2759.85
	ALIN	7.01	35.85	52.74	500.05
	BREGMAN	92.71	129.98	172.36	92.67
$\lambda = 10^{-3}$	*ADMM*	51.94	327.29	597.86	963.79
	SQOPT	1025.00	NA	NA	NA
	SLEP	59.65	570.25	1059.80	2750.72
	ALIN	11.45	67.69	72.04	632.32
	BREGMAN	90.86	129.08	158.72	91.88
$\lambda = 10^{-2}$	*ADMM*	52.37	322.74	599.56	1274.93
	SQOPT	1019.00	NA	NA	NA
	SLEP	59.24	576.82	1053.05	2729.88
	ALIN	27.68	100.29	225.30	810.70
	BREGMAN	86.70	129.29	153.66	90.47
$\lambda = 0.1$	*ADMM*	52.03	326.55	574.75	1528.20
	SQOPT	956.00	NA	NA	NA
	SLEP	56.58	572.34	1004.90	2679.26
	ALIN	31.42	292.73	466.84	597.30
	BREGMAN	93.93	125.65	136.92	77.98
$\lambda = 0.5$	*ADMM*	51.57	317.28	565.74	1482.76
	SQOPT	1015.00	NA	NA	NA
	SLEP	53.10	541.02	983.12	2420.88
	ALIN	22.74	211.35	473.82	908.08
	BREGMAN	97.38	129.68	151.67	71.19
$\lambda = 1$	*ADMM*	51.45	322.37	569.20	1397.77
	SQOPT	1029.00	NA	NA	NA
	SLEP	46.80	470.14	879.51	2404.60
	ALIN	18.15	169.31	477.02	983.31
	BREGMAN	85.14	136.82	170.35	61.48

Table 3: Run time (in CPU seconds) comparison for combinations of dimension p and tuning parameter λ for fused lasso problems

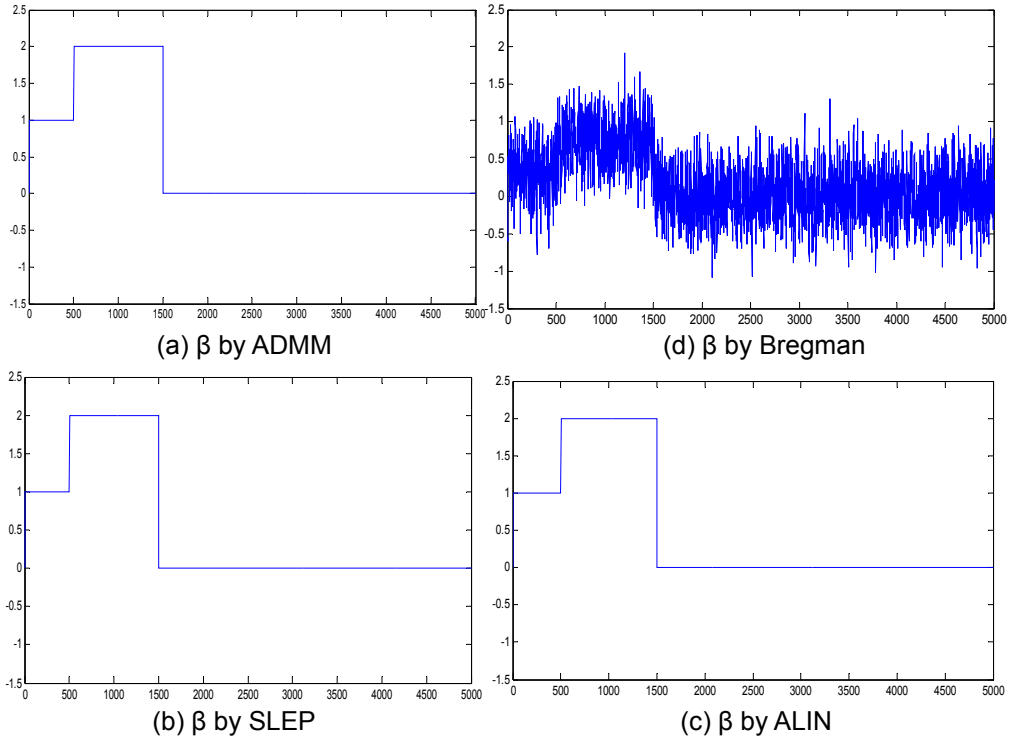


Figure 2: Results of using fused lasso penalty on a simulated data set with $n = 1000$, $p = 5000$, and $\lambda = 0.5$. Plots (a), (b), (c), and (d) correspond to results from ADMM, SLEP, ALIN, and BREGMAN, respectively.

The update test (8) is an essential element of the ALIN method. For example, in the case with $n = 1000$, $p = 5000$, and $\lambda = 0.1$, the update of $\hat{\beta}$ occurred in about 80% of the total of 70 iterations, while other iterations consisted only of improving alternating linearizations. If we allow updates of $\hat{\beta}$ at every step, the algorithm takes more than 5000 iterations to converge in this case. Similar behavior was observed in all other cases. These observations clearly demonstrate the difference between the alternating linearization method and the operator splitting methods.

5.3 CGH Data Example

In this study we present the results on analyzing the CGH data using fused lasso penalty. CGH is a technique for measuring DNA copy numbers of selected genes on the genome. The CGH array experiments return the log ratio between the number of DNA copies of the gene in the tumor cells and the number of DNA copies in the reference cells. A value greater

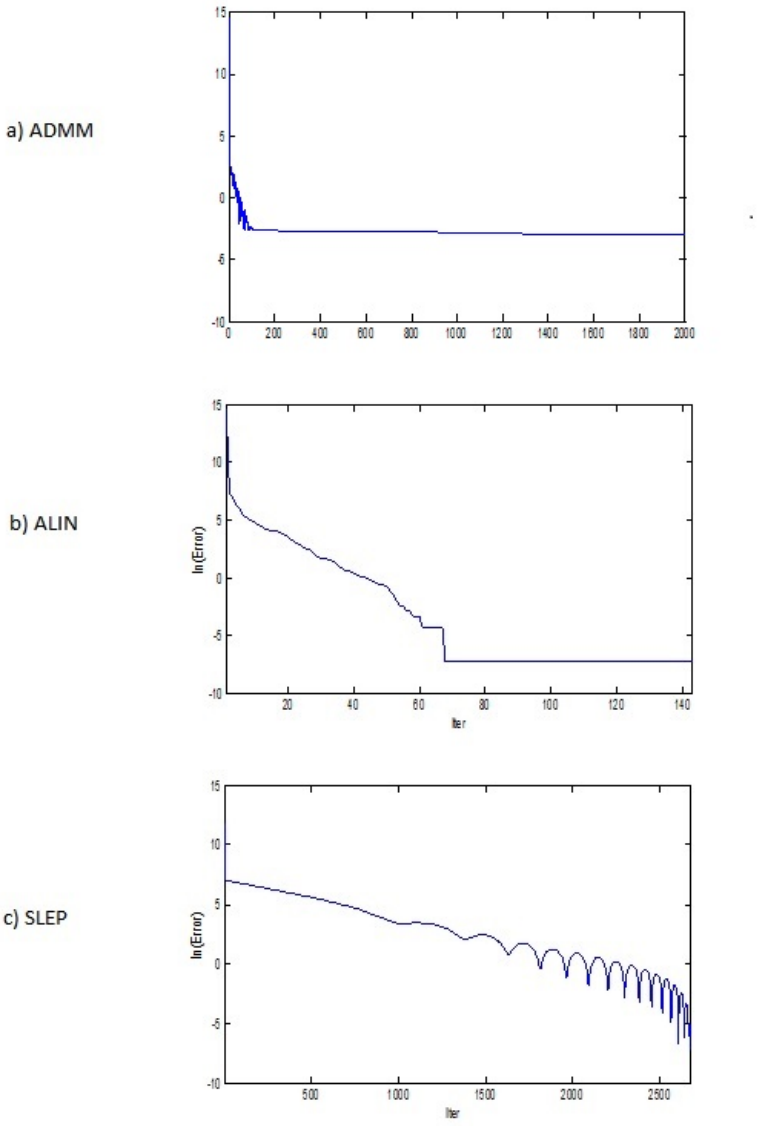


Figure 3: Simulated data set with $n = 1000$, $p = 5000$, $\lambda = 0.1$. Plots (a), (b) and (c): ln(Error) versus number of iterations for ADMM,ALIN, and SLEP respectively. Error is defined as the difference between the optimal value $\mathcal{L}^*$ (obtained by SQOPT) and those obtained by ADMM, ALIN, and SLEP respectively.

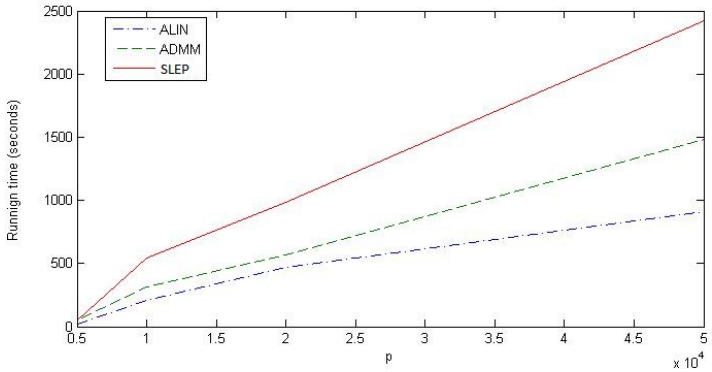


Figure 4: Running time of ADMM, SLEP and ALIN as dimension changes. The vertical axis is the run time in seconds, and the horizontal axis is the data dimension.

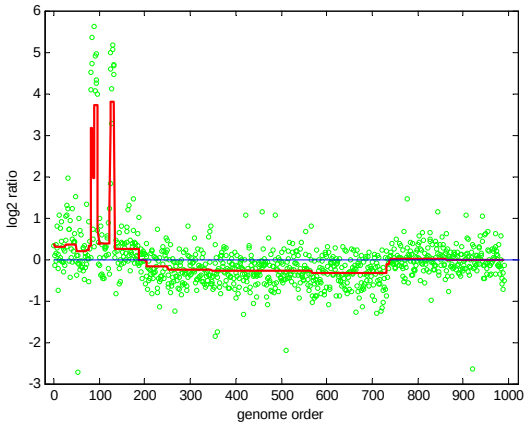


Figure 5: Fused lasso applied to CGH data, $\lambda = 3$.

than zero indicates a possible gain, while a value less than zero suggests possible losses. Tibshirani and Wang (2008) applied the fused lasso signal approximator for detecting such copy number variations. This is a simple one-dimensional signal approximation problem with the design matrix X being the identity matrix. Thus the advantage of ALIN over the other three methods is not significant, due to the overhead that ALIN has during the conjugate gradient method implemented in MATLAB. Indeed the solution time of ALIN is comparable to that of Bregman and SLEP.

Figure 5 presents the estimation results obtained by our ALIN method. The green dots shows the original CNV number, and the red line presents the fused lasso penalized estimates.

In Table 4, the running time comparisons between ADMM, SLEP, ALIN, and Bregman on the CGH data are reported. The fused lasso signal approximator problem has been solved successfully by many methods. Although the main focus of ALIN is on a general design matrix X, ALIN's performance is still comparable with other methods for this example.

	SLEP	ADMM	Alin	Bregman
λ=0.1	0.01	0.06	0.01	0.02
λ=0.5	0.01	0.05	0.01	0.02
λ=1	0.01	0.05	0.03	0.02
λ=3	0.01	0.06	0.06	0.02

Table 4: Average run time (in CPU seconds) comparison for different values of tuning parameter λ on the CGH data example.

5.4 Total Variation Based Image Reconstruction

In image recovery literature, two classes of regularizers are well known. One is the Tikhonov type operator, where the regularizing term is quadratic, and the other is the discrete total variation (TV) regularizer. The resulting objective function from the first type is relatively easy to minimize, but it tends to over-smooth the image, thus failing to preserve its sharpness (Wang et al., 2008). In the following experiment, we demonstrate the effectiveness of ALIN in solving TV-based image deblurring problems, with discrete TV (Rudin et al., 1992), as well as a comparison to the Tikhonov regularizer.

Although of similar form, higher-order fused lasso models are fundamentally different from the one-dimensional fused lasso, as the structural matrix R appearing in (2) is not full-rank and R^TR is ill-conditioned. This additional complication introduces considerable challenges in the path type algorithms (Tibshirani and Taylor, 2011), and additional computational steps need to be implemented to guarantee convergence. The ALIN algorithm does not suffer from complications due to the singularity of R, because the dual problem (18) is always well-defined and has a solution. Even if the solution is not unique, (17) is still an optimal solution of the h-subproblem, and the algorithm proceeds unaffected.

Let y be an $m \times n$ observed noisy image; one attempts to minimize the following objective function:

$$\mathcal{L}(\beta) = \tfrac{1}{2}\|y - \mathcal{A}(\beta)\|_2^2 + \lambda h(\beta), \tag{20}$$

where $h(\beta)$ is an image variation penalty, and $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ is a linear transformation. When $\mathcal{A}$ is the identity transformation, the problem is to *denoise* the image y, but we are rather interested in a significantly more challenging problem of *deblurring*, where $\mathcal{A}$ replaces each pixel with the average of its neighbors and itself (typically, a 3 by 3 block, except for the border).

The penalty can be defined as the ℓ_1-norm of the differences between neighboring pixels (ℓ_1-TV),

$$h(\beta) = \sum_{i=1}^{m-1}\sum_{j=1}^{n-1}\big(|\beta_{i,j}-\beta_{i+1,j}|+|\beta_{i,j}-\beta_{i,j+1}|\big)+\sum_{i=1}^{m-1}|\beta_{i,n}-\beta_{i+1,n}|+\sum_{j=1}^{n-1}|\beta_{m,j}-\beta_{m,j+1}|, \tag{21}$$

or as follows (ℓ_2-TV):

$$h(\beta) = \sum_{i=1}^{m-1}\sum_{j=1}^{n-1}\left(|\beta_{i,j}-\beta_{i+1,j}|^2+|\beta_{i,j}-\beta_{i,j+1}|^2\right)^{1/2}+\sum_{i=1}^{m-1}|\beta_{i,n}-\beta_{i+1,n}|+\sum_{j=1}^{n-1}|\beta_{m,j}-\beta_{m,j+1}|. \tag{22}$$

It is clear that both cases can be cast into the general form (2), with the operator R representing the evaluation of the differences $\beta_{i,j}-\beta_{i+1,j}$ and $\beta_{i,j}-\beta_{i,j+1}$. The regularizing function (21) corresponds to the ℓ_1-norm of $R\beta$, while the function (22) corresponds to a norm of form (19). In the latter case, we have mn blocks, each of dimension two, except for the border blocks, which are one-dimensional.

In the following experiments, we apply the ℓ_1-TV to recover noisy and blurred images to their original forms. The resulting regularization problems are rather complex. Deblurring a 256 by 256 image results in solving a very large generalized lasso problem (the matrix R has dimensions of about 262000×66000). The f-subproblem is solved using the block coordinate descent method and the h-subproblem is solved using the preconditioned conjugate gradient method with "tol" $= 10^{-5}$, as discussed previously. The fact that $\mathcal{A}$ and R are sparse matrices makes the implementation very efficient, as demonstrated in the numerical study.

First, we *blur* the image, by replacing each pixel with the average of its neighbors and itself. This operation defines the kernel operator $\mathcal{A}$ used in the loss function $\frac{1}{2}\|y-\mathcal{A}(\beta)\|_2^2$. Then we add $N(0, 0.02)$ noise to each pixel. Clearly, for image deblurring, the design matrix is no longer the identity matrix, thus the problem is more complicated than the image denoising problem. The deblurring results on a standard example ("Lena") are shown in Figure 6; similar deblurring results from ALIN and FISTA are observed.

Next, we run the image deblurring on a 1 Megapixel image. We compare the result of image deblurring using the ℓ_1-TV and a quadratic Tiknonov regularization approach, which corresponds to formula (22) without the square root operations:

$$h(\beta) = \sum_{i=1}^{m-1}\sum_{j=1}^{n-1}|\beta_{i,j}-\beta_{i+1,j}|^2+|\beta_{i,j}-\beta_{i,j+1}|^2+\sum_{i=1}^{m-1}|\beta_{i,n}-\beta_{i+1,n}|^2+\sum_{j=1}^{n-1}|\beta_{m,j}-\beta_{m,j+1}|^2. \tag{23}$$

The results are shown in Figure 7. It is seen that the ℓ_1-TV recovers a sharper image than the quadratic penalty. Deblurring with the two-dimensional fused lasso penalty yields an MSE of 7.2 with respect to the original image, while that of Tikhonov regularization is 9.3. Deblurring with the regularizer (21) has an almost identical effect as with (22).

There have been many efficient iterative methods proposed to solve this problem. Two outstanding general frameworks are a variation of Nesterov's gradient method (Nesterov, 2007) and the method of alternating direction (ADMM). SLEP is a variation of Nesterov method like FISTA, although it was specifically implemented for fused-lasso penalty. It is not directly applicable for total variation deblurring problem. We pick two algorithms to compare with ALIN in this numerical study: FISTA of Beck and Teboulle (2009), a very efficient first-order method for discrete total variation based image processing; and TVAL (Li et al., 2013), a method based on Augmented Lagrangian and Alternating Direction algorithm. TVAL solves a model equivalent to (20), but with a coefficient μ in front of the least-squares term, instead of λ at the regularization.

Figure 6: Results of deblurring using fused lasso penalty. Plots (a), (b), (c), and (d) correspond to the original image, the blurred image, the ALIN de-blurred image, and the FISTA de-blurred image, respectively.

Figure 7: Image deblurring on the "lion" data. The left plot is the result from the ℓ_1-TV penalty; the middle plot is from the Tikhonov penalty. The rightmost plot is the noisy and blurred image.

In the first comparison, we pick 10 random gray scale images with small size, typically 205×205, or approximately $40,000$ pixels. Following the same procedure as described by Beck and Teboulle (2009), the image is blurred using a 3×3 kernel and a Gaussian noise with variance 10^{-2} is added. The deblurring procedure is run with a few different values for λ. We let FISTA run for 100 iterations with the *monotone* feature, which results in a monotonic decrease of the objective function decrease, and the tolerance is set to 10^{-5}. Then we run ALIN to the same objective function value. TVAL, unfortunately, cannot reach the same objective function value. Thus we let TVAL run $10,000$ iterations or to "tol" $= 10^{-5}$, whichever comes first. To compare the quality of the restored image, we use the signal-to-noise (SNR) ratio defined as

$$SNR = 10 \log 10 \frac{\|u^0 - \tilde{u}\|^2}{\|u^0 - u\|^2},$$

where u^0 is the original image, $\tilde{u}$ is the mean intensity of the original image, and u is the restored image. In Table 5.4, we report the running time to produce the best quality restored image, where the regularization parameter $\lambda = 10^{-4}$, similar to what was suggested by (Li et al., 2013). We also report SNR and the mean squared error (MSE).

Although TVAL has superior performance in terms of running time, when compared to FISTA and ALIN, it produces an image of lower quality. With the same value of parameter λ, TVAL was not able to obtain the same objective function value as FISTA and ALIN. This makes the SNR of the TVAL-restored image lower and the error higher than those of ALIN and FISTA. ALIN and FISTA have similar performance in terms of image quality, but ALIN is more efficient than FISTA. In Figure 8, we plot the progression in terms of the objective function values for all three methods. TVAL takes only 52 iterations to terminate. In this plot, ALIN and FISTA are set to terminate in 52 iterations.

In the second comparison, we pick 10 random gray scale images with medium size, ranging from $200,000$ to $500,000$ pixels. The experiment is carried out in the same manner as the previous one. The results are reported in Table 6, and we observe the same pattern as in the previous comparison.

Method	CPU time (secs)	SNR	MSE
FISTA	9.19	11.00	4.21
ALIN	6.85	11.03	4.18
TVAL	3.03	10.56	4.57

Table 5: Run time comparison on image deblurring: small size images.

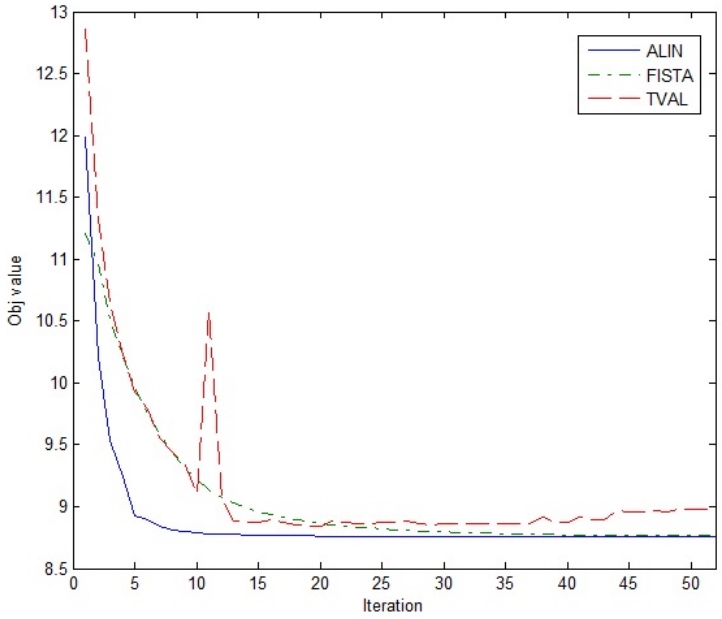


Figure 8: Progression in terms of objective function values of ALIN, FISTA, and TVAL

Method	CPU time (secs)	SNR	MSE
FISTA	65.41	12.14	5.98
ALIN	41.28	12.14	5.97
TVAL	7.18	8.30	14.36

Table 6: Run time comparison on image deblurring: medium size images.

5.5 Application to a Narrative Comprehension Study for Children

With high dimensional fused lasso penalty, the constrained optimization problem with identity design matrix is already difficult to solve, and a large body of literature has been devoted to solving this problem. When the design matrix is not full rank, the problem becomes much more difficult. In this section, we apply the three-dimensional fused lasso penalty to an regression problem where the design matrix X contains many more columns than rows.

Specifically, we perform regularized regression between the measurement of children's language ability (the response y) and voxel level brain activity during a narrative comprehension task (the design matrix X). Children develop a variety of skills and strategies for narrative comprehension during early childhood years (Karunanayaka et al., 2010). This is a complex brain function that involves various cognitive processes in multiple brain regions. We are not attempting to solve the challenging neurological problem of identifying all such brain regions for this cognitive task. Instead, the goal of this study is to demonstrate ALIN's ability for solving constrained optimization problems of this type and magnitude.

The functional MRI data are collected from 313 children with ages 5 to 18 (Schmithorst et al., 2006). The experimental paradigm is a 30-second block design with alternating stimulus and control. Children are listening to a story read by adult female speaker in each stimulus period, and pure tones of 1-second duration in each resting period. The subjects are instructed to answer ten story-related multiple-choice questions upon the completion of the MRI scan (two questions per story). The fMRI data were preprocessed and transformed into the Talairach stereotaxic space by linear affine transformation. A uniform mask is applied to all the subjects so that they have measurements on the same set of voxels.

The response variable y is the oral and written language scale (OWLS). The matrix X records the activity level for all the 8000 voxels measured. The objective function is the following:

$$\mathcal{L}(\beta) = \tfrac{1}{2}\|y - X\beta\|_2^2 + \lambda_1 h_1(\beta) + \lambda_2 h_2(\beta),$$

where

$$\begin{aligned}
h_1(\beta) = &\sum_{i=1}^{m-1}\sum_{j=1}^{n-1}\sum_{k=1}^{p-1}\{|\beta_{i,j,k} - \beta_{i+1,j,k}| + |\beta_{i,j,k} - \beta_{i,j+1,k}| + |\beta_{i,j,k} - \beta_{i,j,k+1}|\} \\
&+ \sum_{i=1}^{m-1}\sum_{k=1}^{p-1}\{|\beta_{i,n,k} - \beta_{i+1,n,k}| + |\beta_{i,n,k} - \beta_{i,n,k+1}|\} + \sum_{j=1}^{n-1}\{|\beta_{m,j,p} - \beta_{m,j+1,p}|\} \\
&+ \sum_{j=1}^{n-1}\sum_{k=1}^{p-1}\{|\beta_{m,j,k} - \beta_{n,j+1,k}| + |\beta_{m,j,k} - \beta_{m,j,k+1}|\} + \sum_{i=1}^{m-1}\{|\beta_{i,n,p} - \beta_{i+1,n,p}|\} \\
&+ \sum_{i=1}^{m-1}\sum_{j=1}^{n-1}\{|\beta_{i,j,p} - \beta_{i+1,j,p}| + |\beta_{i,j,p} - \beta_{i,j+1,p}|\} + \sum_{k=1}^{p-1}\{|\beta_{m,n,k} - \beta_{m,n,k+1}|\}, \\
h_2(\beta) = &\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{p}|\beta_{i,j,k}|,
\end{aligned}$$

and $m = 31$, $n = 35$, and $p = 15$.

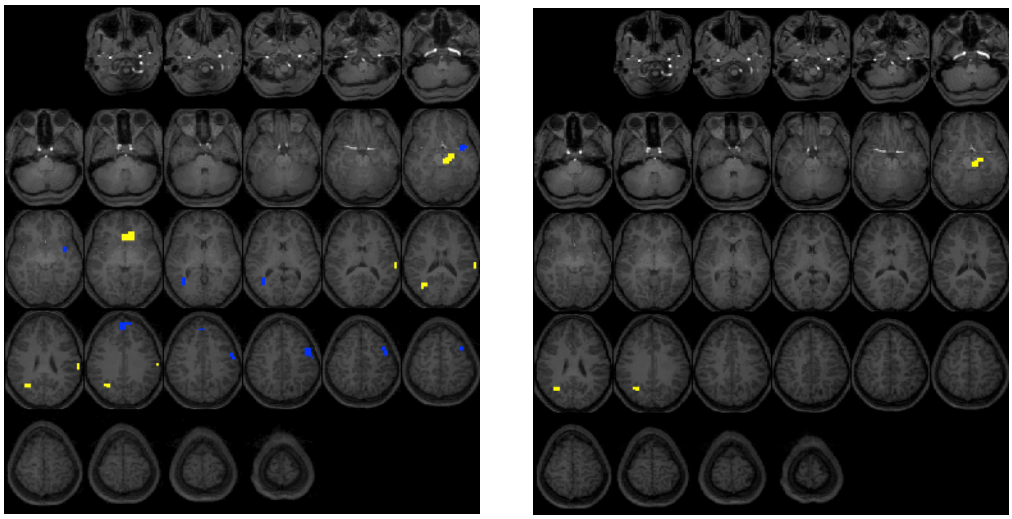


Figure 9: Results of regularization regression with combined lasso and 3-d fused lasso penalty. The tuning parameters of fused lasso is 0.2 for both figures. The tuning parameter for lasso is 0.2 for the left and 0.6 for the right.

While the main purpose of this study is to demonstrate the capability of the ALIN algorithm for solving penalized regression problems with 3-d fused lasso, there are also some interesting neurological observations. One objective of this study is to identify the voxels that are significant for explaining the performance score y. These voxels constitute active brain regions that are closely related to the OWLS. Figure 9 presents the results of fitted coefficients using combined lasso and fused lasso penalty. The highlighted regions shown in the maps are areas with more than 10 voxels (representing clusters of size 10 and above). The left plot in the figure is the optimal solution obtained using ten-fold cross validation. The optimal tuning parameters are 0.2 for both fused lasso and lasso penalties. Roughly speaking, five brain regions have been identified. The yellow area to the rightmost side of the brain is situated in the wernicke area, which is one of the two parts of the cerebral cortex linked to speech. It is involved in the understanding of written and spoken language. The only difference between the left and right plots is the value of the tuning parameter for the lasso penalty, which is 0.2 and 0.6 respectively. Clearly, the right plot shrinks more coefficients to zero, which results in a reduced number of significant regions, as compared to the left plot.

We further study this regularization problem using only lasso penalty and Tiknonov type penalty similar to (23). Figure 10 shows the fitted maps. The left plot is the case where only lasso penalty is applied. Comparing this with Figure 9, we see that the 3-d fused lasso penalty imposes smoothing constraints on the neighboring coefficients, thus allowing to identify larger areas significant for the response variable y. The simple lasso penalty imposes shrinkage on the coefficients individually, resulting in rather disjoint significant voxels. Such scatterness is much less informative for neurologists than larger areas identified

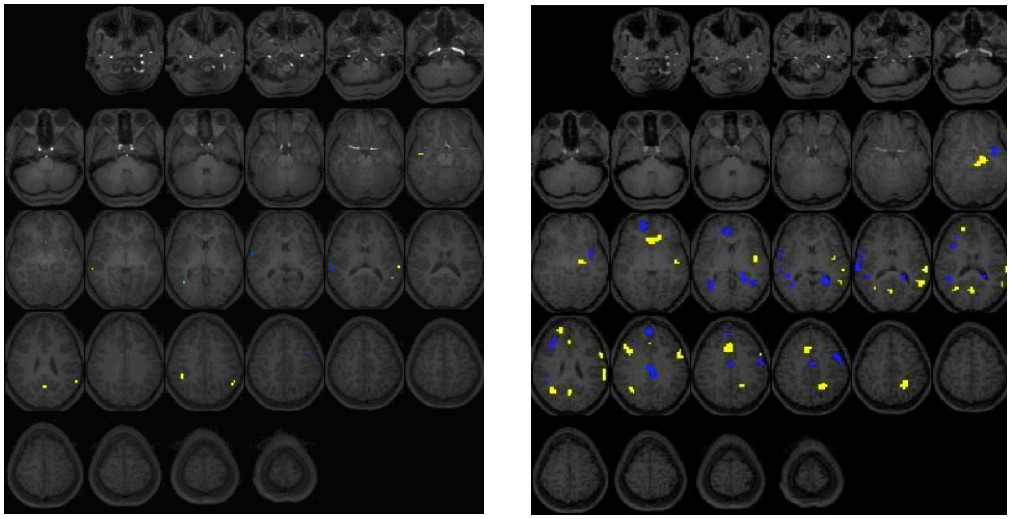


Figure 10: Results of regularization regression with lasso penalty (left plot) and Tiknonov type penalty (right plot).

by the three-dimensional fused lasso penalty. Meanwhile, the Tiknonov type regularization generates too many significant regions as shown in the right plot. This is partially due to the over smoothing of the image as discussed in the previous section.

For comparison, we have considered a couple of methods designed to solve this particular problem. *Genlasso* is the path algorithm designed for the Generalized Lasso in the original paper. However, it was unable to handle an instance of this magnitude.

Another method is the Augmented Lagrangian and Alternating Direction method. The problem of interest

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|R\beta\|_1, \quad \lambda > 0$$

can be reformulated as

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|z\|_1 \quad s.t : R\beta - z = 0.$$

The Augmented Lagrangian for this problem has the form:

$$\mathcal{L}(\beta, z, u) = \frac{1}{2}\|y - A\beta\|_2^2 + \lambda\|z\|_1 + \mu^T(R\beta - z) + \frac{\rho}{2}\|R\beta - z\|_2^2,$$

where μ is the vector of Lagrange multipliers, and $\rho > 0$ is the penalty coefficient. This formulation can be solved by the Alternating Direction method, in which alternating minimization with respect to β and z are followed by an update of μ. We implemented this method in MATLAB. The update step for β requires minimizing a quadratic function. The iterative method of choice to solve the sub-problems is the conjugate gradient method built in MATLAB. The performance of this method strongly depends on the choice of the penalty

parameter ρ. As suggested by Wahlberg et al. (2012), we choose $\rho = \lambda$ to keep the algorithm stable for the implementation. It is known that the method has a nice decrease in the function values but slow tail convergence and an iteration of ADMM for this particular problem is rather expensive. Because of these reasons, we let the method run for 30 iterations and then let ALIN run until it reached the same objective function value. The running times for different values of λ are reported in Table 7.

Method	$\lambda = 0.001$	$\lambda = 0.01$	$\lambda = 0.05$	$\lambda = 0.1$	$\lambda = 1$
ALIN	20.68	12.19	18.58	23.45	129.81
ADMM	72.86	94.69	68.13	74.43	267.01

Table 7: Run time (in seconds) comparison on 3D fused lasso.

We also implemented FISTA for this particular problem. For each iteration k of FISTA, the following optimization problem is solved:

$$\min_{\beta} \mathcal{Q}_L(\beta, \beta^k) = f(\beta^k) + \langle \nabla f(\beta^k), \beta - \beta^k \rangle + \frac{L}{2}\|\beta - \beta^k\|^2 + g(\beta),$$

where f is Gaussian loss function and $g(\beta) = \|R\beta\|_1$. The parameter L is the Lipschitz constant of ∇f. This problem is similar to the f-subproblem of ALIN, so we utilize our own block-coordinate descent method to solve this problem. Since the Lipschitz constant L cannot be computed efficiently, we use back-tracking to find the proper L. Back-tracking is a popular method to find the right step size for FISTA iterations, however it can slow down the algorithm significantly. We observe that in each iteration, back-tracking will have to solve the sub-problem 20 to 30 times to find a good approximation to the Lipschitz constant of ∇f. Normally, this quantity is approximated by the maximum eigenvalue of $X^T X$. However, in the $p \gg n$ setting, it is not computationally efficient to estimate eigenvalues. When FISTA is applied to this data set with 3-d fused lasso penalty, it needs around $10,000$ iterations to reach the same objective function value as ADMM, and the running time is about one hour. This is also in line with what we observe from the implementation of FISTA for 1-d fused lasso penalty in the package SPAMS (Jenatton et al., 2010).

6. Conclusion

The alternating linearization method is a specialized nonsmooth optimization method for solving structured nonsmooth optimization problems. It combines the ideas of bundle methods and operator splitting methods, to define a descent algorithm in terms of the values of the function that is minimized. We have adapted the alternating linearization method to structured regularization problems by introducing the idea of diagonal quadratic approximation and developing specialized methods for solving subproblems. As a result, a new general method for a variety of regularization problems has been obtained, which has the following theoretical features:

- It deals with nonsmoothness directly, not via approximations,
- It is monotonic with respect to the values of the function that is minimized,
- Its convergence is guaranteed theoretically.

Our numerical experiments on a variety of structured regularization problems illustrate the applicability of the alternating linearization method and indicate its practically important virtues: speed, scalability, and accuracy. While other specialized methods compare to ALIN in some cases, none of them exhibits equal speed and accuracy for the broad range of problems discussed in the paper.

Its efficacy and accuracy follow from the use of the diagonal quadratic approximation and from a special test, which chooses in an implicit way the best operator splitting step to be performed. The current approximate solution is updated only if it leads to a significant decrease of the value of the objective function.

Its scalability is due to the use of highly specialized algorithms for solving its two subproblems. The algorithms do not require any explicit matrix formation or inversion, but only matrix–vector multiplications, and can be efficiently implemented with sparse data structures.

Our study of image denoising and deblurring in Section 5.4, as well as the narrative comprehension for children in Section 5.5 are illustrations of broad applicability of the alternating linearization method.

Acknowledgements

The authors are grateful to the Associate Editor and four anonymous Referees, whose insightful comments helped to make significant improvements in the results reported in this manuscript. The work of the third author was partially supported by the National Science Foundation (award CMMI-0965689).

References

H. H. Bauschke and P. L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. CMS Books in Mathematics/Ouvrages de Mathématiques de la SMC. Springer, New York, 2011.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

E. G. Birgin and J. M. Martínez. Large-scale active-set box-constrained optimization method with spectral projected gradients. *Comput. Optim. Appl.*, 23(1):101–125, 2002.

J.-F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. Sagastizábal. *Numerical Optimization. Theoretical and Practical Aspects*. Springer-Verlag, Berlin, 2003.

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):1–122, 2010.

X. Chen, S. Kim, Q. Lin, J. Carbonell, and E. Xing. Graph-structured multi-task regression and an efficient optimization method for general fused lasso. Technical report 1005.3579v1, arXiv, 2010.

X. Chen, Q. Lin, S. Kim, J. Carbonell, and E. Xing. Smoothing proximal gradient method for general structured sparse regression. *Ann. Appl. Stat.*, 6(2):719–752, 2012.

P. L. Combettes. Iterative construction of the resolvent of a sum of maximal monotone operators. *J. Convex Anal.*, 16(3-4):727–748, 2009.

P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, Springer Optimization and Its Applications, pages 185–212. Springer, 2011.

I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. Pure Appl. Math*, 57(11):1413–1457, 2004.

E. Van den Berg and M. P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM J. on Scientific Computing.*, 2008.

J. Douglas and H. H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Trans. Amer. Math. Soc.*, 82:421–439, 1956.

J. Eckstein. Augmented Lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results. *RUTCOR Research Report, Rutgers University*, RRR 32-2012, 2012.

J. Eckstein and D. P. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Math. Programming*, 55(3, Ser. A):293–318, 1992.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–451, 2004.

J.M. Fadili and G. Peyré. Total variation projection with first order schemes. *IEEE Transactions on Image Processing*, 20(3):657 –669, 2011.

A. Friedlander and J. M. Martínez. On the maximization of a concave quadratic function with box constraints. *SIAM J. Optim.*, 4(1):177–192, 1994.

J. Friedman, T. Hastie, H. Hoefling, and R. Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1(2):302–332, 2007.

W. Fu. Penalized regressions: the bridge vs. the lasso. *Journal of Computational and Graphical Statistics*, 7(3):397–416, 1998.

D. Goldfarb, S. Ma, and K. Scheinberg. Fast alternating linearization methods for minimizing the sum of two convex functions. *Mathematical Programming*, (141):349–382, 2013.

T. Goldstein and S. Osher. The split Bregman method for $L1$-regularized problems. *SIAM J. Imaging Sci.*, 2(2):323–343, 2009.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

L. Grosenick, B. Klingenberg, B. Knutson, and J. Taylor. A family of interpretable multivariate models for regression and classification of whole-brain fmri data. *arXiv/1110.4139*, 2011.

L. Grosenick, B. Klingenberg, K. Katovich, B. Knutson, and J. Taylor. Interpretable whole-brain prediction analysis with graphnet. *Neuroimage*, 2013.

B. He and X. Yuan. On the $O(1/t)$ convergence rate of alternating direction method. Technical report, 2011.

J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms. II*, volume 306 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1993.

H. Hoefling. A path algorithm for the fused lasso signal approximator. *Journal of Computational and Graphical Statistics*, 19(4), 2010.

R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for sparse hierarchical dictionary learning. *International Conference on Machine Learning*, 2010.

P. Karunanayaka, V. J. Schmithorst, J. Vannest, J. P. Szaflarski, E. Plante, and S. K. Holland. A group independent component analysis of covert verb generation in children: A functional magnetic resonance imaging study. *Neuroimage*, 51:472–487, 2010.

K. Kiwiel, C. Rosa, and A. Ruszczyński. Proximal decomposition via alternating linearization. *SIAM Journal on Optimization*, 9:153–172, 1999.

K. C. Kiwiel. *Methods of Descent for Nondifferentiable Optimization*, volume 1133 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1985.

C. Li, W. Yin, H. Jiang, and Y. Zhang. An efficient augmented Lagrangian method with applications to total variation minimization. *Computational Optimization and Applications*, 56(3):507–530, 2013.

P.-L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM J. Numer. Anal.*, 16(6):964–979, 1979.

J. Liu, L. Yuan, and J. Ye. An efficient algorithm for a class of fused lasso problems. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2010.

J. Liu, L. Yuan, and J. Ye. SLEP: Sparse learning with efficient projections. Technical report, Computer Science Center, Arizona State University, 2011.

V. Michel, A. Gramfort, G. Varoquaux, E. Eger, and B. Thirion. Total variation regularization for fMRI-based prediction of behavior. *IEEE Transactions on Medical Imaging*, 30(7):1328–1340, 2011.

Yu. Nesterov. Gradient methods for minimizing composite objective function. Discussion paper 2007/76, ECORE, 2007.

D. W. Peaceman and H. H. Rachford. The numerical solution of parabolic and elliptic differential equations. *J. Soc. Indust. Appl. Math.*, 3:28–41, 1955.

Z. Qin and D. Goldfarb. Structured sparsity via alternating direction methods. *Journal of Machine Learning Research*, pages 1435–1468, 2012.

L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena.*, 1992.

A. Ruszczyński. On convergence of an augmented Lagrangian decomposition method for sparse convex optimization. *Math. Oper. Res.*, 20(3):634–656, 1995.

A. Ruszczyński. *Nonlinear Optimization*. Princeton University Press, Princeton, NJ, 2006.

V. Schmithorst, S. Holland, and E. Plante. Cognitive modules utilized for narrative comprehension in children: a functional magnetic resonance imaging study. *Neuroimage*, 29: 254–266, 2006.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of The Royal Statistical Society Series B*, 58(1):267–288, 1996.

R. Tibshirani and J. Taylor. The solution path of the generalized lasso. *Annals of Statistics*, 39(3), 2011.

R. Tibshirani and P. Wang. Spatial smoothing and hot spot detection for CGH data using the fused lasso. *Biostatistics*, 9(1):18–29, 2008.

R. Tibshirani, M. Saunders, J. Zhu, and S. Rosset. Sparsity and smoothness via the fused lasso. *Journal of The Royal Statistical Society Series B*, 67:91–108, 2005.

P. Tseng. Convergence of block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109:474–494, 2001.

B. Wahlberg, S. Boyd, M. Annergren, and Y. Wang. An ADMM algorithm for a class of total variation regularized estimation problems. *arXiv:1203.1828*, 2012.

Y. Wang, J. Yang, W. Yin, and Y. Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM Journal on Imaging Sciences*, 1(3):248–272, 2008.

S. Wright, R. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7), 2009.

L. Xiao and T. Zhang. A proximal-gradient homotopy method for the sparse least-squared problem. *SIAM Journal on Optimization.*, 23(2):1062–1091, 2013.

G.-B. Ye and X. Xie. Split Bregman method for large scale fused Lasso. *Comput. Statist. Data Anal.*, 55(4):1552–1569, 2011.

Z. Zhang, K. Lange, and C. Sabatti. Reconstructing DNA copy number by joint segmentation of multiple sequences. *BMC Bioinformatics*, 13(205), 2012.

Journal of Machine Learning Research 15 (2014) 3483-3487 Submitted 2/14; Revised 5/14; Published 10/14

The Gesture Recognition Toolkit

Nicholas Gillian NGILLIAN@MEDIA.MIT.EDU
Joseph A. Paradiso JOEP@MEDIA.MIT.EDU
Responsive Environments Group, Media Lab
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Editor: Isabelle Guyon, Vassilis Athitsos, Sergio Escalera

Abstract

The Gesture Recognition Toolkit is a cross-platform open-source C++ library designed to make real-time machine learning and gesture recognition more accessible for non-specialists. Emphasis is placed on ease of use, with a consistent, minimalist design that promotes accessibility while supporting flexibility and customization for advanced users. The toolkit features a broad range of classification and regression algorithms and has extensive support for building real-time systems. This includes algorithms for signal processing, feature extraction and automatic gesture spotting.

Keywords: gesture recognition, machine learning, C++, open source, classification, regression, clustering, gesture spotting, feature extraction, signal processing

1. Introduction

Gesture recognition is a powerful tool for human-computer interaction. It is increasingly redefining how we interact with our smartphones, wearable devices, televisions and gaming consoles. In addition to the increasing prevalence of gesture-based interactions in consumer devices, a diverse range of individuals are gaining access to affordable sensor technology and rapid-prototyping tools that facilitate non-specialists to build custom gesture-based applications. Commercial sensors such as the Microsoft Kinect or easy-to-use hardware platforms like Arduino (Mellis et al., 2007), combined with prototyping environments, such as Processing or Openframeworks,[1] are empowering professional developers, students, researchers, hobbyists, creative coders, interaction designers, musicians and artists to create novel-interactive systems that are playful, poignant and expressive.

Nevertheless, while a diverse range of individuals now have access to powerful sensors and rapid-prototyping tools, performing *real-time* gesture recognition can pose a challenge, even to accomplished developers and engineers (Patel et al., 2010). This is despite the large number of sophisticated machine-learning applications currently available, such as MATLAB, R and WEKA (Hall et al., 2009). Many of these applications are primarily designed for offline analysis of prerecorded data sets by domain experts, and require substantial effort to recognize real-time signals. There are accessible machine-learning libraries in Java (Abeel et al., 2009) and Python (Pedregosa et al., 2011) that can be used to prototype real-time

1. More information on Processing and Openframeworks can be found on their respective websites: `http://processing.org` and `http://www.openframeworks.cc`.

systems. However, many users need to build their systems in C++ due to the computational overhead of the sensor data and interactive visualizations and therefore benefit from C++ tools for real-time machine learning. While there are a number of powerful C++ libraries that can be adapted for gesture recognition (King, 2009; Sonnenburg et al., 2010; Gashler, 2011), these tools still require the user to develop the supporting infrastructure needed to build real-time systems and can have steep learning curves for non-specialists. This leaves C++ users with a sizable gulf of execution, specifically the gap between their goals and the actions needed to attain those goals with the system (Hutchins et al., 1985). This gap can significantly impede the process of building novel gesture-based interfaces for technologists, researchers, artists and beyond.

2. Gesture Recognition Toolkit

To address this issue, we have created the Gesture Recognition Toolkit (GRT), a cross platform open source C++ machine-learning library for real-time gesture recognition. The toolkit was developed with the following core design principles:

Accessibility: The GRT is a general-purpose tool for facilitating non-specialists to create their own machine-learning based systems. Emphasis is placed on ease of use, with a clear and consistent coding convention applied throughout the toolkit. The GRT provides a minimal code footprint for the user, reducing the need for arduous and error-prone boilerplate code to perform common functionality, such as passing data between algorithms or to preprocess data sets. This consistent, minimalist design significantly lowers the entry barrier for a new user because the same subset of core functions apply throughout the toolkit.

Flexibility: To support flexibility while maintaining consistency, the GRT uses an object-oriented modular architecture. This architecture is built around a set of core **modules** and a central **gesture-recognition pipeline**. The input to both the modules and pipeline consists of an N-dimensional double-precision vector, making the toolkit flexible to the type of input signal. The algorithms in each module can be used as stand-alone classes; alternatively a gesture-recognition pipeline can be used to chain modules together to create a more sophisticated gesture-recognition system. The GRT includes modules for preprocessing, feature extraction, clustering, classification, regression and post processing.

Choice: To date, there is no single machine-learning algorithm that can be used to recognize all gestures. It is therefore crucial for a user to be able to choose from, and quickly experiment with, a number of algorithms to see which might work best for their particular task. The GRT features a broad range of machine-learning algorithms such as AdaBoost, Decision Trees, Dynamic Time Warping, Hidden Markov Models, K-Nearest Neighbor, Linear and Logistic Regression, Naïve Bayes, Multilayer Perceptrons, Random Forests, Support Vector Machines and more.[2] In addition to supporting a broad range of algorithms, the toolkit's architecture facilities a user to seamlessly switch between different algorithms with minimal modifications to the user's code.

Supporting Infrastructure: Building sophisticated machine-learning based systems requires more than just a state-of-the-art classifier. In many real-world scenarios, the input

2. For Support Vector Machines, we provide an easy-to-use wrapper for LibSVM (Chang and Lin, 2011). All other algorithms are custom implementations unless otherwise stated in the source documentation.

to a classification algorithm must first be preprocessed and have salient features extracted. Preprocessing and feature extraction are important because they can significantly improve the predictive performance of a classifier, and also provide faster and more cost-effective predictors (Guyon and Elisseeff, 2003). The GRT therefore supports a wide range of pre/post processing, feature extraction and feature selection algorithms, including popular preprocessing filters (e.g., Moving Average Filter), embedded feature extraction algorithms (e.g., AdaBoost), dimensionality reduction techniques (e.g., Principal Component Analysis), and unsupervised quantizers (e.g., K-Means Quantizer, Self-Organizing Map Quantizer). Accurate labeling of data sets is also critical for building robust machine-learning based systems. The toolkit therefore contains extensive support for recording, labeling and managing supervised and unsupervised data sets for classification, regression and time series analysis.[3]

Customizability: In addition to using the wide range of existing GRT algorithms, more advanced users commonly want to test or deploy their own algorithms when building novel recognition systems, such as using a custom feature-extraction algorithm. The GRT is therefore designed to facilitate users to easily incorporate their own algorithms within the toolkit's framework by inheriting from one of the GRT base classes. The toolkit leverages advanced object-orientated concepts, such as polymorphism and abstract base-class pointers, facilitating custom algorithms to be used alongside any of the existing GRT algorithms.

Real-time Support: The GRT supports common techniques for performing offline analysis on pre-recorded data sets, such as partitioning data into validation and test data sets, running cross validation and computing accuracy metrics. In addition to these offline techniques, the toolkit is designed to enable a user to seamlessly move from the offline analysis phase to the real-time recognition phase. One significant challenge involved in moving from offline analysis to real-time gesture recognition is automatically segmenting valid gestures from a continuous stream of data (Junker et al., 2008). This is a nontrivial task because the input data might consist of generic movements that are not valid gestures in the model. To support real-time gesture recognition, the GRT features algorithms that automatically perform gesture spotting. These algorithms, such as the Adaptive Naïve Bayes Classifier (Gillian et al., 2011a) and N-Dimensional Dynamic Time Warping (Gillian et al., 2011b), learn rejection thresholds from the training data, which are then used to automatically recognize valid gestures from a continuous stream of real-time data.

3. Code Example

The code example below demonstrates the core design principles of the GRT. This example shows how to setup a custom gesture-recognition system consisting of a moving-average filter preprocessing module, a fast Fourier transform and custom feature extraction modules, an AdaBoost classifier and a timeout-filter post processing module. The example also illustrates how to load a training data set from a CSV file, train a classification model, and use this model to predict the class label of a new data sample.

```
   //Setup a custom recognition pipeline
1: GestureRecognitionPipeline pipeline;
2: pipeline.addPreProcessingModule( MovingAverageFilter( 5 ) );
```

3. A detailed description of the GRT data structures can be found at `http://www.nickgillian.com/wiki/pmwiki.php/GRT/Reference`.

```
3: pipeline.addFeatureExtractionModule( FFT( 512 ) );
4: pipeline.addFeatureExtractionModule( MyCustomFeatureAlgorithm() );
5: pipeline.setClassifier( Adaboost( DecisionStump() ) );
6: pipeline.addPostProcessingModule( ClassLabelTimeoutFilter( 1000 ) );

   //Load a labeled data set from a CSV file and train a classification model
7: ClassificationData trainingData;
8: trainingData.load( "TrainingData.csv" );
9: bool success = pipeline.train( trainingData );

    //The following lines would be called each time the user gets a new sample
10: vector< double > sample = getDataFromSenor(); //Custom user function
11: pipeline.predict( sample );
12: unsigned int predictedClassLabel = pipeline.getPredictedClassLabel();
13: double maxLikelihood = pipeline.getMaximumLikelihood();
```

Lines 1 through 6 show how a `GestureRecognitionPipeline` can be used to link several modules together to build a more complex recognition system. Note that the customization of the recognition system is achieved with a minimal code footprint, as the pipeline will automatically connect the output of one module to the next module's input; propagating signals through the entire pipeline at both the training, testing and real-time prediction phases. These six lines also illustrate the flexibility of the toolkit's modular design, and demonstrate how a user can easily experiment with different algorithms from existing modules, or insert a custom algorithm into the pipeline as illustrated on line 4. Line 10 demonstrates how real-time sensor data from a variety of devices can be incorporated; input can consist of something as simple as the three-dimensional data from an accelerometer connected to an Arduino, to more complex inputs, such as the high-dimensional skeleton data from a Kinect.

This example also demonstrates one of the key designs of the GRT that make it more accessible: clean and consistent coding through abstraction. For instance, lines 9 and 11 show respectively how a user can train a model and then predict the class label of a new sample using that model. These key functions are the same, regardless of which algorithms are used. This abstraction significantly reduces the learning curve for new users, because the same key functions are consistent across all the GRT algorithms.

4. Conclusion

The gesture recognition toolkit is open source under the MIT license and has been publicly available since 2012, receiving over ten-thousand hits on the toolkit's website.[4] It has been downloaded several thousand times and has built up a community of over 300 users on the toolkit's forum. To support a diverse range of users, we have established a number of online resources, including detailed examples for each module and a wide range of tutorials and references.[5] Future work includes an interactive graphical user interface, in which a user can record and label training data; configure; train and test a gesture-recognition model; perform real-time prediction and then export their model and pipeline configuration so it can be loaded directly into the user's program, using the C++ API.

4. The toolkit's website can be found at: `http://www.nickgillian.com/software/grt`.
5. Online tutorials, references and examples can be found at: `http://www.nickgillian.com/wiki`.

References

T. Abeel, Y. Van de Peer, and Y. Saeys. Java-ML: A machine learning library. *Journal of Machine Learning Research*, 10:931–934, 2009.

C.C. Chang and C.J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

M. Gashler. Waffles: A machine learning toolkit. *Journal of Machine Learning Research*, 12:2383–2387, 2011.

N. Gillian, R. B. Knapp, and S. O'Modhrain. An adaptive classification algorithm for semiotic musical gestures. In *Proceedings of the 8th Sound and Music Computing Conference*, 2011a.

N. Gillian, R. B. Knapp, and S. O'Modhrain. Recognition of multivariate temporal musical gestures using n-dimensional dynamic time warping. In *Proceedings of the 2011 International Conference on New Interfaces for Musical Expression*, 2011b.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

E. L. Hutchins, J. D. Hollan, and D. A. Norman. Direct manipulation interfaces. *Human–Computer Interaction*, 1(4):311–338, 1985.

H. Junker, O. Amft, P. Lukowicz, and G. Tröster. Gesture spotting with body-worn inertial sensors to detect user activities. *Pattern Recognition*, 41(6):2010–2024, 2008.

D.E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

D. Mellis, M. Banzi, D. Cuartielles, and T. Igoe. Arduino: An open electronics prototyping platform. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2007.

K. Patel, N. Bancroft, S. M. Drucker, J. Fogarty, A. J. Ko, and J. Landay. Gestalt: integrated support for implementation and analysis in machine learning. In *Proceedings of the 23rd Annual ACM symposium on User Interface Software and Technology*, 2010.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

S. Sonnenburg, G. Rätsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. Bona, A. Binder, C. Gehl, and V. Franc. The SHOGUN machine learning toolbox. *Journal of Machine Learning Research*, 11:1799–1802, 2010.

Journal of Machine Learning Research 15 (2014) 3489-3511 Submitted 6/13; Revised 5/14; Published 10/14

Convolutional Nets and Watershed Cuts for Real-Time Semantic Labeling of RGBD Videos

Camille Couprie CAMILLE.COUPRIE@IFPEN.FR
IFP Energies Nouvelles
Technology, Computer Science and Applied Mathematics Division
Rueil Malmaison, France

Clément Farabet CFARABET@TWITTER.COM
Twitter, Inc.
1355 Market St, Suite 900
San Francisco, CA 94103, USA

Laurent Najman L.NAJMAN@ESIEE.FR
Université Paris-Est
Laboratoire d'Informatique Gaspard-Monge
Équipe A3SI - ESIEE Paris, France

Yann LeCun YANN@CS.NYU.EDU
New York University & Facebook AI Research
Courant Institute of Mathematical Sciences
New York, NY 10003, USA

Editors: Aaron Courville, Rob Fergus, and Christopher Manning

Abstract

This work addresses multi-class segmentation of indoor scenes with RGB-D inputs. While this area of research has gained much attention recently, most works still rely on hand-crafted features. In contrast, we apply a multiscale convolutional network to learn features directly from the images and the depth information. Using a frame by frame labeling, we obtain nearly state-of-the-art performance on the NYU-v2 depth data set with an accuracy of 64.5%. We then show that the labeling can be further improved by exploiting the temporal consistency in the video sequence of the scene. To that goal, we present a method producing temporally consistent superpixels from a streaming video. Among the different methods producing superpixel segmentations of an image, the graph-based approach of Felzenszwalb and Huttenlocher is broadly employed. One of its interesting properties is that the regions are computed in a greedy manner in quasi-linear time by using a minimum spanning tree. In a framework exploiting minimum spanning trees all along, we propose an efficient video segmentation approach that computes temporally consistent pixels in a causal manner, filling the need for causal and real-time applications. We illustrate the labeling of indoor scenes in video sequences that could be processed in real-time using appropriate hardware such as an FPGA.

Keywords: deep learning, optimization, convolutional networks, superpixels, depth information

1. Introduction

The recent release of the Kinect allowed for progress in indoor computer vision. Most approaches have focused on object recognition (Janoch et al., 2011) or point cloud semantic labeling (Koppula et al., 2009), finding their applications in robotics or games (Cruz et al., 2012). The pioneering work of Silberman and Fergus (2011) was the first to deal with the task of semantic full image labeling using depth information. This task implies the joint detection, recognition, and delineation at once (See Figure 1), closer to what our vision system perceives. Semantic segmentation results are still possible to improve, specifically in indoor environments where many objects may be similar, while belonging to different categories. Learning to segment full images in contrast to object recognition systems results in learning a context for the objects, such as co-occurrence relationships, that may help getting good results. Using RGB information, the deep learning approach of Farabet et al. (2013) achieves state-of-the-art semantic labeling performances while being an order of magnitude faster than competing approaches. In contrast with previous works employing hand-crafted features, the feature learning and the semantic predictions are performed using a unique model: a multiscale convolutional network, described in Section 2. The goal of this paper is to improve the result of this feature learning approach by using depth and temporal information, that is nowadays easily available in indoor environments.

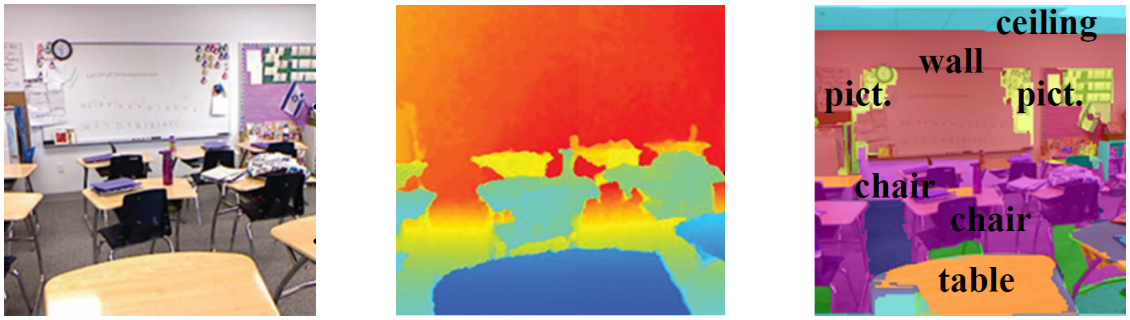


Figure 1: From color and depth images, our goal is to predict semantic labels for the entire image, in other words, a semantic delineation of its components.

The first contribution of our work consists in adapting Farabet et al.'s network to learn more effective features for indoor scene labeling. Our work is, to the best of our knowledge, the first exploitation of depth information in a feature learning approach for full scene labeling (Couprie et al., 2013b).

Currently, training data sets containing fully labeled videos do not exist. Therefore, as the learning takes place on 2D images, the results show flickering objects between subsequent frames. In order to avoid this effect, in addition to the use of depth information, we enforce spatial consistency of the final labeling using superpixels. Whereas oversegmentation methods producing such superpixels are a rather well studied problem, video segmentation into consistent spatio-temporal segments is largely unsolved. While there have been attempts at video segmentation, most methods are non causal and non real-time. The second contribution of this paper is to propose a fast method for real-time video segmentation,

including semantic segmentation as an application (Couprie et al., 2013a). We employ to that goal minimum spanning trees that are suitable to compute superpixels in quasi-linear time, therefore well adapted to real time applications.

The paper contains two main parts, the first dealing with feature learning using depth information, and the second dealing with the temporal smoothing approach. It is organized as follows. After presenting the related work on each parts, we introduce the algorithms employed in Section 2 for the two different aspects separately. We finally present in Section 3 our results using depth on images, independently from the temporal aspects, then introduce our temporal smoothing results independently from depth information, and combine the two aspects in the last results.

1.1 Related Work

Before detailing previous works on temporal smoothing methods, we introduce the existing background around depth-based segmentation approaches.

1.1.1 Learning To Segment Using Depth Information

The first results of Silberman and Fergus (2011) on the NYU depth data set employ sift features on the depth maps in addition to the RGB images. The depth is then used in the gradient information to refine the predictions using graph cuts. Alternative MRF-like approaches have also been explored to improve the computation time performance such as Couprie (2012). The results on NYU data set v1 have been improved by Ren et al. (2012) using elaborate kernel descriptors and a post-processing step that employs gPb superpixels MRFs, involving large computation times.

A second version of the NYU depth data set was released by Silberman et al. (2012), and improves the labels categorization into 894 different object classes. Furthermore, the size of the data set did also increase, it now contains hundreds of video sequences (407024 frames) acquired with depth maps. Recently, shortly after the submission of this paper, several groups designed features adapted to the NYU depth v2 data set, and present their results on the 4-class grouping of Silberman et al. (2012). Müller and Behnke (2014) use random forests averaging predictions within superpixels, but also explicitly learning interactions between neighboring superpixels using CRFs. They obtain state-of-the-art performance on NYU depth v2. The work of Cadena and Košecka (2013) also uses handcrafted features and a CRF framework. Despite the better performance of these methods in comparison to our results, their computation times remain higher. Furthermore, the complexity of their inference constrains their methods to a low number of different classes to identify. The work of (Stückler et al., 2013), based on random decision forests and implemented on GPUs, is achieving fine performance in terms of speed and accuracy because it combines, like our work, depth and temporal information.

The present work aims at not only combining all available information (depth, spatial, and temporal consistency) to answer the challenging problem of indoor scene segmentation, but also avoiding the expense of designing data specific features.

Feature learning, or deep learning approaches are particularly adapted to the addition of new image modalities such as depth information. Its recent success for dealing with various types of data is manifest in speech recognition (Jaitly et al., 2012), molecular activity

prediction, object recognition (Hinton et al., 2012), mitosis detection (Ciresan et al., 2013) and many more applications. In computer vision, the approach of Farabet et al. (2012, 2013); Farabet (2014) has been specifically designed for full scene labeling and has proven its efficiency for outdoor scenes. The key idea is to learn hierarchical features by the mean of a multiscale convolutional network. Training networks using multiscale representations appeared also the same year in Ciresan et al. (2012); Schulz and Behnke (2012).

When the depth information was not yet available, there have been attempts to use stereo image pairs to improve the feature learning of convolutional networks as in LeCun et al. (2004). Now that depth maps are easy to acquire, deep learning approaches started to be considered for improving object recognition such as in Socher et al. (2012). Similarly, the work of Lenz et al. (2013) experiments group regularization to differentiate between the color and depth channels, treated as different modalities.

1.1.2 Temporal Smoothing

A large number of approaches in computer vision makes use of super-pixels at some point in the process, for example, semantic segmentation (Farabet et al., 2013), geometric context identification (Hoiem et al., 2005), extraction of support relations between object in scenes (Silberman et al., 2012), etc. Among the most popular approaches for super-pixel segmentation, two types of methods are distinguishable. Regular shape super-pixels may be produced using normalized cuts or graph cuts, see the works of Shi and Malik (1997); Veksler et al. (2010) for instance. More object—or part of object—shaped super-pixels can be generated from watershed based approaches. In particular, the method of Felzenszwalb and Huttenlocher (2004) produces such results.

It is a real challenge to obtain a decent delineation of objects from a single image. When it comes to real-time data analysis, the problem is even more difficult. However, additional cues may be used to constrain the solution to be temporally consistent, thus helping to achieve better results. Since many of the underlying algorithms are in general super-linear, there is often a need to reduce the dimensionality of the video. To this end, developing low level vision methods for video segmentation is necessary. Currently, most video processing approaches are non-causal, that is to say, they make use of future frames to segment a given frame, sometimes requiring the entire video as in (Grundmann et al., 2010). This prevents their use for real-time applications.

Some approaches have been designed to address the causal video segmentation problem such as (Paris, 2008; Miksik et al., 2013). Paris (2008) employs the mean shift algorithm of Comaniciu and Meer (2002). As this method works in a feature space, it does not necessary clusters spatially consistent super-pixels. A more recent approach, specifically applied for semantic segmentation, is the one of Miksik et al. (2013). The authors employ an optical flow method to enforce the temporal consistency of the semantic segmentation. Our approach is different because it aims to produce general purpose super-pixels, and possibly uses the produced super-pixels for smoothing semantic segmentation results. Furthermore, we do not use any optical flow pre-computation that would prevent us having real-time performance on a CPU.

Some works (Glasner et al., 2011; Lee et al., 2005; Joulin et al., 2012; Gomila and Meyer, 2003; Xu et al., 2012) use the idea of enforcing some consistency between different segmentations. Glasner et al. (2011) formulate a co-clustering problem as a Quadratic Semi-Assignment Problem. However solving the problem for a pair of images takes about a minute. Alternatively, Lee et al. (2005) and Gomila and Meyer (2003) identify the corresponding regions using graph matching techniques. Xu et al. (2012) propose like us to exploit Felzenszwalb et al. superpixels in causal video processing. The complexity of this approach is super-linear because of a hierarchical segmentation, preventing the current implementation from real-time applications.

Our strategy to produce temporal superpixels is to perform independent segmentations and match the produced super-pixels to define markers. The markers are then used to produce the final segmentation by minimizing a global criterion defined on the image. We show how Minimum Spanning Trees can be used at every step of the process, leading to gains in speed, and real-time performance on a single core CPU.

2. Full Scene Labeling

We introduce in this section the employed feature learning approach from RGBD images, as well as our temporal smoothing methodology to improve the segmentation results on video.

2.1 Multi-Scale Feature Extraction

Good internal representations are compact, and fast to process. Multi-scale approaches such as hierarchies meet these requirements. In vision, pixels are assembled into edglets, edglets into motifs, motifs into parts, parts into objects, and objects into scenes. This suggests that recognition architectures for vision (and for other modalities such as audio and natural language) should have multiple trainable stages stacked on top of each other, one for each level in the feature hierarchy.

Convolutional Networks (ConvNets) provide a simple framework to learn such hierarchies of features. Introduced by Fukushima (1980), they became usable thanks to LeCun et al. (1998) who proposed the first back-propagation algorithm. Riesenhuber and Poggio (1999) replaced Fukushima's competition layers by max-pooling layers. Since the first GPU implementation of Ciresan et al. (2011), ConvNets have been used with GPUs in virtually all non-recurrent, feedforward, competition-winning or benchmark record-setting deep learners (for object detection, image segmentation, traffic signs, ImageNet, Pascal).

Convolutional Networks are trainable architectures composed of multiple stages. The input and output of each stage are sets of arrays called feature maps. In our case, the input is a color (RGB) image plus a depth (D) image and each feature map is a 2D array containing a color or depth channel of the input RGBD image. At the output, each feature map represents a particular feature extracted at all locations on the input. Each stage is composed of three layers: a filter bank layer, a non-linearity layer, and a feature pooling layer. A typical ConvNet is composed of one, two or three such 3-layer stages, followed by a classification module. Because they are trainable, arbitrary input modalities can be modeled, such as the depth modality that is added to the input channel in this work.

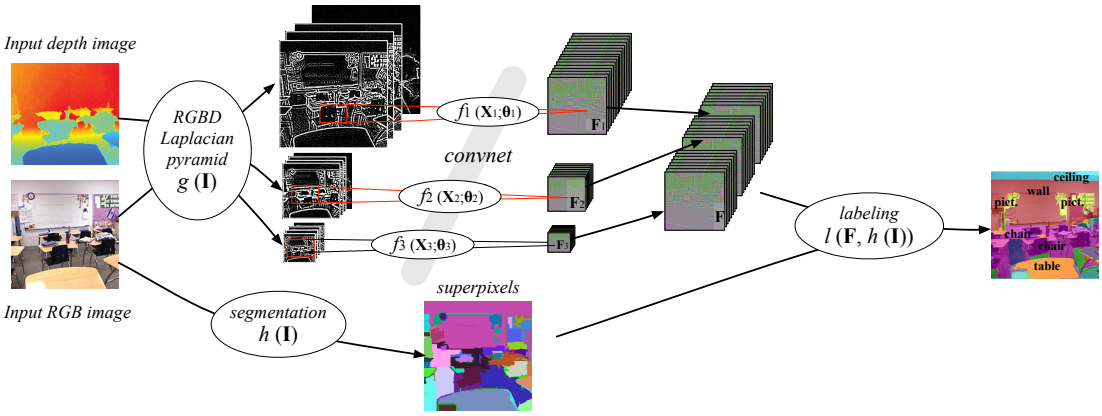


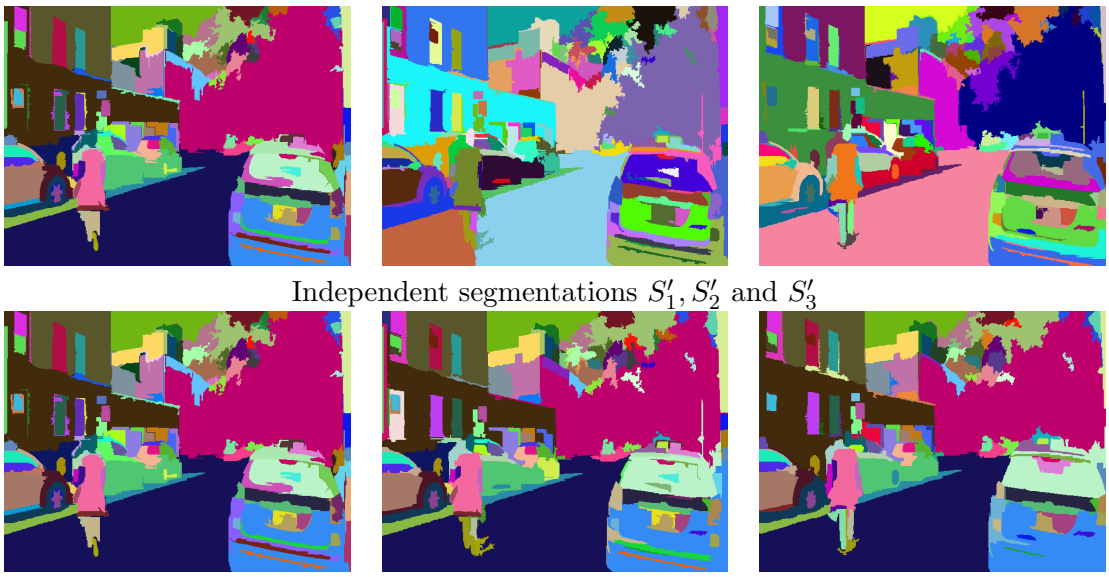
Figure 2: Scene parsing (frame by frame) using a multiscale network and superpixels. The RGB channels of the image and the depth image are transformed through a Laplacian pyramid. Each scale is fed to a 3-stage convolutional network, which produces a set of feature maps. The feature maps of all scales are concatenated, the coarser-scale maps being upsampled to match the size of the finest-scale map. Each feature vector thus represents a large contextual window around each pixel. In parallel, a single segmentation of the image into superpixels is computed to exploit the natural contours of the image. The final labeling is obtained by the aggregation of the classifier predictions into the superpixels.

A great gain has been achieved with the introduction of the *multiscale* convolutional network described in Farabet et al. (2013). The multi-scale, dense feature extractor produces a series of feature vectors for regions of multiple sizes centered around every pixel in the image, covering a large context. The multi-scale convolutional net contains multiple copies of a single network that are applied to different scales of a Laplacian pyramid version of the RGBD input image.

The RGBD image is first pre-processed, so that local neighborhoods have zero mean and unit standard deviation. The depth image, given in meters, is treated as an additional channel similarly to any color channel. The overview scheme of our model appears in Figure 2.

Beside the input image which is now including a depth channel, the parameters of the multi-scale network (number of scales, sizes of feature maps, pooling type, etc.) are identical to Farabet et al. (2013). The feature maps sizes are 16, 64, 256, multiplied by the three scales. The size of convolutions kernels are set to 7 by 7 at each layer, and sizes of subsampling kernels (max pooling) are 2 by 2. In our tests we rescaled the images to the size 240×320.

As in Farabet et al. (2013), the feature extractor followed by a classifier was trained to minimize the negative log-likelihood loss function. The classifier that follows feature extraction is a 2-layer multi-perceptron, with a hidden layer of size 1024. Once a labeling is obtained, we use the superpixels of Felzenszwalb and Huttenlocher (2004) to smooth the



Independent segmentations S'_1, S'_2 and S'_3

Temporally consistent segmentations $S_1 (= S'_1), S_2$, and S_3

Figure 3: Segmentation results on three consecutive frames of the NYU-Scene data set.

ConvNet predictions as a post-processing step, by aggregating the classifiers predictions in each superpixel.

2.2 Movie Processing

While the training is performed on single images, we are able to perform scene labeling of video sequences. In order to improve the performance of our frame-by-frame predictions, a temporal smoothing may be applied. Instead of using the frame by frame superpixels as in the previous section, we compute temporally consistent superpixels. Our approach works in quasi-linear time and reduces the flickering of objects that may appear in the video sequences.

The temporal smoothing method presented in this section is introduced for RGB images and may trivially be extended to use depth information.

Given a segmentation S_t of an image at time t, we wish to compute a segmentation S_{t+1} of the image at time $t+1$ which is consistent with the segments of the result at time t. For that purpose, we first compute a superpixel segmentation of the image at time $t+1$ using only this frame, and denote the resulting independent segmentation S'_{t+1}, as detailed in Section 2.2.1.

2.2.1 Independent Image Segmentation

The super-pixels produced by Felzenszwalb and Huttenlocher (2004) have been shown to satisfy the global properties of being not too coarse and not too fine according to a particular region comparison function. In order to generate temporal superpixels close to the ones produced by Felzenszwalb and Huttenlocher (2004), we first generate independent segmentations of the 2D images using Felzenszwalb et al.'s algorithm. We name these seg-

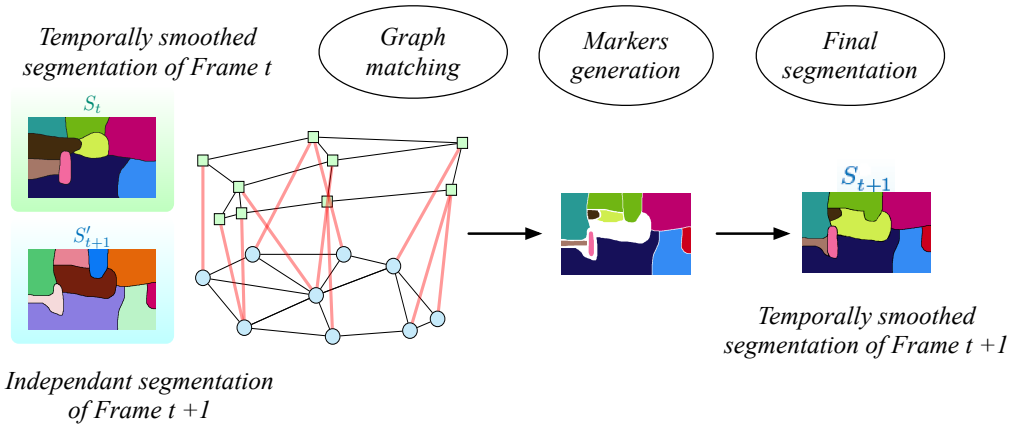


Figure 4: Illustration of the temporal segmentation procedure

mentations $S'_1, ..., S'_T$, where T denotes the index of the last frame of the video sequence. The principle of segmentation is fairly simple. We define a graph G_t, where the nodes correspond to the image pixels, and the edges link neighboring nodes in 8-connectivity. The edge weights ω_{ij} between nodes i and j are given by a color gradient of the image, or a color and depth gradient if depth is available.

A Minimum Spanning Tree (MST) is build on G_t, and regions are merged according to a criterion taking into account the regions sizes and a scale parameter k.

Once an image is independently segmented, resulting in S'_{t+1}, we then face the question of the propagation of the temporal consistency given the non overlapping contours of S_t and S'_{t+1}. Our solution is the development of a cheap graph matching technique to obtain correspondences between segments from S_t and these of S'_{t+1}. This first step is described in Section 2.2.2. We then mine these correspondences to create markers (also called seeds) to compute the final labeling S_{t+1} by solving a global optimization problem. This second step is detailed in Section 2.2.3.

2.2.2 Graph Matching Procedure

The basic idea is to use the segmentation S_t and segmentation S'_{t+1} to produce markers before a final segmentation of image at time $t+1$. Therefore, in the process of computing a new segmentation S_{t+1}, a graph G is defined. The vertices of G comprise two sets of vertices: V_t that corresponds to the set of regions of S_t and V'_{t+1} that corresponds to the set of regions of S'_{t+1}. Edges link regions characterized by a small distance between their centroids. The edges weights between vertex $i \in V_t$ and $j \in V'_{t+1}$ are given by a similarity measure taking into account distance and differences between shape and appearance

$$w_{ij} = \frac{(|r_i| + |r_j|)d(c_i, c_j)}{|r_i \cap r_j|} + a_{ij}, \tag{1}$$

where $|r_i|$ denotes the number of pixels of region r_i, $|r_i \cap r_j|$ the number of pixels present in r_i and r_j with aligned centroids, and a_{ij} the appearance difference of regions r_i and r_j.

In our experiments a_{ij} was defined as the difference between mean color intensities of the regions. It may also include depth information if available.

The graph matching procedure is illustrated in Figure 4 and produces the following result: For each region of S'_{t+1}, its best corresponding region in image S_t is identified. More specifically, each node i of V_t is associated with the node j of V'_{t+1} which minimizes w_{ij}. Symmetrically, for each region of S_t, its best corresponding region in image S'_{t+1} is identified, that is to say each node i of V'_{t+1} is associated with the node j of V_t which minimizes w_{ij}. This step may also be viewed as the construction of two minimum spanning trees, one spanning V_t, and the other V_{t+1}.

2.2.3 Final Segmentation Procedure

The final segmentation S_{t+1} is computed using a minimum spanning forest procedure. This seeded segmentation algorithm that produces watershed cuts as shown by Cousty et al. (2009) is strongly linked to global energy optimization methods such as graph-cuts as detailed by (Allène et al., 2010; Couprie et al., 2011) and Section 2.3. In addition to theoretical guarantees of optimality, this choice of algorithm is motivated by the opportunity to reuse the sorting of edges that is performed in Section 2.2.1 and constitutes the main computational effort. Consequently, we reuse here the graph $G_{t+1}(V, E)$ built for the production of independent segmentation S'_{t+1}.

Algorithm 1: Minimum Spanning Forest Algorithm

Data: A weighted graph $G(V, E)$ and a set of labeled nodes makers L. Nodes of $V \setminus L$ have unknown labels initially.
Result: A labeling x associating a label to each vertex.
Sort the edges of E by increasing order of weight.
while *any node has an unknown label* **do**
  Find the edge e_{ij} in E of minimal weight;
  if v_i *or* v_j *have unknown label values* **then**
    Merge v_i and v_j into a single node, such that when the value for this merged node becomes known, all merged nodes are assigned the same value of x and considered known.

The minimum spanning forest algorithm is recalled in Algorithm 1. The first appearance of such forest in image processing dates from 1986 with the work of Morris et al. (1986). It was later introduced in a morphological context by Meyer (1994).

The seeds, or markers, are defined using the regions correspondences computed in the previous section, according to the procedure detailed below. For each segment s' of S'_{t+1} four cases may appear:

1. s' has one and only one matching region s in S_t: propagate the label l_s of region s. All nodes of s' are labeled with the label l_s of region s.

2. s' has several corresponding regions $s_1, ..., s_r$: propagate seeds from S_t. The coordinates of regions $s_1, ..., s_r$ are centered on region s'. The labels of regions $s_1, ..., s_r$ whose coordinates are in the range of s' are propagated to the nodes of s'.

3. s' has no matching region : The region is labeled by the label l'_s itself.

4. If none of the previous cases is fulfilled, it means that s' is part of a larger region s in S_t. If the size of s' is small, a new label is created. Otherwise, the label l_s is propagated in s' as in case 1.

Before applying the minimum spanning forest algorithm, a safety test is performed to check that the map of produced markers does not differ two much from the segmentation S'_{t+1}. If the test shows large differences, an eroded map of S'_{t+1} is used to correct the markers. The eroded regions in the conflict areas are added as markers.

2.3 Global Optimization Guarantees

The minimum spanning forest computation offers by definition a global optimality guarantee, the sum of the edges weights of the forest being maximal. In addition, less commonly known, there is a probabilistic interpretation (as continuous Markov random fields) of the labeling of a particular case of such forests.

Several graph-based segmentation problems, including minimum spanning forests, graph cuts, random walks and shortest paths have recently been shown to belong to a common energy minimization framework, see the work of (Allène et al., 2010; Sinop and Grady, 2007; Couprie et al., 2011). The considered problem is to find a labeling $x^* \in \mathbb{R}^{|V|}$ defined on the nodes of a graph that minimizes

$$E(x) = \sum_{e_{ij} \in E} w_{ij}^p |x_j - x_i|^q + \sum_{v_i \in V} w_i^p |l_i - x_i|^q, \tag{2}$$

where l represents a given configuration and x represents the target configuration. The result of $\lim_{p \to \infty} \arg\min_x E(x)$ for values of $q \geq 1$ always produces a cut by maximum (equivalently minimum) spanning forest. The reciprocal is also true if the weights of the graph are all different.

In the case of our application, the pairwise weights w_{ij} is given by an inverse function of the original weights ω_{ij}. The pairwise term thus penalizes any unwanted high-frequency content in x and essentially forces x to vary smoothly within an object, while allowing large changes across the object boundaries. The second term enforces fidelity of x to a specified configuration l, w_i being the unary weights enforcing that fidelity.

The enforcement of markers l_s—defined as described in Section 2.2.3 according to four cases—as hard constrained may be viewed as follows: A node of label l_s is added to the graph, and linked to all nodes i of V that are supposed to be marked. The unary weights ω_{i,l_s} are set to arbitrary large values in order to impose the markers.

2.3.1 Applications To Optical Flow And Semantic Segmentation

An optical flow map may be easily estimated from two successive segmentations S_t and S_{t+1}. For each region r of S_{t+1}, if the label of r comes from a label present in a region s of the segmentation S_t, the optical flow in r is computed as the distance between the centroid of r and the centroid of s. The optical flow map may be used as a sanity check for

region tracking applications. In principle, a video sequence will not contain displacements of objects greater than a certain value.

For each superpixel s of S_{t+1}, if the label of region s comes from the previous segmentation S_t, then the semantic prediction from S_t is propagated to S_{t+1}. Otherwise, in case the label of s is a new label, the semantic prediction is computed using the prediction at time $t+1$. As some errors may appear in the regions tracking, labels of regions having inconsistent large values in optical flow maps are not propagated. For the specific task of semantic segmentation, results can be improved by exploiting the contours of the recognized objects. Semantic contours such as for example transition between a building and a tree for instance, might not be present in the gradient of the raw image. Thus, in addition to the pairwise weights ω described in Section 2.2.1, we add a constant in the presence of a semantic contour.

3. Results

We used for our experiments the NYU depth data set—version 2—of Silberman et al. (2012), composed of 407024 couples of RGB images and depth images. Among these images, 1449 frames have been labeled. The object labels cover 894 categories. The data set is provided with the original raw depth data that contain missing values, with code from Dani et al. (2004) to inpaint the depth images.

3.1 Validation Of Our RGBD Network On Images

The training has been performed using the 894 categories directly as output classes. The frequencies of object appearances have not been changed in the training process. However, we established 14 clusters of classes categories to evaluate our results more easily. The distributions of number of pixels per class categories are given in Table 1. We used the train/test splits as provided by the NYU depth v2 data set, that is to say 795 training images and 654 test images. Please note that no jitter (rotation, translations or any other transformation) was added to the data set to gain extra performances. However, this strategy could be employed in future work. The code consists of Lua scripts using the Torch machine learning software of Collobert et al. (2011) available online at `http://www.torch.ch/`.

To evaluate the influence of the addition of depth information, we trained a multiscale ConvNet only on the RGB channels, and another network using the additional depth information. Both networks were trained until the achievement of their best performances, that is to say for 105 epochs and 98 epochs respectively, taking approximately 2 days on a regular server (1.8 GHz Intel Xeon) using 1 core. For superpixel smoothing, following the implementation of Felzenszwalb and Huttenlocher (2004), we pre-process the images using a Gaussian filtering step with a kernel of variance σ. A post-processing step that removes regions of small size, that is to say below a threshold δ is also performed. As Felzenszwalb and Huttenlocher (2004), we denote the scale of observation parameter by k. To produce our frame by frame results on NYU depth v2 , we used as in the work of (Farabet et al., 2013) $k = 450, \delta = 30$ and $\sigma = 0.5$. We report in Table 1 two different performance measures:

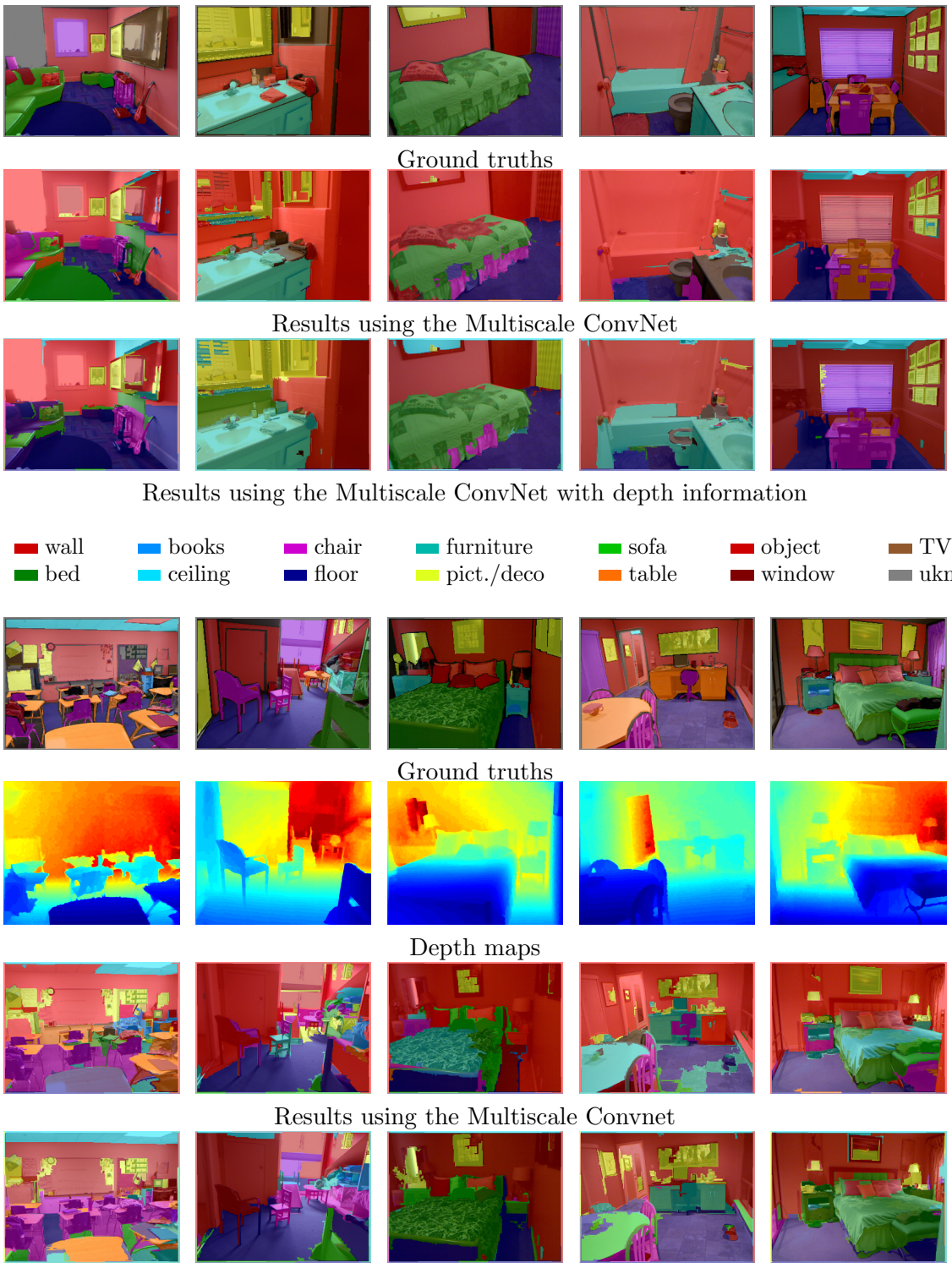


Figure 5: Some scene labelings using our Multiscale Convolutional Network trained on RGB and RGBD images. We observe in Table 1 that adding depth information helps to recognize objects that have low intra-class variance of depth appearance.

- the "classwise accuracy", counting the number of correctly classified pixels divided by the number of false positive, averaged for each class. This number corresponds to the mean of the confusion matrix diagonal.

- the "pixelwise accuracy", counting the number of correctly classified pixels divided by the total number of pixels of the test data.

We tested to scale the depth image with a log scale, however it did not improved the results.

We observe that considerable gains (15% or more) are achieved for the classes 'floor', 'ceiling', and 'furniture'. This result is logical since these classes are characterized by a globally constant appearance of their depth map. Objects such as TV, table, books can either be located in the foreground as well as in the background of images. On the contrary, the floor and ceiling will almost always lead to a depth gradient always oriented in the same direction: Since the data set has been collected by a person holding a Kinect device at a his chest, floors and ceiling are located at a distance that does not vary to much through the data set. Figure 5 provides examples of depth maps that illustrate these observations. Large drops of performances when using depth information are also observed for some classes (chair, table, window, books and TV). We explain this result partly by the very weak frequency of appearance of such classes in the data set that is not sufficient to cover the variety of possible depth appearance for these objects. One should also note that for specific applications, object frequencies may be weighted in the training step as in Farabet et al.'s work. Overall, improvements induced by the depth information exploitation are present and statistically significant, as checked using paired t-tests. In the next section, these improvements are more apparent.

3.2 Comparison to the State-Of-The-Art

In order to compare our results to the state-of-the-art on the NYU depth v2 data set, we adopted a different selection of outputs instead of the 14 classes employed in the previous section. The work of Silberman et al. (2012) defines the four semantic classes Ground, Furniture, Props and Structure. This class selection is adopted by Silberman et al. (2012) to use semantic labelings of scenes to infer support relations between objects. We recall that the recognition of the semantic categories is performed by (Silberman et al., 2012) using the definition of diverse features including SIFT features, histograms of surface normals, 2D and 3D bounding box dimensions, color histograms, and relative depth.

As reported in Table 2, the results achieved using the Multiscale convnet are improving the structure class predictions, resulting in an overall 4% relative gain in pixelwise accuracy over Silberman *et al.*'s approach. Adding the depth information results in a considerable improvement of the ground prediction, and better performance over the other classes. This approach achieves a 4% relative gain in classwise accuracy over previous works and improves by almost 6% the pixelwise accuracy compared to Silberman et al.'s results.

We note that the class 'furniture' in the 4-classes evaluation is different than the 'furniture' class of the 14-classes evaluation. The furniture-4 class encompasses chairs and beds but not desks, and cabinets for example, explaining a drop of performance here using the depth information.

	Class Occurrences	Multiscale Convnet Acc. Farabet et al. (2013)	MultiScale Convnet +depth Acc.
bed	4.4%	30.3	**38.1**
objects	7.1 %	**10.9**	8.7
chair	3.4%	**44.4**	34.1
furnit.	12.3%	28.5	**42.4**
ceiling	1.4%	33.2	**62.6**
floor	9.9%	68.0	**87.3**
deco.	3.4%	38.5	40.4
sofa	3.2%	25.8	24.6
table	3.7%	**18.0**	10.2
wall	24.5%	**89.4**	86.1
window	5.1%	**37.8**	15.9
books	2.9%	**31.7**	13.7
TV	1.0%	**18.8**	6.0
unkn.	17.8%	-	-
Avg. Class Accuracy	-	35.8	**36.2**
Pixel Acc. (mean)	-	51.0	**52.4**
Pixel Acc. (median)	-	51.7	**52.9**
Pixel Acc. (std. dev.)	-	15.2	15.2

Table 1: Class occurrences in the test set—Performance per class and per pixel. Results in bold indicate statistically significant better performances (above probability of 95% being statistically different)

	Ground	Furniture	Props	Structure	Class Acc.	Pixel Acc.	Comput. time (s)
Silberman et al. (2012)	68	**70**	**42**	59	59.6	58.6	>3
Cadena and Košecka (2013)	**87.9**	**64.1**	31.0	77.8	**65.2**	**66.9**	1.7
Multiscale convnet	61.7	49.4	30.8	86.1	57.0	61.3	**0.6**
Multiscale convnet+ superpixels	68.1	51.1	29.9	**87.8**	59.2	63.0	**0.7**
Multiscl.+depth convnet+superpixels	**87.3**	45.3	**35.5**	**86.1**	**63.5**	**64.5**	**0.7**

Table 2: Accuracy of the multiscale convnet compared to the state-of-the-art. The reported computation times of Cadena and Košecka (2013) were obtained on an architecture similar to ours. Bold numbers indicate the best or second best performance.

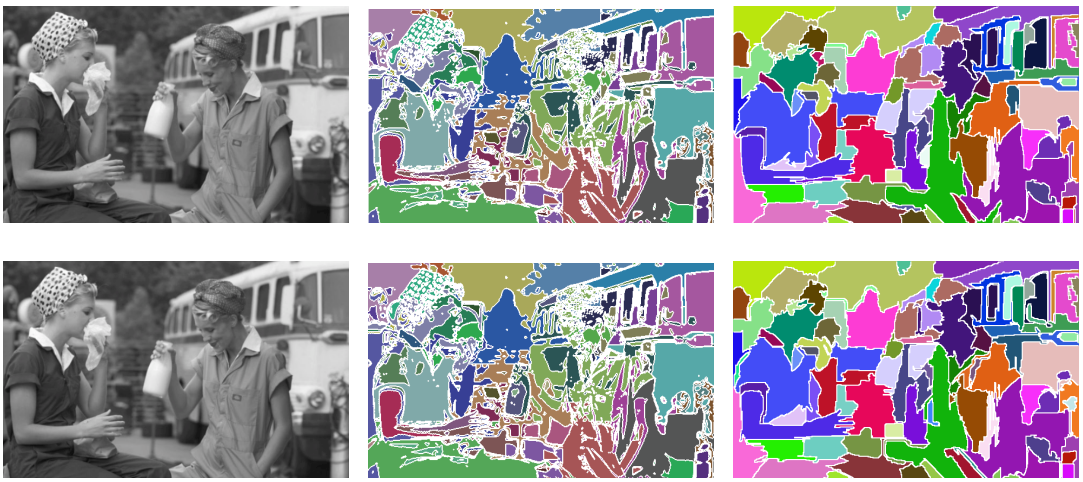
Very recently, the approach of Cadena and Košecka (2013) achieved about two percent improvement over our results. Contrary to our system, they use handcrafted features, resulting therefore in a less versatile method.

A great advantage of our approach is its real-time capabilities. Processing a 320x240 frame takes 0.7 seconds on a laptop using 4 cores, see the work of Farabet et al. (2013) for

details. The temporal smoothing only requires an additional 0.1s per frame, as we develop in the next section.

3.3 Super-Pixel Segmentation On Videos Using RGB Information

Next experiments demonstrate the efficiency and versatility of our temporal smoothing approach by performing simple super-pixel segmentation and semantic scene labeling.



Original frames Mean shift results of Paris (2008) Our results

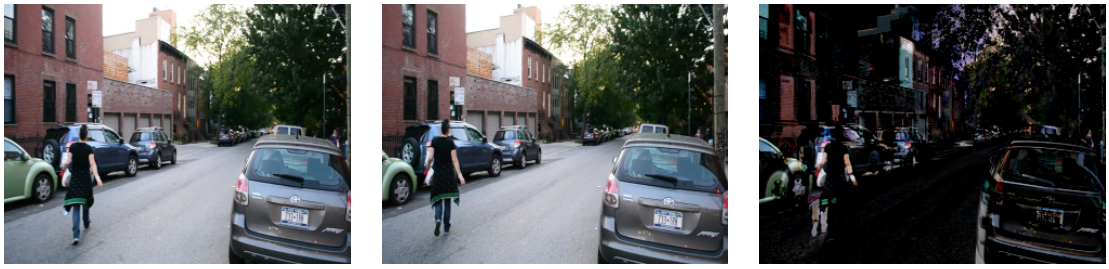
Figure 6: Comparison of our temporal superpixels with the mean-shift segmentation method of Paris (2008) on Frame 19 and 20. $k = 200, \delta = 400, \sigma = 0.5$.

Experiments are performed on two different types of videos: videos where the camera is static, and videos where the camera is moving. The robustness of our approach to large variations in the region sizes and large movements of camera is illustrated in Figure 7, where the camera is moving.

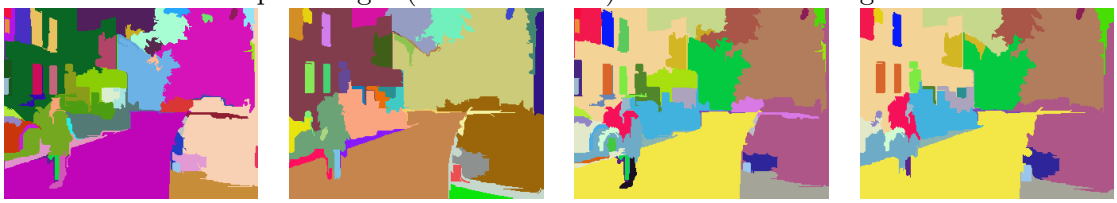
A comparison with the temporal mean shift segmentation of Paris (2008) is displayed in Figure 6. The super-pixels produced by Paris (2008) are not spatially consistent as the segmentation is performed in the feature (color) space in their case. Our approach is slower, although qualified for real-time application, but computes only spatially consistent super-pixels.

3.4 Semantic Video Labeling

We present our semantic video labeling results in indoor scenes using RGBD images, and outdoor environments, where the depth information is not available.



Input images (frames 4 and 5) and difference image



Independent segmentations S'_4, S'_5 (left) and temporally consistent segmentations S_4, S_5 (right)

Figure 7: Segmentation results on 2 consecutive frames of the NYU-Scene data set.

(a) Independent segmentations with no temporal smoothing

(b) Result using the temporal smoothing method of Miksik et al. (2013)

(c) Our temporally consistent segmentation

Figure 8: Comparison with the temporal smoothing method of Miksik et al. (2013). Parameters used: $k = 1200, \delta = 100, \sigma = 1.2$.

3.4.1 Outdoor Segmentation Results

The goal of this section is to assess the gain of using only the additional temporal information. Thus, we do not use depth information in our evaluation. We compare our results

(a) Output of the Multiscale convnet trained with RGB information - frame by frame

(b) Output of the Multiscale convnet trained using depth information - frame by frame

(c) Results from (b) smoothed using our temporal superpixels

Figure 9: Some results on video sequences of the NYU v2 depth data set.

with the results of Miksik et al. (2013) on the NYU-Scene data set. The data set consists in a video sequence of 73 frames provided with a dense semantic labeling and ground truths. The provided dense labeling being performed with no temporal exploitation, it suffers from sudden large object appearances and disappearances. As illustrated in Figure 8 our approach reduces this effect, and improves the classification performance of more than 5% as reported in Table 3.

	Multiscale RGBD Convnet Frame by frame	Temporal smoothing of Miksik et al. (2013)	Our temporal smoothing
Accuracy	71.11	75.31	**76.27**
#Frames/sec	1.43	1.33*	**10.5**

Table 3: Overall pixel accuracy (%) for the semantic segmentation task on the NYU Scene video and computation times *Note that the reported timing does not take into account the optical flow computation needed by Miksik et al. (2013).

3.4.2 Indoor Segmentation Results

On the NYU Depth data set by Silberman et al. (2012), we compare independent segmentation performance with our temporally smoothed results on four video sequences of indoor scenes. The videos scenes were chosen randomly among these of the NYU data set containing the largest number of ground truth frames (typically five ground truth frames for two thousands unlabeled frames). Unless specified, the same choice of parameters was performed in all our comparisons.

	Multiscale convnet	Multiscale convnet + depth	Multiscale convnet + depth + temp. smoothing
dining room	54.5	**63.8**	58.5
living room	63.6	65.4	**72.1**
classroom	52	56.5	**58.3**
office	**58.8**	56.3	57.4
mean	57.2	60.5	**61.6**

Table 4: Overall pixel accuracy (%) for the semantic segmentation task on the NYU Depth data set. Parameter used for the temporal smoothing: $\delta = 100, \sigma = 1.2$, $k = 800, 1000, 1000, 920$.

The results obtained in Table 4 show that in most videos scenes, our temporal superpixels allow us to obtain better results. An example of results appears in Figure 9.

3.5 Computation Time of the Temporal Smoothing Approach

The experiments were performed on a laptop with 2.3 GHz Intel core i7-3615QM. Our method is implemented on CPU only, in C/C++, and makes use of only one core of the processor. Super-pixel segmentations take 0.1 seconds per image of size 320×240 and 0.4 seconds per image of size 640×380, thus demonstrating the scalability of the pipeline. All computations are included in the reported timings. The mean segmentation time using the work of Xu et al. (2012) for a frame of size 320×240 is 4 seconds. The timings of the temporal smoothing method of Miksik et al. (2013) are reported in Table 3. We note that the processor used for the reported timings of Miksik et al. (2013) has similar characteristics as ours. Furthermore, Mistik et al. use an optical flow procedure that takes only 0.02 seconds per frame when implemented on GPU, but takes seconds on CPU. Our approach is thus more adapted to real-time applications for instance on embedded devices where a GPU is often not available. The code, as well as some data and videos are available at the website of Couprie (2013).

4. Discussion and Conclusion

This works combines two independent contributions devoted to improve scene labeling results. It introduces the addition of depth information into feature trainable architectures, which brings, for a 4-class segmentation task, a class accuracy improvement of more than 7% relatively to the training using only RGB data. The speed of our system is mainly allowed by the multiscale nature and the sparse connectivity of the employed convolutional network, allowing us to parse an image in 0.7s on CPU.

Our model is easier to implement without the need to design specific features adapted to depth information. In contrast to CRF approaches that do not necessary scale well with a large number of different classes to address, the complexity of our model does not increase with the number of classes. We also introduce a 14-classes clustering for the NYU depth data set that allows us to point out some limitations of our model and the data set. For

instance, poorly represented classes in the data set do not allow our model to sense the variability of the objects with the additional depth dimension.

In terms of evaluation, the 4-class split was designed for a specific application: to perform inference on support relationships between objects, but we deplore that all systems now use it for general purpose semantic labeling with depth. Indeed, it does not give insight about how well the concurrent systems discriminate objects in a world more complex than with four object categories.

With the temporal smoothing strategy, we demonstrate the ability of minimum spanning trees to fulfill accuracy and competitive timing requirements in a global optimization framework. Unlike many video segmentation techniques, our algorithm is causal and does not require any computation of optical flow. Our experiments on challenging videos show that the obtained super-pixels are robust to large camera or objects displacements. Their use in semantic segmentation applications demonstrate that significant gains can be achieved and lead to state-of-the-art results.

Our feature learning and temporal smoothing approaches were developed independently from each other, this presents the advantages of possibly being used in different applications, and also having real time capabilities by decorrelating temporal aspects in the post-treatment. However the best approach for us would be to combine them in one fully trainable system, which will probably be possible in a near future with nowadays hardware computational advances.

Acknowledgments

We would like to thank Nathan Silberman for his useful input for handling the NYU depth v2 data set, and fruitful discussions.

References

Cédric Allène, Jean-Yves Audibert, Michel Couprie, and Renaud Keriven. Some links between extremum spanning forests, watersheds and min-cuts. *Image and Vision Computing*, 28(10):1460–1471, 2010.

César Cadena and Jana Košecka. Semantic parsing for priming object detection in RGB-D scenes. In *3rd Workshop on Semantic Perception, Mapping and Exploration*, 2013.

Dan Claudiu Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Proc. of the 22nd International Joint Conference on Artificial Intelligence*, pages 1237–1242, 2011.

Dan Claudiu Ciresan, Alessandro Giusti, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Conference on Neural Information Processing Systems*, pages 2852–2860, 2012.

Dan Claudiu Ciresan, Alessandro Giusti, Luca Maria Gambardella, and Jürgen Schmidhuber. Mitosis detection in breast cancer histology images using deep neural networks. In *MICCAI*, 2013.

R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A Matlab-like environment for machines learning. In *NIPS Big Learning Workshop, Sierra Nevada, Spain*, 2011.

Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.

Camille Couprie. Multi-label energy minimization for object class segmentation. In *20th European Signal Processing Conference*, Bucharest, Romania, August 2012.

Camille Couprie. Source code for causal graph-based video segmentation, 2013. `www.esiee.fr/~coupriec/code.html`.

Camille Couprie, Leo J. Grady, Laurent Najman, and Hugues Talbot. Power watershed: A unifying graph-based optimization framework. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 33(7):1384–1399, 2011.

Camille Couprie, Clément Farabet, Yann LeCun, and Laurent Najman. Causal graph-based video segmentation. In *Proc. of IEEE International Conference on Image Processing*, 2013a.

Camille Couprie, Clément Farabet, Laurent Najman, and Yann LeCun. Indoor semantic segmentation using depth information. In *International Conference on Learning Representations*, 2013b.

Jean Cousty, Gilles Bertrand, Laurent Najman, and Michel Couprie. Watershed cuts: Minimum spanning forests and the drop of water. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31(8):1362–1374, 2009.

Leandro Cruz, Djalma Lucio, and Luiz Velho. Kinect and RGBD images: Challenges and applications. *SIBGRAPI Tutorial*, 2012.

Anat Levin Dani, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23:689–694, 2004.

Clément Farabet. *Towards Real-Time Image Understanding with Convolutional Networks.* PhD thesis, Université Paris Est, 2014.

Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Scene Parsing with Multiscale Feature Learning, Purity Trees, and Optimal Covers. In *Proc. of International Conference on Machine Learning*, Edinburgh, Scotland, June 2012.

Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, Aug 2013.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:167–181, 2004.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4): 193–202, 1980.

Daniel Glasner, Shiv Naga Prasad Vitaladevuni, and Ronen Basri. Contour-based joint clustering of multiple segmentations. In *Proc. of IEEE Computer Vision and Pattern Recognition*, pages 2385–2392, Washington, DC, USA, 2011.

Cristina Gomila and Fernand Meyer. Graph-based object tracking. In *Proc. of IEEE International Conference on Image Processing*, 2003.

Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient hierarchical graph-based video segmentation. In *Proc. of IEEE Computer Vision and Pattern Recognition*, 2010.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

Derek Hoiem, Alexei A. Efros, and Martial Hebert. Geometric context from a single image. In *Proc. of IEEE International Conference on Computer Vision*, volume 1, pages 654–661, 2005.

Navdeep Jaitly, Patrick Nguyen, Andrew Senior, and Vincent Vanhoucke. Application of pretrained deep neural networks to large vocabulary speech recognition. In *Proc. of Interspeech*, 2012.

Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T. Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A category-level 3-D object dataset: Putting the kinect to work. In *IEEE International Conference on Computer Vision Workshops*, pages 1168–1174, 2011.

Armand Joulin, Francis Bach, and Jean Ponce. Multi-class cosegmentation. In *Proc. of IEEE Computer Vision and Pattern Recognition*, pages 542–549, 2012.

Hema S. Koppula, Abhishek Anand, Thorsten Joachims, and Ashutosh Saxena. Cornell-RGBD-dataset, 2009. `http://pr.cs.cornell.edu/sceneunderstanding/data/data.php`.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278 –2324, nov 1998.

Yann LeCun, Fu-Jie Huang, and Leon Bottou. Learning Methods for generic object recognition with invariance to pose and lighting. In *Proc. of IEEE Computer Vision and Pattern Recognition*, 2004.

J. Lee, JungHwan Oh, and Sae Hwang. Clustering of video objects by graph matching. In *Proc. of IEEE International Conference on Multimedia and Expo*, pages 394–397, 2005.

Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasp, corr abs/1301.3592s. In *Robotics: Science and Systems*, 2013.

Fernand Meyer. Minimum spanning forests for morphological segmentation. In *Proc. of International Symposium on Mathematical Morphology*, pages 77–84, 1994.

Ondrej Miksik, Daniel Munoz, J. Andrew Bagnell, and Martial Hebert. Efficient temporal consistency for streaming video scene analysis. In *Proc. of the IEEE International Conference on Robotics and Automation, Karlsruhe, Germany*, 2013.

O.J. Morris, M. de Jersey Lee, and A.G. Constantinides. Graph theory for image analysis: an approach based on the shortest spanning tree. *Communications, Radar and Signal Processing, IEE Proceedings F*, 133(2):146–152, April 1986.

Andreas C Müller and Sven Behnke. Learning depth-sensitive conditional random fields for semantic segmentation of RGB-D images. In *IEEE International Conference on Robotics and Automation, Hong Kong, May*, 2014.

Sylvain Paris. Edge-preserving smoothing and mean-shift segmentation of video streams. In *Proc. of IEEE European Conference on Computer Vision*, pages 460–473, Marseille, France, 2008.

Xiaofeng Ren, Liefeng Bo, and D. Fox. RGB-(D) scene labeling: Features and algorithms. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2759 –2766, june 2012.

Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025, 1999.

Hannes Schulz and Sven Behnke. Learning object-class segmentation with convolutional neural networks. In *11th European Symposium on Artificial Neural Networks*, 2012.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, volume 22, pages 888–905, 1997.

Nathan Silberman and Rob Fergus. Indoor scene segmentation using a structured light sensor. In *3DRR Workshop, IEEE International Conference on Computer Vision Workshops*, 2011.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *Proc. of IEEE European Conference on Computer Vision*, 2012.

Ali Kemal Sinop and Leo Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *Proc. of International Conference of Computer Vision*, 2007.

Richard Socher, Brody Huval, Bharath Bhat, Christopher D. Manning, and Andrew Y. Ng. Convolutional-Recursive Deep Learning for 3D Object Classification. In *Advances in Neural Information Processing Systems 25*, 2012.

Jörg Stückler, Benedikt Waldvogel, Hannes Schulz, and Sven Behnke. Dense real-time mapping of object-class semantics from RGB-D video. *Journal of Real-Time Image Processing*, pages 1–11, 2013.

Olga Veksler, Yuri Boykov, and Paria Mehrani. Superpixels and supervoxels in an energy optimization framework. In *Proc. of IEEE European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11 (5)*, pages 211–224, 2010.

Chenliang Xu, Caiming Xiong, and Jason J. Corso. Streaming hierarchical video segmentation. In *Proc. of IEEE European Conference on Computer Vision, Florence, Italy, October 7-13*, pages 626–639, 2012.

Journal of Machine Learning Research 15 (2014) 3513-3568 Submitted 10/13; Revised 6/14; Published 11/14

On the Bayes-Optimality of F-Measure Maximizers

Willem Waegeman WILLEM.WAEGEMAN@UGENT.BE
Department of Mathematical Modelling, Statistics and Bioinformatics
Ghent University, Ghent 9000 Belgium

Krzysztof Dembczyński KRZYSZTOF.DEMBCZYNSKI@CS.PUT.POZNAN.PL
Arkadiusz Jachnik ARKADIUSZ.JACHNIK@CS.PUT.POZNAN.PL
Institute of Computing Science
Poznan University of Technology, Poznan 60-695 Poland

Weiwei Cheng ROYWWCHENG@GMAIL.COM
Amazon Development Center Germany, Berlin 10707 Germany

Eyke Hüllermeier EYKE@UPB.DE
Department of Computer Science
University of Paderborn, Paderborn 33098 Germany

Editor: Charles Elkan

Abstract

The F-measure, which has originally been introduced in information retrieval, is nowadays routinely used as a performance metric for problems such as binary classification, multi-label classification, and structured output prediction. Optimizing this measure is a statistically and computationally challenging problem, since no closed-form solution exists. Adopting a decision-theoretic perspective, this article provides a formal and experimental analysis of different approaches for maximizing the F-measure. We start with a Bayes-risk analysis of related loss functions, such as Hamming loss and subset zero-one loss, showing that optimizing such losses as a surrogate of the F-measure leads to a high worst-case regret. Subsequently, we perform a similar type of analysis for F-measure maximizing algorithms, showing that such algorithms are approximate, while relying on additional assumptions regarding the statistical distribution of the binary response variables. Furthermore, we present a new algorithm which is not only computationally efficient but also Bayes-optimal, regardless of the underlying distribution. To this end, the algorithm requires only a quadratic (with respect to the number of binary responses) number of parameters of the joint distribution. We illustrate the practical performance of all analyzed methods by means of experiments with multi-label classification problems.

Keywords: F-measure, Bayes-optimal predictions, regret, statistical decision theory, multi-label classification, structured output prediction

1. Introduction

Being rooted in information retrieval (van Rijsbergen, 1974), the so-called F-measure is nowadays routinely used as a performance metric for different types of prediction problems, including binary classification, multi-label classification (MLC), and certain applications of structured output prediction. Amongst others, examples of such applications include

chunking or named entity recognition in natural language processing (Sang and De Meulder, 2003), image segmentation or edge detection in computer vision (Martin et al., 2004) and detection of geographic coincidence in social networks (Zhuang et al., 2012).

Compared to measures like error rate in binary classification and Hamming loss in multi-label classification, the F-measure enforces a better balance between performance on the minority and the majority class, respectively. Therefore, it is more suitable in the case of imbalanced data, as it does not take the true negative rate into account. Given a prediction $\boldsymbol{h} = (h_1, \ldots, h_m) \in \{0,1\}^m$ of an m-dimensional binary label vector $\boldsymbol{y} = (y_1, \ldots, y_m)$ (e.g., the class labels of a test set of size m in binary classification or the label vector associated with a single instance in MLC or the binary vector indicating named entities in a text document in a structured output prediction task), the F-measure is defined as follows:

$$F(\boldsymbol{y}, \boldsymbol{h}) = \frac{2\sum_{i=1}^m y_i h_i}{\sum_{i=1}^m y_i + \sum_{i=1}^m h_i} \in [0,1] \ , \tag{1}$$

where $0/0 = 1$ by definition. This measure essentially corresponds to the harmonic mean of precision *prec* and recall *recl*:

$$prec(\boldsymbol{y}, \boldsymbol{h}) = \frac{\sum_{i=1}^m y_i h_i}{\sum_{i=1}^m h_i}, \quad recl(\boldsymbol{y}, \boldsymbol{h}) = \frac{\sum_{i=1}^m y_i h_i}{\sum_{i=1}^m y_i} \ .$$

One can generalize the F-measure to a weighted harmonic average of these two values, but for the sake of simplicity, we stick to the unweighted mean, which is often referred to as the F1-score or the F1-measure.

Despite its popularity in experimental settings, very few theoretical studies of the F-measure can be found. This paper intends to bridge this gap by analyzing existing methods and, moreover, by presenting a new algorithm that exhibits the desirable property of statistical consistency. To this end, we will adopt a decision-theoretic viewpoint. Modeling the ground-truth as a random variable $\boldsymbol{Y} = (Y_1, Y_2, \ldots, Y_m)$, i.e., assuming an underlying probability distribution P over $\{0,1\}^m$, the prediction $\boldsymbol{h}$ that maximizes the expected F-measure is given by

$$\boldsymbol{h}_F = \operatorname*{arg\,max}_{\boldsymbol{h} \in \{0,1\}^m} \mathbb{E}\left[F(\boldsymbol{Y}, \boldsymbol{h})\right] = \operatorname*{arg\,max}_{\boldsymbol{h} \in \{0,1\}^m} \sum_{\boldsymbol{y} \in \{0,1\}^m} P(\boldsymbol{y})\, F(\boldsymbol{y}, \boldsymbol{h}). \tag{2}$$

The corresponding optimization problem is non-trivial and cannot be solved in closed form. Moreover, a brute-force search is infeasible, as it would require checking all 2^m combinations of prediction vector $\boldsymbol{h}$ and summing over an exponential number of terms in each combination. As a result, many researchers who report the F-measure in experimental studies rely on optimizing a surrogate loss as an approximation of (2). For problems such as multi-label classification and structured output prediction, the Hamming loss and the subset zero-one loss are immediate candidates for such surrogates. However, as will be shown in Section 3, these surrogates do not yield a statistically consistent model and, more importantly, they manifest a high regret. As an intermezzo, we present results for the Jaccard index, which has recently gained an increased popularity in areas such as multi-label classification. This measure is closely related to the F-measure, and its optimization appears to be even more difficult.

Apart from optimizing surrogates, a few more specialized approaches for finding the F-measure maximizer (2) have been presented in the last decades (Lewis, 1995; Chai, 2005; Jansche, 2007; Ye et al., 2012; Quevedo et al., 2012). These algorithms will be revisited in Section 4. They typically require the assumption of independence of the Y_i, i.e.,

$$P(\boldsymbol{Y} = \boldsymbol{y}) = \prod_{i=1}^{m} p_i^{y_i} (1 - p_i)^{1-y_i} \,, \tag{3}$$

with $p_i = P(Y_i = 1)$. While being natural for problems like binary classification, this assumption is indeed not tenable in domains like MLC and structured output prediction. We will show in Section 4 that algorithms based on independence assumptions or marginal probabilities are not statistically consistent when arbitrary probability distributions P are considered. Moreover, we also show that the worst-case regret of these algorithms is very high.

Looking at (2), it seems that information about the entire joint distribution P is needed to maximize the F-measure. Yet, as will be shown in this paper, the problem can be solved more efficiently. In Section 5, we present a general algorithm that requires only a quadratic instead of an exponential (with respect to m) number of parameters of the joint distribution. If these parameters are given, then, depending on their form, the exact solution can be obtained in quadratic or cubic time. This result holds regardless of the underlying distribution. In particular, unlike algorithms such as Chai (2005); Jansche (2007); Ye et al. (2012) and Quevedo et al. (2012), we do not require independence of the binary response variables (labels).

Our theoretical results are specifically relevant for applications in multi-label classification and structured output prediction. In these application domains, three different aggregation schemes of the F-measure can be distinguished, namely instance-wise, micro- and macro-averaging. One should carefully distinguish these versions, since algorithms optimized with a given objective are usually performing suboptimally for other (target) evaluation measures (e.g., Dembczyński et al., 2012a; Luaces et al., 2012). In Section 7, we present extensive experimental results to illustrate the practical usefulness of our findings. More specifically, all examined methods are compared for a series of multi-label classification problems. One particular data set originates from a recent data mining competition, in which we obtained the second place using some of the algorithms presented in this article. Let us anticipate that our experimental results will not determine a clear winner. This is not at all surprising: while enjoying the advantage of consistency, our algorithm requires the estimation of more parameters than existing approximate algorithms. As a consequence, exact optimization is not necessarily superior to approximate optimization. Instead, the relative performance of exact and approximate optimization depends on several factors, such as the sample size, the length of $\boldsymbol{Y}$, the shape of the distribution P, etc.

As mentioned above, we adopt a decision-theoretic point of view: assuming a probabilistic model to be given, the problem of F-measure maximization is interpreted as an inference problem. Before going into technical details, we like to stress that this is only one way of looking at F-measure maximization. A second, somewhat orthogonal approach is to optimize the F-measure during the training phase. This is sometimes referred to as empirical utility maximization. In general, optimality in this framework is different from our definition of optimality, but connections between the two paradigms have recently been

discussed by Ye et al. (2012). These authors establish asymptotic equivalence results under the assumption of independence and infinitely large vectors $\boldsymbol{Y}$. They focus on binary classification problems, for which such assumptions are realistic, because the vector $\boldsymbol{Y}$ then represents an entire test set of i.i.d. observations. The same assumptions are made in another recent work that provides an interesting theoretical analysis for binary classification problems (Zhao et al., 2013). However, in structured output prediction and multi-label classification, independence does not hold and the length of $\boldsymbol{Y}$ might be small, especially if the instance-wise F-measure needs to be optimized.

Algorithms that optimize the F-measure during training will hence not be discussed further in this article. Nevertheless, we briefly mention some of them here for the sake of completeness. In binary classification, such algorithms are extensions of support vector machines (Musicant et al., 2003; Joachims, 2005), logistic regression (Jansche, 2005) or boosting (Kokkinos, 2010). However, the most popular methods, including that of Keerthi et al. (2007), rely on explicit threshold adjustment. A few specific algorithms have also been proposed for certain applications in structured output prediction (Tsochantaridis et al., 2005; Suzuki et al., 2006; Daumé III et al., 2009) and multi-label classification (Fan and Lin, 2007; Zhang et al., 2010; Petterson and Caetano, 2010, 2011). During training, some of these methods, especially those based on structured SVMs, need to solve an inference problem that is closely related but not identical to (2). In a recent paper, we have presented a theoretical and experimental comparison of approaches that optimize the F-measure during training or inference, respectively, in the context of multi-label classification (Dembczyński et al., 2013). Since we focus on the decision-theoretic point of view in this work, we do not discuss the theoretical results obtained in that article, but for completeness we report some experimental results.

Parts of this article have already been published in previous conference papers (Dembczyński et al., 2011, 2013). Here, we summarize the results of these papers in a unifying framework, provide a much more detailed theoretical analysis, and complement them by additional formal results as well as novel experimental studies. The rest of the article is organized as follows. Section 2 gives a formal definition of the notion of regret, which will serve as a key element for showing that most existing algorithms are suboptimal. Section 3 contains a regret analysis of algorithms that optimize other loss functions than the F-measure. In Section 4, we perform a similar analysis for F-measure inference algorithms that have been proposed in the literature. Subsequently, our own algorithm is presented in Section 5. In Section 6, we test the inference algorithms on synthetic data, while practical considerations, applications and experimental results on benchmark data are further discussed in Section 7. The proofs of all theorems in Sections 3 and 4 can be found in an appendix.

2. Formal Presentation of our Mathematical Framework

In order to show formally that many existing algorithms are sub-optimal w.r.t. optimising the F-measure, we will use a mathematical framework that is closely related to frameworks encountered in classical machine learning papers. However, our analysis will slightly differ from the analysis performed in such papers, as we are investigating inference algorithms instead of training algorithms. Let us start with a formal definition of what we will call the

regret of an inference algorithm.

Definition 2.1 *Given a probability distribution P, the regret of a vector of predictions $\boldsymbol{h}$ w.r.t. the F-measure is formally defined as*

$$R_F(\boldsymbol{h}) = \mathbb{E}\big[F(\boldsymbol{Y}, \boldsymbol{h}_F) - F(\boldsymbol{Y}, \boldsymbol{h})\big] = \sum_{\boldsymbol{y} \in \{0,1\}^m} \big[F(\boldsymbol{y}, \boldsymbol{h}_F) - F(\boldsymbol{y}, \boldsymbol{h})\big] P(\boldsymbol{y})\,,$$

with $\boldsymbol{h}_F$ the F-measure maximizer in (2).

The above definition of regret (aka *excess risk* in the statistical literature) can be considered as a classical tool in the framework of Bayes-risk consistency (Devroye et al., 1997; Bartlett et al., 2006). However, let us emphasize that the theoretical analysis presented below differs from traditional techniques for investigating the consistency of machine learning algorithms. Typically, a training algorithm is considered consistent if its risk converges in probability to the Bayes risk as the training sample grows to infinity. Since many training algorithms optimize a convex and differentiable surrogate of the target loss of interest, such an analysis is often performed by bounding the risk of the target loss as a function of the surrogate *ϕ-risk* of the surrogate loss (e.g., Breiman, 2000; Steinwart, 2001; Zhang, 2004; Bartlett et al., 2006; Tewari and Bartlett, 2007; Duchi et al., 2010; Gao and Zhou, 2013).

In this article, we start from a different perspective, since we are analyzing the Bayes-optimality of inference algorithms. As such, we call a given loss a surrogate loss for the F-measure if an inference algorithm optimizes this loss instead of the F-measure. This is different from, for example, the above papers, which analyse the consistency of surrogate losses during the training phase of a machine learning algorithm, using the surrogate loss as the internal training loss, i.e., as a convex and differentiable approximation of the target loss that is optimized during training. Furthermore, a second notable difference to other papers is that sample size convergence is less important in our analysis, as we are starting from a trained probabilistic model that is assumed to deliver consistent estimates. A similar analysis to the one presented here has been performed for the subset 0/1 loss and the Hamming loss in (Dembczyński et al., 2012a).

We will consider the regret of various types of loss functions and algorithms under any arbitrary probability distribution P. By searching for probability distributions that maximize the regret, we are mainly considering the worst-case scenario. In the case of surrogate losses, we restrict this search to probability distributions that deliver unique risk minimizers; the reasons for this restriction are of technical nature. Similar to the F-measure maximizer, let us introduce the risk minimizer of a loss $L : \{0,1\}^m \times \{0,1\}^m \to \mathbb{R}_+$ as

$$\boldsymbol{h}_L = \operatorname*{arg\,min}_{\boldsymbol{h} \in \{0,1\}^m} \mathbb{E}\left[L(\boldsymbol{Y}, \boldsymbol{h})\right] = \operatorname*{arg\,min}_{\boldsymbol{h} \in \{0,1\}^m} \sum_{\boldsymbol{y} \in \{0,1\}^m} P(\boldsymbol{y})\, L(\boldsymbol{y}, \boldsymbol{h}). \tag{4}$$

This allows us to introduce the worst-case regret formally.

Definition 2.2 *Let $L : \{0,1\}^m \times \{0,1\}^m \to \mathbb{R}_+$ be a loss, let $\mathcal{P}$ be the set of all probability distributions over $\{0,1\}^m$, and let $\mathcal{P}_L^u$ be the subset of $\mathcal{P}$ that delivers unique solutions to*

(4). Then, the worst-case regret is defined as

$$\sup_{P \in \mathcal{P}_L^u} \mathbb{E}\big[F(\boldsymbol{Y}, \boldsymbol{h}_F) - F(\boldsymbol{Y}, \boldsymbol{h}_L)\big] \ , \tag{5}$$

with $\boldsymbol{h}_F$ and $\boldsymbol{h}_L$ defined by (2) and (4), respectively.

Note that, in the above definition, we restrict the worst-case analysis to probability distributions with a unique risk minimizer for L. Technically, the problem would otherwise become more difficult, as it would require the comparison of the F-measure maximizer with a *set* of risk minimizers $\boldsymbol{H}_L$ instead of a unique minimizer $\boldsymbol{h}_L$. This could be done in different ways, for example, by looking at the most favorable case for L, leading to

$$\sup_{P \in \mathcal{P}} \min_{\boldsymbol{h}_L \in \boldsymbol{H}_L} \mathbb{E}\Big[F(\boldsymbol{Y}, \boldsymbol{h}_F) - F(\boldsymbol{Y}, \boldsymbol{h}_L)\Big] \ ,$$

or the least favorable one, leading to

$$\sup_{P \in \mathcal{P}} \max_{\boldsymbol{h}_L \in \boldsymbol{H}_L} \mathbb{E}\Big[F(\boldsymbol{Y}, \boldsymbol{h}_F) - F(\boldsymbol{Y}, \boldsymbol{h}_L)\Big] \ .$$

To avoid a more or less arbitrary decision, we prefer to exclude these cases from the beginning. In any case, it is clear that the regret (5) provides a lower bound for any other definition of regret that maximizes over the entire set $\mathcal{P}$ of distributions. Furthermore, remark that non-uniqueness of the F-measure maximizer is unproblematic, since for the definition of the regret, only the *value* of the F-measure is important, not the maximizer itself.

Using classical notions such as Fisher consistency (Wu et al., 2010), one can say that a sufficient condition for inconsistency is encountered if (5) does not evaluate to zero. However, since exact solutions or lower bounds will be derived, we are able to give much more precise information on the degree of incorrectness.

3. The F-Measure and Related Loss Functions

Given the difficulty of maximizing the F-measure, we start our analysis by investigating a few related loss functions that have been used as surrogates in some multi-label classification and structured output prediction papers. We will analyze the Hamming loss, the subset zero-one loss and the Jaccard index. For the former two loss functions, we perform a regret analysis to show that optimizing these loss functions is not consistent if the F-measure is our performance metric of interest. For the Jaccard index, we derive a simple upper bound on the regret when optimizing the F-measure instead.

3.1 The Hamming Loss

The Hamming loss can be considered as the most standard loss for multi-label classification problems (e.g., Schapire and Singer, 2000; Tsoumakas and Katakis, 2007; Hariharan et al., 2010). It is also widely used in many structured output prediction methods (e.g., Taskar et al., 2004; Daumé III et al., 2009; Finley and Joachims, 2008). Using the general notation

that was introduced above, the Hamming loss simply corresponds to the error rate in binary classification, and it can be formally defined as follows:[1]

$$L_H(\boldsymbol{y}, \boldsymbol{h}) = \frac{1}{m} \sum_{i=1}^{m} [\![y_i \neq h_i(\boldsymbol{x})]\!] \,. \tag{6}$$

For the Hamming loss, the risk minimizer is

$$\boldsymbol{h}_H = \operatorname*{arg\,min}_{\boldsymbol{h} \in \{0,1\}^m} \mathbb{E}\left[L_H(\boldsymbol{Y}, \boldsymbol{h})\right] = \operatorname*{arg\,min}_{\boldsymbol{h} \in \{0,1\}^m} \sum_{\boldsymbol{y} \in \{0,1\}^m} P(\boldsymbol{y})\, L_H(\boldsymbol{y}, \boldsymbol{h}). \tag{7}$$

This is obtained by $\boldsymbol{h}_H(\boldsymbol{x}) = (h_{H,1}, \ldots, h_{H,m})$, where

$$h_{H,i}(\boldsymbol{x}) = \operatorname*{arg\,max}_{b \in \{0,1\}} P(Y_i = b) \qquad \forall i \in \{1, \ldots, m\}. \tag{8}$$

Thus, in order to optimize the Hamming loss, one should select the marginal modes of P. The following theorem presents our main result for the Hamming loss.

Theorem 1 *Let $\boldsymbol{h}_H$ be a vector of predictions obtained by minimizing the Hamming loss, Then for $m > 2$ the worst-case regret is given by:*

$$\sup_{P \in \mathcal{P}^u_{L_H}} \left(\mathbb{E}\big[F(\boldsymbol{Y}, \boldsymbol{h}_F) - F(\boldsymbol{Y}, \boldsymbol{h}_H)\big] \right) = 0.5 \,,$$

where the supremum is taken over all possible distributions P that result in a unique Hamming loss minimizer.

In other words, the theorem indicates that optimizing the Hamming loss as a surrogate for the F-measure results in a prediction that is far from optimal. This claim will be further confirmed by experimental results in Sections 6 and 7.

3.2 The Subset Zero-One Loss

The next multi-label loss function we analyze is the subset 0/1 loss, which generalizes the well-known 0/1 loss from the conventional to the multi-label setting:

$$L_s(\boldsymbol{y}, \boldsymbol{h}) = [\![\boldsymbol{y} \neq \boldsymbol{h}]\!] \tag{9}$$

Admittedly, this loss function may appear overly stringent, especially in the case of many labels. Moreover, since making a mistake on a single label is punished as hardly as a mistake on all labels, it does not discriminate well between "almost correct" and "completely wrong" predictions. Still, a lot of existing frameworks for multi-label classification and structured output prediction optimize a convex upper bound on this loss in a direct or approximate manner. For example, conditional random fields optimize the log-loss as a surrogate for the subset zero-one loss (Lafferty et al., 2001), structured support vector machines consider the

1. We use $[\![c]\!]$ to denote the indicator function, equal to 1 if predicate c holds and 0 otherwise.

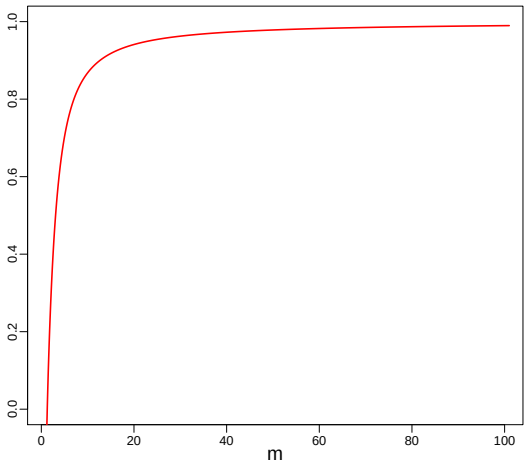


Figure 1: Plot of the worst-case regret for the subset zero-one loss (11) as a function of the number of labels m.

structured hinge loss as a surrogate of the subset 0/1 loss when no margin/slack rescaling is performed (Tsochantaridis et al., 2005) and probabilistic classifier chains with logistic base classifiers optimize the log-loss approximately by means of pseudo-likelihood maximization (Dembczyński et al., 2010, 2012b). Moreover, maximum a posteriori (MAP) estimation techniques in Bayesian statistics and graphical models are also known to minimize the subset 0/1 loss.

As for any other 0/1 loss, the risk-minimizing prediction for (9) is simply given by the mode of the distribution:

$$\boldsymbol{h}_s = \underset{\boldsymbol{y}\in\{0,1\}^m}{\arg\max}\, P(\boldsymbol{y}) \tag{10}$$

Thus, unlike the Hamming loss, looking at marginal probabilities does not suffice to minimize the subset 0/1 loss. When the independence assumption is violated, information about the joint distribution over labels is needed, similar as for the F-measure. Our interest in the subset 0/1 loss is primarily fueled by this connection. Summarized in the following theorem, we perform a similar type of regret analysis as for the Hamming loss.

Theorem 2 *Let $\boldsymbol{h}_s$ be a vector of predictions obtained by minimizing the subset 0/1 loss, then for $m > 2$ the worst-case regret is given by:*

$$\sup_{P\in\mathcal{P}^u_{L_s}} \left(\mathbb{E}\big[F(\boldsymbol{Y},\boldsymbol{h}_F) - F(\boldsymbol{Y},\boldsymbol{h}_s)\big]\right) = \frac{(-m-2+2m^2)m}{(2m-1)(4+m+m^2)}\,, \tag{11}$$

where the supremum is taken over all possible distributions P.

Let us remark that the worst-case regret converges rapidly to one as a function of the number of labels m, as illustrated in Figure 1. Similar to the result for the Hamming loss, the above theorem confirms that using the subset zero-one loss as an alternative for the

F-measure can potentially yield a high regret. Optimizing the subset zero-one loss might hence be not a valid alternative. Our experimental results in Sections 6 and 7 will indeed make clear that such an approach performs suboptimal for several data sets.

3.3 The Jaccard Index

The F-measure was originally defined by set operators, as a measure for expressing the similarity of sets. In this literature, it is known as the dice coefficient (Dice, 1945). Another well-known measure for expressing similarity of sets is the Jaccard index. The two measures are very related, since both belong to a more general parametric family of similarity measures for sets (De Baets et al., 2009). The Jaccard index computes the ratio of intersection to union:

$$J(\boldsymbol{y}, \boldsymbol{h}) = \frac{|\{i \mid y_i = 1 \wedge h_i = 1, i = 1, \ldots, m\}|}{|\{i \mid y_i = 1 \vee h_i = 1, i = 1, \ldots, m\}|} \tag{12}$$

Owing to a simple transformation, it can also be written as follows:[2]

$$J(\boldsymbol{y}, \boldsymbol{h}) = \frac{\sum_{i=1}^{m} y_i h_i}{\sum_{i=1}^{m} y_i + \sum_{i=1}^{m} h_i - \sum_{i=1}^{m} y_i h_i} \tag{13}$$

In recent years, the Jaccard index has gained popularity in the machine learning community. In the context of kernel methods, it is often used as an alternative to the linear kernel for binary feature vectors, such as fingerprints of molecules in cheminformatics and bioinformatics. In these application domains, one often speaks of the Tanimoto kernel (Swamidass et al., 2005).

As a utility function the Jaccard index is often considered in multi-label classification. It remains an open question whether or not a closed-form solution for the risk minimizer of the Jaccard similarity exists, but the maximization is far from straightforward. Under the assumption of label independence, which allows one to transform many loss functions to a contingency table, Quevedo et al. (2012) have recently proposed an exact algorithm for maximizing the instance-wise Jaccard similarity, as well as other loss functions that can be computed from such a contingency table. However, without this assumption, one commonly believes that exact optimization is intractable (Chierichetti et al., 2010). Even though the F-measure and the Jaccard are monotonically related, it is not the case that the F-measure maximizer is necessarily also the Jaccard maximizer, because the summation in (4) in general breaks the monotonicity property. As a result, the analysis that we report for the Jaccard index differs from the one reported for the Hamming loss and the subset 0/1 loss. Given that the maximization of the Jaccard index is believed to be much harder, it does not make sense to use this measure as a surrogate for the F-measure during optimization. In contrast, one might think of maximizing the F-measure as a surrogate for the Jaccard index. The following result characterizes what we can lose with such a strategy.

Theorem 3 *Let $\boldsymbol{h}_J$ and $\boldsymbol{h}_F$ be vectors of predictions obtained by maximizing the Jaccard index and the F-measure, respectively. Let the utility of the F-measure maximizer be given*

2. Similar as the F-measure, note that the denominator is 0 if $y_i = h_i = 0$ for all $i = 1, \ldots, m$. In this case, the utility is 0 by definition.

by

$$\delta(P) = \max_{\boldsymbol{h}\in\{0,1\}^m} \mathbb{E}\left[F(\boldsymbol{Y},\boldsymbol{h})\right] = \max_{\boldsymbol{h}\in\{0,1\}^m} \sum_{\boldsymbol{y}\in\{0,1\}^m} P(\boldsymbol{y})\, F(\boldsymbol{y},\boldsymbol{h}).$$

The regret of the F-measure maximizer with respect to the Jaccard index is then upper bounded by

$$\mathbb{E}\big[J(\boldsymbol{Y},\boldsymbol{h}_J) - J(\boldsymbol{Y},\boldsymbol{h}_F)\big] \leq 1 - \delta(P)/2$$

for all possible distributions P.

Notwithstanding that the above upper bound on the regret remains rather loose, the observation is interesting because the upper bound on the regret decreases as a function of the utility of the F-measure maximizer $\delta(P)$. Due to this relationship, a high utility for the F-measure implies that optimizing this measure as a surrogate for the Jaccard similarity might be a reasonable thing to do. In other words, on data sets for which an F-measure maximizing algorithm gets good results, one can expect that this algorithm will also get good results in terms of the Jaccard index.

4. Existing Algorithms for F-Measure Maximization

The previous section revealed that optimizing more conventional loss functions as surrogates for the F-measure might result in a poor predictive performance. In this section, we perform a similar type of analysis for more specialized algorithms that intend to solve (2). These algorithms make different types of assumptions to simplify the problem. First of all, the algorithms operate on a constrained hypothesis space, sometimes justified by theoretical arguments. Secondly, they only guarantee optimality for specific distributions P.

4.1 Algorithms Based on Label Independence

By assuming independence of the random variables $Y_1, ..., Y_m$, optimization problem (2) can be substantially simplified. It has been shown independently in (Lewis, 1995) and (Jansche, 2007) that the optimal solution then always contains the labels with the highest marginal probabilities, or no labels at all. As a consequence, only a few hypotheses $\boldsymbol{h}$ ($m+1$ instead of 2^m) need to be examined, and the computation of the expected F-measure can be performed in an efficient way.

Theorem 4 [(Lewis, 1995)] *Let $Y_1, Y_2, \ldots, Y_m$ be independent Bernoulli variables with parameters $p_1, p_2, \ldots, p_m$ respectively. Then, for all $j, k \in \{1, \ldots, m\}$, $h_{F,j} = 1$ and $h_{F,k} = 0$ implies $p_j \geq p_k$.*

In addition, Lewis (1995) showed that the expected F-measure can be approximated by the following expression under the assumption of independence:[3]

$$\mathbb{E}\left[F(\boldsymbol{Y},\boldsymbol{h})\right] \simeq \begin{cases} \prod_{i=1}^{m}(1-p_i), & \text{if } \boldsymbol{h}=\boldsymbol{0}_m \\ \frac{2\sum_{i=1}^{m} p_i h_i}{\sum_{i=1}^{m} p_i + \sum_{i=1}^{m} h_i}, & \text{if } \boldsymbol{h}\neq\boldsymbol{0}_m \end{cases}$$

This approximation is exact for $\boldsymbol{h}=\boldsymbol{0}_m$, while for $\boldsymbol{h}\neq\boldsymbol{0}_m$, an upper bound of the error can easily be determined (Lewis, 1995).

However, Chai (2005), Jansche (2007) and Quevedo et al. (2012) have independently proposed exact procedures for computing the F-maximizer. To this end, independence is assumed and marginal probabilities $p_1, p_2, \ldots, p_m$ serve as input for the algorithms. The method of Jansche runs in $\mathcal{O}(m^4)$, while the other two approaches solve the same problem more efficiently in $\mathcal{O}(m^3)$.

As a starting point for explaining the three algorithms, notice that (2) can be solved via outer and inner maximization. Namely, (2) can be transformed into an inner maximization

$$\boldsymbol{h}^{(k)} = \operatorname*{arg\,max}_{\boldsymbol{h}\in H_k} \mathbb{E}\left[F(\boldsymbol{Y},\boldsymbol{h})\right], \tag{14}$$

where $H_k = \{\boldsymbol{h}\in\{0,1\}^m \mid \sum_{i=1}^{m} h_i = k\}$, followed by an outer maximization

$$\boldsymbol{h}_F = \operatorname*{arg\,max}_{\boldsymbol{h}\in\{\boldsymbol{h}^{(0)},\ldots,\boldsymbol{h}^{(m)}\}} \mathbb{E}\left[F(\boldsymbol{Y},\boldsymbol{h})\right]. \tag{15}$$

The outer maximization (15) can be done by simply checking all $m+1$ possibilities. The main effort is then devoted to solving the inner maximization (14). According to Lewis' theorem, to solve (14), one needs to check only one vector $\boldsymbol{h}$ for a given k, in which $h_i = 1$ for the k labels with highest marginal probabilities p_i. The remaining problem is the computation of the expected F-measure in (14). This expectation cannot be computed naively, as the sum is over exponentially many terms. But the F-measure is a function of integer counts that are bounded, so it can normally only assume a much smaller number of distinct values. It has been further shown that the expectation has a domain with a cardinality exponential in m; but since the cardinality of its range is polynomial in m, it can be computed in polynomial time. As a result, Jansche (2007) obtains an algorithm that is cubic in m for computing (14), resulting in an overall $\mathcal{O}(m^4)$ time complexity. He also presents an approximate version of this procedure, reducing the complexity from cubic to quadratic. This approximation leads to an overall complexity of $\mathcal{O}(m^3)$, but it does no longer guarantee optimality of the prediction.

As a more efficient alternative, the procedure of Chai (2005) is based on ordering the labels according to the marginal probabilities. For $\boldsymbol{h}^{(k)} \in H_k$, thus $h_i = 1$ for k labels with the greatest marginal probabilities, he derives the following expression:

$$\mathbb{E}\left[F(\boldsymbol{Y},\boldsymbol{h}^{(k)})\right] = 2\prod_{i=1}^{m}(1-p_i)I_1(m)\ ,$$

3. We use $\boldsymbol{0}_m$ and $\boldsymbol{1}_m$ to denote m-element vectors of all zeros or ones, respectively.

where $I_1(m)$ is given by the following recurrence equations and boundary conditions:

$$\begin{aligned} I_t(a) &= I_{t+1}(a) + r_t I_{t+1}(a+1) + r_t J_{t+1}(a+1) \\ J_t(a) &= J_{t+1}(a) + r_t J_{t+1}(a+1) \\ & \quad I_{k+1}(a) = 0 \quad J_{m+1}(a) = a^{-1} \end{aligned}$$

with $r_i = p_i/(1-p_i)$. These equations suggest a dynamic programming algorithm of space $\mathcal{O}(m)$ and time $\mathcal{O}(m^2)$ in computing the expected F-measure for given k. This yields an overall time complexity of $\mathcal{O}(m^3)$.

In a more recent follow-up paper, Ye et al. (2012) further improved the dynamic programming algorithm to an $\mathcal{O}(m^2)$ complexity by additional sharing of internal representations. The old and the new version of the algorithm both rely on Lewis' theorem and a factorization of the probability mass for constructing recurrence equations, hence leaving few hope for extending the algorithm to situations where label independence cannot be assumed. In another recent paper, Quevedo et al. (2012) propose a general inference procedure that utilizes similar recurrence equations and dynamic programming techniques. In contrast to Chai (2005), Jansche (2007) and Ye et al. (2012), the authors primarily address multi-label classification problems, focusing on a wide range of loss functions that can be computed from a contingency table in an instance-wise manner. As a result, the instance-wise F-measure is maximized as a special case, while assuming label independence.

If the independence assumption is violated, none of the above methods is able to guarantee optimality. In the most general case, the F-maximizer needs to be computed by analyzing the joint distribution. The above methods rely on modeling or ordering marginal probabilities, which is not sufficient to compute the F-maximizer for many distributions. This is illustrated by the following example, in which two joint distributions with identical marginal probabilities have different F-measure maximizers:

$\boldsymbol{y}$	$P(\boldsymbol{y})$	$\boldsymbol{y}$	$P(\boldsymbol{y})$
0001	0.1	0000	0.5
0010	0.2	1001	0.1
0100	0.2	1010	0.2
1000	0.5	1100	0.2

The non-specified configurations have zero probability mass. For both distributions, we have $p_1 = P(Y_1 = 1) = 0.5$, $p_2 = P(Y_2 = 1) = 0.2$, $p_3 = P(Y_3 = 1) = 0.2$, $p_4 = P(Y_4 = 1) = 0.1$, but one can easily check that the F-measure maximizers are $\boldsymbol{h} = (1000)$ and $\boldsymbol{h} = (0000)$, respectively. The regret is small for this simple example, but methods that assume independence may produce predictions being far away from the optimal one. The following result shows this concretely.

Theorem 5 *Let $\boldsymbol{h}_I$ be a vector of predictions obtained by assuming label independence as defined in (3), then the worst-case regret is lower-bounded by:*

$$\sup_P \left(\mathbb{E}\big[F(\boldsymbol{Y}, \boldsymbol{h}_F) - F(\boldsymbol{Y}, \boldsymbol{h}_I)\big] \right) \geq 2q - 1,$$

for all $q \in [1/2, 1]$ *satisfying* $\sum_{s=1}^{m} \left(\frac{2m!}{(m-s)!(s-1)!(m+s)} q^{m-s}(1-q)^s \right) - q^m > 0$ *and the supremum taken over all possible distributions* P.

For increasing m, the condition is satisfied for q close to one (see the appendix for details). In such a scenario, the worst-case regret is lower bounded by $R_q = 2q - 1$, so that $\lim_{q \to 1, m \to \infty} R_q = 1$. As a consequence, the lower bound becomes tight in the limit of m going to infinity, as summarized in the following corollary.

Corollary 1 *Let* $\boldsymbol{h}_I$ *be a vector of predictions obtained by assuming independence, then the worst-case regret converges to* 1 *in the limit of* m, *i.e.,*

$$\lim_{m \to \infty} \sup_{P} \left(\mathbb{E}\big[F(\boldsymbol{Y}, \boldsymbol{h}_F) - F(\boldsymbol{Y}, \boldsymbol{h}_I)\big] \right) = 1,$$

where the supremum is taken over all possible distributions P.

Again we will show by means of experiments in Sections 6 and 7 that algorithms based on label independence can be suboptimal on real-world data sets.

4.2 Algorithms Based on the Categorical Distribution

Solving (2) becomes straightforward in the case of a specific distribution in which the probability mass is distributed over vectors $\boldsymbol{y}$ containing only a single positive label, i.e., $\sum_{i=1}^{m} y_i = 1$, corresponding to the categorical distribution. This was studied by del Coz et al. (2009) in the setting of so-called *non-deterministic classification*.

Theorem 6 [(del Coz et al., 2009)] *Denote by* $\boldsymbol{y}(i)$ *a vector for which* $y_i = 1$ *and all the other entries are zeros. Assume that* P *is a joint distribution such that* $P(\boldsymbol{Y} = \boldsymbol{y}(i)) = p_i$. *The maximizer* $\boldsymbol{h}$ *of (2) consists of the* k *labels with the highest marginal probabilities, where* k *is the first integer for which*

$$\sum_{j=1}^{k} p_j \geq (1+k) p_{k+1};$$

if there is no such integer, then $\boldsymbol{h} = \mathbf{1}_m$.

The categorical distribution reflects the case of multi-class classification problems, as a special case of multi-label classification problems. The above approach is only applicable to such problems.

4.3 Algorithms that Use Both the Marginal and the Joint Distribution

Since all the methods so far rely on the fact that the optimal solution contains ones for the labels with the highest marginal probabilities (or consists of a vector of zeros), one may expect that thresholding on the marginal probabilities ($h_i(\theta) = 1$ for $p_i \geq \theta$, and $h_i(\theta) = 0$ otherwise) will provide a solution to (2) in general. Practically, despite using marginal probabilities for the thresholds, such a scenario does not assume label independence anymore,

because also the joint probability distribution P must be provided.[4] When the labels are ordered according to the marginal probabilities, thus $p_i \geq p_{i+1}$ for all $i \in \{1, ..., m-1\}$, this approach resembles the following optimization problem:

$$\boldsymbol{h}_T = \underset{\theta \in \{1, p_1, ..., p_m\}}{\arg\max} \; \mathbb{E}\left[F(\boldsymbol{Y}, \boldsymbol{h}(\theta))\right].$$

Thus, to find an optimal threshold θ, access to the entire joint distribution is needed. However, this is not the main problem here, since in the next section, we will show that only a polynomial number of parameters of the joint distribution is needed. What is more interesting is the observation that the F-maximizer is in general not consistent with the order of marginal label probabilities. In fact, the regret can be substantial, as shown by the following result.

Theorem 7 *Let $\boldsymbol{h}_T$ be a vector of predictions obtained by putting a threshold on sorted marginal probabilities, then the worst-case regret is lower bounded by*

$$\sup_P \left(\mathbb{E}\big[F(\boldsymbol{Y}, \boldsymbol{h}_F) - F(\boldsymbol{Y}, \boldsymbol{h}_T)\big]\right) \geq \max\left(0, \frac{1}{6} - \frac{2}{m+4}\right),$$

where the supremum is taken over all possible distributions P.

Finding the exact value of the supremum in the worst case is for the above formulation an interesting open question. The statement is a surprising result in light of the existence of many algorithms that rely on finding a threshold for maximizing the F-measure (Keerthi et al., 2007; Fan and Lin, 2007; Zhang et al., 2010; Lipton et al., 2014)—remark that those methods rather seek for a threshold on scoring functions instead of marginal probabilities. While being justified by Theorems 4, 5 and 6 for specific applications, thresholding does not yield optimal predictions in general. Let us illustrate this with an example for which $m = 12$:

$\boldsymbol{y}$	$P(\boldsymbol{y})$
000000000000	0.21
100000000000	0.39
011111100000	0.2
010000011111	0.2

The non-specified configurations have zero probability mass. The F-measure maximizer is given by (1000000000000); yet, not the first label but the second one exhibits the highest marginal probability. The regret remains rather low in this case, but higher values can be easily obtained by constructing more complicated examples from (41)—see the appendix.

4. In fact most thresholding methods that optimize the F-measure during training do not use the joint distribution, and define a threshold based on marginals only. However, that is in practice the same as assuming independence, and resembles the same conclusions as in Section 4.1. In contrast, the regret will be lower when also the joint distribution is used to define the threshold. When the F-measure is optimized in an inference phase, starting from a trained probabilistic model, access to the joint distribution is of course needed.

5. An Exact Algorithm for F-Measure Maximization

We now introduce an exact and efficient algorithm for computing the F-maximizer without using any additional assumption on the probability distribution P. While adopting the idea of decomposing the problem into an outer and an inner maximization, our algorithm differs in the way the inner maximization is solved.[5] For convenience, let us introduce the following quantities:

$$s_{\boldsymbol{y}} = \sum_{i=1}^{m} y_i, \quad \Delta_{ik} = \sum_{\boldsymbol{y}: y_i = 1} \frac{2P(\boldsymbol{y})}{s_{\boldsymbol{y}} + k}.$$

The first quantity gives the number of ones in the label vector $\boldsymbol{y}$, while Δ_{ik} is a specific marginal value for i-th label, which for $u = 1$ corresponds to weighted true positives. Using these quantities, we show that only $m^2 + 1$ parameters of the joint distribution $P(\boldsymbol{y})$ are needed to compute the F-maximizer.

Theorem 8 *The solution of (2) can be computed by solely using $P(\boldsymbol{y} = \mathbf{0}_m)$ and the values of Δ_{ik}, for $i, k \in \{1, \ldots, m\}$, that constitute an $m \times m$ matrix Δ.*

Proof. The inner optimization problem (14) can be formulated as follows:

$$\boldsymbol{h}^{(k)} = \operatorname*{arg\,max}_{\boldsymbol{h} \in H_k} \mathbb{E}\left[F(\boldsymbol{Y}, \boldsymbol{h})\right] = \operatorname*{arg\,max}_{\boldsymbol{h} \in \mathcal{H}_k} \sum_{\boldsymbol{y} \in \{0,1\}^m} P(\boldsymbol{y}) \frac{2\sum_{i=1}^{m} y_i h_i}{s_{\boldsymbol{y}} + k} \ .$$

The sums in arg max can be swapped, resulting in

$$\sum_{\boldsymbol{y} \in \{0,1\}^m} P(\boldsymbol{y}) \frac{2\sum_{i=1}^{m} y_i h_i}{s_{\boldsymbol{y}} + k} = \sum_{i=1}^{m} h_i \sum_{\boldsymbol{y} \in \{0,1\}^m} \frac{2P(\boldsymbol{y}) y_i}{s_{\boldsymbol{y}} + k} = \sum_{i=1}^{m} h_i \sum_{\boldsymbol{y}: y_i = 1} \frac{2P(\boldsymbol{y})}{s_{\boldsymbol{y}} + k}\,.$$

Finally, we obtain

$$\boldsymbol{h}^{(k)} = \operatorname*{arg\,max}_{\boldsymbol{h} \in H_k} \sum_{i=1}^{m} h_i \Delta_{ik}\,. \tag{16}$$

As a result, one does not need the whole distribution to find the maximizer of the F-measure, but the values of Δ_{ik}, which can be given in the form of an $m \times m$ matrix Δ. For the special case of $k = 0$, we have $\boldsymbol{h}^{(k)} = \mathbf{0}_m$ and $\mathbb{E}_{\boldsymbol{y} \sim P(\boldsymbol{y})}\left[F(\boldsymbol{y}, \mathbf{0}_m)\right] = P(\boldsymbol{y} = \mathbf{0}_m)$. ■

If the matrix Δ is given, the solution of the F-measure maximization (2) is straightforward, since for each inner maximization the problem boils down to selecting the k labels with the highest Δ_{ik}. The resulting algorithm, referred to as General F-measure Maximizer (GFM), is summarized in Algorithm 1 and its time complexity is analyzed in the following theorem.

Theorem 9 *Algorithm 1 solves problem (2) in time $\mathcal{O}(m^2)$ assuming that the matrix Δ of m^2 parameters and $P(\boldsymbol{y} = \mathbf{0}_m)$ are given.*

5. The description of the method slightly differs from the previous paper (Dembczyński et al., 2011), and it is concordant with Dembczyński et al. (2013).

Algorithm 1 General F-measure Maximizer

INPUT: matrix Δ and probability $P(\boldsymbol{y} = \mathbf{0}_m)$

for $k = 1$ to m **do**

solve the inner optimization problem (14):

$$\boldsymbol{h}^{(k)} = \underset{\boldsymbol{h} \in H_i}{\arg\max} \sum_{i=1}^{m} h_i \Delta_{ik}$$

by setting $h_i = 1$ to k labels with the highest Δ_{ik} (in case of ties take any k top labels), and $h_i = 0$ for the rest;

store a value of

$$\mathbb{E}\left[F(\boldsymbol{Y}, \boldsymbol{h}^{(k)})\right] = \sum_{i=1}^{m} h_i^{(k)} \Delta_{ik};$$

end for

define $\boldsymbol{h}^{(0)} = \mathbf{0}_m$, and $\mathbb{E}\left[F(\boldsymbol{Y}, \mathbf{0}_m)\right] = P(\boldsymbol{y} = \mathbf{0}_m)$;

solve the outer optimization problem (15):

$$\boldsymbol{h}_F = \underset{\boldsymbol{h} \in \{\boldsymbol{h}^{(0)}, \ldots, \boldsymbol{h}^{(m)}\}}{\arg\max} \mathbb{E}\left[F(\boldsymbol{Y}, \boldsymbol{h})\right];$$

return $\boldsymbol{h}_F$ and $\mathbb{E}\left[F(\boldsymbol{Y}, \boldsymbol{h}_F)\right]$;

Proof. To solve (16), it is enough to find the top k elements (i.e., the elements with the highest values) in the k-th column of matrix Δ, which can be carried out in linear time (Cormen et al., 2001). This step has to be repeated for all k. Therefore, the overall complexity of the inner maximization is quadratic. The solution of the outer optimization problem (15) is then straight-forward and requires linear time. ■

In light of combining the inference algorithm with particular training algorithms, like multinomial regression as we discuss it in Section 7.1.4, it could be reasonable to redefine the formulation in the following way. Consider the probabilities

$$p_{is} = P(y_i = 1, s_{\boldsymbol{y}} = s), \quad i, s \in \{1, \ldots, m\}, \tag{17}$$

that constitute an $m \times m$ matrix P.[6] Let us also introduce an $m \times m$ matrix W with elements

$$w_{sk} = \frac{2}{(s+k)} \ , \qquad s, k \in \{1, \ldots, m\}. \tag{18}$$

It can be easily shown that

$$\Delta = \mathrm{PW}, \tag{19}$$

since

$$\Delta_{ik} = \sum_{\boldsymbol{y}: y_i = 1} \frac{2P(\boldsymbol{y})}{s_{\boldsymbol{y}} + k} = \sum_{s=1}^{m} \frac{2p_{is}}{s+k}.$$

6. We use capital letters to denote matrices.

If the matrix P is taken as an input by the algorithm, then its complexity is dominated by the matrix multiplication (19) that is solved naively in $\mathcal{O}(m^3)$, but faster algorithms working in $\mathcal{O}(m^{2.376})$ are known (Coppersmith and Winograd, 1990).[7]

Interestingly, the above results clearly suggest that the F-measure maximizer is more affected by the number of 1s in the $\boldsymbol{y}$-vectors than by the interdependence between particular labels. In other words, modeling of pairwise or higher degree dependencies between labels is not necessary to obtain an optimal solution, but a proper estimation of marginal quantities (Δ_{ik}, or p_{is}) that take the number of co-occurring labels into account.

In the reminder of this section, we discuss the properties of the GFM algorithm in comparison to the other algorithms discussed in Section 4. The methods presented by Chai (2005); Jansche (2007) and Ye et al. (2012) all assume label independence and produce exactly the same result, apart from small numerical instabilities that might always occur. These methods, contrary to GFM, will not deliver an exact F-maximizer if the assumption of independence is violated. On the other hand, the disadvantage of GFM is the quadratic number of parameters it requires as input, while the other methods only need m parameters. Since the estimation of a larger number of parameters is statistically more difficult, it is a priori unclear which method performs better in practice. We are facing here a common trade-off between an approximate method on better estimates (we need to estimate a smaller number of parameters from a given sample) and an exact method on potentially weaker estimates. Nonetheless, if the joint distribution is concentrated on a small number of different label combinations $\boldsymbol{y}$, the estimates of Δ or P can be as good as the estimates of the marginal probabilities p_i.

From the computational perspective, Jansche's method is characterized by a much higher time complexity, being respectively $\mathcal{O}(m^4)$ and $\mathcal{O}(m^3)$ for the exact and the approximate versions. The method of Chai has a cubic complexity, and the enhanced version presented in Ye et al. (2012) is more efficient, since it solves the problem in $O(m^2)$ time. The GFM algorithm is quite competitive, as its complexity is of $O(m^2)$ or $O(m^3)$, depending on the setting. Moreover, the cubic complexity of GFM, which follows from the matrix multiplication, can be further decreased if the number of distinct values of $s_{\boldsymbol{y}}$ with non-zero probability mass is smaller than m.

6. Simulations

In the previous sections, we gave theoretical results concerning the performance of different inference methods in the worst case scenarios. Here, we verify the methods empirically on synthetic data to check the difference in average performance on two large classes of distributions. The first class assumes independence of labels, while the second class uses a model with strong label dependencies.

We test four inference methods optimal for different performance measures. The first one is suited for Hamming loss. It estimates the marginal probabilities by simple counting from a given sample and gives empirical marginal modes as output. We denote this method MM, since it estimates marginal modes. The second one is tailored for subset 0/1 loss. It

7. The complexity of the Coppersmith-Winograd algorithm (Coppersmith and Winograd, 1990) is more of theoretical significance, since practically this algorithm outperforms the naïve method only for huge matrices.

seeks for the joint mode by checking the frequencies of label combinations appearing in the sample. We refer to this method as JM, since it estimates the joint mode. The two remaining methods are suited for F-measure maximization. We use the dynamic programming method of Ye et al. (2012) that assumes label independence, denoted by FM, and the exact GFM method described in the previous section, which performs exact inference. All parameters required by these algorithms are also estimated from the sample by simple counting. We verify the performance of the inference methods by using Hamming loss, subset 0/1 loss, the Jaccard index, and the F-measure.

We run these simulations, as well as the other experiments described later in this paper, on a Debian virtual machine with 8-core x64 processor and 5GB RAM.

6.1 Label Independence

The independent data are generated according to:

$$P(\boldsymbol{y}) = \prod_{i=1}^{m} P(y_i)\,,$$

where the probabilities $P(y_i)$ are given by the logistic model:

$$P(y_i = 1) = \frac{1}{1 + \exp(-w_i)}, \qquad \text{where } w_i \sim N(0, 3)\,.$$

In experiments we set the number of labels to 25 and vary the number of observations using the following values $\{5, 10, 20, 30, 40, 50, 75, 100, 200, 500, 1000, 2000, 5000, 10000\}$. We repeat the experiment for 30 different models, i.e., sets of values w_i. For each model, we use 50 different training sets of the same size, but to reduce the variance of the results we use for testing one set of 100,000 observations. The results are given in Figure 2. The right column presents the performance with respect to different measures as a function of the number of training observations. The left column gives the same results, but zoomed to the range from 5 to 100 training observations. We see from the plots that MM and JM get the best results for Hamming loss and subset 0/1 loss. Since the labels are independent, the marginal modes and joint mode are the same. Therefore, for large enough training samples, these two algorithms converge to the same prediction that should be optimal for Hamming and subset 0/1. However, JM converges much slower, since it directly estimates the joint mode, by checking the frequencies of label combinations in the training set. FM and GFM perform very similarly for each performance measure. Since the labels are independent, FM and GFM should converge to the same prediction, being optimal for the F-measure. GFM, however, may get slightly worse results for small sample sizes, since it needs to estimate a larger number of parameters than FM. We also see that algorithms maximizing the F-measure perform the best for Jaccard index.

6.2 Strong Label Dependence

We perform a similar experiment for a data model with strong label dependencies. The data are generated by the chain rule of probability, i.e.,

$$P(\boldsymbol{y}) = \prod_{i=1}^{m} P(y_i \,|\, y_1, \ldots, y_{i-1}),$$

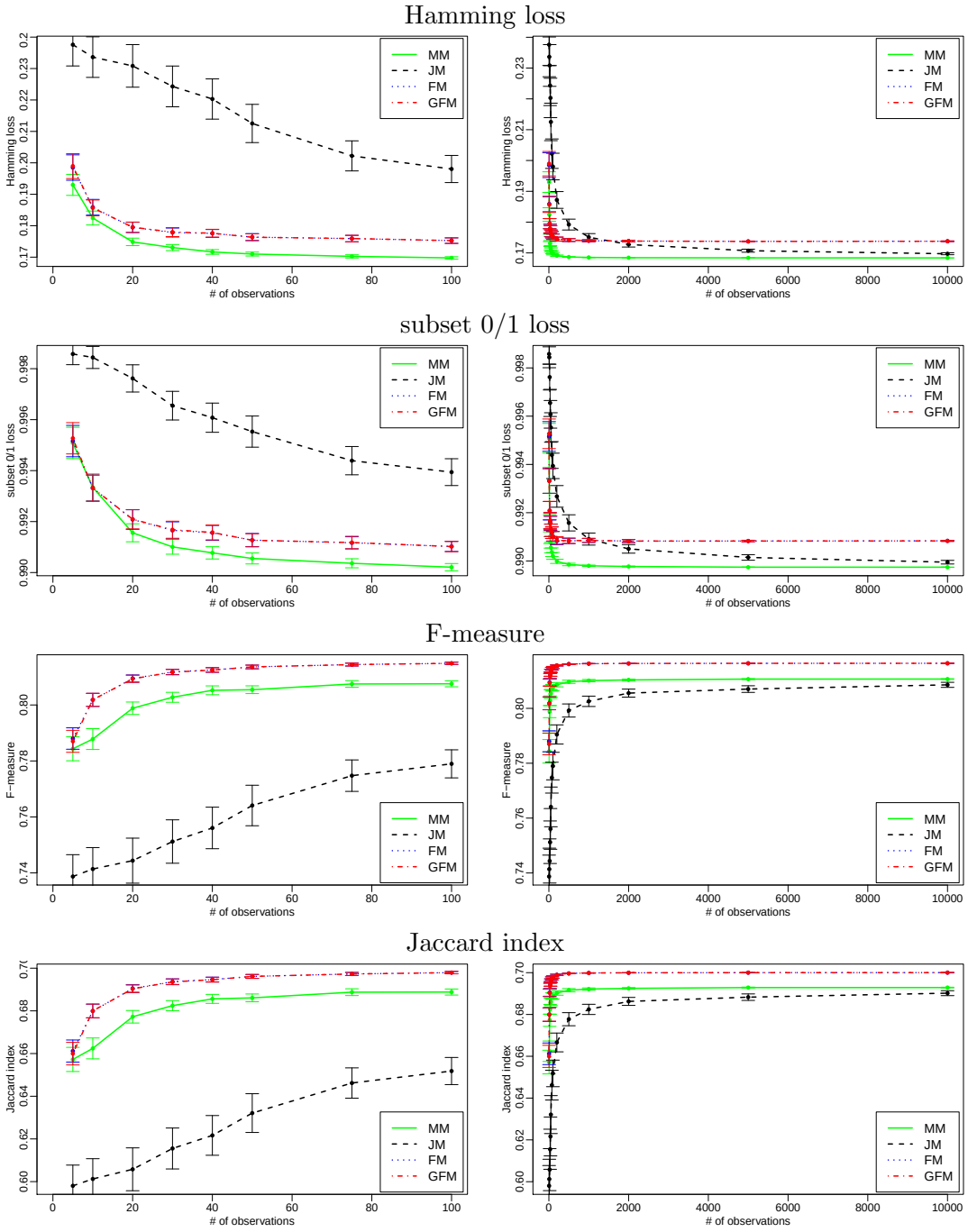


Figure 2: Performance of inference methods in case of label independence as a function of the number of training observations. Left: the performance up to 100 training observations. Right: the performance up to 10000 training observations. The error bars show the standard error of the measured quantities.

where the probabilities $P(y_i \,|\, y_1, \ldots, y_{i-1})$ are coming from a logistic model of the form:

$$P(y_i \,|\, y_1, \ldots, y_{i-1}) = \frac{1}{1 + \exp(-\sum_{j=1}^{i-1} 2w_{ij}(y_j - \frac{1}{2}) - w_{i0})},$$

with all $w_{ij} \sim N(1, 3)$ and $w_{i0} \sim N(1, 3)$. This model tends to produce for a given label y_i a value that appeared more often on previous labels. The results are presented in Figure 3. In this case, the marginal modes and the joint mode are not the same. Therefore MM performs the best for Hamming loss and JM for subset 0/1 loss. More importantly, we can see that FM performs suboptimally for F-measure, and a clear winner in this case is GFM. This result confirms our theoretical analysis and shows the benefits of the GFM inference method. Also GFM performs the best for the Jaccard index, followed by the JM. The FM method in this case performs the worst. Similarly to the F-measure we might expect here that methods that assume label independence will not get good results with respect to the Jaccard index.

There is of course a price we have to pay for a good performance of GFM. Figure 4 presents running times of parameter estimation and inference of the algorithms as a function of the number of labels. GFM is the slowest method. The running times increase quadratically with the number of labels. The inference time of FM grows also quadratically, but with a lower rate. Moreover, this algorithm needs only to estimate marginal probabilities, therefore its estimation time is exactly the same as for the MM method.

7. Application to Multi-Label Classification Problems

The inference methods for F-measure maximization can be used whenever an estimation of required parameters is possible. In this section, we focus on the application of the inference methods in the multi-label setting. Thus, we consider a task of predicting a vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_m) \in \{0, 1\}^m$ given another vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ as input attributes. To this end, we use a training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ to estimate the required parameters and perform inference for a given test vector $\boldsymbol{x}$ so as to deliver an optimal prediction under the F-measure (1). Thus, we optimize the performance for each instance individually (instance-wise F-measure), in contrast to macro- and micro-averaging of the F-measure (Yang, 1999; Tsoumakas et al., 2010).

7.1 Learning Algorithms

The inference methods for F-measure maximization can be combined with many conventional learning approaches. In the following, we mainly focus on algorithms in which the final decision is made based on an empirical distribution, like in nearest-neighbors or decision trees. In these algorithms all parameters required by F-measure maximization methods are estimated from the empirical distribution. At the end of this section we also discuss another approach in which the parameters are obtained from a parametric model, for example, from logistic regression. We present the algorithms with the GFM method for F-measure maximization; however, it should be clear from the description how to obtain corresponding variants of the algorithms for the methods that assume label independence.

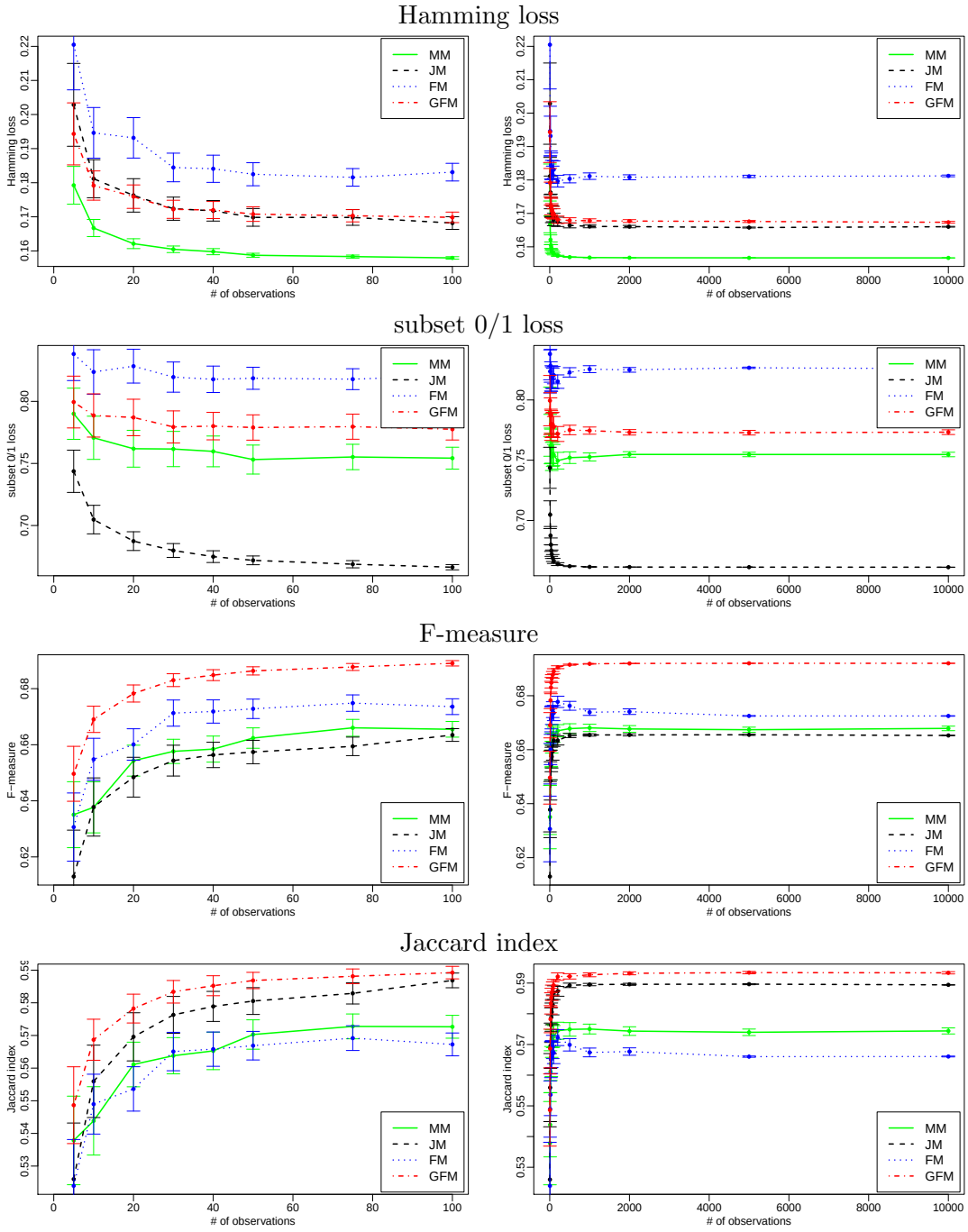


Figure 3: Performance of inference methods in case of label dependence as a function of the number of training observations. Left: the performance up to 100 training observations. Right: the performance up to 10000 training observations. The error bars show the standard error of the measured quantities.

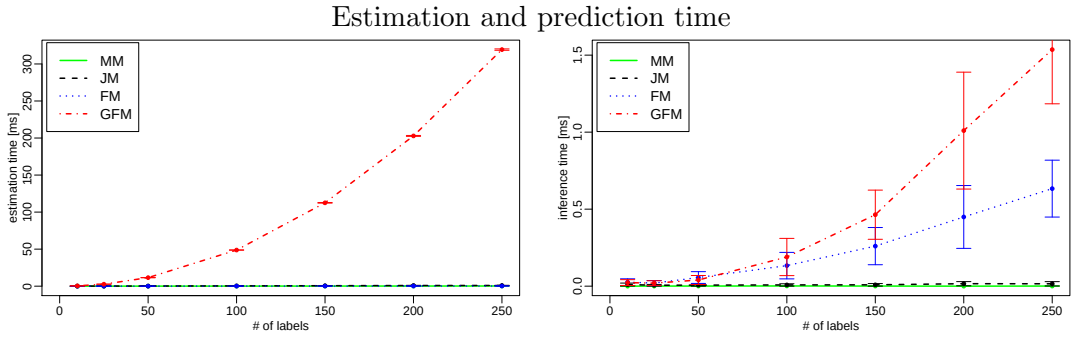


Figure 4: Running times in milliseconds of the inference methods for label dependent data as a function of the number of labels. Left plot shows estimation time of parameters required by the method. Right plot shows the inference time. The error bars show the standard error of the measured quantities.

7.1.1 Instance-Based Learning

Several instance-based methods for multi-label classification have been proposed in the past (e.g., Zhang and Zhou, 2007; Cheng and Hüllermeier, 2009), but none of these methods is tailored for optimizing the F-measure during an inference phase. Consider a query instance $\boldsymbol{x} \in \mathcal{X}$ and let $\{\boldsymbol{y}_j, \boldsymbol{x}_j\}_{j=1}^{l}$ denote the l nearest neighbors of $\boldsymbol{x}$ with respect to a distance measure on $\mathcal{X}$ in the training set. The number l is a fixed parameter of the method. Instance-based learning can be extended for maximizing the F-measure in a straight-forward way, namely by replacing the distribution $P(\boldsymbol{y})$ with the empirical distribution in the neighborhood of the query. Correspondingly, the values Δ and $P(\boldsymbol{y} = \mathbf{0}_m)$ are estimated through simple counting:

$$\hat{\Delta}_{ik} = \frac{1}{l} \sum_{j=1}^{l} \frac{[\![y_{ik} = 1]\!]}{s_{\boldsymbol{y}_j} + k}, \quad \hat{P}(\boldsymbol{y} = \mathbf{0}_m) = \frac{1}{l} \sum_{j=1}^{l} [\![\boldsymbol{y}_j = \mathbf{0}_m]\!].$$

By using these estimates in the GFM algorithm, we obtain an estimate of the F-measure maximizer.

7.1.2 Decision Trees

Decision tree methods have been extensively studied in standard classification and regression settings, and have also been generalized to multi-output problems like MLC and multivariate regression (see e.g., Zhang, 1998; Lee, 2006). An adaptation of decision trees for maximizing the F-measure is slightly more complicated than for instance-based learning. The method we present here resembles the ideas used in predictive clustering trees (Vens et al., 2008). For simplicity, we only consider binary trees. Moreover, since decision tree induction is well-known in the machine learning field, we restrict our discussion to the main differences to conventional (classification or regression) tree learning. In a decision tree, each leaf node represents a (typically rectangular) part of the instance space $\mathcal{X}$ and is labeled with a local model. Typically, a local model consists of a single prediction, namely the prediction that

minimizes the average loss among the associated training examples (e.g., the mean value in regression and the most frequent label in classification). Applying the same principle in the case of the F-measure in MLC comes down to computing the maximizer of this measure over the instances in a leaf node. This is similar to the case of instance-based learning, except that the examples used for estimating Δ and $P(\boldsymbol{y} = \mathbf{0}_m)$ originate from a rectangular region of the instance space $\mathcal{X}$ and not from the neighborhood of the query instance $\boldsymbol{x}$.

The more demanding part is the induction of the tree, i.e., finding optimal splits with respect to the F-measure. For a given node of the tree, we search over all attributes and possible split points, just like in regular decision tree algorithms. Let us denote by $\mathcal{N}$ a set of training examples $\{(\boldsymbol{y}_j, \boldsymbol{x}_j)\}_{j=1}^{l}$ in a node. The task is then to find a split of $\mathcal{N}$ into two subsets, $\mathcal{N}_L$ and $\mathcal{N}_R$, that maximize some purity criterion. Our approach is analogous to the one of conventional decision trees, but based on the F-measure:

$$Q = \frac{\#(\mathcal{N}_L)}{\#(\mathcal{N})} F(\mathcal{N}_\mathcal{L}) + \frac{\#(\mathcal{N}_R)}{\#(\mathcal{N})} F(\mathcal{N}_\mathcal{R})$$

where $\#(\mathcal{A})$ is a cardinality of a set $\mathcal{A}$, and

$$F(\mathcal{A}) = \max_{\boldsymbol{h}} \frac{1}{\#(\mathcal{A})} \sum_{\boldsymbol{y}_i \in \mathcal{A}} F(\boldsymbol{y}_i, \boldsymbol{h}).$$

In order to speed up computations of $F(\cdot)$, we notice that searching a split is usually performed in an example-by-example manner, which means that we can easily update our estimates of Δ and $P(\boldsymbol{y} = \mathbf{0}_m)$. Moreover, assuming that a given training example has a lower number of relevant labels, we do not have to recompute the whole matrix Δ, but only update some of its rows and columns. Finally, the search for the top k elements in each column of Δ can be made faster by checking local changes in the current rankings of labels. We repeat the above step recursively until a stop condition is reached, for example the F-measure becomes maximal or the number of examples in the leaf node falls below a threshold. Of course, more sophisticated approaches are conceivable.

Let us also mention that it is possible to generalize bagging (Breiman, 1996) with these decision trees. Then, GFM can easily be applied over the bootstrap sample of the weak hypotheses returned by the trees.

7.1.3 Probabilistic Classifier Chains

Probabilistic classifier chains (PCCs) (Dembczyński et al., 2010) is an approach similar to maximum entropy Markov models (McCallum et al., 2000) and to conditional random fields (CRFs) (Lafferty et al., 2001; Ghamrawi and McCallum, 2005). All these approaches estimate the joint conditional distribution $P(\boldsymbol{y} \,|\, \boldsymbol{x})$. PCC has an additional advantage that one can easily sample from the estimated distribution. The underlying idea is to repeatedly apply the product rule of probability to the joint distribution of the labels:

$$P(\boldsymbol{y} \,|\, \boldsymbol{x}) = \prod_{i=1}^{m} P(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}) \tag{20}$$

Learning in this framework can be considered as a simple procedure. According to (20), we decompose the joint distribution into a sequence of marginal distributions that depend

on a subset of the labels. These marginal distributions can be learned by m functions $f_i : \mathcal{X} \times \{0,1\}^{i-1} \to [0,1]$ on an augmented input space $\mathcal{X} \times \{0,1\}^{i-1}$, taking $y_1, \ldots, y_{i-1}$ as additional input attributes:

$$f_i : (\boldsymbol{x}, y_1, \ldots, y_{i-1}) \mapsto P(y_i = 1 \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}) \tag{21}$$

By plugging the log-linear model into (20), it can be shown that pairwise dependencies between labels y_i and y_j are modeled (see also (Kumar et al., 2013)).

The algorithm is mainly suitable for subset 0/1 loss. Exploration of the structure of the chain in the inference phase boils down to search for the most probable label combination in a resulting probabilistic binary tree. A greedy algorithm follows only one path choosing always only the most probable label in each position in the chain (Read et al., 2009). This algorithm, however, may lead to suboptimal results. It has been shown (Dembczyński et al., 2012b), however, that an exact method based on a variant of uniform-cost search with a cut-off list finds the joint mode in a linear time of $1/p_{max}$, where p_{max} is the probability of the joint mode. For reasonable values of p_{max}, this method works very fast.

To optimize the response of PCC for other loss functions, we need to obtain a sample of observations from the conditional joint distribution $P(\boldsymbol{y} \,|\, \boldsymbol{x})$. To get a single observation, we can follow the chain and pick the value of label y_i by tossing a biased coin with probabilities given by the i-th classifier. Such a procedure is sometimes referred to as ancestral sampling (Bishop, 2006, Chapter 8). From the sample of such observations, we can estimate all the parameters required by inference methods like GFM, similarly as in the case of nearest neighbors and decision trees. More precisely, let $\{\boldsymbol{y}_j\}_{j=1}^n$ denote a set of sampled observations for a given test example $\boldsymbol{x}$. Then, the values Δ and $P(\boldsymbol{y} = \mathbf{0}_m)$ can be estimated through simple counting:

$$\hat{\Delta}_{ik} = \frac{1}{n} \sum_{j=1}^{n} \frac{[\![y_{ik} = 1]\!]}{s_{\boldsymbol{y}_j} + k}, \quad \hat{P}(\boldsymbol{y} = \mathbf{0}_m) = \frac{1}{n} \sum_{j=1}^{n} [\![\boldsymbol{y}_j = \mathbf{0}_m]\!].$$

By plugging these estimates into the GFM algorithm, we obtain for a given $\boldsymbol{x}$ a prediction optimized for the F-measure.

7.1.4 Parametric Models

Alternatively to the approaches described above, which estimate the parameters required by the F-measure maximization methods on empirical distributions, we discuss here an approach in which the parameters are efficiently obtained from a parametric model (Dembczyński et al., 2013). Unfortunately, there is no easy way to estimate directly the matrix Δ, since the elements of this matrix do not correspond to a proper probability distribution. However, we can estimate matrix P, defined in (17), which elements are probabilities:

$$p_{is} = P(y_i = 1, s_{\boldsymbol{y}} = s), \quad i, s \in \{1, \ldots, m\}\,.$$

Multiplying the matrix P by a weight matrix W with elements (18) results in the estimate of Δ, as shown in (19).

To estimate the elements of P we can use a simple reduction to m multi-class probability estimation (i.e., multinomial regression) problems, each with at most $m + 1$ classes. We

define one multinomial regression model for each row of matrix P. Let us observe that for $t = (\llbracket y_i = 1 \rrbracket \cdot s_{\boldsymbol{y}})$, $t \in \{0, \ldots, m\}$, we have:

$$\sum_{t \in \{0,\ldots,m\}} P(t \,|\, \boldsymbol{x}) = 1 \,.$$

Therefore, we can define the i-th multinomial regression problem as:

$$f_i : \boldsymbol{x} \mapsto P(t \,|\, \boldsymbol{x}), \text{ for } t \in \{0, \ldots, m\} \,. \tag{22}$$

In a similar way, we can estimate $P(\boldsymbol{y} = \mathbf{0}_m \,|\, \boldsymbol{x})$ by performing an additional reduction to binary probability estimation with $t = \llbracket \boldsymbol{y} = \mathbf{0}_m \rrbracket$ as an output variable:

$$f_0 : \boldsymbol{x} \mapsto P(t \,|\, \boldsymbol{x}) \,,$$

and solving it via logistic regression.

The decomposition of the original problem into independent multinomial regression tasks has computational advantages. Moreover, since the number of distinct values of $s_{\boldsymbol{y}}$ is usually small, the number of classes in a single multinomial regression task is much smaller than $m + 1$; only in the worst case, we end up with a quadratic complexity in the number of labels m.

Let us, however, remark that the elements of matrix P estimated across different tasks are not fully independent of each other (e.g., p_{im} is the same for all i, since $P(y_i = 1, s_{\boldsymbol{y}} = m) = P(\boldsymbol{y} = \mathbf{1}_m)$). Consequently, learning on a finite training set may lead to conflicting estimates that are not in agreement with any valid distribution. To avoid such conflicts, one may include additional constraints in the learning problem or calibrate the estimates afterwards. However, Dembczyński et al. (2013) have shown that with the sample size growing to infinity this approach is statistically consistent.

To summarize this approach we notice that learning of the probabilistic model has a time complexity that is at most quadratic in m. In the inference phase for a test instance $\boldsymbol{x}$, we first get estimates of $P(\mathbf{0}_m \,|\, \boldsymbol{x})$ and P from the probabilistic model, again in at most quadratic time. Then, we need to multiply matrices P and W to get Δ. Finally, all the parameters are plugged into the GFM method. This approach has in the original paper (Dembczyński et al., 2013) been referred to as Exact-F-measure-Plug-in classifier (EFP).

Under the assumption of label independence, one can simplify this approach. Since we only need marginal probabilities p_i, it is enough to reduce the problem to m binary probability estimation tasks that can be solved, for example, by logistic regression. Then, for each test instance $\boldsymbol{x}$, we obtain a vector of marginal probabilities p_i, to which one might apply, for example, the inference method of Ye et al. (2012). We refer to this approach as Label-independence F-measure-Plug-in classifier (LFP), similarly as in (Dembczyński et al., 2013).

7.2 Experimental Results

We test some of the algorithms described above on four commonly used multi-label benchmark data sets with known training and test sets. We take these data sets from the MU-

LAN[8] and LibSVM[9] repositories. Table 1 contains basic statistics of these data sets. We also relate the obtained results to the results of a variant of structured SVMs that moves the effort of maximizing the F-measure to the training phase.

We run the experiments on the machine that was also used for the simulations described earlier, i.e., on a Debian virtual machine with 8-core x64 processor and 5GB RAM.

data set	#train	#test	m	d
Scene	1211	1196	6	294
Yeast	1500	917	14	103
Enron	1123	579	53	1001
Mediamill	30993	12914	101	120

Table 1: Data sets and their properties. The number of training and test observations is denoted by #train and #test, respectively, m is the number of labels, d is the number of features.

7.2.1 Instance-Based Learning

We first present results of instance-based learning. We use a different number of nearest neighbors, $l \in \{10, 20, 50, 100\}$. For each test example, we seek for its nearest neighbors and apply different inference methods. We use exactly the same methods we applied in our experiments on synthetic data given in Section 6. The first method, MM, estimates the marginal modes, JM estimates the joint mode, FM approximates the F-measure by assuming label independence, and the introduced GFM performs exact inference for the F-measure. The nearest-neighbor search is performed by using the Weka (Hall et al., 2009) and Mulan (Tsoumakas et al., 2011) implementation of instance-based learning.

The results are given in Table 2. Similarly as in Section 6, we report the performance in terms of Hamming loss, subset 0/1 loss, F-measure and Jaccard index.

We can generally confirm our previous results on synthetic data: an inference method tailored for a given performance measure obtains the best results. This is clear for Hamming loss, for which MM has the smallest error throughout. JM performs the best for subset 0/1 loss on all data sets with some exceptions on Enron. Both methods tailored for F-measure maximization, FM and GFM, substantially outperform MM and JM on this performance criterion. FM seems to beat GFM on Enron, while the latter method produces better results on the other data sets. There are, however, no clear results for the Jaccard index.

Let us underline that in the case of nearest neighbor methods, we are dealing with a specific trade-off between the size of the neighborhood and its volume. In general, increasing the sample size in the inference methods should improve the results. But in this case, by increasing the number of neighbors, we simultaneously increase the volume of the space that contains the neighbors. In other words, some of the neighbors can be far away from

8. This repository can be found at: `http://mulan.sourceforge.net/datasets.html`.
9. This repository can be found at: `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html`.

	Hamming Loss [%]				Subset 0/1 Loss [%]				F-measure [%]				Jaccard [%]			
	10	20	50	100	10	20	50	100	10	20	50	100	10	20	50	100
Scene																
MM	**10.28**	**10.62**	**11.52**	13.03	40.47	45.15	54.01	66.05	65.80	59.53	49.39	36.40	64.23	58.36	48.54	35.79
JM	11.19	10.99	11.97	**12.28**	**36.54**	**36.29**	**39.13**	**40.05**	68.26	68.73	65.89	**64.97**	**67.06**	**67.47**	**64.63**	**63.71**
FM	13.80	14.40	16.74	19.83	53.09	56.10	64.55	75.17	70.80	70.12	67.20	62.65	64.59	63.24	58.76	52.48
GFM	13.03	14.09	16.15	19.12	49.75	54.68	63.13	72.91	**71.42**	**70.29**	**67.85**	64.31	65.95	63.80	59.73	54.41
Yeast																
MM	**20.72**	**20.00**	**20.02**	**20.38**	81.57	81.03	82.77	85.71	62.88	61.93	60.31	58.26	52.31	51.49	49.74	47.47
JM	23.09	21.78	21.84	22.04	**76.66**	**76.23**	**77.54**	**78.30**	59.14	60.53	60.97	60.68	49.75	51.06	51.20	50.72
FM	22.94	23.26	22.83	23.21	83.21	83.21	85.93	88.11	65.29	65.06	65.23	64.85	54.14	53.74	53.71	**53.10**
GFM	22.94	23.17	22.87	23.61	82.99	83.97	86.37	88.66	**65.49**	**65.47**	**65.75**	**64.98**	**54.31**	**54.09**	**54.11**	53.08
Enron																
MM	**5.73**	**5.94**	**6.28**	**6.46**	88.08	88.60	89.46	89.64	35.56	27.91	23.23	23.71	28.70	22.97	19.36	19.60
JM	6.56	6.51	6.70	6.73	**86.18**	87.39	89.29	89.46	33.38	29.52	25.52	24.68	27.68	24.51	21.05	20.38
FM	6.51	6.23	6.34	6.48	87.56	88.60	**87.22**	**88.26**	**47.50**	**47.01**	**42.10**	**37.86**	**37.31**	**36.82**	**33.65**	**30.17**
GFM	6.61	6.28	6.54	6.63	88.08	**86.87**	88.43	89.29	44.43	42.45	34.76	29.27	35.12	34.10	27.93	23.70
Mediamill																
MM	**3.29**	**3.18**	**3.16**	**3.19**	89.17	89.66	90.30	91.13	54.98	53.92	52.68	51.81	43.32	43.34	43.10	42.74
JM	4.17	4.03	3.93	3.91	**88.93**	**88.82**	**89.24**	**89.05**	48.87	48.58	47.32	47.07	43.55	**43.66**	**43.30**	**42.80**
FM	3.97	3.81	3.72	3.68	91.30	91.99	92.64	93.39	55.47	55.65	55.51	**55.24**	**43.60**	42.64	41.49	40.58
GFM	3.87	3.74	3.66	3.65	90.48	91.15	91.99	92.81	**55.52**	**55.80**	**55.54**	55.16	38.56	38.40	37.39	37.28

Table 2: Empirical results on 4 benchmark data sets. Instance-based methods are used with a different number of neighbors ($l \in \{10, 20, 50, 100\}$) and with different inference methods: MM—estimates marginal modes, JM—the joint mode, FM—approximates the F-measure maximizer by assuming independence of labels, and GFM—computes the exact F-measure maximizer over the nearest neighbors. Results are reported for Hamming loss, subset 0/1 loss, F-measure and Jaccard index. The best results for a given number of neighbors and a performance measure are marked in bold.

	Inference time [s]					Inference time [s]			
	10	20	50	100		10	20	50	100
	Scene					Yeast			
MM	7.622	8.110	9.078	10.064	MM	3.662	3.907	4.352	4.793
JM	7.519	8.052	9.093	10.010	JM	3.687	3.958	4.362	4.812
FM	7.488	8.023	9.106	10.126	FM	3.728	3.920	4.404	4.788
GFM	8.311	8.171	9.169	10.475	GFM	3.634	3.944	4.467	4.875
	Enron					Mediamill			
MM	1.601	1.618	1.978	2.172	MM	336.550	374.207	435.171	499.977
JM	1.477	1.607	1.929	2.236	JM	338.855	373.297	431.354	492.079
FM	1.483	1.645	1.931	2.145	FM	339.704	375.230	433.660	493.424
GFM	1.518	1.681	2.022	2.228	GFM	347.402	382.663	442.356	502.060

Table 3: Computation time in seconds for the instance-based methods for different $l = \{10, 20, 50, 100\}$ and inference methods: MM, JM, FM, and GFM. Computation time includes searching time and inference time

the test example, which usually deteriorates the quality of the estimates. This is usually the case for high-dimensional problems. This may partially explain the results on the Enron data set. The performance under the F-measure decreases substantially with the number of neighbors. Also when comparing the instance-based methods with other methods, presented later in this section, we see that the overall performance on this data set is much worse for the former methods.

In Table 3 we present the computation time of the instance-based methods. The computation time includes both the search of nearest-neighbors and the inference. We can easily observe that the nearest-neighbor search dominates inference in terms of computational complexity, since there is only a small difference in computational times between the inference methods for a given data set and l. We do not present the inference times separately here, since their characteristics are exactly the same as presented in Section 6.

7.2.2 Probabilistic Classifier Chains

In the next experiments, we use PCC. We train PCC by using linear regularized logistic regression. We use the implementation of logistic regression from Mallet (McCallum, 2002). We tune the regularization parameter for each base classifier independently by minimizing the logistic loss, hoping to thereby produce better probability estimates. We use 5-fold cross-validation and choose the regularization parameter from the following set of possible values $\{10^{-4}, 10^{-3}, \ldots, 10^{3}\}$. Similarly as in the previous experiments, we use four different inference mechanisms. For MM, FM, and GFM methods, we obtain the estimates of the required parameters by performing ancestral sampling from the conditional joint distribution of each test example $\boldsymbol{x}$. The JM method, instead of estimating the joint mode from the sample, applies the efficient exact search method (Dembczyński et al., 2012b).

Table 4 contains the results of the experiment. As before, we report the Hamming loss, subset 0/1 loss, F-measure and the Jaccard index. We also give the training and inference times. The training time concerns the entire procedure that consists of tuning

of the regularization parameter in cross-validation and training of a model with the best value of the regularization parameter. The results for MM, FM, and GFM are given for sample size of 1000. From the results, we can clearly state that approaches tailored for the F-measure obtain better results on this performance criterion. It seems that GFM obtains slightly better results, but also needs a little bit more time. In Figure 5, we additionally present the F-measure and inference times of FM and GFM as a function of the number of observations obtained from ancestral sampling. These results are computed over 5 runs of the inference methods to decrease the impact of the randomness of the sampling method. The plots confirm our theoretical results concerning the predictive performance and time complexity. However, the inference times reported here include all three steps: sampling, estimation of parameters, and inference based on these parameters. As we can see in Table 4 and Figure 5 the differences between GFM, FM, and MM are not substantial here, since sampling is the most expensive step. The exact method used for joint mode estimation works much faster than the other methods based on sampling. From the other results in Table 4 we can also observe that MM is the best for Hamming loss, and JM for the subset 0/1 loss. The results for the Jaccard index show that maximization of the F-measure can be used as a proxy for this performance criterion, at least FM and GFM perform better than MM.

7.2.3 Parametric Models

To complete the picture we also present the experimental results of parametric models that have been previously published in (Dembczyński et al., 2013). We compare EFP and its simplified variant LFP. We also include to the comparison the binary relevance (BR) approach that learns and predicts for each label independently. Such a model should perform well under the Hamming loss. All the methods use linear models and are trained, similarly as PCC, by logistic regression. The regularization parameter and its tuning is exactly the same as in the experiment with PCC.

The results are summarized in Table 5. We show the results for Hamming loss, F-measure and report training and inference times. Not surprisingly, BR achieves the best results for Hamming loss, but it is outperformed by all the other methods on the F-measure. EFP is the best method in this regard.

BR is the most efficient in inference. Nevertheless, the inference times of LFP and EFP are quite comparable to those of BR, despite their quadratic (for LFP) and cubic (for EFP) complexity. Admittedly, however, the data sets used in the experiments only contain a small to moderate number of labels (up to 100). For data sets with thousands of labels, the difference is likely to become substantially larger. The training of BR and LFP (these are exactly the same procedures) is the most effective. Training of EFP leads to m multinomial regression models. One should note, however, that the number of classes in each multinomial regression models can be much less than the highest possible value $m+1$. Therefore, the training of EFP is still quite effective and takes only a few times longer than the training of LFP. The training time includes here also the tuning time, similarly as in the case of PCC.

	Hamming Loss [%]	Subset 0/1 Loss [%]	F-Measure [%]	Jaccard [%]	Training Time [s]	Inference Time [s]
			Scene			
MM	**9.89**	42.69	62.73	61.37	39	1.839
JM	10.40	**34.87**	70.93	**69.48**	39	0.255
FM	12.83	50.44	**72.78**	66.71	39	1.858
GFM	12.74	49.86	**72.78**	66.89	39	1.927
			Yeast			
MM	**19.56**	81.98	61.78	51.32	32	7.120
JM	20.91	**76.34**	63.36	53.56	32	0.220
FM	22.31	84.22	65.53	54.29	32	7.161
GFM	22.54	84.73	**65.63**	**54.32**	32	7.513
			Enron			
MM	**4.63**	86.88	52.62	42.51	81	120.636
JM	4.81	**82.56**	55.67	45.80	81	1.974
FM	5.59	90.33	58.54	46.19	81	121.172
GFM	5.53	89.12	**59.08**	**46.89**	81	121.700
			Mediamill			
MM	**3.18**	92.44	51.21	39.60	6150	2226.455
JM	3.57	**90.02**	44.99	35.62	6150	28.856
FM	3.62	94.81	**55.39**	**42.57**	6150	2230.661
GFM	3.61	94.51	55.18	42.44	6150	2293.945

Table 4: Empirical results on 4 benchmark data sets. PCC is used with different inference methods: MM - estimates marginal modes, JM - the joint mode, FM - approximates the F-measure maximizer by assuming independence of labels, and GFM - estimates the exact F-measure maximizer over the conditional distribution obtained from the model. For MM, FM, and GFM, we sample 1000 observations from the conditional joint distribution for each test example. Results are reported for Hamming loss, subset 0/1 loss, F-measure and Jaccard distance. The best results for a performance measure are marked in bold.

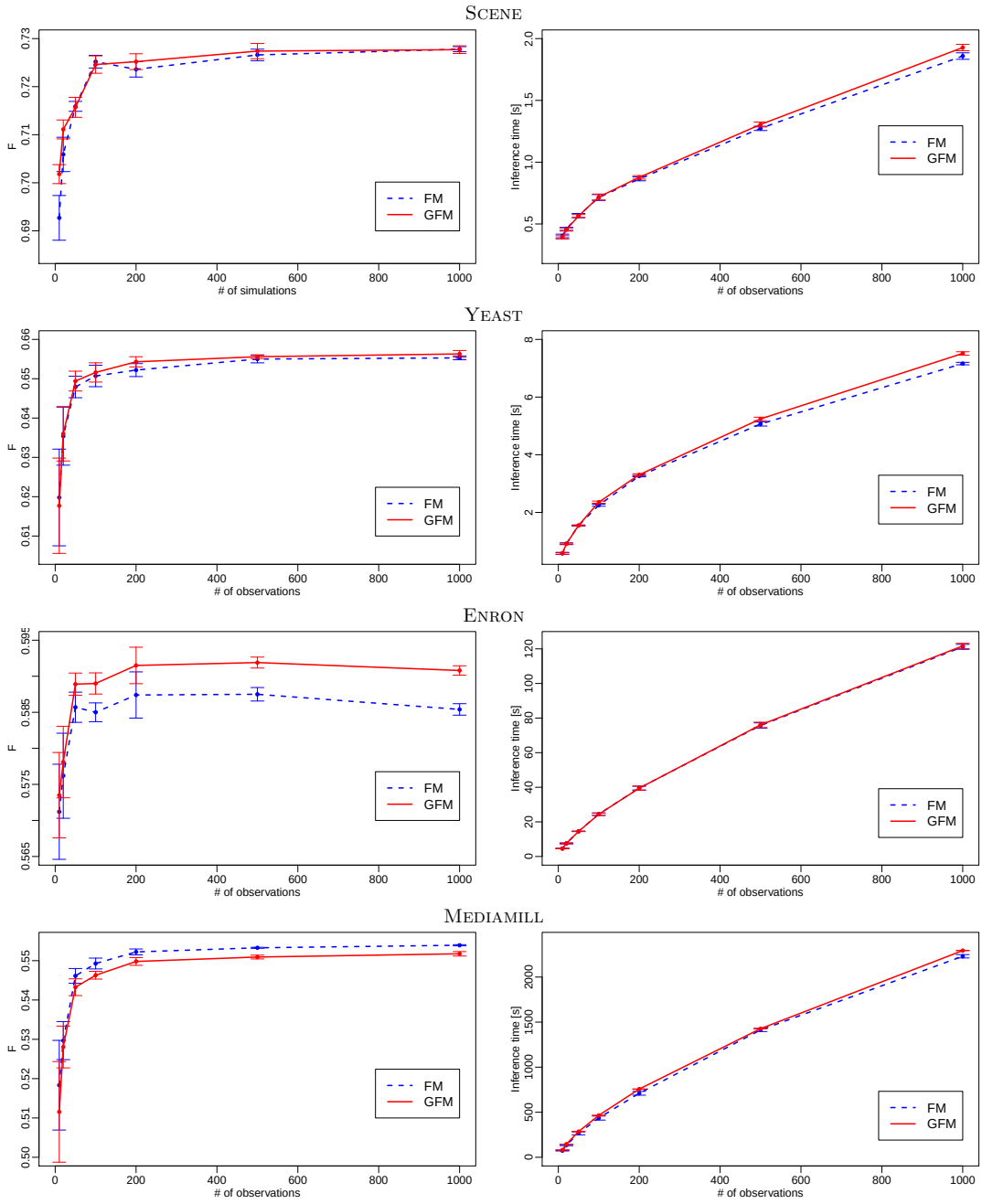


Figure 5: Performance of FM and GFM inference methods used in PCC with different number of observations obtained from ancestral sampling. The results are averaged over 5 runs of the inference methods to eliminate the randomness of sampling. Left plots show F-measure, while right plots inference times. The inference time includes sampling, estimation of parameters, and inference based on these parameters. The error bars show the standard error of the measured quantities.

	Hamming Loss [%]	F-Measure [%]	Training Time [s]	Inference Time [s]
		Scene		
BR	**10.51**	55.73	29	0.241
LFP	12.18	74.38	29	0.270
EFP	12.22	**74.44**	72	0.399
		Yeast		
BR	**20.03**	60.59	26	0.128
LFP	22.24	65.02	26	0.146
EFP	22.82	**65.47**	101	0.367
		Enron		
BR	**4.54**	55.49	52	1.016
LFP	6.09	56.86	52	1.519
EFP	5.34	**61.04**	214	2.628
		Mediamill		
BR	**3.19**	51.21	3238	13
LFP	3.67	55.15	3238	20
EFP	3.63	**55.16**	24620	30

Table 5: Empirical results on 4 benchmark data sets of parametric models: BR, LFP, and EFP. We report the Hamming loss, F-measure and training and inference times in seconds. The best results for a performance measure are marked in bold.

7.2.4 Comparison with Structured Support Vector Machines

In this subsection we gather results of all F-measure maximization methods presented so far and compare them with the results of a variant of structured SVMs (Tsochantaridis et al., 2005) in which the effort of maximizing the F-measure is moved to the training phase. Two methods of that kind, referred to as RML and SML, have been introduced by Petterson and Caetano (2010, 2011). We present here only the results of RML, previously published in (Dembczyński et al., 2013).[10] Basically, this method trains one model for each label, but in a way that the margin, appropriately rescaled by the F-measure, is maximized jointly over all labels. RML uses a variant of the cutting-plane algorithm for optimization. So, in each iteration a most violating constraint for each training example is generated. This step has quadratic complexity in terms of the number of labels. In the prediction phase the models are independently applied to corresponding labels. Surprisingly, this method produces usually better results than the more complex SML method (Petterson and Caetano, 2011), which additionally models pairwise label dependencies.[11] In the experiment the RML method uses a linear model. The regularization parameter is tuned in 5-fold cross-validation using a range of values corresponding to the one used for PCC, EFP, and LFP. The maximal number of iterations in the cutting-plane algorithm has been set to 1000.

10. The results were obtained by using the software available at `http://users.cecs.anu.edu.au/~jpetterson/`.
11. The comparison of these two methods is given in (Petterson and Caetano, 2011) and Dembczyński et al. (2013).

Table 6 present the results for F-measure, training and inference times. For instance-based methods we report the variant with the number of neighbors which achieves the best results for a given data set. Similarly for PCC, we take the number of sampled observations which leads to the best performance. Observing the results we can say that in general the methods that maximize the F-measure in the inference phase outperform the structured SVM approach. RML is only competitive on the Scene data set, on which it wins against instance-based methods and PCC, but loses from EFP and LFP. On Yeast and Mediamill it achieves the worst performance. On the latter data set the difference is the most substantial. On Enron it wins only against the instance-based method. As we already pointed out, on this data set all variants of nearest neighbors perform weakly, probably because of the high-dimensional feature space.

The comparison of the running times between RML and the other methods should be interpreted with caution, due to the use of different programming languages (RML is implemented in C++, while the other algorithms in Java) and differences in the implementations (different data structures). Therefore, the evaluation times may not be fully comparable. For example, the inference times for BR (Table 5) and RML should basically be very similar, as in both cases there is a single linear model for each label. Yet, the implementation of RML is much more efficient. Nevertheless, we are still able to derive several important conclusions.

Not surprisingly RML is the most efficient in inference. However, the cutting-plane algorithm and the constraint generation step therein slow down significantly the training of RML. This also makes tuning very costly. For PCC, EFP and LFP, the tuning can be performed independently for each base classifier. In that way we hope to obtain a good probabilistic model and there is no need to perform neither a costly training nor inference. Unfortunately, this is not the case of RML. For example, tuning on the Mediamill data set has not finished in reasonable time. The F-measure result reported in the table is the best one among those obtained on the test set for different values of the regularization parameter.

Each of the methods maximizing the F-measure in the inference phase has its advantages and disadvantages. By comparing the results, we see that parametric models get the best results on Scene and Enron followed by PCC. On Yeast and Mediamill all the methods perform very similarly, but the best is the instance-based learning here. Learning of EFP is the most costly. PCC and LFP train a single model for each label, but in the case of the former algorithm the feature space is enhanced by the preceding labels, therefore, the training time is longer for this procedure. We do not report training time for instance-based methods, as in the simplest case there is no learning in this kind of methods. In general, however, one should consider the time needed for tuning the number of neighbors and optionally learning the metric, which we have not performed here. From the inference point of view, LFP seems to be the most efficient. As we already discussed in the previous subsection, EFP is still quite competitive in comparison with LFP, but for data sets with a large number of labels this difference shall be more substantial. Instance-based methods are also very efficient for data sets with a small number of training examples. Because of the sampling procedure applied for each testing examples, PCC is the most time demanding procedure. However, we can see from Figure 5 that the good performance with respect to the F-measure can be obtained with a smaller number of sampled observations. In many

applications a set of 100 observations should be sufficient, resulting in a sample generation that is approximately 10 times faster. The main advantage of PCC is that once we have a trained model we can apply inference for many different performance measures without any additional training.

	F-Measure [%]	Training Time [s]	Inference Time [s]
		Scene	
IB FM ($l = 10$)	70.80	-	3.746
IB GFM ($l = 10$)	71.42	-	3.749
PCC FM ($n = 1000$)	72.78	39	1.858
PCC GFM ($n = 1000$)	72.77	39	1.927
LFP	74.38	**29**	0.270
EFP	**74.44**	72	0.399
RML	73.92	73	**0.118**
		Yeast	
IB FM ($l = 10$)	65.29	-	1.518
IB GFM ($l = 50$)	**65.75**	-	1.795
PCC FM ($n = 1000$)	65.53	39	7.161
PCC GFM ($n = 1000$)	65.63	39	7.513
LFP	65.02	**26**	0.146
EFP	65.47	101	0.367
RML	64.78	206	**0.056**
		Enron	
IB FM ($l = 10$)	47.50	-	0.787
IB GFM ($l = 10$)	44.43	-	0.810
PCC FM ($n = 500$)	58.75	81	75.677
PCC GFM ($n = 500$)	59.19	81	76.056
LFP	56.86	**52**	1.519
EFP	**61.04**	214	2.628
RML	57.69	3897	**0.143**
		Mediamill	
IB FM ($l = 20$)	55.65	-	164
IB GFM ($l = 20$)	**55.80**	-	167
PCC FM ($n = 1000$)	55.39	6150	2230
PCC GFM ($n = 1000$)	55.18	6150	2293
LFP	55.15	**3238**	20
EFP	55.16	24620	30
RML	49.35	-	**7**

Table 6: Comparison of RML, a variant of structured SVMs for F-measure maximization, with other F-measure maximization methods given in the previous tables: instance-based methods (IB), PCC, LFP and EFP. We present here the best variant of instance-based methods and PCC for a given data set. The number l of nearest neighbors for the best variant of IB is given in parentheses. Similarly the number of sampled observations in PCC is also given in parentheses. The F-measure, training and inferences time in seconds are reported. The best results are marked in bold.

7.3 Results in the JRS 2012 Data Mining Competition

We used the algorithms maximizing the F-measure in the inference phase in our solution for the JRS 2012 Data Mining Competition (Janusz et al., 2012).[12] This competition considered topical classification of bio-medical articles. In essence, it consisted of a multi-label learning problem, where the objective was to optimize the instance-based F-measure. We decided to participate in this competition to showcase the practical relevance of our theoretical findings regarding the F-measure maximization. Similar to many of our competitors, our final predictions in the competition were produced by a blend of several methods, and they achieved a very satisfactory result, namely the second place in the competition with more than 100 participants. In this paragraph, we briefly explain the methodology that led to this result.

Our solution was mainly based on PCC with FM and GFM inference methods and the LFP algorithm. The methods were tuned and run in a similar way as described in the previous experiments (with small differences: we used 10-fold cross validation, and considered a wider range for the regularization parameter, namely $\{10^{-5}, 10^{-4}, \ldots, 10^{5}\}$). At this time the EFP algorithm was not yet developed. We used neither instance-based nor decision tree methods. Our preprocessing on the competition data was quite straightforward. We simply deleted all the empty columns (i.e., zero vectors) in the training data, then the corresponding columns in the test data. The values of features were normalized to $[0, 1]$.

The results of the methods are presented in Table 7. The F-measure is computed over the entire test set delivered by the organizers after the competition. This is a minor difference in comparison to the competition results, which are computed over 90% of test examples. The remaining 10% of test examples constitute a validation set that served for computing the scores for the leader board during the competition. The results of PCC we show for different sizes of samples generated from the conditional joint distribution of a given test example. In the last row in the table, we also give a result of the final method we used in the competition. It relies on averaging over all predictions we computed during the competition. These predictions were results of different parameterization of PCC and BR. In total we gathered 16 predictions that we aggregated via voting. In this voting procedure, we tested different thresholds on the validation set and selected the best one. The solution is described in more detail in (Cheng et al., 2012).

From the results we can see that there is no big difference among the methods. The voting procedure improves only slightly over LFP and PCC with the GFM inference. The results of these methods would be enough to obtain at least the third place in the competition. It shows that a quite simple model, without any blending, but with an appropriate inference method suited for a given performance measure is enough for solving complex tasks. Interestingly, LFP performs here better than PCC with GFM, which suggests independence of the labels. However, one can also observe that PCC with FM loses against other methods. This may suggest that PCC with the sampling procedure has problems with accurate estimation of marginal probabilities. Increasing the sample size improves the results (for both, FM and GFM), but it still seems that LFP is the most appropriate method in this case. It is the most cheapest one, since it does not require additional sampling in the inference step as PCC does, and gives results only slightly worse than the voting method

12. More info can be found at: `http://tunedit.org/challenge/JRS12Contest`.

Method	F-measure
PCC FM ($n = 50$)	0.48650
PCC FM ($n = 200$)	0.51979
PCC FM ($n = 1000$)	0.52995
PCC GFM ($n = 50$)	0.52286
PCC GFM ($n = 200$)	0.53005
PCC GFM ($n = 1000$)	0.53146
LFP	0.53279
Voting (final submission)	0.53327

Table 7: The results on the JRS 2012 Competition data set. The number n in parentheses denotes the number of sampled observations in PCC.

that averages over many predictions. As we already said before, there is no clear answer which of the two inference methods, GFM or FM, will get better results on a given data set. GFM provides an exact solution, but needs to estimate more parameters, so FM may get better results, particularly in the case of no or weak label dependencies.

8. Discussion

In contrast to other performance measures commonly used in experimental studies, such as error rate, squared loss, and AUC, the F-measure has been investigated less thoroughly from a theoretical point of view, and only few papers have been devoted to that kind of analysis so far (e.g. Lewis (1995); Chai (2005); Jansche (2007); Dembczyński et al. (2011); Ye et al. (2012); Zhao et al. (2013)). In this paper, we analyzed the problem of optimal predictive inference from the joint distribution under the F-measure. While partial results were already known from the literature, we completed the picture by presenting the solution for the general case without any distributional assumptions and by analyzing the relations between F and other performance measures. Our GFM algorithm requires only a polynomial number of parameters of the joint distribution and delivers the exact solution in polynomial time. From a theoretical perspective, GFM should be preferred to existing approaches, which typically perform threshold maximization on marginal probabilities, often relying on the assumption of (conditional) independence of labels. Focusing on optimizing the instance-wise F-measure, empirical results on synthetic and real-world multi-label data sets show a competitive performance for our approach.

The algorithms discussed here optimize the F-measure in the inference phase. Alternatively, one can move the effort of maximizing the F-measure to the training phase, as in structured SVMs (Tsochantaridis et al., 2005), SEARN (Daumé III et al., 2009), or in a specific variant of CRFs (Suzuki et al., 2006). These algorithms, however, are usually based on additional assumptions, and their original formulation does not directly concern multi-label problems. In the experiments, we performed a comparison to the adaptation of structured SVMs to the multi-label setting introduced by Petterson and Caetano (2010,

2011). This algorithm also maximizes the F-measure, but produces worse results than the approaches based on the GFM inference. However, its prediction time is much faster, giving an interesting alternative in time-critical applications.

Let us also mention that the GFM algorithm can be easily tailored for maximizing the instance-wise F-measure in structured output prediction problems. If the structured output classifier is able to model the joint distribution, from which we can easily sample observations, then the use of the algorithm is straight-forward. An application of this kind is planned as future work.

The GFM algorithm could also be considered for maximizing the macro F-measure, for example, in a similar setting as in (Zhang et al., 2010), where a specific Bayesian on-line model is used. In order to maximize the macro F-measure, the authors sample from the graphical model to find an optimal threshold. The GFM algorithm may solve this problem optimally, since, as stated by the authors, the independence of labels is lost after integrating out the model parameters. Theoretically, one may also consider a direct maximization of the micro F-measure with GFM, but the computational burden is rather high in this case. We would also like to emphasize that maximization of instance-based F-measure leads to suboptimal results for the micro F-measure. Despite being related to each other, these two measures coincide only in a specific case when $\sum_{i=1}^{m}(y_i+h_i)$ is constant for all test examples. The discrepancy between these measures strongly depends on the nature of the data and the classifier used. For high variability in $\sum_{i=1}^{m}(y_i + h_i)$, a significant difference between the values of these two measures is to be expected. Surprisingly, experimental results are quite often reported in terms of micro F-measure, although the algorithms maximize the instance-wise F-measure on the training set.

The use of the GFM algorithm in binary classification seems to be superfluous, since in this case, the assumption of label independence is rather reasonable. The algorithm of Ye et al. (2012) seems to be an interesting alternative for probabilistic classifiers. Thresholding methods (Keerthi et al., 2007; Ye et al., 2012; Zhao et al., 2013) or learning algorithms optimizing the F-measure directly (Musicant et al., 2003; Joachims, 2005; Jansche, 2005; Ye et al., 2012) are also appropriate solutions here.

Acknowledgments

We thank Volkmar Welker for carefully checking the proofs of some theorems. Willem Waegeman was for this work supported as a postdoctoral fellow by the Research Foundation of Flanders (FWO Vlaanderen). Krzysztof Dembczyński and Arkadiusz Jachnik were supported by the Foundation of Polish Science under the Homing Plus programme, co-financed by the European Regional Development Fund. Weiwei Cheng and Eyke Hüllermeier were supported by the German Research Foundation.

Appendix A. Proofs of Theorems

Theorem 1 *Let $\boldsymbol{h}_H$ be a vector of predictions obtained by minimizing the Hamming loss, Then for $m > 2$ the worst-case regret is given by:*

$$\sup_{P \in \mathcal{P}^u_{L_H}} \left(\mathbb{E}\big[F(\boldsymbol{Y}, \boldsymbol{h}_F) - F(\boldsymbol{Y}, \boldsymbol{h}_H)\big] \right) = 0.5\,,$$

where the supremum is taken over all possible distributions P that result in a unique Hamming loss minimizer.

Proof. For a fixed Hamming loss minimizer $\boldsymbol{h}_H$ it follows from (8) that any probability distribution $P \in \mathcal{P}^u_{L_H}$ should satisfy the following constraint for all $i \in \{1, ..., m\}$:

$$\sum_{\boldsymbol{y} \in \{0,1\}^m : y_i \neq h_{H,i}} P(\boldsymbol{y}) \leq 0.5 - \epsilon$$

with $\epsilon > 0$. Practically, we will choose ϵ arbitrarily close to zero, implying that its contribution vanishes in the limit, but this construction allows to rewrite the constraint in traditional mathematical programming form. Let us also define

$$\eta_{\boldsymbol{y}}(\boldsymbol{h}, \boldsymbol{h}_H) = F(\boldsymbol{y}, \boldsymbol{h}) - F(\boldsymbol{y}, \boldsymbol{h}_H)$$

for all $\boldsymbol{y} \in \{0,1\}^m$. Finding the supremum over all probability distributions then becomes equivalent to solving the following mixed integer nonlinear program:

$$\max_{\boldsymbol{h}, \boldsymbol{h}_H, P \in \mathcal{P}^u_{L_H}} \sum_{\boldsymbol{y} \in \{0,1\}^m} \eta_{\boldsymbol{y}}(\boldsymbol{h}, \boldsymbol{h}_H) P(\boldsymbol{y}) \tag{23}$$

$$\text{subject to} \begin{cases} \sum_{\boldsymbol{y} \in \{0,1\}^m} P(\boldsymbol{y}) = 1\,, \\ \forall i \in \{1, ..., m\} : \sum_{\boldsymbol{y} \in \{0,1\}^m : y_i \neq h_{H,i}} P(\boldsymbol{y}) \leq 0.5 - \epsilon\,, \\ \forall \boldsymbol{y} \in \{0,1\}^m : 0 \leq P(\boldsymbol{y}) \leq 1\,, \\ \boldsymbol{h}, \boldsymbol{h}_H \in \{0,1\}^m \end{cases}$$

By definition, the solution $\boldsymbol{h}$ for which the maximum is obtained corresponds to the F-measure maximizer in (2).

To reduce the number of integer variables in the optimization problem, we introduce the following equivalence classes for the indices of labels:

$$\begin{aligned} A &= \{i \in \{1, ..., m\} : h_i = 1 \wedge h_{H,i} = 0\}\,, \\ B &= \{i \in \{1, ..., m\} : h_i = 0 \wedge h_{H,i} = 1\}\,, \\ C &= \{i \in \{1, ..., m\} : h_i = 1 \wedge h_{H,i} = 1\}\,, \\ D &= \{i \in \{1, ..., m\} : h_i = 0 \wedge h_{H,i} = 0\}\,. \end{aligned}$$

We also adopt the shorthand notation $a = |A|, b = |B|, c = |C|, d = |D|$ and

$$s_{\boldsymbol{y}} = \sum_{i=1}^{m} y_i\,, \quad s^A_{\boldsymbol{y}} = \sum_{i \in A} y_i\,, \quad s^B_{\boldsymbol{y}} = \sum_{i \in B} y_i\,, \quad s^C_{\boldsymbol{y}} = \sum_{i \in C} y_i\,, \quad s^D_{\boldsymbol{y}} = \sum_{i \in D} y_i\,. \tag{24}$$

The coefficients in (23) can then be simplified to:

$$\eta_{\boldsymbol{y}}(\boldsymbol{h}, \boldsymbol{h}_H) = \eta_{\boldsymbol{y}}(a,b,c,d) = \frac{2s^A_{\boldsymbol{y}}(s_{\boldsymbol{y}}+b+c) - 2s^B_{\boldsymbol{y}}(s_{\boldsymbol{y}}+a+c) + 2s^C_{\boldsymbol{y}}(b-a)}{(s_{\boldsymbol{y}}+a+c)(s_{\boldsymbol{y}}+b+c)}\,. \tag{25}$$

As a consequence, only four integer variables remain present in the optimization problem, which is for simplicity converted to a minimization problem in standard mixed-integer linear program form:

$$\min_{a,b,c,d,P} -\sum_{\boldsymbol{y}\in\{0,1\}^m} \eta_{\boldsymbol{y}}(a,b,c,d)P(\boldsymbol{y})$$

$$\text{subject to} \begin{cases} \sum_{\boldsymbol{y}\in\{0,1\}^m} P(\boldsymbol{y}) = 1\,, \\ \forall i \in \{1,...,m\} : \sum_{\boldsymbol{y}\in\{0,1\}^m : y_i \neq h_{H,i}} P(\boldsymbol{y}) \leq 0.5-\epsilon\,, \\ \forall \boldsymbol{y} \in \{0,1\}^m : 0 \leq P(\boldsymbol{y}) \leq 1\,, \\ a+b+c+d = m\,, \\ a,b,c,d \in \mathbb{N}\,. \end{cases}$$

This new optimization problem is a relaxation of (23), since the F-maximizer of the probability distribution found as solution will not necessarily comply with the definition of the sets A, B, C and D. However, this will not cause any trouble, because the oracle solution that is derived below will obey this additional constraint.

One arrives at a standard linear program formulation by keeping the four integer variables fixed. As the key element of our proof, we show that for every allowed value of (a,b,c,d) a solution of the linear program is given by the following probability distribution:

$$P_A(\boldsymbol{y}) \;=\; \begin{cases} 0.5-\epsilon & \text{if } \boldsymbol{y} = \boldsymbol{y}^A \\ 0.5-(2d-1)\epsilon & \text{if } \boldsymbol{y} = \boldsymbol{y}^{BCD} \\ 2\epsilon & \text{if } \boldsymbol{y} \in \Omega^D_m \\ 0 & \text{otherwise} \end{cases}\,,$$

where $\boldsymbol{y}^A$ is defined as a vector containing ones at positions $i \in A$ and zeros at all other positions. Similarly, $\boldsymbol{y}^{BCD}$ contains zeros at positions $i \in A$ and ones at all other positions:

$$\begin{aligned} y^A_i &= 1 + y^{BCD}_i = 1, \qquad \forall i \in A\,, \\ y^A_i &= 1 - y^{BCD}_i = 0, \qquad \forall i \in B \cup C \cup D\,. \end{aligned}$$

The set Ω^D_m is defined as:

$$\Omega^D_m = \{\boldsymbol{y} \in \{0,1\}^m \mid \sum_{i\in A} y_i = 0 \wedge \sum_{i\in B\cup C} y_i = b+c \wedge \sum_{i\in D} y_i = d-1\}\,,$$

so this set contains d vectors, which differ only in one position with $\boldsymbol{y}^{BCD}$.

We verify the Karush-Kuhn-Tucker (KKT) conditions to prove that the above probability distribution yields the optimum of the linear program for every (a,b,c,d). For

linear programs, which represent a specific case of optimizing an invex function, the KKT-conditions are not only necessary but also sufficient for optimality (Hanson, 1981). Let us define the primal Lagrangian as:

$$\begin{aligned}\mathfrak{L}_p &= -\sum_{\boldsymbol{y}\in\{0,1\}^m} \eta_{\boldsymbol{y}}(a,b,c,d)P(\boldsymbol{y}) + \nu \sum_{\boldsymbol{y}\in\{0,1\}^m} \big(P(\boldsymbol{y})-1\big) \\ &\quad + \sum_{i=1}^{m} \mu_i \big[\sum_{\boldsymbol{y}\in\{0,1\}^m : y_i \neq h_{H,i}} P(\boldsymbol{y}) - 0.5 + \epsilon \big] \\ &\quad - \sum_{\boldsymbol{y}\in\{0,1\}^m} \lambda_y^- P(\boldsymbol{y}) + \sum_{\boldsymbol{y}\in\{0,1\}^m} \lambda_{\boldsymbol{y}}^+ (P(\boldsymbol{y})-1)\,.\end{aligned}$$

with $\nu, \mu_i, \lambda_{\boldsymbol{y}}^+$ and $\lambda_{\boldsymbol{y}}^-$ Lagrange multipliers. For the above-mentioned probability distribution, the complementary slackness conditions imply that $\lambda_{\boldsymbol{y}}^+ = 0$ for all $\boldsymbol{y} \in \{0,1\}^m$ and $\lambda_{\boldsymbol{y}}^- = 0$ for all $\boldsymbol{y}$ contained in $\Omega_m^D \cup \{\boldsymbol{y}^A, \boldsymbol{y}^{BCD}\}$. Hence, the zero-gradient condition results in the following system of equations:

$$\begin{aligned}\eta_{\boldsymbol{y}}(a,b,c,d) &= \nu + \sum_{i:y_i \neq h_{H,i}} \mu_i\,, \qquad \forall \boldsymbol{y} \in \Omega_m^D \cup \{\boldsymbol{y}^A, \boldsymbol{y}^{BCD}\}\,, \\ \eta_{\boldsymbol{y}}(a,b,c,d) &= \nu + \sum_{i:y_i \neq h_{H,i}} \mu_i - \lambda_{\boldsymbol{y}}^-\,, \qquad \forall \boldsymbol{y} \notin \Omega_m^D \cup \{\boldsymbol{y}^A, \boldsymbol{y}^{BCD}\}\,.\end{aligned}$$

First, we show that a solution always exists for this system, and additionally, we check the dual feasibility conditions $\mu_i \geq 0, \lambda_{\boldsymbol{y}}^+ \geq 0$ and $\lambda_{\boldsymbol{y}}^- \geq 0$. For all $\boldsymbol{y} \notin \Omega_m^D \cup \{\boldsymbol{y}^A, \boldsymbol{y}^{BCD}\}$, the equations have an individual variable $\lambda_{\boldsymbol{y}}^-$, which only occurs in one equation, so these equations do not impose any further restrictions, apart from the non-negativity constraint on the respective Lagrange multipliers. Furthermore, since the sum over μ_i is always positive, we have enough degrees of freedom to ignore those equations. For the other equations, one arrives at a new system of equations by solving for ν:

$$\begin{aligned}\sum_{i:y_i \neq h_{H,i}} \mu_i - \sum_{j:y'_j \neq h_{H,j}} \mu_j &= \eta_{\boldsymbol{y}}(a,b,c,d) - \eta_{\boldsymbol{y}'}(a,b,c,d)\,, \\ &\qquad \forall \boldsymbol{y}, \boldsymbol{y}' \in \Omega_m^D \cup \{\boldsymbol{y}^A, \boldsymbol{y}^{BCD}\}\,.\end{aligned}$$

We continue by writing out this system of equations more explicitly, resulting in four cases. For the pair corresponding to $\boldsymbol{y} = \boldsymbol{y}^A$ and $\boldsymbol{y}' = \boldsymbol{y}^{BCD}$ we obtain

$$\sum_{i \in A \cup B \cup C} \mu_i - \sum_{i \in D} \mu_i = \eta_{\boldsymbol{y}}(a,b,c,d) - \eta_{\boldsymbol{y}'}(a,b,c,d)\,. \tag{26}$$

For all pairs having $\boldsymbol{y} = \boldsymbol{y}^A$ and $\boldsymbol{y}' \in \Omega_m^D$, so d pairs in total with $j \in D$, we obtain

$$\sum_{i \in A \cup B \cup C} \mu_i - \sum_{i \in D \setminus \{j\}} \mu_i = \eta_{\boldsymbol{y}}(a,b,c,d) - \eta_{\boldsymbol{y}'}(a,b,c,d)\,. \tag{27}$$

For all pairs having $\boldsymbol{y} = \boldsymbol{y}^{BCD}$ and $\boldsymbol{y}' \in \Omega_m^D$, so again d pairs with $i \in D$, we obtain

$$\mu_i = \eta_{\boldsymbol{y}}(a,b,c,d) - \eta_{\boldsymbol{y}'}(a,b,c,d)\,. \tag{28}$$

For all pairs having $\boldsymbol{y}, \boldsymbol{y}' \in \Omega_m^D$, so $d(d-1)$ pairs with $i, j \in D$, we obtain

$$\mu_i = \mu_j \,. \tag{29}$$

Let us observe that the d equations (27) and the d equations (28) are equivalent due to (26). Moreover, the $d(d-1)$ equations in (29) are trivially satisfied, resulting in a system that can be reduced to $d+1$ equations and m variables. Hence, a solution always exists if the dual feasibility conditions are satisfied. Plugging (25) into (28) yields for $i \in D$:

$$\begin{aligned}
\mu_i &= \frac{2c(b-a)}{(m+c)(2b+2c+d)} - \frac{2b}{2b+2c+d} \\
&\qquad - \frac{2c(b-a)}{(m+c-1)(2b+2c+d-1)} + \frac{2b}{2b+2c+d-1} \\
&\geq \frac{2cb(m-2)}{(m+c)(m-1+c)(2b+2c+d)} \,.
\end{aligned}$$

So, $\mu_i \geq 0$ as soon as $m > 1$. Due to the denominator it is also required that $2b+2c+d > 1$. For the μ_i having $i \notin D$ we find:

$$\begin{aligned}
\sum_{i \in A \cup B \cup C} \mu_i &\geq \eta_{\boldsymbol{y}}(a,b,c,d) - \eta_{\boldsymbol{y}'}(a,b,c,d) \\
&= \frac{2a}{2a+c} + \frac{2b}{2b+2c+d} \\
&\qquad - \frac{2c(b-a)}{(m+c)(2b+2c+d)} \\
&\geq \frac{2mb}{(m+c)(2b+2c+d)} \geq 0 \,.
\end{aligned}$$

In other words, these are the conditions that remain for the μ_i for which $i \in D$. From the inequality it follows that the individual μ_i can be made greater than zero in this case, implying that also the active Lagrange multipliers $\lambda_{\boldsymbol{y}}^-$ can be chosen in such a way that they are greater than zero.

Consequently, when $m > 1$, all KKT conditions are satisfied for the oracle solution that we provide. Let us compare the value of the objective function for all allowed values of the four integer variables a, b, c and d. By omitting ϵ-dependent terms, which vanish when ϵ approaches zero, we obtain:

$$\begin{aligned}
\sum_{\boldsymbol{y} \in \Omega_m^D \cup \{\boldsymbol{y}^A, \boldsymbol{y}^{BCD}\}} \eta_{\boldsymbol{y}}(a,b,c,d) P(\boldsymbol{y}) &= \frac{a}{(2a+c)} - \frac{ca}{(m+c)(2b+2c+d)} \\
&\qquad - \frac{mb}{(m+c)(2b+2c+d)} \,.
\end{aligned}$$

This function is decreasing in b and c, it is constant in a as soon as $c = 0$ and $a > 0$. It is constant in d when $b + c = 0$, thus its maximum of 0.5 is not unique. This maximum is for example obtained for $(a = 1, b = 0, c = 0, d = m-1)$ and it corresponds to the worst-case regret mentioned in the theorem. Remark as well that the solution imposes as additional constraint $m > 2$, as a result of the previous constraint $2b + 2c + d > 1$. Two cases were

excluded from our analysis: $(a = m, b = 0, c = 0, d = 0)$ and $(a = m-1, b = 0, c = 0, d = 1)$. These cases do not deserve further attention, since they lead to a worst-case regret that is always upper bounded by 0.5. ■

Theorem 2 *Let $\boldsymbol{h}_s$ be a vector of predictions obtained by minimizing the subset 0/1 loss, then for $m > 2$ the worst-case regret is given by:*

$$\sup_{P \in \mathcal{P}^u_{L_s}} \left(\mathbb{E}\big[F(\boldsymbol{Y}, \boldsymbol{h}_F) - F(\boldsymbol{Y}, \boldsymbol{h}_s)\big] \right) = \frac{(-m - 2 + 2m^2)m}{(2m-1)(4+m+m^2)} \,,$$

where the supremum is taken over all possible distributions P.

Proof. It follows from (9) that optimization problem (4) has a unique solution for the subset zero-one loss if and only if the underlying probability distribution has a unique mode. Translating this requirement into a mathematical programming formulation implies that any probability distribution $P \in \mathcal{P}^u_{L_s}$ should satisfy the following constraint:

$$P(\boldsymbol{y}) + \epsilon \leq P(\boldsymbol{h}_s) \,,$$

for all $\boldsymbol{y} \in \{0,1\}^m \setminus \boldsymbol{h}_s$ and any $\epsilon > 0$. Practically, the contribution of ϵ will again vanish by choosing it arbitrarily close to zero.

As a result, the supremum can be interpreted as the solution of a mixed integer nonlinear program:

$$\max_{\boldsymbol{h}, \boldsymbol{h}_s, P} \sum_{\boldsymbol{y} \in \{0,1\}^m} (F(\boldsymbol{y}, \boldsymbol{h}) - F(\boldsymbol{y}, \boldsymbol{h}_s)) P(\boldsymbol{y}) \tag{30}$$

$$\text{subject to} \begin{cases} \sum_{\boldsymbol{y} \in \{0,1\}^m} P(\boldsymbol{y}) = 1 \,, \\ \forall \boldsymbol{y} \in \{0,1\}^m \setminus \boldsymbol{h}_s : P(\boldsymbol{y}) + \epsilon \leq P(\boldsymbol{h}_s) \,, \\ \forall \boldsymbol{y} \in \{0,1\}^m : 0 \leq P(\boldsymbol{y}) \leq 1 \,, \\ \boldsymbol{h}_s, \boldsymbol{h}_F \in \{0,1\}^m \quad \epsilon \geq 0 \,. \end{cases}$$

Recall that by construction the solution for $\boldsymbol{h}$ again coincides with the F-measure maximizer in (2). The mixed integer nonlinear program contains 2^{2m} integer variables and 2^m real variables. In what follows, we assume that ϵ is arbitrarily close to zero, so that all ϵ-dependent terms cancel out, while guaranteeing unique risk minimizers. The only consequence of this decision is that the presented solution acts as a supremum (the maximum is not reached).

Despite a similar formulation as the previous theorem, our proving techniques will be quite different, because, unlike the previous theorem, it is impossible to derive an oracle solution for the entire mixed integer nonlinear program. Alternatively, we will look for a solution of the optimization problem that emerges when ϵ-dependent terms become zero. The only practical problem with this solution is the non-uniqueness of the corresponding subset zero-one loss minimizer, but we will show that this solution can be approximated by an arbitrarily close solution with a unique subset zero-one loss minimizer. This technique will suffice to prove the theorem. However, let us remark that the same technique would not

work to prove Theorem 1, because there it would not be possible to approach the solution of the optimization problem without ϵ-terms by an ϵ-approximate extension with unique Hamming loss minimizer. This explains why the proving techniques of Theorem 1 and 2 are different.

We start by providing an oracle solution for the linear program that is obtained by fixing the integer variables to $\boldsymbol{h}_F = \mathbf{1}_m$ and $\boldsymbol{h}_s = \mathbf{0}_m$. The first part of the proof is a bit similar to a proof given for the regret of the subset 0/1 loss minimizer w.r.t. the Hamming loss (Dembczyński et al., 2012a). While omitting ϵ-dependent terms, optimization problem (30) can be reformulated in standard linear program form as:

$$\min_P \sum_{\boldsymbol{y}\in\{0,1\}^m} -\eta(\boldsymbol{y})P(\boldsymbol{y})$$

$$\text{subject to} \begin{cases} \sum_{\boldsymbol{y}\in\{0,1\}^m} P(\boldsymbol{y}) - 1 = 0\,, \\ \forall \boldsymbol{y} \in \{0,1\}^m \setminus \mathbf{0}_m : P(\boldsymbol{y}) - P(\mathbf{0}_m) \leq 0\,, \\ \forall \boldsymbol{y} \in \{0,1\}^m : -P(\boldsymbol{y}) \leq 0\,, \\ \forall \boldsymbol{y} \in \{0,1\}^m : P(\boldsymbol{y}) - 1 \leq 0\,, \end{cases}$$

with

$$\eta(\boldsymbol{y}) \;=\; \begin{cases} \frac{2s_{\boldsymbol{y}}}{s_{\boldsymbol{y}}+m} & \text{if } \boldsymbol{y} \neq \mathbf{0}_m\,, \\ -1 & \text{if } \boldsymbol{y} = \mathbf{0}_m\,. \end{cases}$$

In the next paragraphs we will show that the following probability distribution corresponds to the solution of the linear program:

$$P_A(\boldsymbol{y}) \;=\; \begin{cases} \frac{2}{m^2+m+4} & \text{if } d(\boldsymbol{y},\mathbf{0}_m) \geq m-2 \vee \boldsymbol{y} = \mathbf{0}_m\,, \\ 0 & \text{otherwise }, \end{cases}$$

where $d_H(\boldsymbol{y},\boldsymbol{y}') = \sum_{i=1}^m |y_i - {y'}_i|$ denotes the Hamming distance. This solution represents a case where the subset zero-one loss minimizer is not unique, but it could be easily extended to an ϵ-approximate solution with unique subset zero-one loss minimizer:

$$P_A(\boldsymbol{y}) \;=\; \begin{cases} \frac{2}{m^2+m+4} + \frac{m^2+m+2}{m^2+m+4}\epsilon & \text{if } \boldsymbol{y} = \mathbf{0}_m\,, \\ \frac{2}{m^2+m+4} - \frac{2}{m^2+m+4}\epsilon & \text{if } d(\boldsymbol{y},\mathbf{0}_m) \geq m-2\,, \\ 0 & \text{otherwise }, \end{cases}$$

We verify the KKT conditions to prove that the above probability distribution is indeed the solution of the optimization problem. The primal Lagrangian of the linear program can be defined as:

$$\begin{aligned} \mathfrak{L}_p \;=\; & -\sum_{\boldsymbol{y}\in\{0,1\}^m} \eta(\boldsymbol{y})P(\boldsymbol{y}) + \nu \sum_{\boldsymbol{y}\in\{0,1\}^m} \big(P(\boldsymbol{y}) - 1\big) + \sum_{\boldsymbol{y}\neq\mathbf{0}_m} \lambda^2_{\boldsymbol{y}}\big(P(\boldsymbol{y}) - P(\mathbf{0}_m)\big) \\ & - \sum_{\boldsymbol{y}\in\{0,1\}^m} \lambda^0_{\boldsymbol{y}} P(\boldsymbol{y}) + \sum_{\boldsymbol{y}\in\{0,1\}^m} \lambda^1_{\boldsymbol{y}} P(\boldsymbol{y})\,, \end{aligned}$$

with $\nu, \lambda_{\boldsymbol{y}}^0, \lambda_{\boldsymbol{y}}^1$ and $\lambda_{\boldsymbol{y}}^2$ Lagrange multipliers. The stationarity condition for optimality leads to the following system of linear equations:

$$-\eta(\boldsymbol{y}) + \nu + \lambda_{\boldsymbol{y}}^1 - \lambda_{\boldsymbol{y}}^0 + \lambda_{\boldsymbol{y}}^2 = 0 \qquad \forall \boldsymbol{y} \neq \mathbf{0}_m\,, \tag{31}$$

$$1 + \nu + \lambda_{\boldsymbol{y}}^1 - \lambda_{\boldsymbol{y}}^0 - \sum_{\boldsymbol{y} \neq \mathbf{0}_m} \lambda_{\boldsymbol{y}}^2 = 0 \qquad \boldsymbol{y} = \mathbf{0}_m \tag{32}$$

Other conditions that need to be satisfied are dual feasibility

$$\forall \boldsymbol{y}: \qquad \lambda_{\boldsymbol{y}}^0 \geq 0\,, \tag{33}$$

$$\forall \boldsymbol{y}: \qquad \lambda_{\boldsymbol{y}}^1 \geq 0\,, \tag{34}$$

$$\forall \boldsymbol{y}: \qquad \lambda_{\boldsymbol{y}}^2 \geq 0\,, \tag{35}$$

and the complementary slackness conditions of our oracle solution $P_A(\boldsymbol{y})$:

$$\begin{aligned} \forall \boldsymbol{y} \in \Omega^u \cup \{\mathbf{0}_m\}: \qquad & \lambda_{\boldsymbol{y}}^0 = 0\,, \\ \forall \boldsymbol{y}: \qquad & \lambda_{\boldsymbol{y}}^1 = 0\,, \\ \forall \boldsymbol{y} \notin \Omega^u: \qquad & \lambda_{\boldsymbol{y}}^2 = 0\,, \end{aligned}$$

where $\Omega^u = \Omega(m) \cup \Omega(m-1) \cup \Omega(m-2)$ and $\Omega(t) = \{\boldsymbol{y} \in \{0,1\}^m \mid d_H(\mathbf{0}_m, \boldsymbol{y}) = t\}$. Plugging the latter three conditions into (31) and (32) yields

$$\begin{aligned} &-\eta(\boldsymbol{y}) + v - \lambda_{\boldsymbol{y}}^0 = 0\,, \qquad && \forall \boldsymbol{y} \notin \Omega^u \cup \{\mathbf{0}_m\}\,, \\ &-\eta(\boldsymbol{y}) + v + \lambda_{\boldsymbol{y}}^2 = 0\,, \qquad && \forall \boldsymbol{y} \in \Omega^u\,, \\ &v = \textstyle\sum_{\boldsymbol{y} \neq \mathbf{0}_m} \lambda_{\boldsymbol{y}}^2 - 1\,. \end{aligned}$$

Solving the last equation for v results in

$$v = \frac{(-m - 2 + 2m^2)m}{(2m-1)(4+m+m^2)}\,.$$

Subsequently, one can verify that this solution for v obeys the non-negativity conditions for all $\lambda_{\boldsymbol{y}}$. The non-negativity of $\lambda_{\boldsymbol{y}}^2$ turns out to be most restrictive for the equivalence class $\boldsymbol{y} \in \Omega(m-2)$. In this case we obtain:

$$\lambda_{\boldsymbol{y}}^2 = \frac{2m-4}{2m-2} - v = \frac{2(3m^2 - 10m + 4)}{(m-1)(2m-1)(4+m+m^2)}\,. \tag{36}$$

Analyzing this function more thoroughly reveals that it is strictly positive in the interval $[+3, +\infty[$. Similarly, we find that the most restrictive condition on $\lambda_{\boldsymbol{y}}^0$ is obtained for the elements in the equivalence class $\boldsymbol{y} \in \Omega(m-3)$, leading to the following equality:

$$\lambda_{\boldsymbol{y}}^0 = v - \frac{2m-6}{2m-3} = \frac{-9m^2 + 2m^3 + 56m - 24}{(2m-3)(2m-1)(4+m+m^2)}\,. \tag{37}$$

This function is also strictly positive in the interval $[+3, +\infty[$, so all non-negativity conditions are satisfied. Plots of the functions are shown in Figure 6.

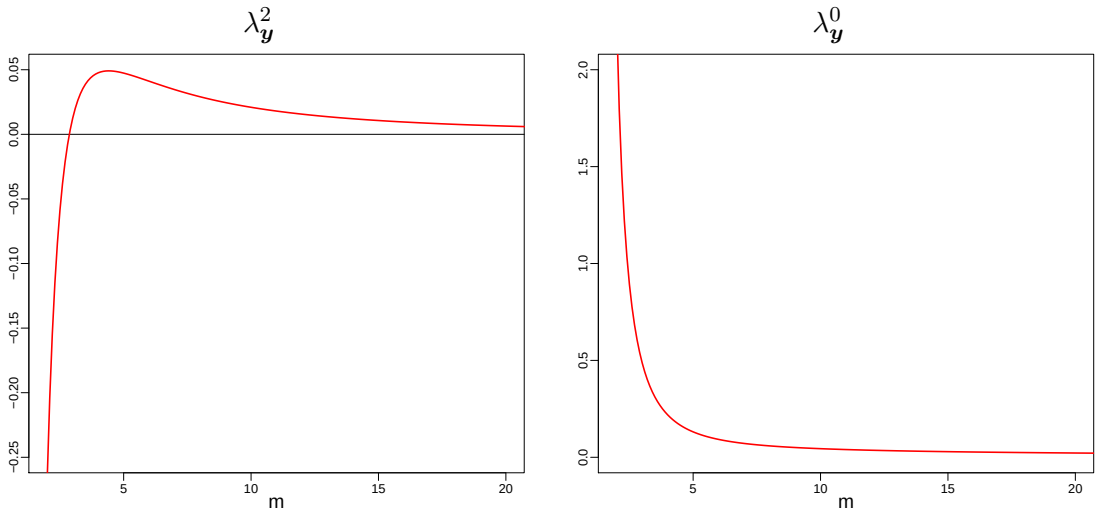


Figure 6: Plots of the functions $\lambda^2_{\boldsymbol{y}}$ and $\lambda^0_{\boldsymbol{y}}$ as defined by equations (36) and (37).

One can observe that $P_A(\boldsymbol{y})$ yields the regret mentioned in the theorem. Thus, what we found so far is a lower bound on the worst-case regret. The tightness of the bound is further proven by showing that the supremum is always obtained by $\boldsymbol{h}_s = \mathbf{0}_m$ and $\boldsymbol{h}_F = \mathbf{1}_m$ as soon as $m > 2$. Since it is impossible to enumerate all solutions for the 2^{2m} possible values of the integer variables, we analyse the properties of the objective function to prove that the optimum is obtained for $\boldsymbol{h}_s = \mathbf{0}_m$ and $\boldsymbol{h}_F = \mathbf{1}_m$. Similar to the previous theorem, let us introduce

$$\begin{aligned} A &= \{i \in \{1,...,m\} : h_i = 1 \wedge h_{s,i} = 0\}\,, \\ B &= \{i \in \{1,...,m\} : h_i = 0 \wedge h_{s,i} = 1\}\,, \\ C &= \{i \in \{1,...,m\} : h_i = 1 \wedge h_{s,i} = 1\}\,, \\ D &= \{i \in \{1,...,m\} : h_i = 0 \wedge h_{s,i} = 0\}\,. \end{aligned}$$

and the shorthand notations $a = |A|, b = |B|, c = |C|, d = |D|, s_{\boldsymbol{y}}, s^A_{\boldsymbol{y}}, s^B_{\boldsymbol{y}}, s^C_{\boldsymbol{y}}, s^D_{\boldsymbol{y}}$, as defined in (24).

Optimization problem (30) can then be relaxed to the following standard mixed-integer nonlinear program form:

$$\min_{a,b,c,d,P} - \sum_{\boldsymbol{y} \in \{0,1\}^m} \eta_{\boldsymbol{y}}(a,b,c,d) P(\boldsymbol{y})$$

$$\text{subject to} \begin{cases} \sum_{\boldsymbol{y} \in \{0,1\}^m} P(\boldsymbol{y}) = 1\,, \\ \forall \boldsymbol{y} \in \{0,1\}^m \setminus \boldsymbol{h}_s : P(\boldsymbol{y}) - P(\boldsymbol{h}_s) \leq 0\,, \\ \forall \boldsymbol{y} \in \{0,1\}^m : 0 \leq P(\boldsymbol{y}) \leq 1\,, \\ a + b + c + d = m\,, \\ a, b, c, d \in \mathbb{N}\,, \end{cases}$$

writing down the coefficients of the objective function as:

$$\eta_{\boldsymbol{y}}(\boldsymbol{h}, \boldsymbol{h}_s) = \eta_{\boldsymbol{y}}(a, b, c, d) = \frac{s^A_{\boldsymbol{y}} + s^C_{\boldsymbol{y}}}{s_{\boldsymbol{y}} + a + c} - \frac{s^B_{\boldsymbol{y}} + s^C_{\boldsymbol{y}}}{s_{\boldsymbol{y}} + b + c}\,.$$

Recall that the relaxation again originates from the fact that the solution not necessarily complies with the definitions of the sets A, B, C and D.

Now observe that the objective function of the mixed-integer nonlinear program is strictly decreasing in b, independent of the other variables, so we can fix $b = 0$. Subsequently, observe that for $b = 0$ the objective function is also strictly decreasing in c, independent of the other variables, so we also fix $c = 0$. As a result, the coefficients can be further simplified to:

$$\eta_{\boldsymbol{y}}(\boldsymbol{h}, \boldsymbol{h}_s) = \eta_{\boldsymbol{y}}(a, b, c, d) = \begin{cases} \frac{s^A_{\boldsymbol{y}}}{s_{\boldsymbol{y}}+a} & \text{if } \boldsymbol{y} \neq \mathbf{0}_m\,, \\ -1 & \text{if } \boldsymbol{y} = \mathbf{0}_m\,. \end{cases} \tag{38}$$

To complete the proof we show by contradiction that the optimum is obtained for $a = m$. In order to construct the recurrence equations below let us introduce $q \in \{1, ..., m-1\}$ and let

$$\begin{aligned} \Omega^0_q &= \{\boldsymbol{y} \in \{0,1\}^m \mid y_{q+1} = 0 \wedge \boldsymbol{y} \neq \mathbf{0}_m\}\,, \\ \Omega^1_q &= \{\boldsymbol{y} \in \{0,1\}^m \mid y_{q+1} = 1 \wedge \boldsymbol{y} \neq \mathbf{0}^1_{q+1}\}\,, \end{aligned}$$

where $\mathbf{0}^1_{q+1}$ denotes a vector of $m-1$ zeros apart from a single one at position $q+1$. Furthermore, let us introduce the mapping $\Psi_q : \{0,1\}^m \to \{0,1\}^m$, which, in any binary vector of length m, toggles the bit at position $q+1$: a zero at that position becomes a one and vice versa. The mapping Ψ_q hence defines a unique correspondence between any element in Ω^0_q and its sister element in Ω^1_q. For $a = q$, the objective function can then be written as:

$$\begin{aligned} \delta^q(P) = \sum_{\boldsymbol{y} \in \{0,1\}^m} \eta_{\boldsymbol{y}}(a, b, c, d) P(\boldsymbol{y}) &= -P(\mathbf{0}_m) \\ &\quad + \sum_{\boldsymbol{y} \in \Omega^0_q} \frac{2 s^A_{\boldsymbol{y}}}{s_{\boldsymbol{y}} + q} P(\boldsymbol{y}) + \sum_{\boldsymbol{y} \in \Omega^1_q} + \frac{2 s^A_{\boldsymbol{\Psi_q(y)}}}{s_{\boldsymbol{\Psi_q(y)}} + q + 1} P(\boldsymbol{y}) \end{aligned}$$

while for $a = q + 1$, it can be written as:

$$\begin{aligned} \delta^{q+1}(P) = \sum_{\boldsymbol{y} \in \{0,1\}^m} \eta_{\boldsymbol{y}}(a, b, c, d) P(\boldsymbol{y}) &= -P(\mathbf{0}_m) + \frac{2}{a+1} P(\mathbf{0}^1_{q+1}) \\ &\quad + \sum_{\boldsymbol{y} \in \Omega^0_q} \frac{2 s^A_{\boldsymbol{y}}}{s_{\boldsymbol{y}} + q + 1} P(\boldsymbol{y}) + \sum_{\boldsymbol{y} \in \Omega^1_q} + \frac{2 s^A_{\boldsymbol{\Psi_q(y)}} + 2}{s_{\boldsymbol{\Psi_q(y)}} + q + 2} P(\boldsymbol{y}) \end{aligned}$$

Let us assume that the global optimum is obtained for $a < m$. Furthermore, let $P^q(\boldsymbol{y})$ be the probability distribution that delivers this optimum for $(a = q, d = m - q)$. We construct a new probability distribution $P^{q+1}(\boldsymbol{y})$ as follows:

$$P^{q+1}(\boldsymbol{y}) = \begin{cases} P^q(\Psi_q(\boldsymbol{y})) & \text{if } (\boldsymbol{y} \in \Omega^1_q \wedge P^q(\boldsymbol{y}) > P^q(\Psi_q(\boldsymbol{y}))) \vee (\boldsymbol{y} \in \Omega^0_q \wedge P^q(\boldsymbol{y}) < P^q(\Psi_q(\boldsymbol{y})))\,, \\ P(\boldsymbol{y}) & \text{otherwise}\ , \end{cases}$$

If $P^q(\boldsymbol{y})$ is feasible, then $P^{q+1}(\boldsymbol{y})$ is feasible, too. It follows from

$$\frac{2s^A_{\boldsymbol{\Psi_q(y)}} + 2}{s_{\boldsymbol{\Psi_q(y)}} + q + 2} \geq \frac{2s^A_{\boldsymbol{\Psi_q(y)}}}{s_{\boldsymbol{\Psi_q(y)}} + q}\,, \quad \forall \boldsymbol{y} \in \{0,1\}^m$$

that $\delta^{q+1}(P^{q+1}) \geq \delta^q(P^q)$ for all q. This is a contradiction. Consequently, the global optimum of the optimization problem is given by P^A. ■

Side note about the difference in proving technique for Theorems 1 and 2. In this paragraph we give some additional explanation why the proving techniques for Theorems 1 and 2 are different. The proof of Theorem 2 cannot proceed with ϵ-terms because we are not able to derive an oracle solution for the mixed integer linear program when ϵ-terms are considered. However, for Theorem 2 this is not needed because we are able to find a probability distribution that has unique risk minimizers, while being arbitrarily close to the oracle solution that we present. The claim of the theorem then immediately follows from taking the limit when ϵ approaches zero. Let us consider the example of $m = 4$, for which we obtain the following probability distribution:

$\boldsymbol{y}$	$P(\boldsymbol{y})$
0000	1/12
1100	1/12
1010	1/12
1001	1/12
0110	1/12
0101	1/12
0011	1/12
1110	1/12
1011	1/12
1101	1/12
0111	1/12
1111	1/12

The subset zero-one loss minimizer is not unique in this case, but let us choose 0000 as subset zero-one loss minimizer. The F-measure maximizer is 1111. The regret should then be 13/24 - see fraction in Theorem 2. We can easily extend this to an ϵ-case with unique subset zero-one loss minimizer:

$\boldsymbol{y}$	$P(\boldsymbol{y})$
0000	$1/12 + 11\epsilon$
1100	$1/12 - \epsilon$
1010	$1/12 - \epsilon$
1001	$1/12 - \epsilon$
0110	$1/12 - \epsilon$
0101	$1/12 - \epsilon$
0011	$1/12 - \epsilon$
1110	$1/12 - \epsilon$
1011	$1/12 - \epsilon$
1101	$1/12 - \epsilon$
0111	$1/12 - \epsilon$
1111	$1/12 - \epsilon$

So, this suffices as a mathematically correct proof. However, the same trick cannot be used for Theorem 1. If we would omit the ϵ-terms in the Lagrangian there, we would end up with a solution that cannot be ϵ-approached by a probability distribution with unique risk minimizer. As an example, let us again consider the case $m = 4$ and the following probability distribution:

$\boldsymbol{y}$	$P(\boldsymbol{y})$
1000	.5
0111	.5

One of the risk minimizers for Hamming loss is vector 0000 with zero F-measure. Another vector, 1110, which is also a Hamming loss minimizer, gets an F-measure of 0.5833. This is a regret that is higher than the supremum mentioned in Theorem 1, but it is a value that cannot be ϵ-approached when restricting to unique Hamming loss minimizers.

Theorem 3 *Let $\boldsymbol{h}_J$ and $\boldsymbol{h}_F$ be vectors of predictions obtained by maximizing the Jaccard index and the F-measure, respectively. Let the utility of the F-measure maximizer be given by*

$$\delta(P) = \max_{\boldsymbol{h} \in \{0,1\}^m} \mathbb{E}\left[F(\boldsymbol{Y}, \boldsymbol{h})\right] = \max_{\boldsymbol{h} \in \{0,1\}^m} \sum_{\boldsymbol{y} \in \{0,1\}^m} P(\boldsymbol{y})\, F(\boldsymbol{y}, \boldsymbol{h}).$$

The regret of the F-measure maximizer with respect to the Jaccard index is then upper bounded by

$$\mathbb{E}\big[J(\boldsymbol{Y}, \boldsymbol{h}_J) - J(\boldsymbol{Y}, \boldsymbol{h}_F)\big] \leq 1 - \delta(P)/2$$

for all possible distributions P

Proof. The proof follows immediately from the following double inequality:

$$\frac{\sum_{i=1}^m y_i h_i(\boldsymbol{x})}{\sum_{i=1}^m y_i + \sum_{i=1}^m h_i(\boldsymbol{x})} \leq \frac{\sum_{i=1}^m y_i h_i(\boldsymbol{x})}{\sum_{i=1}^m y_i + \sum_{i=1}^m h_i(\boldsymbol{x}) - \sum_{i=1}^m y_i h_i(\boldsymbol{x})} \leq \frac{2\sum_{i=1}^m y_i h_i(\boldsymbol{x})}{\sum_{i=1}^m y_i + \sum_{i=1}^m h_i(\boldsymbol{x})},$$

which results in

$$\frac{F(\boldsymbol{y}, \boldsymbol{h}_F)}{2} \leq J(\boldsymbol{y}, \boldsymbol{h}_F) \leq F(\boldsymbol{y}, \boldsymbol{h}_F),$$

for all $\boldsymbol{y}, \boldsymbol{h} \in \{0,1\}^m$. ■

Theorem 5 *Let $\boldsymbol{h}_I$ be a vector of predictions obtained by assuming label independence as defined in (3), then the worst-case regret is lower-bounded by:*

$$\sup_P \left(\mathbb{E}\big[F(\boldsymbol{Y}, \boldsymbol{h}_F) - F(\boldsymbol{Y}, \boldsymbol{h}_I)\big]\right) \geq 2q - 1,$$

for all $q \in [1/2, 1]$ satisfying $\sum_{s=1}^m \left(\frac{2m!}{(m-s)!(s-1)!(m+s)} q^{m-s}(1-q)^s\right) - q^m > 0$ and the supremum taken over all possible distributions P.

Proof. To analyze the potential regret of methods that assume independence, it is sufficient to compare the F-maximizers and their corresponding F-measures for a joint distribution defined on independent random variables and a second joint distribution having the same marginal distributions, but no independence. Below, we analyze two families of probability distributions that are parameterized by a single parameter q, which is defined as $q = P(Y_i = 0)$ for all $i = 1, ..., m$. The first family resembles the case of independent random variables, for which the joint distribution is defined as the product of marginal probabilities:

$$P_A(\boldsymbol{y}) = q^{m-s}(1-q)^s \quad \text{where } s = \sum_{i=1}^m y_i. \tag{39}$$

The second family of distributions captures one particular case of a very strong stochastic dependence:

$$P_B(\boldsymbol{y}) = \begin{cases} q & \text{if } \boldsymbol{y} = \mathbf{0}_m \\ 1-q & \text{if } \boldsymbol{y} = \mathbf{1}_m \\ 0 & \text{otherwise} \end{cases},$$

If $q > 0.5$, then the F-measure maximizer of P_B is given by $\mathbf{0}_m$ and its corresponding F-measure is q. It is less straightforward to find the F-measure maximizer of P_A. Let us first introduce the equivalence classes

$$\Omega_m(s) = \{\boldsymbol{y} \in \{0,1\}^m \mid \sum_{i=1}^m y_i = s\},$$

$$\Omega_m(s, l) = \{\boldsymbol{y} \in \{0,1\}^m \mid \sum_{i=1}^m y_i = s \wedge y_l = 1\},$$

with $s, l \in \{1, ..., m\}$. The cardinality of these equivalence classes is given by

$$\begin{aligned} |\Omega_m(s)| &= \binom{m}{s} = \frac{m!}{(m-s)!s!}\,, \\ |\Omega_m(s,l)| &= \binom{m-1}{s-1} = \frac{(m-1)!}{(m-s)!(s-1)!}\,. \end{aligned}$$

Let $\boldsymbol{h}_k$ be a series of predictions such that $\sum_{i=1}^m h_i = k$. Without loss of generality, we can fix $\boldsymbol{h}_k$ to a vector of k ones that are followed by $m-k$ zeros, because the distributions that we analyze are fully symmetric (i.e., all index permutations of $\{1, ..., m\}$ in the label vectors yield the same values in probability mass). As a result, we can write the expected F-measure of $\boldsymbol{h}_k$ with $k > 0$ as:

$$\begin{aligned} \mathbb{E}_{\boldsymbol{Y}\sim P_A}\left[F(\boldsymbol{Y}, \boldsymbol{h}_k)\right] &= \sum_{\boldsymbol{y}\in\{0,1\}^m} F(\boldsymbol{y}, \boldsymbol{h}_k) P_A(\boldsymbol{y}) \\ &= \sum_{\boldsymbol{y}\in\{0,1\}^m} \frac{\sum_{i=1}^k 2y_i}{s_{\boldsymbol{y}} + k} P_A(\boldsymbol{y}) \\ &= \sum_{s=1}^m \sum_{\boldsymbol{y}\in\Omega_m(s)} \frac{\sum_{i=1}^k 2y_i}{s+k} P_A(\boldsymbol{y}) \\ &= \sum_{s=1}^m \sum_{i=1}^k \sum_{\boldsymbol{y}\in\Omega_m(s)} \frac{2y_i}{s+k} q^{m-s}(1-q)^s \\ &= \sum_{s=1}^m \sum_{i=1}^k \sum_{\boldsymbol{y}\in\Omega_m(s,i)} \frac{2}{s+k} q^{m-s}(1-q)^s \\ &= \sum_{s=1}^m \frac{(m-1)!}{(m-s)!(s-1)!} \frac{2k}{s+k} q^{m-s}(1-q)^s \end{aligned} \tag{40}$$

This is an increasing function of k, which implies that the F-measure maximizer consists of a vector of solely ones (or solely zeros) for P_A. In addition, the expected F-measure for a prediction vector of zeros is given by

$$\mathbb{E}_{\boldsymbol{Y}\sim P_A}\left[F(\boldsymbol{Y}, \boldsymbol{0}_m)\right] = q^m\,.$$

Let us define δ_m as

$$\begin{aligned} \delta_m &= \mathbb{E}_{\boldsymbol{Y}\sim P_A}\left[F(\boldsymbol{Y}, \boldsymbol{1}_m) - F(\boldsymbol{Y}, \boldsymbol{0}_m)\right] \\ &= \sum_{s=1}^m \Big(\frac{(m-1)!}{(m-s)!(s-1)!} \frac{2m}{m+s} q^{m-s}(1-q)^s \Big) - q^m\,. \end{aligned}$$

Then, assuming independence delivers the wrong maximizer for P_B as soon as $\delta_m > 0$. ■

Corollary 1 *Let $\boldsymbol{h}_I$ be a vector of predictions obtained by assuming independence, then the worst-case regret converges to 1 in the limit of m, i.e.,*

$$\lim_{m\to\infty} \sup_P \left(\mathbb{E}\big[F(\boldsymbol{Y}, \boldsymbol{h}_F) - F(\boldsymbol{Y}, \boldsymbol{h}_I)\big] \right) = 1,$$

where the supremum is taken over all possible distributions P.

Proof. For increasing m, the condition is satisfied for q close to one. The easiest way to observe this is computing the limit

$$\lim_{m\to\infty} q^m \quad = \quad 0\,,$$

which implies

$$\lim_{m\to\infty} \sum_{s=1}^{m} \frac{m!}{(m-s)!s!} q^{m-s}(1-q)^s \quad = \quad 1\,.$$

From this last limit and

$$\frac{2m}{m+s} \geq \frac{m}{s}$$

it follows that:

$$\lim_{m\to\infty} \sum_{s=1}^{m} \frac{(m-1)!}{(m-s)!(s-1)!} \frac{2m}{m+s} q^{m-s}(1-q)^s \geq 1\,.$$

By definition, (40) cannot exceed the upper bound of one, so this inequality must hold as an equality. In such a scenario, the worst-case regret is lower bounded by $R_q = 2q-1$, so that $\lim_{q\to 1, m\to\infty} R_q = 1$. As a consequence, the lower bound becomes tight in the limit of m going to infinity.

■

Theorem 7 *Let $\boldsymbol{h}_T$ be a vector of predictions obtained by putting a threshold on sorted marginal probabilities, then the worst-case regret is lower bounded by*

$$\sup_P \left(\mathbb{E}\big[F(\boldsymbol{Y}, \boldsymbol{h}_F) - F(\boldsymbol{Y}, \boldsymbol{h}_T)\big]\right) \geq \max\left(0, \frac{1}{6} - \frac{2}{m+4}\right),$$

where the supremum is taken over all possible distributions P.

Proof. To analyze the regret of thresholding approaches, we have to construct a counterexample for which the F-measure is not consistent with the order of the marginal probabilities. The following family of distributions is such a counterexample:

$$P(\boldsymbol{y}) \quad = \quad \begin{cases} 1/2-\epsilon & \text{if } y_1 = 1 \wedge \sum_{i=1}^{m} y_i = 1 \\ (1/2+\epsilon)/(2m-4) & \text{if } y_2 = 1 \wedge 1 + \sum_{i=3}^{m/2+1} y_i = \sum_{i=3}^{m} y_i = m/2 \\ (1/2+\epsilon)/(2m-4) & \text{if } y_2 = 1 \wedge 1 + \sum_{i=m/2+2}^{m} y_i = \sum_{i=3}^{m} y_i = m/2 \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

where we consider for simplicity that m is even and $\epsilon \in [0, 1/2]$ represents a positive constant. For ϵ close to zero, one can easily show that the F-measure maximizer is given by a vector of predictions consisting of only zeros, apart from a single one at position one. The expected F-measure of this prediction vector is $1/2 - \epsilon$. However, this prediction vector can never be returned by a method that relies on thresholding over marginal probabilities, because

$P(Y_2 = 1) > P(Y_1 = 1)$ in this particular case. By enumerating all candidate solutions examined by thresholding, one will find instead a prediction vector $\boldsymbol{h}_{11}$ consisting of zeros, apart from a one at the first two positions. The expected F-measure of this prediction vector is

$$\mathbb{E}\left[F(\boldsymbol{Y}, \boldsymbol{h}_{11})\right] = (1/2 - \epsilon)(2/3) + (1/2 + \epsilon)(2/(2 + (m/2)))\,.$$

As a consequence, this results in the above-mentioned regret when ϵ approaches zero. ■

References

P. Bartlett, M. Jordan, and J. McAuliffe. Convexity, classification and risk bounds. *Journal of the American Statistical Association*, 101:138–156, 2006.

C. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

L. Breiman. Some infinity theory for predictor ensembles. Technical Report 577, Department of Statistics, University of California, Berkeley, 2000.

K. Chai. Expectation of F-measures: Tractable exact computation and some empirical observations of its properties. In *Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2005.

W. Cheng and E. Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009.

W. Cheng, K. Dembczyński, W. Waegeman A. Jaroszewicz, and E. Hüllermeier. F-measure maximization in topical classification. In *Rough Sets and Current Trends in Computing*, volume 7413 of *Lecture Notes in Computer Science*, pages 439–446, 2012.

F. Chierichetti, R. Kumar, S. Pandey, and S. Vassilvitskii. Finding the Jaccard median. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 293–311, 2010.

D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 3(9):251–280, 1990.

T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms, 2nd edition*. MIT Press, 2001.

H. Daumé III, J. Langford, and D. Marcu. Search-based structured prediction. *Machine Learning*, 75:297–325, 2009.

B. De Baets, S. Janssens, and H. De Meyer. On the transitivity of a parametric family of cardinality-based similarity measures. *Journal of Approximate Reasoning*, 50:104–116, 2009.

J. del Coz, J. Diez, and A. Bahamonde. Learning nondeterministic classifiers. *Journal of Machine Learning Research*, 10:2273–2293, 2009.

K. Dembczyński, W. Cheng, and E. Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 279–286. Omnipress, 2010.

K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. An exact algorithm for F-measure maximization. In *Advances in Neural Information Processing Systems*, volume 25, 2011.

K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88:5–45, 2012a.

K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. An analysis of chaining in multi-label classification. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, 2012b.

K. Dembczyński, A. Jachnik, W. Kotlowski, W. Waegeman, and E. Hüllermeier. Optimizing the F-measure in multi-label classification: Plug-in rule approach versus structured loss minimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, 1997.

L. Dice. Measures of the amount of ecologic associations between species. *Ecology*, 26: 297–302, 1945.

J. Duchi, L. Mackey, and M. I. Jordan. On the consistency of ranking algorithms. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 327–334, 2010.

R. Fan and C. Lin. A study on threshold selection for multi-label classification. Technical report, Department of Computer Science, National Taiwan University, 2007.

T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *Proceedings of the International Conference on Machine Learning (ICML)*. Omnipress, 2008.

Wei Gao and Zhi-Hua Zhou. On the consistency of multi-label learning. *Artificial Intelligence*, 199-200:22–44, 2013.

N. Ghamrawi and A. McCallum. Collective multi-label classification. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 195–200, 2005.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11, 2009.

M.A. Hanson. On sufficiency of the Kuhn-Tucker conditions. *Journal of Mathematical Analysis and Applications*, 80:545–550, 1981.

B. Hariharan, L. Zelnik-Manor, S. Vishwanathan, and M. Varma. Large scale max-margin multi-label classification with priors. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 423–430. Omnipress, 2010.

M. Jansche. Maximum expected F-measure training of logistic regression models. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 736–743, 2005.

M. Jansche. A maximum expected utility framework for binary sequence labeling. In *Proceedings of the Annual Meetings of the Association for Computational Linguistics (ACL)*, pages 736–743, 2007.

A. Janusz, H. Son Nguyen, D. Ślezak, S. Stawicki, and A. Krasuski. JRS'2012 data mining competition: Topical classification of biomedical research papers. In *Rough Sets and Current Trends in Computing*, volume 7413 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 2012.

T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 377–384, 2005.

S. Keerthi, V. Sindhwani, and O. Chapelle. An efficient method for gradient-based adaptation of hyperparameters in SVM models. In *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2007.

I. Kokkinos. Boundary detection using F-measure-, filter- and feature- (f3) boost. In *Proceedings of the European Conference on Computer Vision (ECCV): Part II*, pages 650–663, 2010.

A. Kumar, S. Vembu, A.K. Menon, and C. Elkan. Beam search algorithms for multilabel learning. In *Machine Learning*, 2013.

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)*, pages 282–289. Morgan Kaufmann, 2001.

S. Lee. On classification and regression trees for multiple responses and its application. *Journal of Classification*, 23:123–141, 2006.

D. Lewis. Evaluating and optimizing autonomous text classification systems. In *Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 246–254, 1995.

Z. Lipton, C. Elkan, and B. Naryanaswam. Optimal thresholding of classifiers to maximize F1 measure. In *Proceedings of the European Conference on Machine Learning*, 2014.

O. Luaces, J. Diez, J. Barranquero, J. del Coz, and A. Bahamonde. Binary relevance efficacy for multilabel classification. *Progress in Artificial Intelligence*, 1(4):303–313, 2012.

D. Martin, C. Fowlkes, and J. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:530–549, 2004.

A. K McCallum, 2002. URL `http://mallet.cs.umass.edu`.

A. K. McCallum, D. Freitag, and F. (2000) Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2000.

D. Musicant, V. Kumar, and A. Ozgur. Optimizing F-measure with support vector machines. In *Proceedings of the International FLAIRS Conference*, pages 356–360. Haller AAAI Press, 2003.

J. Petterson and T. Caetano. Reverse multi-label learning. In *Advances in Neural Information Processing Systems*, volume 24, 2010.

J. Petterson and T. Caetano. Submodular multi-label learning. In *Advances in Neural Information Processing Systems*, volume 25, 2011.

J. Quevedo, O. Luaces, and A. Bahamonde. Multilabel classifiers with a probabilistic thresholding strategy. *Pattern Recognition*, 45:876–883, 2012.

J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 254–269. Springer, 2009.

T. Sang and F. De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Human Language Technologies Conference and the Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 142–147, 2003.

R. Schapire and Y. Singer. Boostexter: a boosting-based system for text categorization. *Machine Learning*, 39:135–168, 2000.

I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001.

J. Suzuki, E. McDermott, and H. Isozaki. Training conditional random fields with multivariate evaluation measures. In *Proceedings of the Annual Meetings of the Association for Computational Linguistics (ACL)*, pages 217–224, 2006.

S. Swamidass, J. Chen, J. Bruand, P. Phung, L. Ralaivola, and P. Baldi. Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. *Bioinformatics*, 21 Suppl 1(2):359–368, 2005.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004.

A. Tewari and P. L. Bartlett. On the consistency of multiclass classification methods. *Journal of Machine Learning Research*, 8:1007–1025, 2007.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

G. Tsoumakas and I. Katakis. Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*. Springer, 2010.

G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011.

C. J. van Rijsbergen. Foundation of evaluation. *Journal of Documentation*, 30(4):365–373, 1974.

C. Vens, J. Struyf, L. Schietgat, S. Dzeroski, and H. Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185–214, 2008.

Y. Wu, H. Zhang, and Y. Liu. Robust model-free multiclass probability estimation. *Journal of the American Statistical Association*, 105:424–436, 2010.

Y. Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1(1–2):67–88, 1999.

N. Ye, K. Chai, W. Lee, and H. Chieu. Optimizing F-measures: a tale of two approaches. In *Proceedings of the International Conference on Machine Learning*, 2012.

H. Zhang. Classification trees for multiple binary responses. *Journal of the American Statistical Association*, 93(441):180–193, 1998.

M. Zhang and Z. Zhou. ML-kNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32:56–85, 2004.

X. Zhang, T. Graepel, and R. Herbrich. Bayesian online learning for multi-label and multivariate performance measures. In *Proceedings of the Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 956–963, 2010.

M. Zhao, N. Edakunni, A. Pocock, and G. Brown. Beyond Fano's inequality: Bounds on the optimal F-score, BER, and cost-sensitive risk and their implications. *Journal of Machine Learning Research*, 14:1033–1090, 2013.

H. Zhuang, A. Chin, S. Wu, W. Wang, X. Wang, and J. Tang. Inferring geographic coincidence in ephemeral social networks. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 1–16. Springer, 2012.

Journal of Machine Learning Research 15 (2014) 3569-3573 Submitted 9/13; Revised 6/14; Published 11/14

SPMF: A Java Open-Source Pattern Mining Library

Philippe Fournier-Viger PHILIPPE.FOURNIER-VIGER@UMONCTON.CA
Department of Computer Science
University of Moncton, Moncton, NB E1A 3E9, Canada
Antonio Gomariz AGOMARIZ@UM.ES
Department of Information and Communication Engineering
University of Murcia, Murcia 30100, Spain
Ted Gueniche ETG8697@UMONCTON.CA
Azadeh Soltani SOLTANI.AZ@STU-MAIL.UM.AC.IR
Department of Computer Engineering
Ferdowsi University of Mashhad, Iran
Cheng-Wei Wu SILVEMOONFOX@IDB.CSIE.NCKU.EDU.TW
Vincent S. Tseng TSENGSM@MAIL.NCKU.EDU.TW
Department of Computer Science and Information Engineering
National Cheng Kung University, Taiwan

Editor: Balázs Kégl

Abstract

We present SPMF, an open-source data mining library offering implementations of more than 55 data mining algorithms. SPMF is a cross-platform library implemented in Java, specialized for discovering patterns in transaction and sequence databases such as frequent itemsets, association rules and sequential patterns. The source code can be integrated in other Java programs. Moreover, SPMF offers a command line interface and a simple graphical interface for quick testing. The source code is available under the GNU General Public License, version 3. The website of the project offers several resources such as documentation with examples of how to run each algorithm, a developer's guide, performance comparisons of algorithms, data sets, an active forum, a FAQ and a mailing list.

Keywords: data mining, library, frequent pattern mining, sequence database, transaction database, open-source

1. Introduction

In this paper, we present SPMF (Sequential Pattern Mining Framework), a data mining library that is specialized in *frequent pattern mining*, an important subfield of data mining that aims at discovering interesting patterns and associations in databases. SPMF is an open-source project, started in 2009 to address the lack of large open-source data mining library specialized in frequent pattern mining. There exist several general purpose open-source data mining libraries such as Weka (Witten et al., 2005), Mahout (Mahout, 2013) and Knime (Knime, 2013), which provide a wide range of data mining techniques. However, they offer a very limited set of algorithms for frequent pattern mining. Weka, Knime and Mahout

offer only a few popular pattern mining algorithms such as Apriori (Agrawal and Srikant, 1994), GSP (Srikant et al., 1996) and FPGrowth (Han et al., 2004). Some specialized platforms like Coron (Coron, 2013), LUCS-KDD (LUCS-KDD, 2013) and Illimine (Illimine, 2013) offer a slightly larger choice of pattern mining algorithms. However, the source code of Coron is not public, Illimine provides the source code of only one of its pattern mining algorithms and LUCS-KDD source code cannot be used for commercial purposes. SPMF provides more than 55 algorithms for pattern mining. Implementations of most of these algorithms can only be found in SPMF. For example, only three algorithms from SPMF appear in Weka and Knime (Apriori, FPGrowth and GSP), only one in Mahout (FPGrowth), two in LUCS-KDD (Apriori, FPGrowth), and eight in Coron. Another related project is Galicia (Galicia, 2013), an open-source software focusing on mining lattice-based patterns and visualizing lattices. It has only one algorithm in common with SPMF. Another distinctive feature of SPMF is that it offers more than 17 algorithms for mining sequential patterns, while Weka and Knime only offer a single algorithm (GSP), and other previously mentioned software offer none. Moreover, note that Galicia, Coron, Illimine and LUCS-KDD are projects that have been inactive for several years.

Since its first major release in 2010, SPMF has been used in more than 70 research projects in various domains such as web usage mining, analyzing learner behavior in e-learning, clinical text retrieval, sales forecasting, restaurant recommendation, analyzing nucleic acids sequences, anomaly detection in medical treatment and forecasting crime incidents (see the SPMF website for an up-to-date list of applications). Algorithms offered in SPMF can be applied to two main types of data:

- A *transaction database* (a.k.a. *binary context*) is a set of transactions $T = \{T_1, T_2, ... T_n\}$ and a set of items $I = \{i_1, i_2, ...i_m\}$, where $T_x \subseteq I$ for $1 \leq x \leq n$. For example, each transaction of a transaction database could represent a set of items purchased by a customer at a store, or a set of words appearing in a text document.

- A *sequence database* is a generalization of a transaction database. It is a set of sequences $S = \{S_1, S_2, ...S_p\}$ and a set of items $I = \{i_1, i_2, ...i_q\}$. A sequence is a list of transactions $< T_1, T_2, ...T_r >$ where $T_x \subseteq I$ for $1 \leq x \leq r$. Examples of real-life data that can be represented as sequence databases are sequences of web pages visited by users, bioinformatics data (e.g., protein sequences, microarray data and DNA sequences), stock market data, weather observations and sensor data.

Three main data mining tasks can be performed with SPMF.

- *frequent itemset mining* (Agrawal and Srikant, 1994) consists of discovering frequent itemsets, i.e., sets of items appearing in more than *minsup* transactions of a transaction database, where *minsup* is a parameter set by the user.

- *association rule mining* (Agrawal and Srikant, 1994) consists of discovering the association rules respecting some thresholds *minsup* and $minconf$ in a transaction database. An association rule $X \Rightarrow Y$ is an association between two sets of items X and Y such that $X \cap Y = \emptyset$, $X \cup Y$ appears in more than *minsup* transactions, and that the number of transactions containing $X \cup Y$ divided by the number of transactions containing X is higher than $minconf$.

- *sequential pattern mining* (Agrawal and Srikant, 1995) consists of discovering frequent sequential patterns, i.e., subsequences appearing in more than *minsup* sequences of a sequence database, where *minsup* is a parameter set by the user.

For these three classical data mining tasks with wide applications, SPMF offers implementations of popular algorithms such as Apriori, Eclat (Zaki, M. J.), FPGrowth, GSP, PrefixSpan (Pei et al., 2004), SPAM (Ayres et al., 2000) and BIDE (Wang et al., 2007). But it also offers several algorithms for variations of these problems, for example to discover rare itemsets, closed itemsets, non-redundant association rules, indirect association rules, top-k association rules, to deal with uncertain data or database containing quantity and profit information and to discover *sequential rules* (Fournier-Viger et al., 2011). SPMF offers both classical algorithms and recent state-of-the-art algorithms such as Hui-Miner (Liu et al., 2012), ClaSP (Gomariz et al., 2013) and RuleGrowth (Fournier-Viger et al., 2011).

2. Using SPMF

SPMF is implemented in Java and is cross-platform. The only requirement to run SPMF is to have Java 7 or higher installed. There are two versions of SPMF. The *source code version* offers all algorithms from SPMF. The documentation provides an example of how to run each algorithm. It explains the input and output of each algorithm, its main characteristics and where to obtain more information about the algorithm. Moreover, a sample program and input file is provided in the source code of SPMF to show how to execute each example from Java code. Running an algorithm is just a few lines of code. One needs to create an instance of the algorithm, specify its parameter(s), input file and an output file path (if the result is to be saved to a file). For example, the following code runs the Apriori algorithm on a file "input.txt" with its *minsup* parameter set to 0.4.

```
AlgoApriori apriori = new AlgoApriori();
apriori.runAlgorithm(0.4, "input.txt", "output.txt");
```

The source code can be easily integrated into other Java software programs since (1) the source code of each algorithm implementation is located in its own subpackage and (2) there is no dependency on any other software or library. To support developers and users, extensive resources are provided on the website of SPMF such as an active forum, a FAQ, a developers' guide and a mailing list to be informed of the latest updates to SPMF. The website also provides a set of more than 40 large real-life data sets that can be used with the algorithms offered in SPMF. This can be useful for educational purpose (e.g., for a data mining course) or for comparing the performance of algorithms (for data mining researchers). Finally, the website also provides several performance comparisons of algorithms offered in SPMF, for various data sets, to give a good idea of the relative performance of the algorithms designed for the same task.

The *release version* of SPMF is a runnable JAR file that can be launched with a double-click. It provides a minimalist user interface (see Figure 1), designed to allow quickly testing the behavior of the algorithms. The graphical user interface allows one to select an input file and an output file, choose an algorithm, enter its parameters and run it. For each algorithm, a sample input file is provided and an example is described in the documentation. The

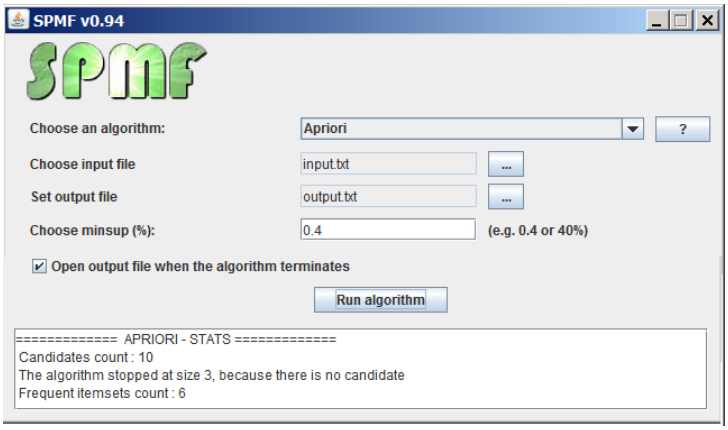


Figure 1: SPMF graphical user interface

runnable JAR file can also be used to run algorithms from the command line. For example, to run Apriori, the following command can be used:

```
java -jar spmf.jar run Apriori input.txt output.txt 0.4
```

Input files for the algorithms are text files. The format that is used is the one from frequent pattern mining competitions such as FIMI (Boyardo et al., 2004) and used by researchers in this domain (files where items are represented by integers). But, to allow greater interoperability, the GUI and command line version of SPMF can also read the popular ARFF file format for itemset and association rule mining (used by Weka and Knime), and tools are provided to convert some selected formats to the SPMF format.

3. Conclusion

We have presented SPMF, a data mining library specialized in frequent pattern mining. The project is active and latest releases can be found on its website. SPMF has been applied in more than 70 research projects. Code submissions are reviewed by the project founder and contributors to see if they meet the requirements before being integrated in SPMF.

Acknowledgments

The authors thank the Natural Sciences and Engineering Research Council for its financial support and users who provided feedback and applied SPMF in their research projects.

References

R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the Twentieth International Conference on Very Large Data Bases*, pages 487-499, Santiago, Chile, 1994.

R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3-14, Taipei, Taiwan, 1995.

J. Ayres, J. Gehrke, T, Yiu, J. Flannick. Sequential pattern mining using bitmaps. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 429-435, Edmonton, Alberta, Canada, July 2002.

R. Bayardo, B. Goethals, M. J. Zaki, Proceedings of the IEEE International Conference on Data Mining Workshop on Frequent Itemset Mining Implementations Brighton, UK, 2004.

Coron. Software available at: `http://coron.loria.fr/site/index.php`

P. Fournier-Viger, R. Nkambou and V.S. Tseng. RuleGrowth: mining sequential rules common to several sequences by pattern-growth. In *Proceedings of the 26th Symposium on Applied Computing.*, ACM Press, pages 954-959, Taitung, Taiwan, 2011.

J. Han, J. Pei, Y. Yin, R. Mao. Mining frequent patterns without candidate generation: a frequent-pattern tree approach. *Data Mining and Know. Discovery*, 8(1):53-87, 2004.

Galicia. Software available at: `http://www.iro.umontreal.ca/~galicia/`

A. Gomariz, M. Campos, R. Marin, B. Goethals. ClaSP: An efficient algorithm for mining frequent closed sequences. *Proceedings of the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 50-61, Gold Coast, Australia, 2013.

Illimine. Software available at: `http://illimine.cs.uiuc.edu/`

Knime. Software available at: `http://www.knime.org/`

M. Liu, J.-F. Qu. Mining high utility itemsets without candidate generation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 55-64, Maui, HI, USA, 2012

LUCS-KDD. Software available at: `http://cgi.csc.liv.ac.uk/~frans/KDD/Software/`

Mahout. Software available at: `http://mahout.apache.org/`

J. Pei, J. Han et al. Mining sequential patterns by pattern-growth: the prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1-17, 2004.

R. Srikant, R. Agrawal. Mining sequential patterns: generalizations and performance improvements. *Proceedings of the 5th International Conference on Extending Database Technology*, pages 3-17, Avignon, France, 1996.

J. Wang, J. Han, C. Li. Frequent closed sequence mining without candidate maintenance. *IEEE Transactions on Knowledge and Data Engineering*, 19(8):1042–1056, 2007.

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, 2005.

M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390.

Journal of Machine Learning Research 15 (2014) 3575-3619 Submitted 12/13; Revised 7/14; Published 11/14

Efficient Learning and Planning with Compressed Predictive States

William Hamilton WILLIAM.HAMILTON2@MAIL.MCGILL.CA
Mahdi Milani Fard MMILANI1@CS.MCGILL.CA
Joelle Pineau JPINEAU@CS.MCGILL.CA
School of Computer Science
McGill University
Montreal, QC, Canada

Editor: Jan Peters

Abstract

Predictive state representations (PSRs) offer an expressive framework for modelling partially observable systems. By compactly representing systems as functions of observable quantities, the PSR learning approach avoids using local-minima prone expectation-maximization and instead employs a globally optimal moment-based algorithm. Moreover, since PSRs do not require a predetermined latent state structure as an input, they offer an attractive framework for model-based reinforcement learning when agents must plan without a priori access to a system model. Unfortunately, the expressiveness of PSRs comes with significant computational cost, and this cost is a major factor inhibiting the use of PSRs in applications. In order to alleviate this shortcoming, we introduce the notion of compressed PSRs (CPSRs). The CPSR learning approach combines recent advancements in dimensionality reduction, incremental matrix decomposition, and compressed sensing. We show how this approach provides a principled avenue for learning accurate approximations of PSRs, drastically reducing the computational costs associated with learning while also providing effective regularization. Going further, we propose a planning framework which exploits these learned models. And we show that this approach facilitates model-learning and planning in large complex partially observable domains, a task that is infeasible without the principled use of compression.[1]

Keywords: predictive state representation, reinforcement learning, dimensionality reduction, random projections

1. Introduction

In the reinforcement learning (RL) paradigm, an agent in a system acts, observes, and receives feedback in the form of numerical signals (Sutton and Barto, 1998). Given this experience, the agent determines an optimal policy (i.e., a guide for its future actions) via value-function based dynamic programming or parameterized policy search. This is conceptually analogous to the 'operant conditioning' postulated to underlie certain forms of

1. An earlier version of this work appeared as: W.L. Hamilton, M. M. Fard, and J. Pineau. Modelling sparse dynamical systems with compressed predictive state representations. In *Proceedings of the Thirtieth International Conference on Machine Learning*, 2013.

animal (and human) learning. Organisms learn to repeat actions that give positive feedback and avoid those with negative results.

1.1 Fully to Partially Observable Domains

In the standard formulation, an RL agent is given prior knowledge of a domain in the form of a state-space, transition probabilities, and an observation (i.e., sensor) model. Formally, the system is described by a Markov decision process (MDP), and given the MDP description, a variety of optimization algorithms may then be used to solve the problem of determining an optimal action policy (Sutton and Barto, 1998). In general, approximate solutions are determined for domains exhibiting large, or even moderate, dimensionality (Gordon, 1999).

The situation is further complicated in domains exhibiting partial observability, where observations are aliased and do not fully determine an agent's state in a system. For example, an agent's sensors may indicate the presence of nearby objects but not the agent's global position within an environment. To accommodate this uncertainty, the MDP framework is extended as partially observable Markov decision processes (POMDPs) (Kaelbling et al., 1998). Here, the true state is not known with certainty, and optimization algorithms must act upon belief states (i.e., probability distributions over the state-space).

1.2 Model-Learning Before Planning

The POMDP extension introduces a measure of uncertainty in the reinforcement learning paradigm. Nevertheless, an agent learning a policy via the POMDP framework has access to considerable a priori knowledge: Most centrally, the agent (which necessarily and implicitly contains the POMDP solver) has access to a description of the system in the form of an explicit state-space representation. Moreover, in a majority of instances, the agent knows the probabilities governing the transitions between states, the observation functions governing the emission of observable quantities from these states, and the reward function specifying some empirical measure of "goodness" for each state (Kaelbling et al., 1998).

Access to such knowledge allows for the construction of optimal (or near-optimal) plans and is useful for real-world applications where considerable domain-specific knowledge is available. However, the converse situation, where a (near)-complete system model is not known a priori, is both important and lags behind in terms of research results. In such a setting, an agent must learn a system model prior to (or while simultaneously) learning an action policy.

At an application level, there are many situations in which expert knowledge is sparse, and it is possible that even application domains with domain-knowledge could benefit from the use of algorithms that learn system models prior to planning and that are thus free from unintended biases introduced via expert-specified system models. At a more theoretical level, the development of general agents that both learn system models and plan using such models is fundamental in the pursuit of creating truly intelligent artificial agents that can learn and succeed independent of prior domain knowledge.

1.3 Learning a Model-based Predictive Agent

In this work we outline an algorithm for constructing a learning and planning agent for sequential decision-making under partial state observability. At a high-level, the algorithm is model-based, specifying an agent that builds a model of its environment through experience and then plans using this learned model. Such a model-based approach is necessary in complicated partially observable domains, where single observations are far from sufficient statistics for the state of the system (Kaelbling et al., 1998). At its core, the algorithm relies on the powerful and expressive model class of predictive state representations (PSRs) (Littman et al., 2002). PSRs (described in detail in Section 2) are an ideal candidate for the construction of an agent that both learns a system model and plans using this model, as they do not require a predetermined state-space as an input.

PSRs have been used as the basis of model-based reinforcement learning agents in a number of recent works (Boots et al., 2010; Rosencrantz et al., 2004; Ong et al., 2012; Izadi and Precup, 2008; James and Singh, 2004). However, for these previous approaches, the time and space complexities of learning scale super-linearly in the maximum length of the trajectories used (see Section 3). In this work we use an approach that simultaneously ameliorates the efficiency concerns related to constructing PSRs and alleviates the need for domain-specific feature construction. The model-learning algorithm, termed compressed predictive state representation (CPSR), uses random projections in order to efficiently learn accurate approximations of PSRs in sparse systems. In addition, the approach makes use of recent advancements in incrementally learning transformed PSRs (TPSRs), providing further optimization (Boots and Gordon, 2011). The details of the model-learning algorithm are provided in Section 3.2. Section 4 presents theoretical results pertaining to the accuracy of the approximate learned model and elucidates how our approach regularizes the learned model, trading off reduced variance for controlled bias.

The planning algorithm used is an extension of the fitted-Q function approximation-based planning algorithm for fully observable systems (Ernst et al., 2005). This approach has been applied to PSRs previously with some success (Ong et al., 2012) and provides a strong alternative to point-based value iteration methods (Izadi and Precup, 2008). The algorithm simply substitutes a predictive state for the observable MDP state in a fitted-Q learning algorithm, and a function approximator is used to learn an approximation of the Q-function for the system (i.e., the function mapping predictive states and actions to expected rewards). The details of the planning approach are outlined in Section 5. The main empirical contribution of this work is the application of this approach to domains and sample-sizes of complexity not previously feasible for PSRs. Section 6 will highlight empirical results demonstrating the performance of the algorithm on some synthetic robot navigation domains and a difficult real-world application task based upon the ecological management of migratory bird species.

This work builds upon the algorithm presented in Hamilton et al. (2013), extending it in a number of ways. Specifically, this work (1) permits a broader class of projection matrices, (2) includes optional compression of both histories and tests, (3) combines compressed sensing with incremental matrix decomposition to facilitate incremental/online learning, (4) provides a more detailed theoretical analysis of the model-learning algorithm, (5) explicitly includes a planning framework, which exploits the learned CPSR models in a principled

manner, and (6) provides extensive empirical results pertaining to both model-learning and planning, including results on a difficult real-world problem.

2. Predictive State Representations

Predictive state representations (PSRs) offer an expressive and powerful framework for modelling dynamical systems and thus provide a suitable foundation for a model-based reinforcement learning agent. In the PSR framework, a predictive model is constructed directly from execution traces, utilizing minimal prior information about the domain (Littman et al., 2002; Singh et al., 2004). Unlike latent state based approaches, such as hidden Markov models or POMDPs, PSR states are defined only via observable quantities. This not only makes PSRs more general, as they do not require a predetermined state-space, but it also increases their expressive power relative to latent state based approaches (Littman et al., 2002). In fact, the PSR paradigm subsumes POMDPs as a special case (Littman et al., 2002). In addition, PSRs facilitate model-learning without the use of local-minima prone expectation-maximization (EM) and allow for the efficient construction of globally optimal models via a method-of-moments based algorithm (James and Singh, 2004). The following section outlines the foundations of the PSR approach and sets the stage for the presentation of compressed predictive state representations in Section 3 and our efficient learning algorithm in Section 3.2. Much of the PSR background material (e.g., the derivation of the PSR model in Sections 2.2 and 2.3) expands upon the presentation in Boots et al. (2010) and uses important results from that work.

2.1 Notation

This section outlines the notation that will be used throughout the paper.

2.1.1 Matrix Algebra Notation

Bold letters denote vectors $\mathbf{v} \in \mathbb{R}^d$ and matrices $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$. Given a matrix $\mathbf{M}$, $\|\mathbf{M}\|$ denotes its Frobenius norm. $\mathbf{M}^\dagger$ is used to denote the Moore–Penrose pseudoinverse of $\mathbf{M}$. Sometimes names are given to the columns and rows of a matrix using ordered index sets $\mathcal{I}$ and $\mathcal{J}$. In this case, $\mathbf{M} \in \mathbb{R}^{\mathcal{I} \times \mathcal{J}}$ denotes a matrix of size $|\mathcal{I}| \times |\mathcal{J}|$ with rows indexed by $\mathcal{I}$ and columns indexed by $\mathcal{J}$. We then specify entries in a matrix (or tensor) using these indices and the bracket notation; e.g., $[\mathbf{M}]_{i,j}$ corresponds to the entry in the row indexed by $i \in \mathcal{I}$ and the column indexed $j \in \mathcal{J}$. Rows or columns of a matrix are specified using this index notation and the $*$ symbol; e.g., $[\mathbf{M}]_{i,*}$, denotes the ith row of $\mathbf{M}$. Finally, given $\mathcal{I}' \subset \mathcal{I}$ and $\mathcal{J}' \subset \mathcal{J}$ we define $[\mathbf{M}]_{\mathcal{I}',\mathcal{J}'}$ as the submatrix of $\mathbf{M}$ with rows and columns specified by the indices in $\mathcal{I}'$ and $\mathcal{J}'$, respectively.

2.1.2 Probability Notation

We denote the probability of an event by $\mathbb{P}(\cdot)$ and use $|$ to denote the usual probabilistic conditioning. To avoid excessive notation, when the $\mathbb{P}(\cdot)$ operator is applied to a vector of events, it is understood as returning a vector of probabilities unless otherwise indicated (i.e., a single operator is used for single events and vectors of events).

For clarity, we use || to denote conditioning upon an agents policy (i.e., plan). That is, || denotes that we are conditioning upon the knowledge that the agent will "intervene" in a system by executing the specified actions.

2.2 Technical Foundations

A PSR model represents a partially observable system's state as a probability distribution over future events. More formally, we maintain a probability distribution over different sequences of possible future action-observation pairs. Such sequences of possible future action-observations are termed *tests* and denoted τ. For example, we could construct a test $\tau_i = [o_{t+1}^{k_1}, o_{t+2}^{k_2}, ..., o_{t+n}^{k_n} || a_{t+1}^{l_1}, a_{t+2}^{l_2}, ..., a_{t+n}^{l_n}]$, where notationally subscripts refer to time, superscripts identify particular actions or observations, and actions following the || symbol denote that we are conditioning upon the agent "intervening" by performing those specified actions at the specified times. We can then say that such a test is *executed* if the agent intervenes and takes the specified actions, and we say the test *succeeded* if the observations received by the agent match those specified by the test. Going further, we can define the probability of success for test τ_i as

$$\mathbb{P}(\tau_i) = \mathbb{P}(o_{t+1}^{k_1}, o_{t+2}^{k_2}, ..., o_{t+n}^{k_n} || a_{t+1}^{l_1}, a_{t+2}^{l_2}, ..., a_{t+n}^{l_n}).$$

Of course, we want to know more than just the unconditioned probabilities of success for each test. A complete model of a dynamical system also requires knowing the success probabilities for each test conditioned on the agent's previous experience, or *history*. We denote such a history $h_j = [a_0^{l_0} o_0^{k_0}, a_1^{l_1} o_1^{k_1} ... a_t^{l_t} o_t^{k_t}]$, where again subscripts denote time and superscripts identify particular actions or observations. Importantly, the || symbol for intervention is absent from the definition of history, as the sequence of actions specified in a history are assumed to have already been executed.

Finally, given that an agent has performed some actions and received some observations, defining some history h_j, we compute

$$\mathbb{P}(\tau_i^{\mathcal{O}} | h_j || \tau_i^{\mathcal{A}}),$$

the probability of τ_i succeeding conditioned upon the agent's current history in the system, where $\tau_i^{\mathcal{A}}$ and $\tau_i^{\mathcal{O}}$ denote the ordered lists of actions and observations, respectively, defined in τ_i.

It is not difficult to see that a partially observable system is completely described by the conditional success probabilities of all tests given all histories. That is, if we have $\mathbb{P}(\tau_i^{\mathcal{O}} | h_j || \tau_i^{\mathcal{A}}) \; \forall i \; \forall j$ then we trivially have all necessary information to characterize the dynamics of a system. Of course, maintaining all such probabilities directly is infeasible, as there is a potentially infinite number of tests and histories (and at the very least an exorbitant number for any system of even moderate complexity) (Littman et al., 2002).

Fortunately, it has been shown that it suffices to remember only the conditional probabilities for a (potentially) small *core set* of tests, and the conditional probabilities for all other tests may be defined as linear functions of the conditional probabilities for the tests in this core set (Littman et al., 2002).[2] More formally, we define the system dynamics ma-

2. In this work, the shortened phrase *core set* is always to be interpreted as *core set of tests*; that is, such sets always correspond to a set of tests.

trix, $\mathbf{H}$, as the (potentially infinite size) matrix, where each row corresponds to a particular test (under some lexicographic ordering), each column to a particular history (under some lexicographic ordering), and a particular $[\mathbf{H}]_{i,j}$ entry to $\mathbb{P}(\tau_i^{\mathcal{O}}|h_j||\tau_i^{\mathcal{A}})$. $\mathbf{H}$ simply organizes $\mathbb{P}(\tau_i^{\mathcal{O}}|h_j||\tau_i^{\mathcal{A}})$, $\forall i \forall j$ in a matrix structure. In Littman et al. (2002) and Singh et al. (2004) it is shown that if $\mathbf{H}$ has rank k then (1) k corresponds to the rank of the partially observable system, as defined by Jaeger (2000) and (2) there exists a *minimal core set* of size k (i.e., the smallest core set of tests is of size k, though there may be larger core sets). Thus, if $\mathbf{H}$ has rank k, it suffices to remember conditional probabilities for only k tests (those that are a part of the minimal core set), and the conditional probabilities for all other tests may be defined as *linear* functions of the conditional probabilities for these tests.

The rank of $\mathbf{H}$ thus describes the complexity of a system. For example, a system with $\text{rank}(\mathbf{H}) = k$ can not be modelled by a POMDP with less than k states; though it may require more than k POMDP states (Singh et al., 2004). In contrast, a PSR can always (exactly) model a system with $\text{rank}(\mathbf{H}) = k$ using a minimal core set of exactly size k (Singh et al., 2004). This demonstrates how PSRs can be more compact than POMDPS.

Thus, for a PSR, given a minimal core set $\mathcal{Q}$ (i.e., $|\mathcal{Q}| = \text{rank}(\mathbf{H})$), we can compute the conditional probability of some test $\tau_i \notin \mathcal{Q}$ as

$$\mathbb{P}(\tau_i^{\mathcal{O}}|h_j||\tau_i^{\mathcal{A}}) = \mathbf{r}_{\tau_i}^{\top}\mathbb{P}(\mathcal{Q}^{\mathcal{O}}|h_j||\mathcal{Q}^{\mathcal{A}}),$$

where $\mathbf{r}_{\tau_i}$ is a vector of weights and $\mathbb{P}(\mathcal{Q}^{\mathcal{O}}|h_j||\mathcal{Q}^{\mathcal{A}})$ an ordered vector of conditional probabilities for each test in the minimal core set $q_i \in \mathcal{Q}$. Integral to this approach is the fact that restricting the model to linear functions of tests in the minimal core set does not preclude the modelling of non-linear systems, as the dynamics implicit in the probabilities may specify non-linear behaviours (Littman et al., 2002).

Thus, given the functions mapping tests in the core set to all other tests, it suffices to maintain, at time t, only the vector $\mathbf{m}_t = \mathbb{P}(\mathcal{Q}^{\mathcal{O}}|h_t||\mathcal{Q}^{\mathcal{A}})$, where h_t is the history of the system at time t. That is, it suffices to maintain only the vector of conditional probabilities for the tests in a core set (which is usually assumed to be minimal) .

2.3 The PSR Model

Formally, a PSR model of a system is defined by $\langle \mathcal{O}, \mathcal{A}, \mathcal{Q}, \mathcal{F}, \mathbf{m}_0 \rangle$, where $\mathcal{O}$ and $\mathcal{A}$ define the possible observations and actions respectively, $\mathcal{Q}$ is a minimal core set of tests, $\mathcal{F}$ defines a set of linear functions mapping success probabilities for tests in the minimal core set to the probabilities for all tests, and $\mathbf{m}_0$ defines the initial state of the system (i.e., $\mathbf{m}_0 = \mathbb{P}(\mathcal{Q}^{\mathcal{O}}||\mathcal{Q}^{\mathcal{A}})$). Since $\mathcal{F}$ contains only linear functions, its elements can be specified as vectors of weights. These vectors, in turn, are specified using a finite set of linear operators (i.e., matrices). Specifically, we define a linear operator $\mathbf{M}_{a^l o^k}$ for each action-observation pair such that

$$\begin{aligned}\mathbb{P}(o_{t+1}^k|h_t||a_{t+1}^l) &= \mathbf{m}_{\infty}^{\top}\mathbf{M}_{a^l o^k}\mathbb{P}(Q^{\mathcal{O}}|h_t||Q^{\mathcal{A}}) \\ &= \mathbf{m}_{\infty}^{\top}\mathbf{M}_{a^l o^k}\mathbf{m}_t,\end{aligned}$$

where $\mathbf{m}_{\infty}$ is a constant normalizer such that $\mathbf{m}_{\infty}^{\top}\mathbf{m}_t = 1$, $\forall t$.

These operators map probabilities of tests in the specified minimal core set to the probabilities for single action-observation pairs and may be recursively combined to generate the full set of linear functions in $\mathcal{F}$. For instance, for the test $\tau_i = [o^{k_1}_{t+1}, o^{k_2}_{t+2}, ..., o^{k_n}_{t+n} || a^{l_1}_{t+1}, a^{l_2}_{t+2}, ..., a^{l_n}_{t+n}]$, we compute

$$\begin{aligned}\mathbb{P}(\tau_i^{\mathcal{O}}|h_t||\tau_i^{\mathcal{A}}) &= \mathbf{r}_{\tau_i}^\top \mathbb{P}(Q^{\mathcal{O}}|h_t||Q^{\mathcal{A}}) \\ &= \mathbf{m}_\infty^\top \mathbf{M}_{a^{l_n}o^{k_n}} \cdots \mathbf{M}_{a^{l_2}o^{k_2}} \mathbf{M}_{a^{l_1}o^{k_1}} \mathbf{m}_t. \end{aligned} \tag{1}$$

These operators can also be used to produce n-step predictions (i.e., the probability $\mathbb{P}(o^k_{t+n}|h_t||a^l_{t+n})$ of seeing an observation, o^k, after taking action, a^l, n-steps in the future) by

$$\mathbb{P}(o^k_{t+n}|h_t||a^l_{t+n}) = \mathbf{m}_\infty^\top \mathbf{M}_{a^l o^k} (\mathbf{M}_\star)^{n-1} \mathbf{m}_t,$$

where $\mathbf{M}_\star = \sum_{a^l o^k \in \mathcal{A}\times\mathcal{O}} \mathbf{M}_{a^l o^k}$ is a matrix that can be computed once and stored as a parameter for quick computation (Wiewiora, 2007).

Lastly, the operators provide a convenient method for updating the predictive state, defined by the prediction vector $\mathbf{m}_t$, as an agent tracks through a system and receives observations. The prediction vector $\mathbf{m}_t$ is updated to $\mathbf{m}_{t+1}$ after an agent takes an action a^l and receives observation o^k using

$$\begin{aligned}\mathbf{m}_{t+1} &= \mathbb{P}(\mathcal{Q}^{\mathcal{O}}|h_{t+1}||\mathcal{Q}^{\mathcal{A}}) \\ &= \mathbb{P}(\mathcal{Q}^{\mathcal{O}}|h_t a^l o^k||\mathcal{Q}^{\mathcal{A}}) \\ &= \frac{\mathbf{M}_{a^l o^k}\mathbf{m}_t}{\mathbf{m}_\infty^\top \mathbf{M}_{a^l o^k}\mathbf{m}_t}. \end{aligned} \tag{2}$$

Together, the elements of $\langle \mathcal{O}, \mathcal{A}, \mathcal{Q}, \mathcal{F}, \mathbf{m}_0 \rangle$ (where $\mathcal{F}$ is understood to contain the linear operators described above and the normalizer) thus provide a succinct model of a system, which allows for the efficient computation of event probabilities and also facilitates conditioning upon observed histories.

2.4 Learning PSRs

There is a considerable amount of literature describing different approaches to learning PSRs. We provide an overview of the standard approaches, as Section 3.2 describes, in detail, the efficient compressed learning approach we propose.[3]

In general, PSR learning approaches may be divided into two distinct classes: discovery-based and subspace-based. In the discovery-based approach, a form of combinatorial search is used to discover the (minimal) core set of tests, and the PSR model is then computed in a straightforward manner given the explicit knowledge of $\mathcal{Q}$ (James and Singh, 2004; James et al., 2005). This method generates an exact PSR model. However, the combinatorial search required to find $\mathcal{Q}$ precludes the use of this approach in domains of even moderate cardinality.

Unlike the discovery-based approaches, subspace-based approaches obviate the need for determining $\mathcal{Q}$ exactly (Hsu et al., 2008; Boots et al., 2010; Rosencrantz et al., 2004).

3. For a slightly more detailed discussion of existing PSR learning approaches see Wiewiora (2007).

Instead, subspace-identification techniques (e.g., spectral methods) are used in order to find a subspace that is a linear transformation of the subspace defined by $\mathcal{Q}$ (Rosencrantz et al., 2004). The linear nature of the PSR model allows the use of this transformed PSR model in place of the exact PSR model without detriment. Specifically, it can be shown that the probabilities obtained via such a transformed model are consistent with those obtained via the true model (Boots et al., 2010).

Formally, one first specifies a large (non-minimal) core set of tests $\mathcal{T}$ and a set of histories $\mathcal{H}$. Next, one defines two *observable matrices* $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}$, $\boldsymbol{\mathcal{P}}_{\mathcal{H}}$, and $|\mathcal{A}| \times |\mathcal{O}|$ *observable matrices* $\boldsymbol{\mathcal{P}}_{\mathcal{T},ao,\mathcal{H}}$ (one for each action-observation pair). $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}$ is a $|\mathcal{T}| \times |\mathcal{H}|$ matrix which contains the joint probabilities of all specified tests and histories. $\boldsymbol{\mathcal{P}}_{\mathcal{H}}$ is a $|\mathcal{H}| \times 1$ vector containing the marginal probabilities of each possible history. And each $\boldsymbol{\mathcal{P}}_{\mathcal{T},ao,\mathcal{H}}$ is a $|\mathcal{T}| \times |\mathcal{H}|$ matrix containing the the joint probabilities of all specified tests and histories where a particular action-observation pair (indicated by the subscript) is appended to the history (Boots et al., 2010). These observable matrices can be viewed as submatrices of $\mathbf{H}$, the system dynamics matrix (e.g., $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}} = [\mathbf{H}]_{\mathcal{T},\mathcal{H}}$). We also define matrices $\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}}$ and $\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,\mathcal{H}}$ $\forall ao \in \mathcal{A} \times \mathcal{O}$ analogously but with $\mathcal{Q}$ replacing $\mathcal{T}$ (e.g., $\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}} = [\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}]_{\mathcal{Q},*}$).

Under the assumption that the empty history occurs first in the lexicographic ordering of $\mathcal{H}$, the discovery-based approach builds a PSR model by

$$\mathbf{m}_0 = [\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}}]_{*,1} \tag{3}$$

$$\mathbf{m}_\infty^\top = \boldsymbol{\mathcal{P}}_{\mathcal{H}}^\top (\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}})^\dagger, \tag{4}$$

$$\mathbf{M}_{ao} = \boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,\mathcal{H}} (\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}})^\dagger, \tag{5}$$

while the subspace-based approach builds a model by

$$\boldsymbol{\beta}_0 = [\mathbf{Z}\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}]_{*,1} \tag{6}$$

$$\boldsymbol{\beta}_\infty^\top = \boldsymbol{\mathcal{P}}_{\mathcal{H}}^\top (\mathbf{Z}\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}})^\dagger, \tag{7}$$

$$\mathbf{B}_{ao} = \mathbf{Z}\boldsymbol{\mathcal{P}}_{\mathcal{T},ao,\mathcal{H}} (\mathbf{Z}\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}})^\dagger, \tag{8}$$

where $\mathbf{Z}$ is the projection matrix defining the subspace used for learning, which satisfies certain conditions. The conditions upon $\mathbf{Z}$ and the standard selection criterion for choosing it are elucidated in Section 2.5 below.

From these equations we see that PSR learning, in both the subspace and discovery paradigms, corresponds to a set of regression problems. The pseudoinverses in (3)-(8) corresponding to solutions to a set regression problems. For example, in the learning of $\mathbf{m}_\infty$ the columns of $\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}}$ correspond to samples in the regression (i.e., each history is a sample), the rows to features (i.e., each test is a feature), and the regression targets are the entries of $\boldsymbol{\mathcal{P}}_{\mathcal{H}}$ (i.e., the marginal history vector).

In general, the complexity of the discovery-based learning approach is dominated by the combinatorial search for the set of core tests. In the worst case this search has time-complexity $O((|\mathcal{A}||\mathcal{O}|)^L)$, where L is the max-length of a trajectory (i.e., execution trace) used to learn the model. If the minimal core set of tests is provided as input, the discovery-based method has complexity $O(|\mathcal{H}||\mathcal{Q}|^2)$; however, the assumption that the minimal core set of tests is known is not realistic in practice. In contrast, the subspace-based approach has time-complexity $O(|\mathcal{H}||\mathcal{T}|d_{\mathbf{Z}})$, where $d_{\mathbf{Z}}$ is the column-dimension of $\mathbf{Z}$. If the size of the minimal core set of tests is known (an unrealistic assumption) then $d_{\mathbf{Z}} = |\mathcal{Q}|$.

2.5 Transformed Representations

PSR models learned via the subspace method are often referred to as transformed PSRs (TPSRs), since they learn a model that is an invertible transform of a standard PSR model. More formally, given the set of linear parameters defining a PSR model and an invertible matrix $\mathbf{J}$, we can construct a TPSR by applying $\mathbf{J}$ as a linear operator to each parameter. That is, we set $\boldsymbol{\beta}_0 = \mathbf{J}\mathbf{m}_0$, $\boldsymbol{\beta}_\infty^\top = \mathbf{m}_\infty^\top \mathbf{J}^{-1}$, and $\mathbf{B}_{ao} = \mathbf{J}\mathbf{M}_{ao}\mathbf{J}^{-1}\ \forall ao \in \mathcal{A} \times \mathcal{O}$, and these new transformed matrices constitute the TPSR model (Boots and Gordon, 2011). It is easy to see that the $\mathbf{J}$'s cancel out in the prediction equation (1) and update equation (2). Intuitively, TPSRs can be thought of as maintaining a predictive state upon an invertible linear transform of the state defined by the tests in the minimal core set.

In practice, the matrix $\mathbf{J}$ is determined by the projection matrix $\mathbf{Z}$, which is used during learning in the subspace-based paradigm. To make the relationship between $\mathbf{J}$ and $\mathbf{Z}$ explicit, we define the following matrices: $\mathbf{R} = (\mathbf{r}_{\tau_i}, \mathbf{r}_{\tau_2}, ..., \mathbf{r}_{\tau_{|\mathcal{T}|}})^\top \in \mathbb{R}^{\mathcal{T} \times \mathcal{Q}}$, with each row i corresponding to the linear function mapping the probabilities of tests in the minimal core set to the probability of test τ_i (i.e., the $\mathbf{r}_{\tau_i}$ as defined in Equation 1); $\mathbf{N} = \text{diag}(\boldsymbol{\mathcal{P}}_\mathcal{H}) \in \mathbb{R}^{\mathcal{H} \times \mathcal{H}}$, with the marginal history probabilities along the diagonal; and, $\mathbf{Q} \in \mathbb{R}^{\mathcal{Q} \times \mathcal{H}}$, with each column j equal to the expected probability vector for the tests in the minimal core set given that history h_j has been observed (i.e., $[\mathbf{Q}]_{*,j} = \mathbf{M}_{h_j}\mathbf{m}_0$). These matrices can then be used to define a factorization of the observable matrices. In particular, Boots et al. (2010) show that

$$\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}} = \mathbf{R}\mathbf{Q}\mathbf{N} \tag{9}$$

and that

$$\boldsymbol{\mathcal{P}}_{\mathcal{T},ao,\mathcal{H}} = \mathbf{R}\mathbf{M}_{ao}\mathbf{Q}\mathbf{N} \tag{10}$$

holds for all $ao \in \mathcal{A} \times \mathcal{O}$.

Examining the equations for the different learning methods (i.e., Equations 3 and 6) and using the factorizations given in (9) and (10), we see first that for the discovery-based method, which learns a true untransformed PSR, we have that

$$\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}} = \mathbf{I}\mathbf{Q}\mathbf{N},$$

where $\mathbf{I}$ is the identity. In this case the set of tests in $\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}}$ is the minimal core set, and thus the core set mapping operator $\mathbf{R}$ is replaced by the identity. Similarly, we have

$$\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,\mathcal{H}} = \mathbf{I}\mathbf{M}_{ao}\mathbf{Q}\mathbf{N}.$$

Thus for the discovery method

$$\begin{aligned}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,\mathcal{H}}(\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}})^\dagger &= \mathbf{M}_{ao}\mathbf{Q}\mathbf{N}(\mathbf{Q}\mathbf{N})^\dagger \\ &= \mathbf{M}_{ao},\end{aligned}$$

where we used the fact that $\mathbf{QN}$ is full column-rank by definition. By contrast, for the subspace learning algorithm we have, assuming that $\mathbf{ZR}$ has full row-rank,

$$\begin{aligned}\mathbf{B}_{ao} &= \mathbf{Z}\boldsymbol{\mathcal{P}}_{\mathcal{T},ao,\mathcal{H}}(\mathbf{Z}\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}})^\dagger \\ &= \mathbf{Z}\mathbf{R}\mathbf{M}_{ao}\mathbf{Q}\mathbf{N}(\mathbf{Z}\mathbf{R}\mathbf{Q}\mathbf{N})^\dagger \\ &= \mathbf{Z}\mathbf{R}\mathbf{M}_{ao}\mathbf{Q}\mathbf{N}(\mathbf{Q}\mathbf{N})^\dagger(\mathbf{Z}\mathbf{R})^\dagger \\ &= \mathbf{Z}\mathbf{R}\mathbf{M}_{ao}(\mathbf{Z}\mathbf{R})^\dagger,\end{aligned} \tag{11}$$

where we again used the fact that $\mathbf{QN}$ has full column-rank. If we further assume that $\mathbf{ZR}$ is invertible (i.e., is square in addition to being full row rank) then (11) simplifies to

$$\mathbf{ZRM}_{ao}(\mathbf{ZR})^{\dagger} = \mathbf{ZRM}_{ao}(\mathbf{ZR})^{-1}.$$

Similar results hold for $\boldsymbol{\beta}_{\infty}$ and $\boldsymbol{\beta}_0$, showing that the subspace learning method does, in fact, return TPSRs in the case where $\mathbf{ZR}$ is invertible, and in this case we have a transformed representation with $\mathbf{J} := \mathbf{ZR}$.

The final piece of a TPSR is the specification of $\mathbf{Z}$, the projection matrix defining the subspace used during learning (and implicitly defining the transformation matrix $\mathbf{J}$). We know from the above derivations that $\mathbf{Z}$ must be chosen such that $\mathbf{ZR}$ is invertible. The standard method for guaranteeing this is by choosing $\mathbf{Z}$ via spectral techniques; that is, $\mathbf{Z}$ is set to be $\mathbf{U}^{\top}$, the transpose of the matrix of right singular vectors (from the thin-SVD of $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}$) (Boots et al., 2010).

The TPSR approach can also be extended to work with features of tests and histories (Boots et al., 2010; Boots and Gordon, 2011) and/or kernelized to work in continuous domains (Boots and Gordon, 2013). This is useful in cases where the observation space is too complex for standard tests to be used (i.e., when the observation space is structured or continuous). When features of tests and histories are used, however, they are usually specified in a domain-specific manner (Boots et al., 2010). Some authors have also used randomized Fourier methods to efficiently approximate kernel-based feature selection (Boots and Gordon, 2011). These methods are quite successful in continuous domains (Boots et al., 2010; Boots and Gordon, 2011, 2013).

In contrast, the benefit of the algorithm presented in Section 3.2 is that it implicitly performs general purpose feature selection (for discrete-domains) using random compression. And this is especially useful in cases where it is difficult to know a sufficient set of features prior to training (e.g., in the case where the model is being learned incrementally). Moreover, the motivation between the compression performed in this work and the above-mentioned feature-based techniques are disjoint in that the goal of this work is to provide compression for efficient learning whereas the above-mentioned feature-based learning strategies are motivated by the need to cope with continuous or structured observation spaces. See Section 7.2 for further discussion on the relationship between this work and these alternative feature-based approaches.

3. Compressed Predictive State Representations

In this section, we describe our extension of PSRs, compressed predictive state representations (CPSRs). The CPSR approach, at its core, combines the state-of-the-art in subspace PSR learning with recent advancements in compressed sensing. This marriage provides an extremely efficient and principled approach for learning accurate transformed approximations of PSRs in complex systems, where learning a full PSR is simply intractable. Section 3.1 motivates the use of compressed sensing techniques in a PSR learning algorithm, and Section 3.2 describes the efficient CPSR learning approach we propose.

3.1 Foundations: Compressed Estimation

Despite the fact that non-compressed subspace-based algorithms, such as TPSR, can specify a small dimension for a transformed space (e.g., by removing the least important singular vectors of $\mathbf{U}$ as in done in Rosencrantz et al. (2004) and analyzed in Kulesza et al. (2014)), there are still a number of computational limitations. To begin, TPSRs require that the $|\mathcal{T}| \times |\mathcal{H}|$ matrix, $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}$, be estimated in its entirety, and that the $\boldsymbol{\mathcal{P}}_{\mathcal{T},ao,\mathcal{H}}$ matrices be partially estimated as well. Moreover, since the naive TPSR approach must compute a spectral decomposition of $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}$ it has computational complexity $O(|\mathcal{H}||\mathcal{T}|^2)$, in the batch (and incremental mini-batch) setting, assuming the observable matrices are given as input. Thus in domains that require many (possibly long) trajectories for learning or that have large observation spaces, such as those described in Section 6, the naive TPSR approach becomes intractable, since $|\mathcal{H}|$ and $|\mathcal{T}|$ both scale as $O(L|Z|)$, where L is the max length of a trajectory in a training set Z of size $|Z|$.[4][5] In order to circumvent these computational constraints (and provide a form of regularization), the CPSR learning algorithm we propose (in the next section) performs *compressed estimation*.

This method is borrowed from the field of compressed sensing and works by projecting matrices down to low-dimensional spaces determined via randomly generated bases. More formally, a $m \times n$ matrix $\mathbf{Y}$ is compressed to a $d \times n$ matrix $\mathbf{X}$ (where $d < m$) by

$$\mathbf{X} = \boldsymbol{\Phi}\mathbf{Y}, \tag{12}$$

where $\boldsymbol{\Phi}$ is a $d \times m$ *Johnson-Lindenstrauss matrix* (i.e., a matrix satisfying the Johnson-Lindenstrauss lemma) (Baraniuk and Wakin, 2009). Intuitively, a Johnson-Lindenstrauss matrix is a random matrix defining a low-dimensional embedding which approximately preserves Euclidean distances between projected points (i.e., the projection preserves the dot-product between vectors). Different choices for $\boldsymbol{\Phi}$ are discussed in Section 6. It is worth noting that in our case, the matrix multiplication in (12) is in fact performed "online", and the matrices corresponding to $\mathbf{X}$ and $\boldsymbol{\Phi}$ are never explicitly held in memory (details in Section 3.2).

The fidelity of this technique depends what is called the *sparsity* of the matrix $\mathbf{Y}$. Sparsity in this context refers to the maximum number of non-zero entries which occur in any column of $\mathbf{Y}$. Formally, if we denote a column vector of $\mathbf{Y}$ by $\mathbf{y}_i$, we say that a matrix is k-sparse if

$$k \geq ||\mathbf{y}_i||_0 \ \ \forall \mathbf{y}_i \in \mathbf{Y},$$

where $||\cdot||_0$ denotes Donoho's zero "norm" (which simply counts the number of non-zero entries in a vector).

The technique is very well suited for application to PSRs. Informally, the sparsity condition is the requirement that for every history h_j, only a subset of all tests have non-zero probabilities (a more formal definition appears in the theory section below). This

4. Note that $|\mathcal{H}|$ and $|\mathcal{T}|$ scale linearly with the number of *observed* test/histories. The $O(L|Z|)$ bound is thus pessimistic in that it assumes each training instance is unique.
5. It is worth noting that no explicit bounds on the sample complexity of PSR learning have been elucidated. However, the sample complexity bounds of Hsu et al. (2008) provide results for a special case of TPSR learning (i.e., no actions and only single length tests and histories). In general, PSR approaches are consistent estimators but cannot be assumed to be data efficient (thus emphasizing the need to accommodate large sample sizes).

seems realistic in many domains. For example, in the PocMan domain described below, we empirically found the average column sparsity of the matrices to be roughly 0.018% (i.e., approximately 0.018% of entries in a column were non-zero). Moreover, as we will demonstrate empirically in Section 6, certain noisy observation models induce sparsity that can be exploited by this approach.

3.2 Efficiently Learning CPSRs

In this section, we present our novel compressed predictive state representation (CPSR) learning algorithm. The algorithm builds upon the work of Hamilton et al. (2013), extending their algorithm in a number of important ways. Specifically, the algorithm presented here (1) permits a broad class of compression matrices (any full-rank projection matrix satisfying the JL lemma), (2) includes optional compression of both histories and tests, and (3) combines compressed sensing with spectral methods in order to provide numerical stability and facilitate incremental (and even online) model-learning. Section 3.2.1 describes the foundational batch-learning algorithm. Section 3.2.2 describes how to incrementally update a learned model with new data efficiently for deployment in online settings.

3.2.1 Batch Learning of CPSRs

To begin, we define two injective functions: $\phi_{\mathcal{T}} : \mathcal{T} \rightarrow \mathbb{R}^{d_{\mathcal{T}}}$ and $\phi_{\mathcal{H}} : \mathcal{H} \rightarrow \mathbb{R}^{d_{\mathcal{H}}}$. These functions are independent mappings from tests and histories, respectively, to columns of independent random full-rank Johnson-Lindenstrauss (JL) projection matrices $\boldsymbol{\Phi}_{\mathcal{T}} \in \mathbb{R}^{d_{\mathcal{T}} \times \mathcal{T}}$ and $\boldsymbol{\Phi}_{\mathcal{H}} \in \mathbb{R}^{d_{\mathcal{H}} \times \mathcal{H}}$, respectively. The matrices are defined via these functions since the full sets $\mathcal{T}$ and $\mathcal{H}$ may not be known a priori, and we can get away with this "lazy" specification since the columns of JL projection matrices are determined by independent random variables.

Next, given a training trajectory z of action-observation pairs of any length, let $\mathbb{I}_{h_j}(z)$ be an indicator function taking a value of 1 if the action-observations pairs in z correspond to h_j. Similarly define $|\cdot|$ as the length of a sequence (e.g., of action-observation pairs) and let $\mathbb{I}_{h_j,\tau_i}(z)$ be an indicator function taking a value of 1 if z can be partitioned such that, starting from some index k within the sequence, there are $|h_j|$ action-observation pairs corresponding to those in $h_j \in \mathcal{H}$ and the next $|\tau_i|$ pairs correspond to those in $\tau_i \in \mathcal{T}$.[6]

Given a batch of of training trajectories Z we compute compressed estimates

$$\begin{aligned} \hat{\boldsymbol{\Sigma}}_{\mathcal{H}} &= \boldsymbol{\Phi}_{\mathcal{H}} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{H}} \\ &= \sum_{z \in Z} \sum_{h_j \in \mathcal{H}} \mathbb{I}_{h_j}(z) \phi_{\mathcal{H}}(h_j) \end{aligned} \tag{13}$$

and

$$\begin{aligned} \hat{\boldsymbol{\Sigma}}_{\mathcal{T},\mathcal{H}} &= \boldsymbol{\Phi}_{\mathcal{T}} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{T},\mathcal{H}} \boldsymbol{\Phi}_{\mathcal{H}}^{\top} \\ &= \sum_{z \in Z} \sum_{t_i, h_j \in \mathcal{T} \times \mathcal{H}} \mathbb{I}_{h_j,t_i}(z) \left[\phi_{\mathcal{T}}(t_i) \oplus \phi_{\mathcal{H}}(h_j) \right] \end{aligned} \tag{14}$$

6. In this work we use $k = 0$. That is we do not use the suffix history estimation algorithm (Wolfe et al., 2005), where k is varied in the range $[0, |z|)$. Using $k = 0$ minimizes dependencies between estimation errors as the same samples are not used to get estimates for multiple histories.

of the observable matrices $\boldsymbol{\mathcal{P}}_{\mathcal{H}}$ and $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}$, respectively, where $\oplus$ denotes the tensor (outer) product of two vectors.[7]

Next, we compute the $\hat{\mathbf{U}}\hat{\mathbf{S}}\hat{\mathbf{V}}^\top$ rank-d' thin SVD of $\hat{\boldsymbol{\Sigma}}_{\mathcal{T},\mathcal{H}}$:

$$(\hat{\mathbf{U}}, \hat{\mathbf{S}}, \hat{\mathbf{V}}) = \text{SVD}(\hat{\boldsymbol{\Sigma}}_{\mathcal{T},\mathcal{H}}). \tag{15}$$

Given these matrices we can construct

$$\mathbf{c}_1 = \hat{\mathbf{S}}\hat{\mathbf{V}}^\top \mathbf{e} \tag{16}$$

and

$$\mathbf{c}_\infty^\top = \hat{\boldsymbol{\Sigma}}_{\mathcal{H}}^\top \hat{\mathbf{V}}\hat{\mathbf{S}}^{-1}, \tag{17}$$

the compressed and transformed estimates of $\mathbf{m}_1$ and $\mathbf{m}_\infty$, respectively, where $\mathbf{e}$ is a vector such that $\boldsymbol{\Phi}_{\mathcal{H}}\mathbf{e} = (1, 0, 0, ..., 0)^\top$. In practice this can be guaranteed by defining a modified history map $\phi'_{\mathcal{H}} : \mathcal{H} \to \mathbb{R}^{d+1}$ such that that for the null history, $\emptyset$, $\phi'_{\mathcal{H}}(\emptyset) = (1, 0, 0, ..., 0)^\top$ and that $\phi'_{\mathcal{H}}(h_j) = [0 \;\; \phi_{\mathcal{H}}(h_j)]$ for all $h_j \neq \emptyset$. This specification of $\mathbf{e}$ assumes that all $z \in Z$ are starting from a unique start state. If this is not the case, then we set $\mathbf{e}$ such that $\boldsymbol{\Phi}_{\mathcal{H}}\mathbf{e} = (1, 1, 1, ..., 1)^\top$, which again can be guaranteed without cost but in this case by simply adding a constant "dummy" column to the front of $\boldsymbol{\Phi}_{\mathcal{H}}$. In this latter scenario, we would, in fact, not be learning $\mathbf{c}_1$ exactly and instead would learn $\mathbf{c}_*$, an arbitrary feasible state as our start state. The uncertainty in our state estimate should decrease, however, as we update and track through our system and the process mixes (Boots et al., 2010). And indeed, the majority of domains without well-defined start-states are those for which there is significant mixing over time, so this technique should introduce only a small amount of error in practice.

Given the SVD of $\hat{\boldsymbol{\Sigma}}_{\mathcal{T},\mathcal{H}}$, we can also estimate the $\mathbf{C}_{ao}$ matrices, the compressed and transformed versions of the $\mathbf{M}_{ao}$ matrices, directly via a second pass over the data. First, however, we must define a third class of indicator functions on $z \in Z$: $\mathbb{I}_{h_j,ao,\tau_i}(z)$ takes value 1 if and only if the training sequence z can be partitioned such that, starting from some index k within the sequence, there are $|h_j| + 1$ action-observation pairs corresponding to h_j appended with a particular $ao \in \mathcal{A} \times \mathcal{O}$ and the next $|\tau_i|$ correspond to those in τ_i. In other words, $\mathbb{I}_{h_j,ao,\tau_i}(z)$ is equivalent to $\mathbb{I}_{h'_j,\tau_i}(z)$, where a particular $ao \in \mathcal{A} \times \mathcal{O}$ is appended to the history h'_j. Using these indicators and the SVD matrices of $\hat{\boldsymbol{\Sigma}}_{\mathcal{T},\mathcal{H}}$, we compute, for each $ao \in \mathcal{A} \times \mathcal{O}$,

$$\mathbf{C}_{ao} = \sum_{z \in Z} \sum_{t_i, h_j \in \mathcal{T} \times \mathcal{H}} \mathbb{I}_{h_j,ao,t_i}(z) \left[\left(\hat{\mathbf{U}}^\top \phi_{\mathcal{T}}(t_i) \right) \oplus \left(\hat{\mathbf{S}}^{-1} \hat{\mathbf{V}}^\top \phi_{\mathcal{H}}(h_j) \right) \right]. \tag{18}$$

Thus, in two passes over the data, we are able to efficiently construct our CPSR model parameters. The primary computational savings engendered by this approach is in the computation of the pseudoinverse of $\hat{\boldsymbol{\Sigma}}_{\mathcal{T},\mathcal{H}}$, which we implicitly compute via an SVD. Since we are performing pseudoinversion (i.e., SVD) on a compressed matrix, the computational

7. We do not normalize our probability estimates in the estimation equations since the normalization constants cancel out during learning.

complexity is uncoupled from the number of tests and histories in the set of observed trajectories Z. Recalling that L denotes the max length of a trajectory in Z and letting $|Z|$ denote the number of trajectories in the set Z, this approach has a computational complexity of

$$O\left(L|Z|d_{\mathcal{H}}d_{\mathcal{T}} + d_{\mathcal{T}}^2 d_{\mathcal{H}}\right) = O\left(L|Z|\right) \tag{19}$$

since $d_{\mathcal{H}}$ and $d_{\mathcal{T}}$ are a user-specified constants (assuming the standard cubic computational cost for the SVD).[8] Without compression (i.e., with naive TPSR), a computational cost of

$$O\left(L|Z| + |\mathcal{H}||\mathcal{T}|^2\right) = O\left(L^3|Z|^3\right) \tag{20}$$

is incurred.

In addition to these computational savings, the above approach has the added benefit of not requiring that $\mathcal{T}$ and $\mathcal{H}$ be known in entirety prior to learning. This is especially important in the case where we want to alternate model-learning and planning/exploration phases using incremental updates (described below), as it is very unlikely that all possible tests and histories are observed in the first round of exploration. Performing SVD on the compressed matrices also induces a form of regularization (similar to L_2 regularization) on the learned model, where variance is reduced at the cost of a controlled bias (details in Section 4).

3.2.2 Incremental Updates to the Model

In addition to straightforward batch learning, it is also possible to incrementally update a learned model, given new training data, Z' (Boots and Gordon, 2011). This is especially useful in that it facilitates alternating exploration and exploitation phases. Of course, if such a non-blind alternating approach is used then the distribution of the training data changes (i.e., it becomes non-stationary), and the sampled trajectories can no longer be assumed to be i.i.d.. Despite this theoretical drawback, Ong et al. (2012) show that non-blind sampling approaches can lead to better planning results in a small sample setting.[9]

Briefly, we obtain a new $\hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}}$ estimate and update our $\hat{\mathbf{\Sigma}}_{\mathcal{H}}$ estimate using using (14) and (13) with Z'. Next, we update our SVD matrices, given our additive update to $\hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}}$, using the methods of (Brand, 2002). The $\mathbf{c}_1$ and $\mathbf{c}_\infty$ vectors are then re-computed exactly as in equations (16) and (17).

To obtain our $\mathbf{C}_{ao}^{\text{new}}$ matrices, we compute

$$\begin{aligned}\mathbf{C}_{ao}^{\text{new}} = &\sum_{z \in Z'} \sum_{t_i, h_j \in \mathcal{T} \times \mathcal{H}} \mathbb{I}_{h_j, ao, t_i}(z) \left[\left(\hat{\mathbf{U}}_{\text{new}}^\top \phi_{\mathcal{T}}(t_i)\right) \oplus \left(\hat{\mathbf{S}}_{\text{new}}^{-1} \hat{\mathbf{V}}_{\text{new}}^\top \phi_{\mathcal{H}}(h_j)\right)\right] \\ &+ \hat{\mathbf{U}}_{\text{new}}^\top \hat{\mathbf{U}}_{\text{old}} \mathbf{C}_{ao}^{\text{old}} \hat{\mathbf{S}}_{\text{old}} \hat{\mathbf{V}}_{\text{old}}^\top \hat{\mathbf{V}}_{\text{new}} \hat{\mathbf{S}}_{\text{new}}^{-1}.\end{aligned} \tag{21}$$

The first term in (21) corresponds to estimating the contribution to the new $\mathbf{C}_{ao}$ matrix from the new data, and the second term is the projection of the old $\mathbf{C}_{ao}$ matrix onto the new basis. Using the results of Brand (2002), the complexity of this update is

$$O(L'|Z'|(d_{\mathcal{T}}d_{\mathcal{H}} + (d')^3 + d'd_{\mathcal{T}}) + d_{\mathcal{T}}d'd_{\mathcal{H}}), \tag{22}$$

8. Section 4 describes how the choice of these constants affects the accuracy of the learned model.
9. In this work, where larger sample sizes were used, we did not find a significant benefit to goal-directed sampling and in fact saw detrimental effects in terms of planning ability and numerical stability during learning. See Section 7 for details.

where L' denotes the maximum length of a trajectory in Z'.

4. Theoretical Analysis of the Learning Algorithm

In the following section, we describe theoretical properties of the CPSR learning approach. Our analysis proceeds in two stages. First, we show that the learned model is consistent in the case where $d_{\mathcal{T}} \geq |\mathcal{Q}|$ and $d_{\mathcal{H}} \geq |\mathcal{Q}|$ (i.e., when no real compression occurs). Following this, we outline results bounding the induced approximation error (bias) and decrease in estimation error (variance) due to learning a compressed model.

The analysis included in this section is intended as a means to justify the compression technique and study the overall consistency of our algorithm. It also provides guidance for the choosing of a theoretically sound range of values for the projection size used in the algorithm.

4.1 Consistency of the Learning Approach

The following adapts the results of Boots et al. (2010) and shows the consistency of our learning approach when the random projection dimension is greater than or equal to the true underling dimension of the system (i.e., the size of the minimal core set of tests, $|\mathcal{Q}|$). We then describe the implications of this result for the case where we are in fact projecting down to a dimension smaller than $|\mathcal{Q}|$.

4.1.1 Consistency in the Non-Compressed Setting

We begin by noting a fundamental result from the TPSR literature. Recall the matrix $\mathbf{R} = (\mathbf{r}_{\tau_1}, \mathbf{r}_{\tau_2}, ..., \mathbf{r}_{\tau_{|\mathcal{T}|}})^\top \in \mathbb{R}^{\mathcal{T} \times \mathcal{Q}}$ where each row, $\mathbf{r}_i$, specifies the linear map

$$\mathbf{r}_{\tau_i}^\top \mathbb{P}(Q^{\mathcal{O}} | h_t || Q^{\mathcal{A}}) = \mathbb{P}(\tau_i^{\mathcal{O}} | h_t || \tau_i^{\mathcal{A}}).$$

Supposing that $d_{\mathcal{T}} \geq |\mathcal{Q}|$ and $d_{\mathcal{H}} \geq |\mathcal{Q}|$ and with $\mathbf{U}$ coming from the SVD of $\mathbf{\Sigma}_{\mathcal{T},\mathcal{H}}$, we have

$$\mathbf{c}_0 = (\mathbf{U}^\top \mathbf{\Phi}_{\mathcal{T}} \mathbf{R}) \mathbf{m}_0, \tag{23}$$

$$\mathbf{c}_\infty^\top = \mathbf{m}_\infty^\top (\mathbf{U}^\top \mathbf{\Phi}_{\mathcal{T}} \mathbf{R})^{-1}, \tag{24}$$

$$\mathbf{C}_{ao} = (\mathbf{U}^\top \mathbf{\Phi}_{\mathcal{T}} \mathbf{R}) \mathbf{M}_{ao} (\mathbf{U}^\top \mathbf{\Phi}_{\mathcal{T}} \mathbf{R})^{-1}. \tag{25}$$

That is, we simply recover a TPSR where $\mathbf{J} = (\mathbf{U}^\top \mathbf{\Phi}_{\mathcal{T}} \mathbf{R})$, and it has been shown that the above implies a *consistent* learning algorithm (Boots et al., 2010; Boots and Gordon, 2011). We note that $\mathbf{\Phi}_{\mathcal{T}}$ appears in these consistency equations, while $\mathbf{\Phi}_{\mathcal{H}}$ does not, emphasizing the different roles these two matrices occupy. This difference will play an important role in the theoretical analysis below.

4.1.2 Extension to the Compressed Case

In the case where $d_{\mathcal{T}} < |\mathcal{Q}|$ and/or $d_{\mathcal{H}} < |\mathcal{Q}|$ things are not as straightforward. Specifically, equations (23)-(25) no longer hold as $(\mathbf{U}^\top \mathbf{\Phi}_{\mathcal{T}} \mathbf{R})$ is no longer invertible (it is in fact, no longer square), since the SVD is taken on $\mathbf{\Sigma}_{\mathcal{T},\mathcal{H}}$ which has rank less than $|\mathcal{Q}|$ when $d_{\mathcal{T}} < |\mathcal{Q}|$ and/or $d_{\mathcal{H}} < |\mathcal{Q}|$ while the column dimension of $\mathbf{R}$ is $|\mathcal{Q}|$. The primary focus of our theoretical

analysis is the effect of this fact, i.e. $(\mathbf{U}^\top \mathbf{\Phi}_\mathcal{T} \mathbf{R})$ not being invertible. We show how we can view $\mathbf{\Phi}_\mathcal{T}$ as inducing a form of compressed linear regression, and we provide bounds on the excess risk of learning within a compressed space.

There is, however, the additional complication of $\mathbf{\Phi}_\mathcal{H}$ when $d_\mathcal{H} < |\mathcal{Q}|$, as in that setting it is no longer possible to remove $\mathbf{\Phi}_\mathcal{H}$ from the consistency equations (23)-(25). From the perspective of regression, $\mathbf{\Phi}_\mathcal{H}$ can be viewed as compressing the number of samples, while $\mathbf{\Phi}_\mathcal{T}$ can be viewed as compressing the features. In this work, we focus on the effect of compressing tests and provide detailed analysis of how compressing tests (i.e., features) affects the implicit linear regression performed. Zhou et al. (2007) discuss the effect of compressing samples during regression, a result that follows naturally from the Johnson-Lindenstrauss lemma, and in Section 7, we discuss these results and their relationship to this work. For completeness, Section 6 also provides an empirical analysis of the effects of compressing histories and tests versus compressing tests alone.

4.2 Effects of Compression

In what follows, we analyse the effects of compression by viewing $\mathbf{\Phi}_\mathcal{T}$ as inducing a form of compressed linear regression, where both the input data and targets are compressed.

4.2.1 Preliminaries

This approach is justified by noting that, as discussed in Section 2.4, in equations (17) and (18) of our learning algorithm we are in fact performing implicit linear regression. That is, for $(\hat{\mathbf{U}}, \hat{\mathbf{S}}, \hat{\mathbf{V}}) = \text{SVD}(\hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}})$,

$$\begin{aligned} \hat{\mathbf{V}}\hat{\mathbf{S}}^{-1} &= (\hat{\mathbf{U}}^\top \hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}})^\dagger \\ &= (\hat{\mathbf{U}}^\top \mathbf{\Phi}_\mathcal{T} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{T},\mathcal{H}} \mathbf{\Phi}_\mathcal{H}^\top)^\dagger. \end{aligned} \tag{26}$$

In other words, $\hat{\mathbf{V}}\hat{\mathbf{S}}^{-1}$ is the Moore-Penrose pseudoinverse of $\hat{\mathbf{U}}^\top \mathbf{\Phi}_\mathcal{T} \hat{\boldsymbol{\mathcal{P}}}_\mathcal{H} \mathbf{\Phi}_\mathcal{H}^\top$, and multiplication by $\hat{\mathbf{V}}\hat{\mathbf{S}}^{-1}$ is thus equivalent to performing least-squares linear regression.

Following the discussion in the previous section and to avoid unnecessary complication, we assume $\mathbf{\Phi}_\mathcal{H}$ has orthonormal columns (i.e., is not compressive) while analyzing the effects of compressing the tests. In the case where $\mathbf{\Phi}_\mathcal{H}$ has orthonormal columns, we define $\mathbf{\Sigma}_{\mathcal{T},ao,\mathcal{H}}$ as the compressed analogue of $\boldsymbol{\mathcal{P}}_{\mathcal{T},ao,\mathcal{H}}$, and see that (18) can be rewritten as

$$\begin{aligned} \mathbf{C}_{ao} &= (\hat{\mathbf{U}}^\top \hat{\mathbf{\Sigma}}_{\mathcal{T},ao,\mathcal{H}})(\hat{\mathbf{U}}^\top \hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}})^\dagger \\ &= (\hat{\mathbf{U}}^\top \mathbf{\Phi}_\mathcal{T} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{T},ao,\mathcal{H}} \mathbf{\Phi}_\mathcal{H}^\top)(\hat{\mathbf{U}}^\top \mathbf{\Phi}_\mathcal{T} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{T},\mathcal{H}} \mathbf{\Phi}_\mathcal{H}^\top)^\dagger \\ &= (\hat{\mathbf{U}}^\top \mathbf{\Phi}_\mathcal{T} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{T},ao,\mathcal{H}}) \mathbf{\Phi}_\mathcal{H}^\top (\mathbf{\Phi}_\mathcal{H}^\top)^\dagger (\hat{\mathbf{U}}^\top \mathbf{\Phi}_\mathcal{T} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{T},\mathcal{H}})^\dagger \qquad (27) \\ &= (\hat{\mathbf{U}}^\top \mathbf{\Phi}_\mathcal{T} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{T},ao,\mathcal{H}})(\hat{\mathbf{U}}^\top \mathbf{\Phi}_\mathcal{T} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{T},\mathcal{H}})^\dagger, \qquad (28) \end{aligned}$$

where (27)-(28) holds since $\mathbf{\Phi}_\mathcal{H}$ is assumed to have orthonormal columns. An analogous result holds for $\mathbf{c}_\infty$ and thus, $\mathbf{\Phi}_\mathcal{H}$ can, indeed, be omitted in our analysis (under the assumption that $\mathbf{\Phi}_\mathcal{H}^\top \mathbf{\Phi}_\mathcal{H} = \mathbf{I}$).

Moreover, we ignore the $\hat{\mathbf{U}}^\top$ term in what follows, which is justified in the case where $d' = d_\mathcal{T}$ (i.e., when the truncated SVD dimension is equal to the test compression dimension). This $d' = d_\mathcal{T}$ condition is very mild in the sense that the use of SVD during learning is

primarily motivated by the need to efficiently compute pseudoinverses, which facilitates the efficient batch and incremental model-learning algorithms. That is, the SVD is not used as a dimensionality reduction technique, as random projections are used in that role.[10] Thus, under the assumption that $d' = d_{\mathcal{T}}$, we have that

$$\mathbf{A}\mathbf{x} = \mathbf{b} \Rightarrow \hat{\mathbf{U}}^\top \mathbf{A}\mathbf{x} = \hat{\mathbf{U}}^\top \mathbf{b}$$

holds, since $\hat{\mathbf{U}}^\top$ is orthonormal for $d' = d_{\mathcal{T}}$. Thus, the appearance of $\hat{\mathbf{U}}$ in the pseudoinverse is inconsequential in an analysis of the effect of compressing prior to regression.

To simplify the analysis one step further, we assume that our test set is a minimal core set $\mathcal{Q}$. Therefore, random projections are applied on $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},\mathcal{H}}$ and $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},ao,\mathcal{H}}$ matrices. The projections from over-complete test sets with rank bigger than $|\mathcal{Q}|$ down to $d_{\mathcal{T}}$ dimensions can be achieved by first projecting to size $|\mathcal{Q}|$ and then projecting from $|\mathcal{Q}|$ to $d_{\mathcal{T}}$. By the results of Section 4.1, this first projection leads to a consistent model, i.e. a model that is a linear transform of the model learned directly from $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},\mathcal{H}}$ and $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},ao,\mathcal{H}}$ matrices, since $\hat{\mathbf{U}}^\top \boldsymbol{\Phi}_{\mathcal{T}} \mathbf{R}$ is invertible with probability 1 when the projected dimension is equal to $|\mathcal{Q}|$ (Boots et al., 2010). The assumption that we work with the $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},\mathcal{H}}$ and $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},ao,\mathcal{H}}$ matrices directly (as apposed to invertible transforms of them) simplifies the analysis below in that we can elucidate our sparsity assumptions etc. directly in terms of the minimal core set of tests instead of random linear functions of tests in the minimal core set. This assumption is mild in that we could work with these random invertible linear transforms and discuss the discrepancy between a "random" TPSR (i.e., a TPSR defined via a random linear transform) and a compressed version of this "random" TPSR, and this discussion would be analogous to that which is provided below, albeit with more cumbersome and unnecessarily complex derivations. The assumption that we work with the minimal core set of tests simply allows for a more interpretable and less cluttered analysis.

Now, we define

$$\mathbf{B}_{ao} = \boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,\mathcal{H}} (\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}})^\dagger, \quad \boldsymbol{\beta}_\infty = (\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}})^\dagger \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{H}}.$$

Since $\mathcal{Q}$ is a minimal core set of tests, the above is a TPSR representation (Boots et al., 2010; Rosencrantz et al., 2004). Assume we have enough histories in $\mathcal{H}$ such that matrices are full rank. Defining $\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}$ and $\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}$ to be the vectors containing the joint probabilities of all tests in the minimal core set and a fixed history h, we have that (by the linearity of PSRs)

$$\forall h : \boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h} = \mathbf{B}_{ao} \boldsymbol{\mathcal{P}}_{\mathcal{Q},h}, \quad \boldsymbol{\mathcal{P}}_h = \boldsymbol{\beta}_\infty^\top \boldsymbol{\mathcal{P}}_{\mathcal{Q},h}.$$

One can thus think of finding the $\mathbf{B}_{ao}$ and $\boldsymbol{\beta}_\infty$ parameters as regression problems, having the estimates of $\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}$s as noisy input features. We also have noisy observations of the outputs $\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}$ and $\boldsymbol{\mathcal{P}}_h$. Since the sample set suffers from the error in variables problem (i.e., is noisy both on the input and output values) direct regression in the original space might result in large estimation error. Therefore, we apply random projections, reducing the

10. As noted in Section 7.1.3 it is sometimes beneficial to use $d' < d_{\mathcal{T}}$ and/or discard very small singular values in order to improve the numerical stability of computing inverses during learning. However, this issue of numerical stability is orthogonal to the analysis presented in this section.

estimation error (variance) at the cost of a controlled approximation error (bias). And we get the added benefit that working in the compressed space also helps with the computation complexity of the algorithm.

Note that there is an inherent difference between our work and the TPSR framework. In TPSR, one seeks to find concise linear transformations of the observation matrices, whereas CPSR seeks to find good approximations in a compressed space (which cannot be linearly transformed to the original model). That said, approximate variants of the TPSR learning algorithm have been analyzed from the perspective of compressed regression (albeit without appealing to the compressed sensing framework we employ) (Kulesza et al., 2014; Boots and Gordon, 2010). For example, Kulesza et al. (2014) analyze low-rank TPSR models where the rank of the learned model is made less than $|\mathcal{Q}|$ by removing the least significant singular vectors of $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}$. We reiterate, however, that these analyses are distinct from the analysis presented in this work, as we analyze low-rank models where the rank is reduced via random projection-based compression (not by removing least-significant singular vectors). The following sections provide an analysis of the error induced by this compression and how the error propagates through the application of several compressed operators.

4.2.2 Error of One Step Regression

When the size of the projections is smaller than the size of the minimal core set, we have the implicit regression performed on a compressed representation. The update operators are thus the result of compressed ordinary least-squares regression (COLS). There are several bounds on the excess risk of regression in compressed spaces (Maillard and Munos, 2009, 2012; Fard et al., 2012, 2013). In this section, we assume the existence of a generic upper bound for the error of COLS.

Assume we have a target function $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + b(\mathbf{x})$ where $\mathbf{x}$ is in a k-sparse D-dimensional space, and $b(\cdot)$ is the bias of the linear fit. We observe an i.i.d. sample set $\{(\mathbf{x}_i, f(\mathbf{x}_i) + \eta_i)\}_{i=1}^n$, where η_i's are independent zero-mean noise terms for which the maximum variance is bounded by σ_η^2, and $\mathbf{x}_i$'s are sampled from a distribution ρ. Let $\hat{f}_d(\mathbf{x})$ be the compressed least-squares solution on this sample with a random projection of size d. That is, $\hat{f}_d(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\Phi}^\top \hat{\mathbf{w}}_d$ with

$$\hat{\mathbf{w}}_d = (\boldsymbol{\Phi}\mathbf{X}^\top \mathbf{X}\boldsymbol{\Phi}^\top)^{-1}(\boldsymbol{\Phi}\mathbf{X}^\top)\mathbf{y} \in \mathbb{R}^d,$$

where $\mathbf{X} \in \mathbb{R}^{n \times D}$ is a design matrix, $\mathbf{y} \in \mathbb{R}^n$ is a vector of training targets, and $\boldsymbol{\Phi} \in \mathbb{R}^{d \times D}$ is a random projection matrix. Define $\|g(\mathbf{x})\|_{\rho(\mathbf{x})} = \sqrt{\mathbb{E}_{\mathbf{x}\sim\rho}(g(\mathbf{x}))^2}$ to be the weighted L^2 norm under the sampling distribution. We assume the existence of a generic upper bound function ϵ, such that with probability no less than $1 - \delta$

$$\|f(\mathbf{x}) - \hat{f}_d(\mathbf{x})\|_{\rho(\mathbf{x})} \leq \epsilon(n, D, d, \|\mathbf{w}\|^2, \|\mathbf{x}\|^2_{\rho(\mathbf{x})}, \|b(\mathbf{x})\|^2_{\rho(\mathbf{x})}, \sigma_\eta^2, \delta). \tag{29}$$

The effectiveness of the compressed regression is largely dependent on how the $\|\mathbf{w}\|\|\mathbf{x}\|_{\rho(\mathbf{x})}$ term behaves compared to the norm of the target values. We refer the reader to the discussions in Maillard and Munos (2009) and Maillard and Munos (2012) on the $\|\mathbf{w}\|\|\mathbf{x}\|_{\rho(\mathbf{x})}$ term. In the case of working with PSRs, we have that the probability of the tests are often highly correlated. Using this property, we will show that $\|\mathbf{w}\|^2$ can be bounded well below its size.

In order to use these bounds, we need to consider the sparsity assumptions in our compressed PSR framework. We formalize the inherent sparsity, discussed in previous sections, as follows: For all h, $\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}$ and $\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}$ are k-sparse. Given that the empirical estimates of zero elements in these vectors are not noisy, for $\Delta_x = \hat{\mathbf{P}}_{\mathcal{Q},h} - \boldsymbol{\mathcal{P}}_{\mathcal{Q},h}$ we have that Δ_x is k-sparse (with a similar argument for $\Delta_y = \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},ao,h} - \boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}$).

To simplify the analysis, in this section we define our $\mathbf{C}_{ao}$ matrices to be slightly different from the ones used in the described algorithm. By forcing the diagonal entries to be 0, we avoid using the ith feature for the ith regression. This removes any dependence between the projection and the target weights and simplifies the discussion. Since we are working with random compressed features as input, all of the features have similar correlation with the output, and thus removing one of them changes the error of the regression by a factor of $O(1/d)$. We can nevertheless change the algorithm to use this modified version of the regression so that the analysis stays sound.

The following theorem bounds the error of a one step update using the compressed operators. We use i.i.d. normal random projection for simplicity. The error bounds for other types of random projections should be similar.[11] Let $[\mathbf{A}]_{-i,*}$ be matrix $\mathbf{A}$ with the ith row removed. We have the following:

Theorem 1 *Let $\mathcal{H}$ be a large collection of sampled histories according to ρ, and let $\boldsymbol{\Phi}^{d\times|\mathcal{Q}|}$ be an i.i.d. normal random projection: $\boldsymbol{\Phi}_{ij} \sim \mathcal{N}(0, 1/d)$. We observe noisy estimate $\hat{\mathbf{P}}_{\mathcal{Q},h} = \boldsymbol{\mathcal{P}}_{\mathcal{Q},h} + \Delta_x$ of input and $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},ao,h} = \boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h} + \Delta_y$ of the output, where elements of Δ_x and Δ_y are independent zero-mean random variables with maximum variance σ_x^2 and σ_y^2 respectively. Let $\sigma_1^2 \ldots \sigma_{|\mathcal{Q}|}^2$ be the decreasing eigenvalues of $E_{\rho(h)}[\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}^\top]$. Choose $1 \le m \le |\mathcal{Q}|$ such that $\sigma_m^2 \le 1$ and define $\nu = \sum_{i=m+1}^{|\mathcal{Q}|} \sigma_i^2$. For $1 \le i \le d$, define*

$$\mathbf{u}_i = \boldsymbol{\Phi}_i \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},ao,\mathcal{H}} (\boldsymbol{\Phi}_{-i} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},\mathcal{H}})^\dagger.$$

Define $\mathbf{C}_{ao}$ to be a $d \times d$ matrix such that

$$(\mathbf{C}_{ao})_i = [\mathbf{u}_{i,1}, \mathbf{u}_{i,2}, \ldots, \mathbf{u}_{i,i-1}, 0, \mathbf{u}_{i,i}, \mathbf{u}_{i,i+1}, \ldots, \mathbf{u}_{i,d-1}].$$

Then with probability no less than $1 - \delta$ we have

$$\|\mathbf{C}_{ao}(\boldsymbol{\Phi}\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}) - \boldsymbol{\Phi}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}\|_{\rho(h)} \le \sqrt{d}\epsilon(|\mathcal{H}|, |\mathcal{Q}|, d, w^2, x^2, b^2, \sigma_\eta^2, \delta/4d), \tag{30}$$

where

$$w^2 = \|\mathbf{B}_{ao}\|^2 (m + 4\sqrt{m}\ln(4d/\delta)), \tag{31}$$

$$x^2 = \|\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}\|_{\rho(h)}^2, \tag{32}$$

$$b^2 = \nu + 4\sqrt{\nu}\ln(4d/\delta), \tag{33}$$

$$\sigma_\eta^2 = \frac{4k\ln(4|\mathcal{Q}|/\delta)}{d}\sigma_y^2 + w^2\sigma_x^2. \tag{34}$$

11. The core modifications necessary are analogous to those used made in Achlioptas (2001) to adapt the Johnson-Lindenstrauss lemma to more general random matrices.

The proof is included in the appendix. The main idea of the theorem is to use the dependence and sparsity of the features to tighten the bound on the error of compressed regression. When most of the variation in the PSR state can be explained using m linear observations, we can substitute the $\mathbf{\Phi}_i \mathbf{B}_{ao}$ target weights having norm $O(\sqrt{|\mathcal{Q}|})$, with a linear approximation having much smaller norm $O(\sqrt{m})$, at the expense of a small bias b. The theorem also describes the overall noise combining the effects of Δ_x and Δ_y.

Theorem 1 has three main implications. One is that the complexity of the compressed regression depends on how fast the eigenvalues drop for the minimal core set covariance matrix. If the eigenvalues drop exponentially fast, as is observed empirically in our experiments, we can guarantee a smaller regression error. Second, if the projection size is of order $O(k \ln |\mathcal{Q}|)$ we can control the variance of the combined noise term. Third, if we use the sparse COLS bound of Fard et al. (2012, 2013), we can can show that regression of size $O(k \ln |\mathcal{Q}|)$ should be enough to decrease the overall estimation error at the expense of a controlled bias.

The following corollary follows immediately from Theorem 1 by union bounding over all action-observation pairs.

Corollary 2 *Using the assumptions of Theorem 1, with probability no less than $1-\delta$ we have, for all $a \in \mathcal{A}$ and $o \in \mathcal{O}$,*

$$\|\mathbf{C}_{ao}(\mathbf{\Phi}\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}) - \mathbf{\Phi}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}\|_{\rho(h)} \leq \sqrt{d}\epsilon(|\mathcal{H}|, |\mathcal{Q}|, d, w^2, x^2, b^2, \sigma_\eta^2, \delta/(4d|\mathcal{A}||\mathcal{O}|)),$$

where $w^2 = \max_{ao} \|\mathbf{B}_{ao}\|^2(m + 4\sqrt{m}\ln(4d/\delta))$, and other factors are as defined in Theorem 1.

4.2.3 Error of the Compressed Normalizer

The $\mathbf{c}_\infty$ operator is the normalization operator for the compressed space. Therefore, for any history h, $\mathbf{c}_\infty^T \mathbf{\Phi}\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}$ should equal $\boldsymbol{\mathcal{P}}_h$. The following theorem provides a bound over the error of such a prediction:

Theorem 3 *Let $\mathcal{H}$ be a large collection of sampled histories according to ρ. We observe noisy estimate $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},\mathcal{H}} = \boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}} + \Delta_x$ of input and $\boldsymbol{\mathcal{P}}_{\mathcal{H}} = \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{H}} + \Delta_z$ of the output, where elements of Δ_x and Δ_z are independent zero-mean random variables with maximum variance σ_x^2 and σ_z^2 respectively. Define $\mathbf{c}_\infty = (\mathbf{\Phi}_i \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},\mathcal{H}})^\dagger \boldsymbol{\mathcal{P}}_{\mathcal{H}}$. Then with probability no less than $1-\delta$ we have*

$$\left\| \mathbf{c}_\infty^\top (\mathbf{\Phi}\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}) - \boldsymbol{\mathcal{P}}_h \right\|_{\rho(h)} \leq \epsilon(|\mathcal{H}|, |\mathcal{Q}|, d, \|\boldsymbol{\beta}_\infty\|^2, \|\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}\|^2_{\rho(\mathbf{h})}, 0, \sigma_\infty^2, \delta),$$

where we define effective noise variance $\sigma_\infty^2 = \sigma_z^2 + \sigma_x^2 \|\boldsymbol{\beta}_\infty\|^2$.

The proof is included in the appendix.

4.2.4 Error Propagation

Once we have the one step errors of compressed operators, we can analyse the propagation of errors as we concatenate the operators. Define $o_{1:n} = o_1 o_2 \dots o_n$ (and similarly for $a_{1:n}$

and $[ao]_{1:n}$). We would like to bound the error between $\mathbb{P}(o_{1:n}||a_{1:n})$ and our prediction $\mathbf{c}_\infty \mathbf{C}_{a_n o_n} \mathbf{C}_{a_{n-1} o_{n-1}} \dots \mathbf{C}_{a_1 o_1} \mathbf{c}_1$.

Since the theorems in the previous sections were in terms of a fixed measure ρ, we have to make distributional assumptions to simplify the derivations. Assume that we fit our model using samples $h \sim \rho$, imposing a distribution $\boldsymbol{\mathcal{P}}_{\mathcal{Q},h} \sim \mu$. Note that as we increase the size of a history h, the norm of $\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}$ becomes smaller. We make the assumption that for all $1 \leq t \leq n$, for a history $[ao]_{1:t} \sim \rho_t$, the implied $\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:t}}$ is sampled from a scaled version of μ (i.e., $\frac{1}{s_t}\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:t}} \sim \mu$). Therefore $\|f(\boldsymbol{\mathcal{P}}_{\mathcal{Q},h})\|_{\rho_t(h)} = \|f(s_t \boldsymbol{\mathcal{P}}_{\mathcal{Q},h})\|_{\rho(h)}$.

Theorem 4 *Let ϵ and ϵ_∞ be the bounds of Corollary 2 and Theorem 3 respectively, for a sample $\mathcal{H}$ according to ρ and failure probability $\delta/2$. Let ρ_n and its marginals $\rho_{n-1} \dots \rho_1$, be distributions over histories of size $n, n-1, \dots 1$ respectively, such that $\|f(\boldsymbol{\mathcal{P}}_{\mathcal{Q},h})\|_{\rho_t(h)} = \|f(s_t \boldsymbol{\mathcal{P}}_{\mathcal{Q},h})\|_{\rho(h)}$ for all measurable f. With probability $1 - \delta$*

$$\left\|\mathbf{c}_\infty \mathbf{C}_{a_n o_n} \mathbf{C}_{a_{n-1} o_{n-1}} \dots \mathbf{C}_{a_1 o_1} \mathbf{c}_1 - \mathbb{P}(o_{1:n}||a_{1:n})\right\|_{\rho_n} \leq \epsilon_\infty s_n + \|\mathbf{c}_\infty\| \epsilon \sum_{t=1}^{n-1} s_t c^{n-t-1},$$

where $c = \max_{a,o} \|\mathbf{C}_{ao}\|$.

The proof is included in the appendix. Note that s_t is exponentially decreasing in t (because longer tests are less probable). The norm of the update operators are expected to be less than 1 (as they shrink the vector of test probabilities). Combining these two, we expect the summation in the bound of Theorem 4 to be over a small exponential function of n.

5. Planning with CPSRs

The learning algorithm presented in Section 3.2 facilitates the construction of accurate predictive models in large complex partially observable domains. In this section, we outline how to plan (near)-optimal sequences of actions using such a learned model. The planning approach we employ was first proposed by Ong et al. (2012). In essence, the approach substitutes a predictive state in place of an observable state in the standard fitted-Q learning algorithm of Ernst et al. (2005).

Unlike point-based value-iteration PSR (PBVI-PSR) planning algorithms, the theoretical convergence of the fitted-Q algorithm does not require that the PSR correspond to a finite-dimensional POMDP. That is, existing error-bounds for PBVI-PSR require that the PSRs used in planning correspond to some finite-dimensional POMDP (Izadi and Precup, 2008), whereas in general PSRs may have no corresponding finite-dimensional POMDP (Denis and Esposito, 2008).[12] In contrast, the fitted-Q approach only requires that the input state-space be sufficient to describe the system, and PSRs satisfy this requirement, meaning that the convergence results for fitted-Q carry over to the PSR setting (when an

12. It is worth noting, however, that the PSR-PBVI error bounds could possibly be modified to alleviate this issue and that PBVI-PSR algorithms have been employed with considerable empirical success (Boots et al., 2010; Izadi and Precup, 2008).

exact PSR model is used) (Ernst et al., 2005).[13] Moreover, the fitted-Q approach does not explicitly require learning a model of rewards prior to the application of the planning algorithm (i.e., the reward model is captured only through the Q-function). We found this to be preferable to explicitly modelling the immediate rewards as a function of the CPSR states prior to planning, as such an explicit model introduces an extra (and unnecessary) level of approximation. In what follows, we briefly review the fitted-Q approach and provide a high-level description of our planning algorithm.

5.1 Fitted-Q with CPSRs

Algorithm 1: Fitted-Q with CPSR

Inputs: A set $\mathcal{D}$ of tuples of the form $(\mathbf{c}_t, a_t, r_t, \mathbf{c}_{t+1})$ constructed using a CPSR model, where r_t is a numerical reward; $\mathcal{R}$, a regression algorithm; γ, a discount factor; and T, a stopping condition

Outputs: A policy π

1: $k \leftarrow 0$
2: Set $\hat{Q}_k(\mathbf{c}_t, a) = 0 \ \forall a \in \mathcal{A}$ and all possible $\mathbf{c}_t$
3: **repeat**
4: $\quad k \leftarrow k + 1$
5: $\quad$ Build training set, $\mathbb{T} = \{(y^l, i^l), l = 1, ..., |\mathcal{D}|\}$ where: $i^l = (\mathbf{c}_t^l, a_t^l)$ and $y^l = r_t^l + \gamma \max_a \hat{Q}_{k-1}(\mathbf{c}_{t+1}^l, a)$
6: $\quad$ Apply $\mathcal{R}$ to approximate $\hat{Q}_k$ from $\mathbb{T}$
7: **until** T is met

output π, where $\pi(\mathbf{c}_t) = \text{argmax}_a\{\hat{Q}_k(\mathbf{c}_t, a)\}$

As stated above, fitted-Q with PSRs is analogous to the MDP case, with the predictive state taking the place of the MDP state in Algorithm 1. The algorithm iteratively builds more and more accurate approximations of the Q-function, which in our case maps predictive states and actions to expected returns. In this work, the *Extra-Trees* algorithm is used as the base regression algorithm (Geurts et al., 2006), as it is a non-linear function approximator for which the fitted-Q convergence results hold (Ernst et al., 2005). For T, the termination condition, we use an iteration limit (instead of an ϵ convergence condition), as this allows for more accurate predictions of runtimes.

Letting $\Psi(T)$ be the expected number of iterations under stopping condition T and assuming that the splitting procedure for nodes in the Extra-Trees algorithm takes constant time, the computational complexity of this fitted-Q approach is (recalling the definitions of Section 3.2)

$$O\left(\Psi(T) \times L|Z|log\left(L|Z|\right)\right), \tag{35}$$

13. The error bounds for PSR-PBVI also require that an exact model is known. In general, current theoretical results on PSR planning ignore the impact of estimation and/or approximation errors incurred during model-learning, though empirical analyses (e.g., the work of Boots et al. (2010) and Section 6 of this paper) suggest that the impact of such errors is small.

which is a factor $\Psi(T) \times log(L|Z|)$ greater than the complexity for the model-learning algorithm of Section 3.2. In practice, we found Algorithm 1 to be several orders of magnitude slower than the CPSR learning algorithm.

5.2 Combined Learning and Planning

Algorithm 2 specifies how CPSR model-learning and the fitted-Q planning algorithm are combined at a high level. This general specification permits a variety of sampling and Q-function approximation strategies. Specifically, it permits pure unbiased random sampling, interleaving exploration and exploitation phases, or even the drawing of samples from some arbitrary (e.g., expert) policy. Of course, if non-blind policies are used then the sample distribution becomes biased (i.e., the samples are no longer i.i.d.), and the analysis of Section 4 no longer holds.

Also note that the number of iterations used by the learner and planner need not be identical. More specifically, more samples may be used to learn the CPSR model than are used in planning. This is a pragmatic specification, as the CPSR learning algorithm can efficiently accommodate orders of magnitude larger sample sets than the fitted-Q planner (by Equations 19 and 35).

Algorithm 2: Combined learning and planning

Inputs: π_s, a sampling policy; N, the number of sampling iterations; I_m, the number of trajectories to use in learning; and I_p, the number of trajectories to use in planning ($I_m \geq I_p$)

Outputs: A CPSR model, $\mathbf{C}$ and policy π

$\mathcal{D}_0 \leftarrow \emptyset$
Initialize the CPSR model, $\mathbf{C}$
for i=1 to N **do**
  Sample I_m trajectories, Z_i, using π_s
  Update $\mathbf{C}$ using Z_i
  Sub-sample I_p trajectories from Z_i and use $\mathbf{C}$ to construct a tuple-set $\mathcal{D}_i$
  $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \mathcal{D}_{i-1}$
  Apply Algorithm 1 with $\mathcal{D}_i$ to learn a policy, π_i
  [Optional] Update π_s (e.g., using π_i)
end for
output $\mathbf{C}$ and π_N

---

6. Empirical Results

We examine empirical results pertaining to both the model quality of compressed models and the proficiency of model-based planning. The goal of this analysis in the model-quality setting is to elucidate (1) the empirical cost (in terms of prediction accuracy) of performing compression (if any), (2) the compute-time reduction engendered by the use of compression, and (3) the impact of the implicit regularization induced by performing compression. We also provide model-quality results explicitly comparing prediction performance when histo-

ries are compressed versus uncompressed, showing that history compression has a negligible effect empirically (and justifying the simplifying assumption that $\mathbf{\Phi}_{\mathcal{H}}^{\top}\mathbf{\Phi}_{\mathcal{H}} = \mathbf{I}$ in Section 4).

In the planning setting, we again seek to elucidate the empirical impact of performing compression, and we do so using three different partially observable domains. First, we use a simple synthetic robot navigation domain (identical to that used in the model-quality experiments) to compare the planning performance of agents trained with CPSR models, agents trained with uncompressed TPSR models, and memoryless (model-free) agents, which serve as a baseline. Next, we examine a massive partially observable domain that is intractable for classic POMDP-based approaches, demonstrating how the use of compression facilitates learning and planning in settings where it would be otherwise intractable. We also provide a qualitative comparison to the Monte-Carlo AIXI algorithm (Veness et al., 2011), a related model-based reinforcement learning approach, using this domain. Lastly, we apply CPSR based learning and planning to the difficult real-world task of adaptive migratory management (Nicol et al., 2013). In this adaptive migratory management problem, a sequential decision-making agent must learn a model of how a certain bird species migrates and how their migration patterns are adversely affected by rising sea-levels (and must do so without prior domain-specific knowledge). Using this learned model the agent must determine an optimal policy for protecting different locations along the birds' migratory route so as to minimize population decline (Martin et al., 2007; Nicol et al., 2013). This difficult real-world domain, which builds upon hand-crafted simulators and ecological data sets (Iwamura, 2011; Nicol et al., 2013), demonstrates both the benefits of compression (in that it is computationally intractable for uncompressed TPSR) and the stark benefits of model-based planning over memoryless (model-free) planning.

6.1 Projection Matrices

In this analysis, we examine three different classes of random projection matrices: spherical, Rademacher, and hashed. The spherical projection matrices contain random Gaussian distributed entries and are identical to those used in Hamilton et al. (2013). The Rademacher are a related class of random matrices where each entry is an independent Rademacher variable; these matrices also satisfy the JL lemma (Baraniuk and Wakin, 2009) and can afford additional efficiencies with low level implementations that exploit the fact that only additions and subtractions are used in the matrix multiplications (this optimization is not used here) (Achlioptas, 2001). The hashed random projection matrices induce a feature-mapping analogous to random hashing; each column of the random projection matrix has a 1 in a random position and the other entries are zero. These random hashing matrices do not directly satisfy the JL lemma, but they have been shown to preserve certain kernel-functions and perform extremely well in practice (Weinberger et al., 2009; Shi et al., 2009).

6.2 Domains

The domains used are based upon previous work on planning with PSRs and on model-based reinforcement learning in large, complex partially observable domains.

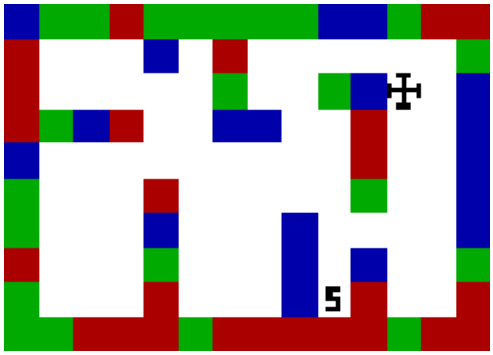


Figure 1: Graphical depiction of *ColoredGridWorld*. The **S** denotes the start position and the target denotes the goal.

6.2.1 ColoredGridWorld

The first domain, *ColoredGridWorld*, is conceptually similar to the simulated robot navigation domains commonly used in the PSR literature and is a direct extension of the *GridWorld* domain used in Hamilton et al. (2013) and Ong et al. (2012). The environment is a 47-state maze with coloured walls. The agent must navigate from a fixed start state to a fixed goal state using only aliased local observations. The action space consists of moves anywhere in the four cardinal directions (moving into walls produces no effect). To simulate noise in the agent's actuators, actions fail with probability 0.2, and if this occurs, the agent moves randomly in a direction orthogonal to that which was specified. The observation space consists of whether or not the agent can see coloured walls in any of the 4 cardinal directions (one observation per wall). There are three possible colors, so there are 3 possible observations per wall and thus 81 possible observations in total. A reward of 1 is returned at the goal state (resetting the environment), and no other states emit rewards.

Though simple, this domain is quintessentially partially observable in that it is impossible to learn how to reach the goal without incorporating memory. Moreover, the added complication of coloured walls exponentially increases the cardinality of the observation space, leading to many possible tests and histories. In essence, the agent cannot know a priori whether the colouring is pertinent to the problem, so it vastly complicates the learning problem.

6.2.2 Partially Observable PacMan

The second domain used is based upon the partially observable PacMan domain, denoted *PocMan*, first proposed by Silver and Veness (2010). It is an extremely large partially observable domain with on the order of 10^{56} states (Veness et al., 2011). The basic dynamics follow that of the video-game PacMan: an agent must navigate a maze-like environment starting from a fixed start-point, collecting food and avoiding coming in contact with any of four ghosts.

In this work, we examine two versions of the domain. The first version is a replica of the *PocMan* domain used by Veness et al. (2011) in their work on a Monte Carlo AIXI

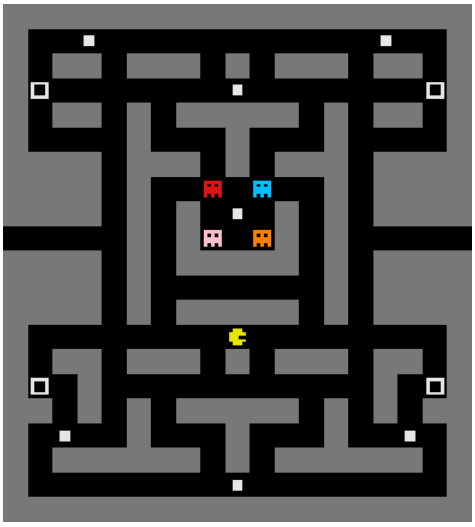


Figure 2: Graphical depiction of *S-PocMan*. The white dots denote food and the white annuli denote power-pills. The yellow PacMan figure denotes the fixed starting position

approximation. In the second version, which we call *S-PocMan*, we further complicate the environment by dropping the parts of the observation vector that allow the agent to sense in what direction food lies, and we sparsify the amount of food in the environment. In the original domain food was placed in each position with probability $\frac{1}{2}$; in *S-PocMan* there are only 7 pieces of food in total, each in a fixed position. The reason for examining this more difficult version of the domain is that, as summarized in Section 6.4, we found that a memoryless controller was able to perform extremely well on the original *PocMan*, achieving results approaching that of the AIXI algorithm. In other words, simply treating the original *PocMan* domain as if it were fully observable led to very good results. This seems to be due to the fact that the food rewards were plentiful and fully observable. In *S-PocMan* we make the problem more partially observable in order to demonstrate the usefulness of a model-based approach.

6.2.3 Adaptive Migratory Management

The last domain we examine is based upon the ecological task of *adaptive migratory management* (*AMM*). The specific goal of *AMM* is to use intervention to protect certain regions in a bird-species' migratory route. In this work, we focus on the Lesser Sand Plover, which is one of many species that uses the East-Asian-Australasian (EAA) migratory route. While migrating, the Lesser Sand Plover stop at staging sites where they feed on invertebrates and gather energy (Martin et al., 2007). These staging sites are located at intertidal mudflats that are especially susceptible to rising sea levels (Iwamura, 2011). The sites can be protected via intervention, but limited resources within the conservation community means that protection can only be implemented at a limited number of sites within a particular year.

By phrasing the task of protecting these intertidal areas as a sequential decision-making problem, the hope is to learn an optimal strategy for intervention.

In Nicol et al. (2013) the *AMM* problem is formalized, and a simulator based dataset is provided (for a number of species including the Lesser Sand Plover). At its core the simulator uses a network-flow model for the migratory routes augmented with handcrafted models for sea-level rises, population declines, and other relevant elements. See Nicol et al. (2013) for a complete description. In this work, we use data generated from the simulator, and we attempt to both learn a succinct model of the domain and optimize decisions using this learned model (i.e., we do not assume access to information contained within the internal simulator state).[14]

Formally, at each time point (which roughly corresponds to a year) the decision-making agent receives a vector of observations, where the first entry corresponds to the population level at the breeding site/node and the next three entries correspond to the protection levels at the three intertidal sites/nodes on the Lesser Sand Plover's migratory route. There are four discretized population levels and three protection levels, corresponding to protection against three increasing states of sea-level rise. There are thus 108 unique possible observations. At each time-step the decision-making agent must increase the protection level at one of the non-breeding nodes, and thus there are three possible actions at each time-step. (If the agent opts to protect a node which is already maximally protected then the action has no effect). Internal dynamics of the underlying system-model determine how protection levels decline over time, but none of this information is available to the agent. At the beginning of a simulation (i.e., in the fixed start-state) the protection and population levels are set to their minimal discretized values.

6.3 Model Quality Results

We examined the model quality of different CPSRs and an uncompressed TPSR on the *ColoredGridWorld* domain. Sample trajectories were generated using a simple ϵ-greedy exploration policy, where the non-random actions were determined by a policy learned via a memoryless controller. All models were set with $d' = 5$, where d' is final model dimension (from Section 3.2) set after performing SVD; however, singular values below a tolerance of 10^{-6} were also discarded. All tests, τ_i, with $|\tau_i| \leq 7$ were included in the estimation process (including longer length tests did not improve performance).[15] For the CPSR models, we set $d_{\mathcal{T}} = d_{\mathcal{H}}$, as preliminary experiments did not reveal any significant benefits to using $d_{\mathcal{T}} \neq d_{\mathcal{H}}$ and examined projection dimensions in the range $[25, 75]$. Only the best performing size (determined through cross-validation) is reported. All models used 10000 train trajectories (of max length 13) and were evaluated with 10000 trajectories. The PacMan-style domains and the *AMM* domain were not examined in this model-quality context as naive TPSRs

14. Note that for the benchmark results presented in Nicol et al. (2013), they use knowledge of the underlying simulator state and cast the planning problem in the POMDP framework, while in this work we solve both the learning and planning problems (rather than just the planning problem).
15. If a particular test was never encountered in the training data, however, it was discarded, as such tests lead to singularities in the observable matrices.

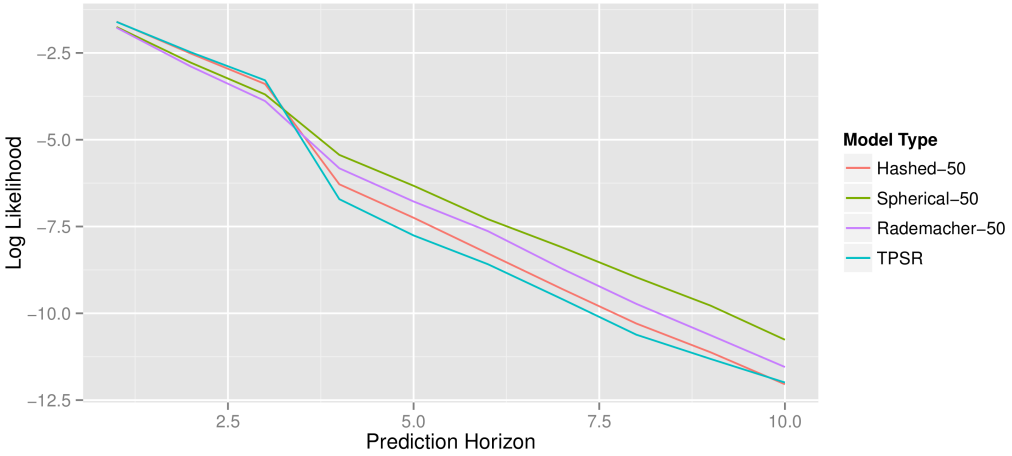


Figure 3: Model-quality results on the *ColoredGridWorld* domain. Plot shows the log-likelihood of the test data given the different models as the prediction horizon is increased. The numbers adjacent to the CPSR projection types correspond to the compressed dimension used. 95% confidence interval error bars are too small to be visible.

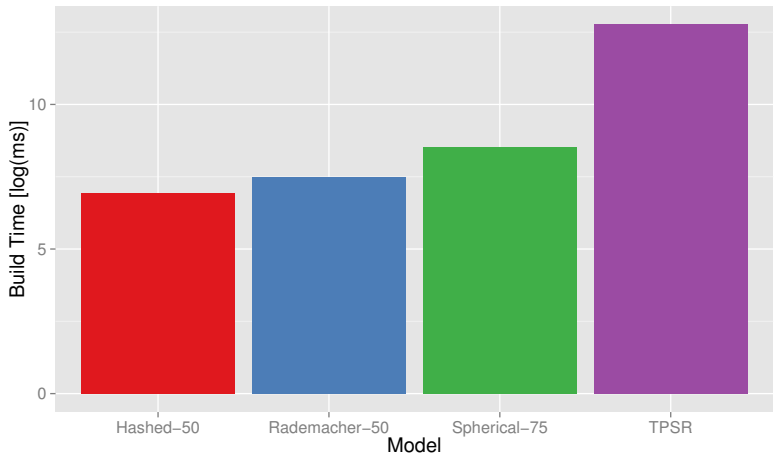


Figure 4: Model build times (on a log-scale) for the different model types on the *Colored-GridWorld* domain. Compressed dimension sizes are listed next to the model names. Times do not include time taken to build the training set. 95% confidence interval error bars are too small to be visible.

exhausted memory limits when tests of length longer than 1 were used, making a rigorous comparison is infeasible.[16]

16. Experiments were run on a machine with a 8-core 3.2 GHz Intel Xeon processor (x64 architecture) and 8GB of RAM.

Figure 3 plots the average log-likelihood of the models as the prediction horizon (i.e., length of the sequences to predict) is increased. The log-likelihood for a single sequence is computed by taking the logarithm of the probability obtained via (1), and this likelihood is averaged over all sequences in the test set. From this figure, we see that the compressed models are not only competitive with the uncompressed TPSR, they actually outperform TPSR at longer prediction horizons. We conjecture that this is due to the regularization induced by the use of random projections. Figure 4 plots the build times for the different models, showing that the compressed models can be built in a fraction of the time required to build the uncompressed TPSR.

Figure 5 shows a focused experiment examining the impact of compressing histories, compared to only compressing tests as was done in Hamilton et al. (2013). These results show $\log(\mathcal{L}(\theta)) - \log(\mathcal{L}(\theta_{HC}))$, the difference between the model-likelihood for a model where histories are not compressed (θ) and where histories are compressed (θ_{HC}). Both the predictive models are constructed using spherical projection matrices and using (identical) samples generated from the *ColoredGridWorld* domain (with the experimental set-up described above). As is evidenced in the plot, there is only a small difference in likelihood between the two models (cf. the likelihood difference seen in Figure 3), and in fact, the model with compressed histories does slightly better for the first few time steps.

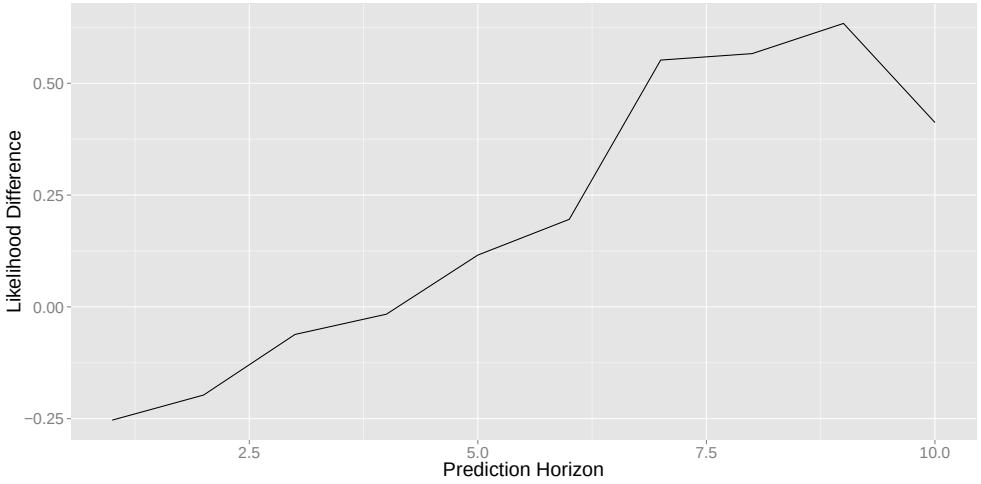


Figure 5: Difference in log-likelihood between a model where histories are not compressed and a model where histories are compressed.

6.4 Planning Results

Next, we apply the full learning and planning approach (Algorithm 2) to the domains *ColoredGridWorld, PocMan*, *S-PocMan*, and *AMM*.

In all experiments, we used 10000 random sampled trajectories to build the models and again used $d_{\mathcal{T}} = d_{\mathcal{H}}$. For planning, we used $I_p = 1000$ with $N = 1$ and a random sampling strategy; this represents the standard unbiased batch-learning setting (Section 7 discusses

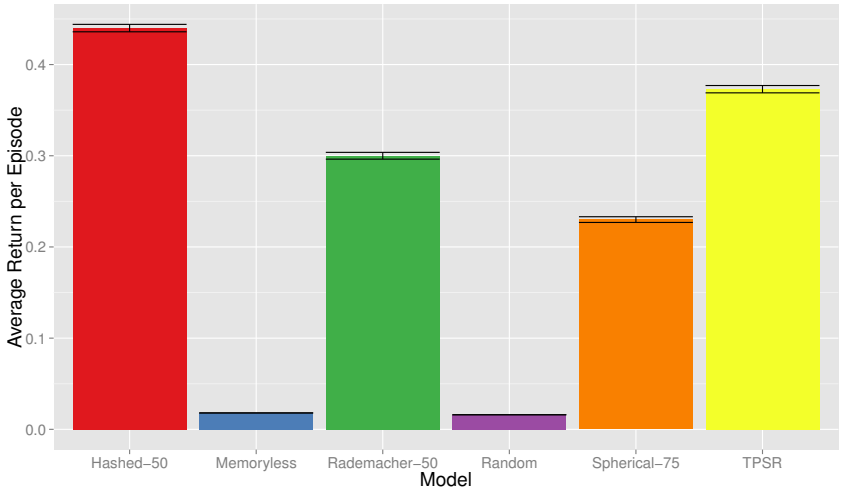


Figure 6: Average return per episode achieved in the *ColoredGridWorld* domain using different models and the baselines. Compressed dimension sizes are listed next to the model names. 95% confidence interval error bars are shown.

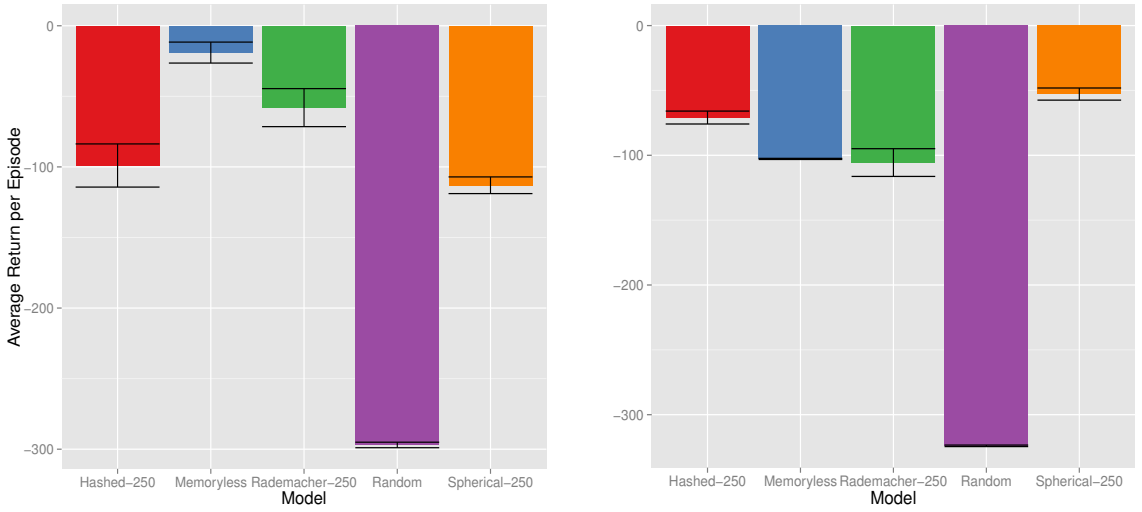
the possibility of using more complex sampling strategies). And for the fitted-Q algorithm, we used 100 fitted-Q iterations, one *Extra-Tree* ensemble of 25 trees per action, and the default settings for the *Extra-Trees* (Geurts et al., 2006). As a baseline, we examined the performance of a memoryless controller on the domains. This controller is analogous to treating the domains as fully observable and running the standard fitted-Q algorithm of Ernst et al. (2005). In order to achieve a fair comparison, the memoryless controller is permitted to use samples that would otherwise be used for model-learning in order to refine its policy (i.e., the memoryless baseline uses the same total number of samples in the experiments as the model-based methods). The use of this baseline is not arbitrary, as its success provides an empirical measure of how partially observable a domain is with respect to planning; if a domain is easily solved by the memoryless controller then it is nearly fully observable in that immediate observations are sufficient for determining near-optimal plans. We also used a simple random planner which selects actions uniformly randomly as a second baseline.

6.4.1 ColoredGridWorld

For *ColoredGridWorld*, the models examined were identical to those described in the model quality experiments above. A discount factor of $\gamma = 0.99$ was used for this domain.

Figure 6 details the performance of the different algorithms on the *ColoredGridWorld* domain. For this domain, the hashed CPSR algorithm achieved the best performance while the TPSR algorithm performed second-best. All the PSR-based approaches vastly outperformed the memoryless-controller baseline. This is expected, as the *ColoredGridWorld* problem is strongly partially observable.

6.4.2 Partially Observable PacMan



(a) *PocMan* (b) *S-PocMan*

Figure 7: Average return per episode achieved in the *PocMan* (a) and *S-PocMan* (b) domains using different models and the baselines. Compressed dimension sizes are listed next to the model names. 95% confidence interval error bars are shown.

For both *PocMan* and *S-PocMan*, we set $d' = 25$ and examined compressed dimensions in the range $[250, 500]$ (selecting only the top performer via a validation set); no TPSR models were used on these domains, as their construction exhausted the memory capacity of the machine used. Following Veness et al. (2011), for these domains we use $\gamma = 0.99999$ as a discount factor.

Figure 7 details the performance of the CPSR algorithms on the *PocMan* and *S-PocMan* domains. In these domains, we see a much smaller performance gap between the CPSR approaches and the memoryless baseline. In fact, in the *PocMan* domain, the memoryless controller is the top-performer. This demonstrates, first and foremost, that the *PocMan* domain is not strongly partially observable. Though the observations do not fully determine the agent's state, the immediate rewards available to an agent (with the exception of reward for eating the power pill and catching a ghost) are discernible through the observation vector (e.g., the agent can see locally where food is). Thus, the memoryless controller is able to formulate successful plans despite the fact that is treating the domain as if it were fully observable. Moreover, a qualitative comparison with the Monte-Carlo AIXI approximation (Veness et al., 2011) reveals that the quality of the memoryless controller's plans are actually quite good. In that work, they use a slightly different optimization criteria of optimizing for average transition reward, and with on the order of 50000 transitions they achieve an average transition reward in the range $[-1, 1]$ (depending on parameter settings). With on the order of 250000 transitions they achieve an average transition reward in the range $[1, 1.5]$. In this work, the memoryless controller achieves an average transition reward of

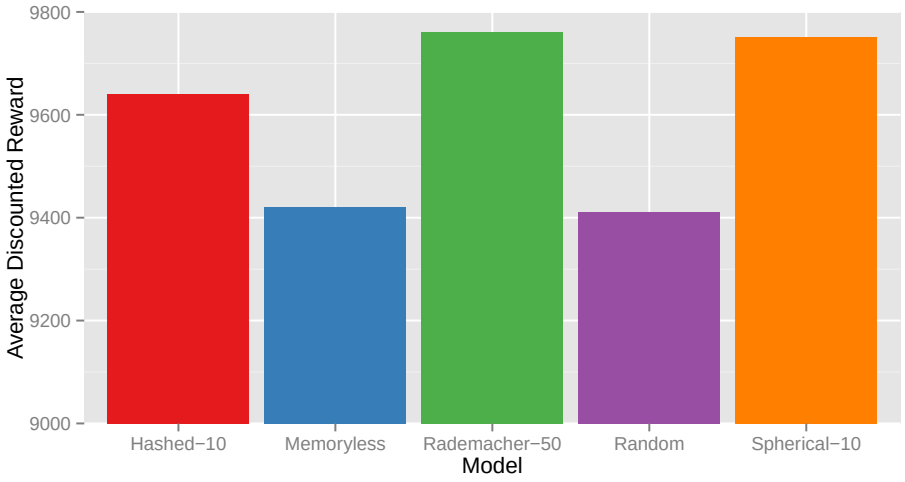


Figure 8: Average discounted reward per episode (i.e., average return per episode) achieved in the *AMM* domain using different methods over 100000 test episodes (each of length 50). The numbers beside the CPSR method names denote the projected dimension size. 95% confidence intervals are too small to be visible.

-0.2 (despite the fact that it is actually optimizing for average return per episode), and it is thus, competitive given the same magnitude of samples, as approximately 50000 transitions were used in this work. It is also important to note that PSR-type models may be combined with memoryless controllers as memory PSRs (described in Section 7.2), and so it should be possible to boost the performance of the CPSR models to match that of the memoryless controller in that way.

Importantly, in *S-Pocman* where part of the observation vector is dropped and the rewards are sparsified, we see that the top-performer is again a CPSR based model (which in this case uses spherical projections). This matches expectations since the food-rewards are no longer fully discernible from the observation vector, and thus the domain is significantly less observable. It is also worth noting that building naive TPSRs (without compression or domain-specific feature selection) is infeasible computationally in these PacMan-inspired domains, and thus the use of a PSR-based reinforcement learning agent (via the compression techniques used) in these domains is a considerable advancement.

A final observation is that the performance is quite sensitive to the choice of projection matrices in these results. For example, in the *S-PocMan* domain, the Rademacher projections perform no better than the memoryless baseline, whereas for *PocMan* the Rademacher outperforms the other projection methods. The exact cause of this performance change is unclear. Nevertheless, this highlights the importance of evaluating different projection techniques when applying this algorithm in practice.

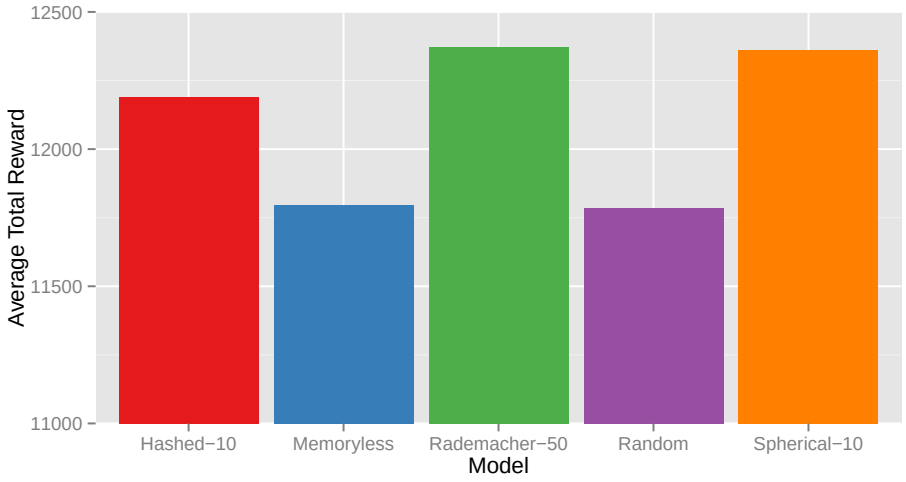


Figure 9: Average total (undiscounted) reward per episode achieved in the *AMM* domain using different methods over 100000 test episodes (each of length 50). The numbers beside the CPSR method names denote the projected dimension size. 95% confidence intervals are too small to be visible.

6.4.3 Adaptive Migratory Management

We used a discount factor of $\gamma = 0.99$ for the *AMM* domain. For model-learning, we set $d' = 10$ and examined compressed dimension in the range $[10, 100]$. The trajectories used during learning are all of maximum length 50 (the simulation may terminate earlier if all the birds perish). Note that since the *AMM* domain is non-stationary (Nicol et al., 2013), the model-learning algorithms must incorporate histories of length up to 50 (i.e., the entire trajectory) (Boots et al., 2010), making the history dimension extremely large (i.e., ≈ 100000) and making uncompressed PSR learning infeasible. Tests up to length 4 were used for this task.

The results obtained are summarized in Figures 8 and 9. Figure 8 shows the average sum of discounted rewards obtained using each method while Figure 9 shows the average total (i.e., undiscounted) sum of rewards obtained by each method. Both these test metrics are included as the discounted return is what the fitted-Q algorithm optimizes for while the average total return is important from an intuitive perspective in that ecological conservationists do not necessarily discount the future. (Note, however, that the discount factor is necessary algorithmically for convergence since this domain technically has an infinite horizon).

Clearly, the CPSR methods are the top-performers with respect to both metrics. In fact, the memoryless baseline does no better than random. We also note that returns achieved by all methods are quite high. The cause of the high return and the fact that the memoryless does no better than random are closely related. Specifically, in the domain all actions are positive in that the agent must increase protection somewhere at each time-step. (The simulator does not allow for no action to be taken). Thus, the random policy still leads to

reasonable results since it will tend to spread its protection actions out uniformly randomly among the candidate nodes. Moreover, without building a model and with access only to the observation vector at each time step, a reasonable strategy is to allocate protection to areas that have relatively low protection levels, compared to the other nodes. That is, a reasonable memoryless strategy is also to simply spread out the protection among the candidate nodes, since without knowledge of the underlying dynamics one must assume that all nodes are equal. Thus, intuitively the optimal memoryless strategy should be close to uniformly random, and this explains the similarity in scores between these two baselines.

Between the different CPSR methods, the Rademacher-projection based method performed the best with the spherical-projection method only performing slightly worse. This result is expected in that there are stronger theoretical guarantees for these methods compared to the hashed projection method.

Lastly, we see that the results are consistent across the two metrics. Interestingly, however, the performance increase between the top CPSR method and the random baseline is greater for the total (undiscounted) reward metric. For that metric, the total reward obtained via the top-performing CPSR method is 4.6% greater than the baseline, whereas for the discounted metric the top-performing method scores 3.7% greater than the random baseline. This makes sense in that the CPSR models should benefit more at longer horizons, since (1) it takes time for the CPSR model to incorporate observed information into its predictions and (2) the non-stationary in the domain, which is captured via the CPSR model, is only a factor at longer time-scales (Nicol et al., 2013).

7. Discussion

The CPSR approach provides a new avenue for model-based reinforcement learning where agents must formulate policies in large, complex partially observable domains without access to a fully-specified prior system model (i.e., where the system model must be learned prior to planning). The compressed learning algorithm allows accurate approximations of PSR models to be constructed in a memory and time efficient manner, and the use of random projections regularizes the learned solutions, preventing high variance models (over-fitting) and potentially leading to more accurate results. We elucidated theoretical guarantees bounding the induced approximation error of this model-learning approach, showing that the low-dimensional embeddings of the models retain predictive accuracy. In addition, we proposed a planning approach which exploits these compressed models in a principled manner, allowing for high-quality plans to be constructed without prior domain knowledge. Finally, we outlined how model-learning and planning can be combined at a high-level.

The empirical results we obtained demonstrate the efficacy of this approach and delineate domains in which its use is beneficial. The model quality experiments demonstrate that CPSR models achieve predictive accuracy competitive to that of uncompressed models, while taking a fraction of the runtime, and the planning results demonstrate that these models can be exploited by efficient planners, providing a novel and powerful framework for model-based reinforcement learning. Moreover, the results highlight the fact that the benefits of such a model-based approach are most stark in domains that are not only partially observable in the traditional sense but that are also strongly partially observable in that the Q-function (or a good approximation of it) is not discernible from the observation vec-

tors. In other words, the results demonstrate that aliased observations (and an unobserved hidden state) alone do not necessitate the need for a model-based learning algorithm. A model-based approach only becomes necessary when the observations are not sufficient for learning a reasonable approximation of the Q-function.

7.1 Practical Concerns

The implementation of complex RL frameworks often reveals practical issues that are not immediately apparent given formal descriptions. In order to facilitate the use of the CPSR algorithm in applications, we outline some pertinent practical issues that arise while implementing the CPSR algorithm and describe our solutions.

7.1.1 Selecting the Projection Matrices

First, it is necessary to reiterate the sensitivity of the approach with respect to both the projection dimension and type of projection used. Empirically, we found that the results could be quite sensitive to these parameters, though this was only the case for some domains. For example, selecting a projection dimension that is too small may lead to suboptimal (near-random) performance. This issue is further exacerbated by the fact that the true dimension of the underlying system is unknown.

The cause of the sensitivity with respect to the projection size is quite evident (smaller dimensions lose information but provide more regularization). However, the underlying cause of the differing performance between the different projection types is not as clear. One would expect the hash-type projection matrices to perform differently than the Rademacher and spherical projections, since the hash-type matrices do not satisfy the JL lemma, but we witnessed substantial variation between all three projection types, especially on the PacMan-type planning domains. Moreover, for the *ColoredGridWorld* domain, the difference between the projection types was more stark for planning performance compared to prediction performance.

The results thus indicate that planning performance is more sensitive to the choice of the projection matrix (compared to prediction performance). One explanation for this is simply that small discrepancies in the prediction performance of the models are amplified when agents must plan using the predictive models. The differing results obtained using the different projection matrices may then be due to the fact that a coarse-grained search (necessitated by computational requirements) for the compressed dimension-size was used and that different random projections may be optimal for slightly different projection sizes (Achlioptas, 2001). For example, a Rademacher projection may be near optimal at one point on the coarse-grained search while a spherical projection may be optimized at a point not included in the coarse-grained search. The slight differences in model-quality induced by the coarse-grained search would then propagate and lead to large variations in planning performance.

In order to cope with the sensitivity of the CPSR approach with respect to the projection sizes and dimension, we recommend using multiple phases of grid search (starting with exponentially separated values). Moreover, it is useful to narrow down the size-range for the projections using model-quality experiments (before performing hyperparameter opti-

mization for planning), since model-quality experiments are not as computational expensive (compared to planning experiments).

7.1.2 Improving Efficiency by Caching

In Section 3.2 we defined the projection operators via the functions $\phi_{\mathcal{T}} : \mathcal{T} \to \mathbb{R}^{d_{\mathcal{T}}}$ and $\phi_{\mathcal{H}} : \mathcal{H} \to \mathbb{R}^{d_{\mathcal{H}}}$. This specification engenders a number of benefits. Specifically, the full projection matrices do not need to be held in memory and the number of tests and histories do not need to be specified in advance. There is a runtime penalty associated with the technique, however, as the mappings must be recomputed each time a particular test or history is encountered while iterating over the sample trajectories. In order to ameliorate this issue, while retaining the benefits of specifying the projections as functions, we implemented a least-recently-used (LRU) cache. By caching the mappings for frequently encountered tests and histories, we improved the empirical runtime of the algorithm considerably.

7.1.3 Numerical Stability Issues

At its core, the CPSR algorithm relies on standard linear algebra techniques, namely SVD and matrix inversions, which are prone to numerical stability issues. If the matrices upon which these operations are performed are ill-formed, suboptimal results will be obtained (or the algorithm will simply fail). In this work, we found one common situation where such stability issues arise.

Since we do not normalize the probability estimates in Section 3.2, the singular values of $\hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}}$ in (15) grow with the size of the training set. This leads to stability issues when inverting the matrix of singular values in order to compute the implicit pseudoinverse in (17) and (18). This stability issue can be alleviated by normalizing the probability estimates, or more generally, by scaling $\hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}}$ by a small constant. Since this constant cancels out during learning, it can be picked arbitrarily, but it should be chosen such that the magnitude of the values in $\hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}}$ are near unity. The most straightforward approach is to simply normalize the probability estimates, though this may not always suffice (e.g., if there are extremely unlikely events, the normalizer may make certain entries too small leading to further stability issues). We also empirically observed that setting $d' < d_{\mathcal{T}}$ and/or removing singular values below a certain threshold (a standard technique) helped with numerical stability.

7.1.4 Q-function Approximation and Sampling Strategies

Algorithm 2 in Section 5 permits a wide-variety of sampling strategies, and the sampling strategy used implicitly constrains the Q-function approximation obtained. In this work, we used an unbiased random sampling strategy in the batch setting. That is, we collected a large batch of random samples, which we used to both learn a model and construct plans. We opted for this framework as (1) our simulators were designed for the batch setting and (2) the theoretical results of Section 4 assume a blind (random) sampling strategy is used.

We did, however, experiment with a goal-directed sampling approach (Ong et al., 2012), where phases of exploration and exploitation are interleaved. In the goal-directed paradigm, a number of mini-batch sampling iterations are used, and the sampling policy (π_s) is updated at each iteration to be ϵ-greedy over the agent's current policy (π_i). Ong et al. (2012) found that this approach led to better performance in the small-sample setting. In our

experiments, where we used larger numbers of samples (on the order of 10000), we found that the goal-directed approach did not improve over random sampling and, in fact, often led to worse results and numerical instabilities. In particular, the bias in the sampling strategy led to an imbalance in the $\hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}}$ matrix in that certain entries dominated in terms of magnitude. As a result of this imbalance, the SVD in (15) became unstable, and poor results were obtained. Such stability problems are likely to be an issue whenever biased sampling strategies are used in the large-sample batch setting. However, in online or small sample settings, such strategies will likely lead to performance increases due to the fact that their exploration is myopic and focuses on areas of the state-space relevant to planning (as shown by Ong et al., 2012).

7.1.5 Compressing Histories

The theoretical analysis of Section 4 assumes that $\mathbf{\Phi}_{\mathcal{H}}$ has orthonormal columns. However, in order to obtain maximal computational benefits, it is necessary to use a compressive $\mathbf{\Phi}_{\mathcal{H}}$, i.e. a $\mathbf{\Phi}_{\mathcal{H}}$ that acts as a feature selector on histories. In fact, for massive domains such as the PacMan-style domains, compressing histories is necessary for tractable learning and planning.

Viewing CPSR learning from the perspective of regression (as was done throughout this paper), the compression of histories is equivalent to compressing the samples used for regression; that is, it is equivalent to linearly mixing the samples. More formally, we use the transformation

$$\mathbf{y} = \mathbf{X}^\top \mathbf{w} + \boldsymbol{\eta} \rightarrow \mathbf{\Phi}_{\mathcal{H}}\mathbf{y} = \mathbf{\Phi}_{\mathcal{H}}\mathbf{X}^\top \mathbf{w} + \mathbf{\Phi}_{\mathcal{H}}\boldsymbol{\eta},$$

where as usual $\mathbf{X}$ is a design matrix, $\mathbf{w}$ a vector of regression weights, $\mathbf{y}$ a vector of targets, and $\boldsymbol{\eta}$ a vector of noise terms. Intuitively, we can view this projection by $\mathbf{\Phi}_{\mathcal{H}}$ as roughly averaging over training samples. The number of samples for the regression will then be reduced, but the averaged samples will have reduced (maximum) variance in their noise terms.

Of course, in this work, we use random $\mathbf{\Phi}_{\mathcal{H}}$ matrices, which do not correspond directly to taking averages over samples. The most important implication of this is that the noise terms of the new combined samples are not independent. This more complex setting has been analyzed in detail by Zhou et al. (2007) (for random Gaussian matrices). In that work, they focus on the more specific setting of l_1 regularized regression, and they prove a number of important results. Of particular relevance is Claim 4.3, which shows (under certain conditions) that the entry-wise discrepancy between $\mathbf{Q}^\top\mathbf{Q}$ and $\mathbf{Q}^\top\mathbf{\Phi}^\top\mathbf{\Phi}\mathbf{Q}$ decreases asymptotically to zero almost surely, where $\mathbf{Q} \in \mathbb{R}^{n\times m}$ and $\mathbf{\Phi} \in \mathbb{R}^{d\times n}$ is a random Gaussian matrix defined as in Theorem 1. This key result facilitates bounding the discrepancy between the compressed training error and the true error of the regressor and does not rely on l_1 regularization assumptions. We refer the interested reader to that work for detailed proofs.

Finally, we reiterate that in this work the compression of histories is a computational necessity, as it allows us to scale the learning algorithm to domains that would be intractable otherwise. And empirical investigations in Section 6 show that the compression of histories to $d_{\mathcal{H}} = d_{\mathcal{T}}$ introduces only a small amount of error during model-learning.

7.2 Related Work

The CPSR algorithm is closely related to work on using features or kernel embeddings with PSRs (Boots et al., 2010; Boots and Gordon, 2011; Boots et al., 2013), where features of tests, histories, and/or observations are employed. Indeed, one view of the CPSR learning approach is that it is an instantiation of the feature-based learning approach where principled random features are employed. However, this view is limited in the sense that the random features used here facilitate an analysis in terms of compression, whereas with other feature-based PSR methods it is simply assumed that the specified features are sufficient to capture the structure of $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}$; that is, the standard feature-based methods assume features that are not compressive (Boots et al., 2010; Boots and Gordon, 2011; Boots et al., 2013).

This distinction of whether or not features are assumed as compressive also highlights the differing motivations between existing feature-based PSR learning and the CPSR approach: in the CPSR approach, compressive random features are employed to increase the efficiency and scalability of learning, whereas in other works (e.g. Boots et al., 2010; Boots and Gordon, 2011; Boots et al., 2013) the features are used to facilitate learning in domains with continuous or structured observation spaces.

It should be noted, however, that since the general PSR learning framework assumes discrete observations, decomposing a continuous domain via feature extraction is necessary for learning in that setting. Moreover, Boots et al. (2013) shows how the well-known "kernel trick" can be employed to learn in feature-spaces of infinite dimension. The penalty associated with this kernel embedded approach is that learning scales cubically with the number of training examples, leading to high computational overhead (Boots et al., 2013). Boots and Gordon (2011) show how to partially alleviate this cost by using random features to approximate certain kernels, a technique that also relies on random projections (though not in the compressed sensing setting).

In a similar vein, the CPSR-based planner is closely related to the goal-directed planning and learning approach of Ong et al. (2012). The primary difference between our work and this goal-directed approach is that we present a more general combined learning and planning framework, which accommodates the use of a wide variety of sampling strategies.

Beyond these works, our approach bears similarities to the memory PSR (mPSR) approach of James et al. (2005), which uses a type of hybrid PSR-MDP model to reduce computational costs and increase predictive accuracy, and the hierarchical PSRs (HPSRs) of Wolfe and Singh (2006), which use the option framework (Sutton et al., 1999) to increase the predictive capacity of PSRs. Importantly, the improvements suggested by both these approaches are not incompatible with our compressed learning algorithm.

Our approach also shares similarities with certain model-based reinforcement learning algorithms, which use adaptive history-based techniques. Examples of these algorithms include *U-Tree* (McCallum, 1996) and the *Monte-Carlo AIXI approximation* (Veness et al., 2011). These approaches share the motivation of developing agents that can learn a model of dynamical system and plan using this model. They differ, however, in the instantiation of their model-based approach, as they use an adaptive history-based approach, which intuitively corresponds to learning mixtures of different k-order MDPs (where k varies adaptively). A key aspect of these approaches is focusing the model-learning on areas of the state-space relevant to achieving goals (similar to the goal-directed sampling routine)

(McCallum, 1996; Veness et al., 2011). Thus, a fundamental difference between Monte-Carlo AIXI-like approaches and the one proposed here is that they efficiently learn myopic models, necessarily constrained by the planning aspect of the problem, whereas in this work we retain the option of learning full-unbiased models of domains (i.e., our model-learning may be decoupled from planning). One implication of this is that the models learned via the CPSR learning approach may be reused in different planning contexts. However, a disadvantage of learning complete (i.e., full and unbiased) models is that it can be impractical in very large and complex domains.

7.3 Future Directions

Given the above discussion, an interesting direction for future work would be an analysis of the inductive bias associated with both the PSR and Monte-Carlo AIXI paradigms. Though these methods bear similarities, their theoretical motivations are quite distinct: PSRs being motivated by the theory of observable operators while certain AIXI-like methods have information-theoretic (and/or Bayesian) motivations (Veness et al., 2011). Recently, there have been a number of theoretical advancements in the understanding of observable operator methods, such as the local loss formulation of Balle et al. (2012) and the method of moments formulation of Anandkumar et al. (2012). These advancements could serve as tools in such an analysis. Perhaps the most interesting question in this area is understanding the regularization induced by these different paradigms (e.g., due to the restriction of the model classes). For example, the Monte-Carlo AIXI method explicitly penalizes model complexity, while this does not explicitly factor into the optimization of PSR-type methods (besides through the hyper-parameter selection of the model-size).

Another interesting avenue for the continuation of this work is exploring the use of different optimization frameworks during learning. In this work, we implicitly use the standard least-squares objective when solving the pseudoinverse in (17) and (18). However, there is no a priori reason to believe that this is the optimal formulation, and in fact, promising results have been obtained by modifying this optimization (e.g., through convex-relaxation) (Balle et al., 2012). Moreover, it is possible that alternative formulations may reveal novel regularization strategies (e.g., regularization on the implicit observable-operator structure) and additional algorithmic efficiencies.

Lastly, the framework presented here provides the necessary ingredients for applying a CPSR-based learning and planning framework to difficult real-world application problems, such as robot navigation problems similar to those solved by U-tree-based approaches (McCallum, 1996). Of course, such applications would introduce certain engineering issues not highlighted here. In particular, the sampling strategy, projection size, and projection type would necessarily be constrained by the problem domain and by hardware limitations; for example, it may be worthwhile to use highly optimized Rademacher projections. Moreover, in domains with extremely large action and observation dimensions, using a distributed implementation (e.g., of Equation 18 in the learning algorithm) would likely engender significant computational benefits. And, in domains with continuous observations, it would be necessary to combine discretization or kernel-based feature extraction with the CPSR compression techniques. These engineering issues, however, should not necessitate altering the core of the CPSR approach.

Acknowledgments

The authors would like to thank Doina Precup, Yuri Grinberg, Sylvie Ong, and Clement Gehring for helpful discussions on this work, and David Silver and Joel Veness for support on the PocMan domain. We are very grateful to our anonymous reviewers for their comments and recommendations. Financial support for this work was provided by NSERC Discovery and CGS-M grants.

Appendix A.

A.1 Proof of Theorem 1

Proof With eigenvalue decomposition we have $E_{\rho(h)}[\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}^\top] = \mathbf{V}\mathbf{D}\mathbf{V}^\top$, where $\mathbf{D}$ is the diagonal matrix containing the eigenvalues and $\mathbf{V}$ is an orthonormal basis. Let $\mathbf{I}_m$ be a $|\mathcal{Q}| \times |\mathcal{Q}|$ matrix with the first m diagonal elements set to 1 and 0 elsewhere. For all $1 \leq i \leq d$, define $[\tilde{\boldsymbol{\Phi}}]_{i,*} = [\boldsymbol{\Phi}]_{i,*}\mathbf{V}\mathbf{I}_m\mathbf{V}^\top$ and $[\boldsymbol{\Phi}']_{i,*} = [\boldsymbol{\Phi}]_{i,*}\mathbf{V}$. Note that since $\mathbf{V}$ is an orthonormal basis and $[\boldsymbol{\Phi}]_{i,*}$ is i.i.d. normal, $[\boldsymbol{\Phi}']_{i,*}$ will also have an i.i.d. normal distribution with the same covariance.

We wish to substitute $[\boldsymbol{\Phi}]_{i,*}$ with $[\tilde{\boldsymbol{\Phi}}]_{i,*}$ which has a small norm and introduces a small bias. We first bound the norm of $[\tilde{\boldsymbol{\Phi}}]_{i,*}$ as follows. With probability no less than $1 - \delta/4$ for all $1 \leq i \leq d$

$$\begin{aligned}
\|[\tilde{\boldsymbol{\Phi}}]_{i,*}\|^2 &= [\boldsymbol{\Phi}]_{i,*}\mathbf{V}\mathbf{I}_m\mathbf{V}^\top\mathbf{V}\mathbf{I}_m\mathbf{V}^\top[\boldsymbol{\Phi}]_{i,*}^\top \\
&= [\boldsymbol{\Phi}']_{i,*}\mathbf{I}_m([\boldsymbol{\Phi}']_{i,*})^\top = \sum_{j=1}^{m}([\boldsymbol{\Phi}']_{ij})^2 \\
&\leq m + 4\sqrt{m}\ln(4d/\delta). \qquad (36)
\end{aligned}$$

The tail bound in last line is union bounding over a corollary of Lemma 1 in Laurent and Massart (2000). The bias induced by using $[\tilde{\boldsymbol{\Phi}}]_{i,*}$ can be bounded as well. Define $b(h) = [\boldsymbol{\Phi}]_{i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h} - [\tilde{\boldsymbol{\Phi}}]_{i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}$. With probability no less than $1 - \delta/4$ for all $1 \leq i \leq d$

$$\begin{aligned}
\|b(h)\|_{\rho(h)}^2 &= E_{\rho(h)}[([\boldsymbol{\Phi}]_{i,*} - [\tilde{\boldsymbol{\Phi}}]_{i,*})\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}^\top([\boldsymbol{\Phi}]_{i,*} - [\tilde{\boldsymbol{\Phi}}]_{i,*})^\top] \\
&= ([\boldsymbol{\Phi}]_{i,*} - [\tilde{\boldsymbol{\Phi}}]_{i,*})\mathbf{V}\mathbf{D}\mathbf{V}^\top([\boldsymbol{\Phi}]_{i,*} - [\tilde{\boldsymbol{\Phi}}]_{i,*})^\top \\
&= ([\boldsymbol{\Phi}]_{i,*} - [\boldsymbol{\Phi}]_{i,*}\mathbf{V}\mathbf{I}_m\mathbf{V}^\top)\mathbf{V}\mathbf{D}\mathbf{V}^\top([\boldsymbol{\Phi}]_{i,*} - [\boldsymbol{\Phi}]_{i,*}\mathbf{V}\mathbf{I}_m\mathbf{V}^\top)^\top \\
&= [\boldsymbol{\Phi}]_{i,*}\mathbf{V}(\mathbf{I} - \mathbf{I}_m)\mathbf{D}(\mathbf{I} - \mathbf{I}_m)\mathbf{V}^\top[\boldsymbol{\Phi}]_{i,*}^\top \\
&= [\boldsymbol{\Phi}']_{i,*}(\mathbf{I} - \mathbf{I}_m)\mathbf{D}(\mathbf{I} - \mathbf{I}_m)([\boldsymbol{\Phi}']_{i,*})^\top \\
&= \sum_{j=m+1}^{|\mathcal{Q}|}([\boldsymbol{\Phi}']_{ij})^2\sigma_j^2 \\
&\leq \nu + 4\sqrt{\nu}\ln(4d/\delta). \qquad (37)
\end{aligned}$$

The tail bound again is due to Lemma 1 in Laurent and Massart (2000) using the assumption $\sigma_m^2 \leq 1$. Using the above bounds, we have for for all $1 \leq i \leq d$

$$\forall h : [\boldsymbol{\Phi}]_{i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h} = [\tilde{\boldsymbol{\Phi}}]_{i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h} + b(h) = ([\tilde{\boldsymbol{\Phi}}]_{i,*}\mathbf{B}_{ao})\boldsymbol{\mathcal{P}}_{\mathcal{Q},h} + b(h). \qquad (38)$$

Therefore, we have a target $[\mathbf{\Phi}]_{i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}$ that is near-linear in the sparse features $\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}$, with expected bias bounded by $b^2 = \nu + 4\sqrt{\nu}\ln(4d/\delta)$, and norm of the weight vector $[\tilde{\mathbf{\Phi}}]_{i,*}\mathbf{B}_{ao}$ bounded by $w^2 = \|\mathbf{B}_{ao}\|^2(m + 4\sqrt{m}\ln(4d/\delta))$.

By definition, $\mathbf{u}_i$ is the COLS estimate with input $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},\mathcal{H}}$, target $[\mathbf{\Phi}]_{i,*}\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},ao,\mathcal{H}}$, and projection $[\mathbf{\Phi}]_{-i,*}$. But in order to use the bound of Equation 29, we need to find the corresponding noise parameters of the COLS algorithm. Since, unlike the assumption of the general COLS bound, both the input and the output of the regression are noisy, we need to derive the effective overall noise variance in the sample output. We have

$$\begin{aligned}
[\mathbf{\Phi}]_{i,*}\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},ao,h} &= [\mathbf{\Phi}]_{i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h} + [\mathbf{\Phi}]_{i,*}\Delta_y \\
&= [\tilde{\mathbf{\Phi}}]_{i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h} + b(h) + [\mathbf{\Phi}]_{i,*}\Delta_y \\
&= [\tilde{\mathbf{\Phi}}]_{i,*}\mathbf{B}_{ao}(\hat{\mathbf{P}}_{\mathcal{Q},h} - \Delta_x) + b(h) + [\mathbf{\Phi}]_{i,*}\Delta_y \\
&= ([\tilde{\mathbf{\Phi}}]_{i,*}\mathbf{B}_{ao})\hat{\mathbf{P}}_{\mathcal{Q},h} + b(h) + ([\mathbf{\Phi}]_{i,*}\Delta_y - [\tilde{\mathbf{\Phi}}]_{i,*}\mathbf{B}_{ao}\Delta_x).
\end{aligned}$$

And thus the sample points are

$$\hat{\mathbf{P}}_{\mathcal{Q},h} \rightarrow ([\tilde{\mathbf{\Phi}}]_{i,*}\mathbf{B}_{ao})\hat{\mathbf{P}}_{\mathcal{Q},h} + b(h) + ([\mathbf{\Phi}]_{i,*}\Delta_y - [\tilde{\mathbf{\Phi}}]_{i,*}\mathbf{B}_{ao}\Delta_x). \tag{39}$$

The effective noise $[\mathbf{\Phi}]_{i,*}\Delta_y - [\tilde{\mathbf{\Phi}}]_{i,*}\mathbf{B}_{ao}\Delta_x$ has mean 0. Since Δ_y is k-sparse and $\|[\tilde{\mathbf{\Phi}}]_{i,*}\mathbf{B}_{ao}\|^2 \leq w^2$, the variance of the effective noise term is bounded by $\max_j([\mathbf{\Phi}]_{ij})^2 k\sigma_y^2 + w^2\sigma_x^2$. Maximization over i and using a tail bound on the maximum of squared normals gives the σ_η^2 defined in the theorem.

We now apply the union bound to Equation 29. With probability no less than $1 - \delta/4$, for all $1 \leq i \leq d$,

$$\|\mathbf{u}_i([\mathbf{\Phi}]_{-i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}) - [\mathbf{\Phi}]_{i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}\|_{\rho(h)} \leq \epsilon(|\mathcal{H}|, |\mathcal{Q}|, d, w^2, x^2, b^2, \sigma_\eta^2, \delta/4d). \tag{40}$$

Note that by our definition of $\mathbf{C}_{ao}$, we have that $\mathbf{u}_i([\mathbf{\Phi}]_{-i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}) = (\mathbf{C}_{ao})_i(\mathbf{\Phi}\boldsymbol{\mathcal{P}}_{\mathcal{Q},h})$, which immediately gives the theorem by combining the error bounds on each row. ■

A.2 Proof of Theorem 3

Proof Similar to Theorem 1, we have $\boldsymbol{\mathcal{P}}_h = \boldsymbol{\beta}_\infty^\top \boldsymbol{\mathcal{P}}_{\mathcal{Q},h}$ for all h. Therefore we have a linear target and by definition $\mathbf{c}_\infty$ is the COLS estimate with projection $\mathbf{\Phi}$. We have

$$\begin{aligned}
\hat{\boldsymbol{\mathcal{P}}}_h &= \boldsymbol{\mathcal{P}}_h + \Delta_z = \boldsymbol{\beta}_\infty^\top \boldsymbol{\mathcal{P}}_{\mathcal{Q},h} + \Delta_z \\
&= \boldsymbol{\beta}_\infty^\top \hat{\mathbf{P}}_{\mathcal{Q},h} - \boldsymbol{\beta}_\infty^\top \Delta_x + \Delta_z.
\end{aligned} \tag{41}$$

Thus the effective variance is bounded by the σ_∞^2 defined in the theorem. We complete the proof by an application of the bound in Equation 29. ■

A.3 Proof of Theorem 4

Proof For all t, define $\mathbf{e}_t = \mathbf{C}_{a_t o_t}\mathbf{C}_{a_{t-1}o_{t-1}}\dots\mathbf{C}_{a_1 o_1}\mathbf{c}_1 - \boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:t}}$. After applying the nth compressed operator we have

$$\begin{aligned}
\|\mathbf{e}_n\|_{\rho_n} &= \|\mathbf{C}_{a_n o_n}\mathbf{C}_{a_{n-1}o_{n-1}}\dots\mathbf{C}_{a_1 o_1}\mathbf{c}_1 - \boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}}\|_{\rho_n} \\
&= \|\mathbf{C}_{a_n o_n}(\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n-1}} + \mathbf{e}_{n-1}) - \boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}}\|_{\rho_n} \\
&\leq \|\mathbf{C}_{a_n o_n}\mathbf{e}_{n-1}\|_{\rho_n} + \|\mathbf{C}_{a_n o_n}\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n-1}} - \boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}}\|_{\rho_n} \\
&\leq \|\mathbf{C}_{a_n o_n}\mathbf{e}_{n-1}\|_{\rho_n} + \max_{o_n,a_n}\|\mathbf{C}_{a_n o_n}\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n-1}} - \boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}}\|_{\rho_{n-1}} \\
&\leq c\|\mathbf{e}_{n-1}\|_{\rho_n} + \max_{o_n,a_n}\|\mathbf{C}_{a_n o_n}s_{n-1}\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n-1}} - s_{n-1}\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}}\|_{\rho} \qquad (42) \\
&\leq c\|\mathbf{e}_{n-1}\|_{\rho_{n-1}} + s_{n-1}\epsilon \\
&\leq \epsilon\sum_{t=1}^{n-1} s_t c^{n-i-1}. \qquad (43)
\end{aligned}$$

Line 42 uses the distribution assumption on ρ_{n-1} and having $\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}}$ linear in $\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n-1}}$. Line 43 follows by induction. We now apply the normalizer operator:

$$\begin{aligned}
&\|\mathbf{c}_\infty\mathbf{C}_{a_n o_n}\mathbf{C}_{a_{n-1}o_{n-1}}\dots\mathbf{C}_{a_1 o_1}\mathbf{c}_1 - \mathbb{P}(o_{1:n}||a_{1:n})\|_{\rho_n} \\
&\quad= \|\mathbf{c}_\infty(\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}} + \mathbf{e}_n) - \mathbb{P}(o_{1:n}||a_{1:n})\|_{\rho_n} \\
&\quad\leq \|\mathbf{c}_\infty\mathbf{e}_n\|_{\rho_n} + \|\mathbf{c}_\infty\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}} - \mathbb{P}(o_{1:n}|a_{1:n})\|_{\rho_n} \\
&\quad\leq \|\mathbf{c}_\infty\|\|\mathbf{e}_n\|_{\rho_n} + \|\mathbf{c}_\infty s_n\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}} - s_n\mathbb{P}(o_{1:n}||a_{1:n})\|_{\rho} \qquad (44) \\
&\quad\leq \|\mathbf{c}_\infty\|\epsilon\sum_{t=1}^{n-1} s_t c^{n-t-1} + \epsilon_\infty s_n. \qquad (45)
\end{aligned}$$

Line 44 uses the distribution assumption on ρ_n and Line 45 uses the bound of Theorem 3. ■

References

D. Achlioptas. Database-friendly random projections. In *Proceedings of the 20th ACM Symposium on Principles of Database Systems*, 2001.

A. Anandkumar, D. Hsu, and S. M. Kakade. A method of moments for mixture models and hidden Markov models. In *Proceedings of the 25th Annual Conference on Learning Theory*, 2012.

B. Balle, A. Quattoni, and X. Carreras. Local loss optimization in operator models: A new insight into spectral learning. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

R. Baraniuk and M. Wakin. Random projections of smooth manifolds. *Foundations of Computational Mathematics*, 9:51–77, 2009.

B. Boots and G. Gordon. Predictive state temporal difference learning. In *Advances in Neural Information Processing Systems*, 2010.

B. Boots and G. Gordon. An online spectral learning algorithm for partially observable dynamical systems. In *Association for the Advancement of Artificial Intelligence*, 2011.

B. Boots and G. Gordon. A spectral learning approach to range-only SLAM. In *Proceedings of the 30th International Conference on Machine Learning*, 2013.

B. Boots, S. Siddiqi, and G. Gordon. Closing the learning-planning loop with predictive state representations. In *Proceedings of Robotics: Science and Systems VI*, 2010.

B. Boots, G. Gordon, and A. Gretton. Hilbert space embeddings of predictive state representations. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*, 2013.

M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *Computer Vision - European Conference on Computer Vision*, pages 707–720. Springer, 2002.

F. Denis and Y. Esposito. On rational stochastic languages. *Fundamenta Informaticae*, 2008.

D. Ernst, P. Geurts, L. Wehenkel, and L. Littman. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

M.M. Fard, Y. Grinberg, J. Pineau, and D. Precup. Compressed least-squares regression on sparse spaces. In *Association for the Advancement of Artificial Intelligence*, 2012.

M.M. Fard, Y. Grinberg, A. Farahmand, J. Pineau, and D. Precup. Bellman error based feature generation using random projections on sparse spaces. In *Advances in Neural Information Processing Systems*, 2013.

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63 (1):3–42, 2006.

G. Gordon. *Approximate Solutions to Markov Decision Processes*. PhD thesis, Robotics Institute, Carnegie Mellon University, 1999.

W. L. Hamilton, M. M. Fard, and J. Pineau. Modelling sparse dynamical systems with compressed predictive state representations. In *Proceedings of the 30th International Conference on Machine Learning*, 2013.

D. Hsu, S. M. Kakade, and T. Zhang. A spectral algorithm for learning hidden Markov models. In *Proceedings of the 21st Annual Conference on Learning Theory*, 2008.

T. Iwamura. *Spatial Conservation Prioritisation Under Global Threats*. PhD thesis, University of Queensland, 2011.

M. T. Izadi and D. Precup. Point-based planning for predictive state representations. In S. Bergler, editor, *Advances in Artificial Intelligence*, volume 5032 of *Lecture Notes in Computer Science*, pages 126–137. 2008.

H. Jaeger. Observable operator models for discrete stochastic time series. *Neural Computation*, 12(6):1371–1398, 2000.

M. James and S. Singh. Learning and discovery of predictive state representations in dynamical systems with reset. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

M. James, B. Wolfe, and S. Singh. Combining memory and landmarks with predictive state representations. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 2005.

L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

A. Kulesza, N. R. Rao, and S. Singh. Low-rank spectral learning. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, 2014.

B. Laurent and P. Massart. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pages 1302–1338, 2000.

M. Littman, Richard S., and Satinder S. Predictive representations of state. In *Advances in Neural Information Processing Systems*, 2002.

O.A. Maillard and R. Munos. Compressed least-squares regression. In *Advances in Neural Information Processing Systems*, 2009.

O.A. Maillard and R. Munos. Linear regression with random projections. *Journal of Machine Learning Research*, 13:2735–2772, 2012.

T. G. Martin, I. Chadès, P. Arcese, P. Marra, H Possingham, and D. Norris. Optimal conservation of migratory species. *Public Library of Science One*, 2(8):e751, 2007.

A McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, The University of Rochester, 1996.

S. Nicol, O. Buffet, T. Iwamura, and I. Chadès. Adaptive management of migratory birds under sea level rise. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 2013.

S. C. W. Ong, Y. Grinberg, and J. Pineau. Goal-directed online learning of predictive models. In S. Sanner and M. Hutter, editors, *Recent Advances in Reinforcement Learning*, volume 7188 of *Lecture Notes in Computer Science*, pages 18–29. 2012.

M. Rosencrantz, G. Gordon, and S. Thrun. Learning low dimensional predictive representations. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

Q. Shi, J. Petterson, G. Dror, J. Langford, A. Smola, and S. V. N. Vishwanathan. Hash kernels for structured data. *The Journal of Machine Learning Research*, 10:2615–2637, 2009.

D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, 2010.

S. Singh, M. James, and M. Rudary. Predictive state representations: a new theory for modeling dynamical systems. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, 2004.

R. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*, volume 1. Cambridge University Press, 1998.

R. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999.

J. Veness, K.S. Ng, M. Hutter, W. Uther, and D. Silver. A Monte-Carlo AIXI approximation. *Journal of Artificial Intelligence Research*, 40(1):95–142, 2011.

K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

E. Wiewiora. *Modeling Probability Distributions with Predictive State Representations*. PhD thesis, University of California at San Diego, 2007.

B. Wolfe and S. Singh. Predictive state representations with options. In *Proceedings of the 23rd International Conference on Machine learning*, 2006.

B. Wolfe, M. James, and S. Singh. Learning predictive state representations in dynamical systems without reset. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

S. Zhou, J. Lafferty, and L. Wasserman. Compressed regression. In *Advances in Neural Information Processing Systems*, 2007.

Journal of Machine Learning Research 15 (2014) 3621-3662 Submitted 8/12; Revised 6/14; Published 11/14

Revisiting Stein's Paradox: Multi-Task Averaging

Sergey Feldman SERGEY.FELDMAN@GMAIL.COM
Data Cowboys
9126 23rd Ave. NE
Seattle, WA 98115, USA
Maya R. Gupta MAYAGUPTA@GOOGLE.COM
Google
1225 Charleston Rd
Mountain View, CA 94301, USA
Bela A. Frigyik FRIGYIK@GMAIL.COM
Institute of Mathematics and Informatics
University of Pécs
H-7624 Pécs, Ifjúság St. 6, Hungary

Editors: Ben Taskar, Massimiliano Pontil

Abstract

We present a multi-task learning approach to jointly estimate the means of multiple independent distributions from samples. The proposed multi-task averaging (MTA) algorithm results in a convex combination of the individual task's sample averages. We derive the optimal amount of regularization for the two task case for the minimum risk estimator and a minimax estimator, and show that the optimal amount of regularization can be practically estimated without cross-validation. We extend the practical estimators to an arbitrary number of tasks. Simulations and real data experiments demonstrate the advantage of the proposed MTA estimators over standard averaging and James-Stein estimation.

Keywords: multi-task learning, James-Stein, Stein's paradox

1. Introduction

The mean is one of the most fundamental and useful tools in statistics (Salsburg, 2001). By the 16th century Tycho Brahe was using the mean to reduce measurement error in astronomical investigations (Plackett, 1958). Legendre (1805) noted that the mean minimizes the sum of squared errors to a set of samples:

$$\bar{y} = \arg\min_{\tilde{y}} \sum_{i=1}^{N} (y_i - \tilde{y})^2. \tag{1}$$

More recently it has been shown that the mean minimizes the sum of any Bregman divergence to a set of samples (Banerjee et al., 2005; Frigyik et al., 2008). Gauss (1857) commented on the mean's popularity in his time:

> *"It has been customary certainly to regard as an axiom the hypothesis that if any quantity has been determined by several direct observations, made under the*

same circumstances and with equal care, the arithmetical mean of the observed values affords the most probable value, if not rigorously, yet very nearly at least, so that it is always most safe to adhere to it."

But the mean is a more subtle quantity than it first appears. In a surprising result popularly called *Stein's paradox* (Efron and Morris, 1977), Stein (1956) showed that it is better (in a summed squared error sense) to estimate *each* of the means of T Gaussian random variables using data sampled from *all* of them, even if the random variables are independent and have different means. That is, it is beneficial to consider samples from seemingly *unrelated* distributions to estimate a mean. Stein's result is an early example of the motivating hypothesis behind multi-task learning: that leveraging data from multiple tasks can yield superior performance over learning from each task independently. In this work we consider a multi-task regularization approach to the problem of estimating multiple means; we call this *multi-task averaging* (MTA).

Multi-task learning is most intuitive when the multiple tasks are *conceptually* similar. But we argue that it is really the *statistical* similarity of the multiple tasks that determines how well multi-task learning works. In fact, a key result of this paper is that proposed multi-task estimation achieves lower total squared error than the sample mean *if* the true means of the multiple tasks are close compared to the variance of the samples (see Equation 12). Of course, in practice cognitive notions of similarity can be a useful guide for multi-task learning, as tasks that seem similar to humans often do have similar statistics.

We begin the paper with the proposed MTA objective in Section 2, and review related work in Section 3. We show that MTA has provably nice theoretical properties in Section 4; in particular, we derive the optimal notion of task similarity for the two task case, which is also the optimal amount of regularization to be used in the MTA estimation. We generalize this analysis to form practical estimators for the general case of T tasks. Simulations in Section 5 verify the advantage of MTA over standard sample means and James-Stein estimation if the true means are close compared to the variance of the underlying distributions. In Section 6 we present four experiments on real data: (i) estimating Amazon customer reviews, (ii) estimating class grades, (iii) forecasting sales, and (iv) estimating the length of kings' reigns. These real-data experiments show that MTA is generally 10-20% better than the sample mean.

A short version of this paper was published in NIPS 2012 (Feldman et al., 2012). This paper substantially differs from that conference paper that it contains more analysis, proofs, and new and expanded experiments.

2. Multi-Task Averaging

Consider the problem of estimating the means of T random variables that have finite means $\{\mu_t\}$ and variances $\{\sigma_t^2\}$ for $t = 1, \ldots, T$. We treat this as a T-task multi-task learning problem, and estimate the T means jointly. We take as given N_t independent and identically distributed (iid) random samples $\{Y_{ti}\}_{i=1}^{N_t}$ for each task t. Key notation is summarized in Table 1.

T	number of tasks
N_t	number of samples for tth task
μ_t	true mean of task t
σ_t^2	variance of the tth task
$Y_{ti} \in \mathbb{R}$	ith random sample from tth task
$\bar{Y}_t \in \mathbb{R}$	sample average for tth task: $\frac{1}{N_t}\sum_i Y_{ti}$, also referred to as the *single-task* mean estimate
$\bar{Y} \in \mathbb{R}^T$	vector with tth component $\bar{Y}_t$
$Y_t^* \in \mathbb{R}$	MTA estimate of tth mean
$Y^* \in \mathbb{R}^T$	vector with tth component Y_t^*
$\hat{Y}_t \in \mathbb{R}$	an estimate of the tth mean
$\tilde{Y}_t \in \mathbb{R}$	dummy variable
$\Sigma \in \mathbb{R}^{T\times T}$	diagonal covariance matrix of $\bar{Y}$ with $\Sigma_{tt} = \frac{\sigma_t^2}{N_t}$
$A \in \mathbb{R}^{T\times T}$	pairwise task similarity matrix
$L = D - A$	graph Laplacian of A, with diagonal D s.t. $D_{tt} = \sum_{r=1}^T A_{tr}$
W	MTA solution matrix, $W = (I + \frac{\gamma}{T}\Sigma L)^{-1}$

Table 1: Key Notation

In this paper, we judge the estimates by total squared error: given T estimates $\{\hat{Y}_t\}$ and T true means $\{\mu_t\}$:

$$\textbf{estimation error}(\{\hat{Y}_t\}) \triangleq \sum_{t=1}^{T} \left(\mu_t - \hat{Y}_t\right)^2, \tag{2}$$

Equivalently (up to an additive factor), the metric can be expressed as the total squared expected error to a random sample Y_t from each task:

$$\textbf{regression error}(\{\hat{Y}_t\}) \triangleq \sum_{t=1}^{T} E\left[\left(Y_t - \hat{Y}_t\right)^2\right]; \tag{3}$$

we use an empirical approximation to (3) in the experiments because the true means are not known but held-out samples from the distributions are available.

Let a $T \times T$ matrix A describe the relatedness or similarity of any pair of the T tasks, with $A_{tt} = 0$ for all t without loss of generality (because the diagonal self-similarity terms are canceled in the objective below). Further we assume each task's variance σ_t^2 is known or already estimated. The proposed MTA objective is

$$\{Y_t^*\}_{t=1}^T = \underset{\{\tilde{Y}_t\}_{t=1}^T}{\arg\min} \ \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N_t}\frac{(Y_{ti} - \tilde{Y}_t)^2}{\sigma_t^2} + \frac{\gamma}{T^2}\sum_{r=1}^{T}\sum_{s=1}^{T} A_{rs}(\tilde{Y}_r - \tilde{Y}_s)^2. \tag{4}$$

The first term of (4) minimizes the multi-task empirical loss, normalizing the contribution of each task's losses by that task's variance σ_t^2 so that high-variance tasks do not disproportionately dominate the loss term. The second term of (4) jointly regularizes the estimates

by tying them together. The regularization parameter γ balances the empirical risk and the multi-task regularizer. If $\gamma = 0$, the MTA objective decomposes into T separate minimization problems, producing the sample averages $\{\bar{Y}_t\}$. If $\gamma = 1$, the balance between empirical risk and multi-task regularizer is completely specified by the task similarity matrix A.

A more general formulation of MTA is

$$\{Y_t^*\}_{t=1}^T = \underset{\{\tilde{Y}_t\}_{t=1}^T}{\arg\min} \ \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N_t} L(Y_{ti}, \tilde{Y}_t) + \gamma J\left(\{\tilde{Y}_t\}_{t=1}^T\right),$$

where L is some loss function and J is some regularization function. If L is chosen to be any Bregman loss, then setting $\gamma = 0$ will produce the T sample averages (Banerjee et al., 2005). For the analysis and experiments in this paper, we restrict our focus to the tractable squared-error formulation given in (4). The MTA objective and many of the results in this paper generalize straightforwardly to samples that are vectors rather than scalars (see Section 4.2), but for most of the paper we restrict our focus to scalar samples $Y_{ti} \in \mathbb{R}$.

The task similarity matrix A can be specified as side information, for example from a domain expert, but often this side information is not available, or it may not be clear how to convert semantic notions of task similarity into appropriate numerical values for the task-similarity values in A. In such cases, A can be treated as a matrix parameter of the MTA objective, and in Section 4 we fix $\gamma = 1$ and derive two optimal choices of A for the $T = 2$ case: the A that minimizes expected squared error, and a minimax A. We use the $T = 2$ analysis to propose practical estimators of A for any number of tasks, removing the need to cross-validate the amount of regularization.

3. Related Work

In this section, we review related and background material: James-Stein estimation, multi-task learning, manifold regularization, and the graph Laplacian.

3.1 James-Stein Estimation

A closely related body of work to MTA is Stein estimation (James and Stein, 1961; Bock, 1975; Efron and Morris, 1977; Casella, 1985), which can be derived as an empirical Bayes strategy for estimating multiple means simultaneously (Efron and Morris, 1972). James and Stein (1961) showed that the maximum likelihood estimate of the task mean can be dominated by a shrinkage estimate given Gaussian assumptions. Specifically, given a single sample drawn from T normal distributions $Y_t \sim \mathcal{N}(\mu_t, \sigma^2)$ for $t = 1, \ldots, T$, Stein showed that the maximum likelihood estimator $\bar{Y}_t = Y_t$ is inadmissible, and is dominated by the James-Stein estimator:

$$\hat{Y}_t^{JS} = \left(1 - \frac{(T-2)\sigma^2}{\bar{Y}^\top \bar{Y}}\right)\bar{Y}_t, \tag{5}$$

where $\bar{Y}$ is a vector with tth entry $\bar{Y}_t$. The above estimator dominates $\bar{Y}_t$ when $T > 2$. For $T = 2$, (5) reverts to the maximum likelihood estimator, which turns out to be admissible (Stein, 1956). James and Stein showed that if σ^2 is unknown it can be replaced by a standard unbiased estimate $\hat{\sigma}^2$ (James and Stein, 1961; Casella, 1985).

Note that in (5) the James-Stein estimator *shrinks* the sample means towards zero (the terms "regularization" and "shrinkage" are often used interchangeably). The form of (5) and its shrinkage towards zero points to the implicit assumption that the μ_t are themselves drawn from a standard normal distribution centered at 0. More generally, the means are assumed to be drawn as $\mu_t \sim \mathcal{N}(\xi, 1)$. The James-Stein estimator then becomes

$$\hat{Y}_t^{JS} = \xi + \left(1 - \frac{(T-3)\sigma^2}{(\bar{Y} - \xi\mathbf{1})^\top(\bar{Y} - \xi\mathbf{1})}\right)(\bar{Y}_t - \xi), \tag{6}$$

where ξ can be estimated (as we do in this work) as the average of means $\xi = \bar{\bar{Y}} = \frac{1}{T}\sum_{r=1}^{T}\bar{Y}_r$, and this additional estimation decreases the degrees of freedom by one.[1] Note that (6) shrinks the estimates towards the mean-of-means ξ rather than shrinking towards zero. Also, the more similar the multiple tasks are (in the sense that individual task means are closer to the mean-of-means ξ), the more regularization occurs in (6).

There have been a number of extensions to the original James-Stein estimator. The James-Stein estimator given in (6) is itself not admissible, and is dominated by the positive part James-Stein estimator (Lehmann and Casella, 1998), which was further theoretically improved by Bock's James-Stein estimator (Bock, 1975). Throughout this work, we compare to Bock's well-regarded positive-part James-Stein estimator for multiple data points per task and independent unequal variances (Bock, 1975; Lehmann and Casella, 1998). In particular, let $Y_{ti} \sim \mathcal{N}(\mu_t, \sigma_t^2)$ for $t = 1, \ldots, T$ and $i = 1, \ldots, N_t$, let Σ be the covariance matrix of the vector of task sample means $\bar{Y}$, and let $\lambda_{\max}(\Sigma)$ be the largest eigenvalue of Σ, then the estimator is

$$\hat{Y}_t^{JS} = \xi + \left(1 - \frac{\frac{\mathbf{tr}(\Sigma)}{\lambda_{\max}(\Sigma)} - 3}{(\bar{Y} - \xi\mathbf{1})^\top\Sigma^{-1}(\bar{Y} - \xi\mathbf{1})}\right)_+(\bar{Y}_t - \xi), \tag{7}$$

where $(x)_+ = \max(0, x)$.

3.2 Multi-Task Learning for Mean Estimation

MTA is an approach to the problem of estimating T means. We are not aware of other work in the multi-task literature that addresses this problem explicitly; most multi-task learning methods are designed for regression, classification, or feature selection, for example, Micchelli and Pontil (2004); Bonilla et al. (2008); Argyriou et al. (2008). Estimating T means can be considered a special case of multi-task regression, where one fits a constant function to each task, since, with a feature space of zero dimensions only the constant offset term is learned. And, similarly to MTA, one of the main approaches to multi-task regression in the literature is tying tasks together with an explicit multi-task parameter regularizer.

Abernethy et al. (2009), for instance, propose to minimize the empirical loss by adding the regularizer

$$||\beta||_*,$$

where the tth column of the matrix β is the vector of parameters for the tth task and $||\cdot||_*$ is the trace norm. Applying this approach to mean estimation, the matrix β has only one row,

1. For more details as to why $T-2$ in (5) becomes $T-3$ in (6), see Example 7.7 on page 278 of Lehmann and Casella (1998).

and $||\beta||_*$ reduces to the ℓ_2 norm on the outputs, thus for mean estimation this regularizer does not actually tie the tasks together.

Argyriou et al. (2008) propose a a different regularizer,

$$\mathbf{tr}(\beta^\top D^{-1}\beta),$$

where D is a learned, shared *feature* covariance matrix. With no features (as in the MTA application of constant function regression), D is just a constant and $\mathbf{tr}(\beta^\top D^{-1}\beta)$ is a ridge regularizer on the outputs. The regularizers in the work of Jacob et al. (2008) and Zhang and Yeung (2010) reduce similarly when applied to mean estimation. These regularizers are similar to the *original* James Stein estimator in that they shrink the estimates towards zero; though more modern James Stein estimators shrink towards a pooled mean (see Section 3.1).

The most closely related work is that of Sheldon (2008) and Kato et al. (2008), where the regularizer or constraint, respectively, is

$$\sum_{r=1}^{T}\sum_{s=1}^{T} A_{rs}\|\beta_r - \beta_s\|_2^2,$$

which is the MTA regularizer if applied to mean estimation. In this paper we do just that: apply this regularizer to mean estimation, show that this special case enables new and useful analytic results, and demonstrate its performance with simulated and real data.

3.3 Multi-Task Learning and the Similarity Between Tasks

A key issue for MTA and many other multi-task learning methods is how to estimate some notion of similarity (or task relatedness) between tasks and/or samples if it is not provided. A common approach is to estimate the similarity matrix jointly with the task parameters (Argyriou et al., 2007; Xue et al., 2007; Bonilla et al., 2008; Jacob et al., 2008; Zhang and Yeung, 2010). For example, Zhang and Yeung (2010) assume that there exists a covariance matrix for the task relatedness, and proposed a convex optimization approach to estimate the task covariance matrix and the task parameters in a joint, alternating way. Applying such joint and alternating approaches to the MTA objective given in (4) leads to a degenerate solution with zero similarity. However, the simplicity of MTA enables us to specify the optimal task similarity matrix for $T = 2$ (see Section 4), which we use to obtain closed-form estimators for the general $T > 1$ case.

3.4 Manifold Regularization

MTA is similar in form to *manifold regularization* (Belkin et al., 2006). For example, Belkin et al.'s Laplacian-regularized least squares objective for semi-supervised regression solves

$$\arg\min_{f\in\mathcal{H}} \quad \sum_{i=1}^{N}(y_i - f(x_i))^2 + \lambda||f||_{\mathcal{H}}^2 + \gamma\sum_{i,j=1}^{N+M} A_{ij}(f(x_i) - f(x_j))^2,$$

where f is the regression function to be estimated, $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS), N is the number of labeled training samples, M is the number of unlabeled training samples, A_{ij} is the similarity (or weight in an adjacency graph) between feature samples

x_i and x_j, and $||f||_{\mathcal{H}}$ is the norm of the function f in the RKHS. In MTA, as opposed to manifold regularization, we are estimating a different function (that is, the constant function that is the mean) for each of the T tasks, rather than a single global function. One can interpret MTA as regularizing the individual task estimates over the task-similarity manifold, which is defined for the T tasks by the $T \times T$ matrix A.

3.5 Background on the Graph Laplacian Matrix

It will be helpful for later sections to review the graph Laplacian matrix. For graph G with T nodes, let $A \in \mathbb{R}^{T \times T}$ be a matrix where component $A_{rs} \geq 0$ is the weight of the edge between node r and node s, for all r, s. The *graph Laplacian matrix* is defined as $L = L(A) = D - A$, with diagonal matrix D such that $D_{tt} = \sum_s A_{ts}$.

The graph Laplacian matrix is analogous to the Laplacian operator, which quantifies how locally smooth a twice-differentiable function $g(x)$ is. Similarly, the graph Laplacian matrix L can be thought of as being a measure of the smoothness of a function defined on a graph. Given a function f defined over the T nodes of graph G, where $f_i \in \mathbb{R}$ is the function value at node i, the total *energy* of a graph is (for symmetric A)

$$\mathcal{E}(f) = \frac{1}{2} \sum_{i=1}^{T} \sum_{j=1}^{T} A_{ij}(f_i - f_j)^2 = f^\top L(A) f,$$

which is small when f is smooth over the graph (Zhu and Lafferty, 2005). If A is asymmetric then the energy can be written as

$$\mathcal{E}(f) = \frac{1}{2} \sum_{i=1}^{T} \sum_{j=1}^{T} A_{ij}(f_i - f_j)^2 = f^\top L((A + A^\top)/2) f. \tag{8}$$

When each $f_i \in \mathbb{R}^d$ is a vector, one can alternatively write the energy in terms of the distance matrix:

$$\mathcal{E}(f) = \frac{1}{2} \mathbf{tr}(\Delta^\top A),$$

where $\Delta_{ij} = (f_i - f_j)^\top (f_i - f_j)$.

As discussed above, the graph Laplacian can be thought of as an operator on a function, but it is useful in and of itself (i.e., without a function). The eigenvalues of the graph Laplacian are all real and non-negative, and there is a wealth of literature showing how the eigenvalues reveal the structure of the underlying graph and can be used for clustering (v. Luxburg, 2007). The graph Laplacian is a common tool in semi-supervised learning literature (Zhu and Goldberg, 2009), and the Laplacian of a random walk probability matrix P (i.e., all the entries are non-negative and the rows sum to 1) is also of interest. For example, Saerens et al. (2004) showed that the pseudo-inverse of the Laplacian of a probability transition matrix is used to compute the square root of the average commute time (the average time taken by a random walker on graph G to reach node j for the first time when starting at node i, and coming back to node i).

Finally, since we will be using this fact occasionally, we note that the graph Laplacian is *homogeneous*, i.e., $L(\gamma A) = \gamma L(A)$, where A is a matrix and γ is a scalar.

4. MTA Theory and Estimators

First, we give a general closed-form solution for the MTA mean estimates and characterize the MTA objective in Sections 4.1 through 4.3. Then in Section 4.4 we analyze the estimation error for the two task $T = 2$ case and give conditions for when MTA is better than the sample means. In Section 4.5, we derive the optimal similarity matrix A for the two task case.

Then in Section 4.7, we generalize our $T = 2$ analysis to propose practical estimators for any number of tasks T, and analyze their computational efficiency. In Section 4.8, we analyze the relationship of different estimators formed by linearly combining the sample means, including the MTA estimators, James Stein, and other estimators that regularize sample means towards a pooled mean. Lastly, we discuss the Bayesian interpretation of MTA in Section 4.9.

Proofs and derivations are in the appendix.

4.1 Closed-form MTA Solution

Without loss of generality, we only deal with symmetric A because in the case of asymmetric A it is equivalent to consider instead the symmetrized matrix $(A^\top + A)/2$.

For symmetric A with non-negative components, the MTA objective given in (4) is continuous, differentiable, and convex; and (4) has closed-form solution (derivation in Appendix A):

$$Y^* = \left(I + \frac{\gamma}{T}\Sigma L\right)^{-1} \bar{Y}, \tag{9}$$

where $\bar{Y}$ is the vector of sample averages with tth entry $\bar{Y}_t = \frac{1}{N_t}\sum_{i=1}^{N_t} Y_{ti}$, L is the graph Laplacian of A, and Σ is the diagonal covariance matrix of the sample mean vector $\bar{Y}$ such that $\Sigma_{tt} = \frac{\sigma_t^2}{N_t}$. The inverse $\left(I + \frac{\gamma}{T}\Sigma L\right)^{-1}$ in (9) always exists:

Lemma 1 *Suppose that $0 \le A_{rs} < \infty$ for all r, s, $\gamma \ge 0$, and $0 < \frac{\sigma_t^2}{N_t} < \infty$ for all t. The MTA solution matrix $W = \left(I + \frac{\gamma}{T}\Sigma L\right)^{-1}$ exists.*

The MTA estimates Y^* converge to the vector of true means μ:

Proposition 2 *As $N_t \to \infty \ \forall\ t$, $Y^* \to \mu$.*

4.2 MTA for Vectors

MTA can also be applied to vectors. Let $\mathbb{Y}^* \in \mathbb{R}^{T\times d}$ be a matrix with Y_t^* as its tth row and let $\bar{\mathbb{Y}} \in \mathbb{R}^{T\times d}$ be a matrix with $\bar{Y}_t \in \mathbb{R}^d$ as its tth row. One can perform MTA on the vectorized form of $\mathbb{Y}^*$:

$$\mathbf{vec}(\mathbb{Y}^*) = \left(I + \frac{\gamma}{T}\Sigma L\right)^{-1} \mathbf{vec}(\bar{\mathbb{Y}}),$$

as long as (the now block-diagonal) $\Sigma \in R^{Td\times Td}$ is invertible. An equivalent formulation for MTA for vectors was proposed in Martínez-Rego and Pontil (2013).

4.3 Convexity of MTA Solution

One sees from (9) that the MTA estimates are linear combinations of the sample averages:

$$Y^* = W\bar{Y}, \text{ where } W = \left(I + \frac{\gamma}{T}\Sigma L\right)^{-1}.$$

Moreover, and less obviously, each MTA estimate is a *convex* combination of the single-task sample averages:

Theorem 3 *If $\gamma \geq 0$, $0 \leq A_{rs} < \infty$ for all r, s and $0 < \frac{\sigma_t^2}{N_t} < \infty$ for all t, then the MTA estimates $\{Y_t^*\}$ given in (9) are convex combinations of the task sample averages $\{\bar{Y}_t\}$.*

This theorem generalizes a result of Chebotarev and Shamis (2006) that the matrix $(I + \gamma L)^{-1}$ is right-stochastic (i.e., the rows are non-negative and sum to 1) if the entries of A are strictly positive. Our proof (given in the appendix) uses a different approach, and extends the result both to the more general form of the MTA solution matrix $\left(I + \frac{\gamma}{T}\Sigma L\right)^{-1}$ and to A with non-negative entries.

4.4 MSE Analysis for the Two Task Case

In this section we analyze the $T = 2$ task case, with N_1 and N_2 samples for tasks 1 and 2 respectively. Suppose random samples drawn for the first task $\{Y_{1i}\}$ are iid with finite mean μ_1 and finite variance σ_1^2, and random samples drawn for the second task $\{Y_{2i}\}$ are iid with finite mean $\mu_2 = \mu_1 + \Delta$ and finite variance σ_2^2. Let the task-relatedness matrix be $A = [0\ a; a\ 0]$, and without loss of generality, we fix $\gamma = 1$. Then the closed-form solution (9) can be simplified:

$$Y_1^* = \left(\frac{2 + \frac{\sigma_2^2}{N_2}a}{2 + \frac{\sigma_1^2}{N_1}a + \frac{\sigma_2^2}{N_2}a}\right)\bar{Y}_1 + \left(\frac{\frac{\sigma_1^2}{N_1}a}{2 + \frac{\sigma_1^2}{N_1}a + \frac{\sigma_2^2}{N_2}a}\right)\bar{Y}_2. \tag{10}$$

The mean squared error of Y_1^* is

$$\mathrm{MSE}[Y_1^*] = \frac{\sigma_1^2}{N_1}\left(\frac{4 + 4\frac{\sigma_2^2}{N_2}a + \frac{\sigma_1^2\sigma_2^2}{N_1N_2}a^2 + \frac{\sigma_2^4}{N_2^2}a^2}{\left(2 + \frac{\sigma_1^2}{N_1}a + \frac{\sigma_2^2}{N_2}a\right)^2}\right) + \frac{\Delta^2\frac{\sigma_1^4}{N_1^2}a^2}{\left(2 + \frac{\sigma_1^2}{N_1}a + \frac{\sigma_2^2}{N_2}a\right)^2}. \tag{11}$$

Next, we compare the MTA estimate Y_1^* to the sample average $\bar{Y}_1$, which is the maximum likelihood estimate of the true mean μ_1 for many distributions.[2] The MSE of the single-task sample average $\bar{Y}_1$ is $\frac{\sigma_1^2}{N_1}$, and comparing that to (11) and simplifying some tedious algebra establishes that

$$\mathrm{MSE}[Y_1^*] < \mathrm{MSE}[\bar{Y}_1] \text{ if } \Delta^2 < \frac{4}{a} + \frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}. \tag{12}$$

Thus the MTA estimate of the first mean has lower MSE than the sample average estimate if the squared mean-separation Δ^2 is small compared to the summed variances of the sample means. See Figure 1 for an illustration.

2. The uniform distribution is perhaps the simplest example where the sample average is *not* the maximum likelihood estimate of the mean. For more examples, see Section 8.18 of Romano and Siegel (1986).

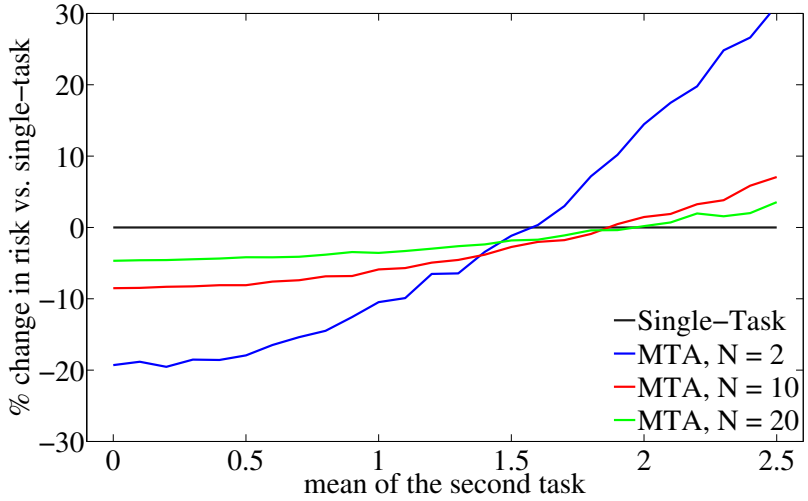


Figure 1: Plot shows the percent change in average risk for two tasks (averaged over 10,000 runs of the simulation). For each task there are N iid samples, for $N = 2, 10, 20$. The first task generates samples from a standard Gaussian. The second task generates samples from a Gaussian with $\sigma^2 = 1$ and different mean value, which is varied as marked on the x-axis. The symmetric task-relatedness value was fixed at $a = 1$ (note this is generally *not* the optimal value). One sees that given $N = 2$ samples from each Gaussian, the MTA estimate is better than the single-task sample if the difference between the true means is less than 1.5. Given $N = 20$ samples from each Gaussian, the MTA estimate is better if the distance between the means is less than 2. In the extreme case that the two Gaussians have the same mean ($\mu_1 = \mu_2 = 0$), then even with this suboptimal choice of $a = 1$, MTA provides a 20% win for $N = 2$ samples, and a 5% win for $N = 20$ samples.

Further, because of the symmetry of (12), if the condition of (12) holds, it is also true that $\mathrm{MSE}[Y_2^*] < \mathrm{MSE}[\bar{Y}_2]$, such that the MSE of each task individually is reduced.

The condition (12) shows that even when the true means are far apart such that Δ is large, there is some tiny amount of MTA regularization a that will improve the estimates.

4.5 Optimal Task Relatedness A for $T = 2$

We analyze the optimal choice of a in the task-similarity matrix $A = [0\ a; a\ 0]$. The risk is the sum of the mean squared errors:

$$R(\mu, Y^*) = \mathrm{MSE}[Y_1^*] + \mathrm{MSE}[Y_2^*], \tag{13}$$

which is a convex, continuous, and differentiable function of a, and therefore the first derivative can be used to specify the optimal value a^*, when all the other variables are fixed.

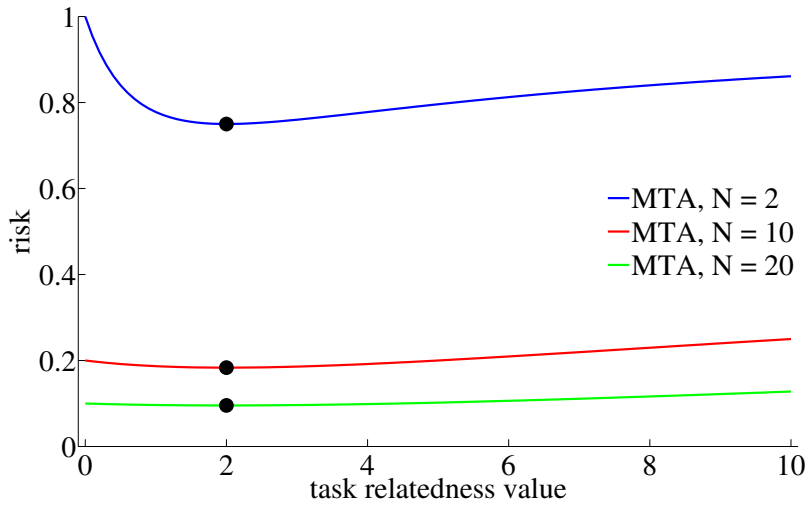


Figure 2: Plot shows the risk for two tasks, where the task samples were drawn iid from Gaussians $\mathcal{N}(0,1)$ and $\mathcal{N}(1,1)$. The task-relatedness value a was varied as shown on the x-axis. The minimum expected squared error is marked by a dot, and occurs for the choice of a given by (14), and is independent of N.

Minimizing (13) w.r.t. a one obtains the optimal:

$$a^* = \frac{2}{\Delta^2}, \tag{14}$$

which is always non-negative, as was assumed. This result is key because it specifies that the optimal task-similarity a^* ideally should measure the inverse of the squared task mean-difference. Further, the optimal task-similarity is independent of the number of samples N_t or the sample variance σ_t^2, as these are accounted for in Σ of the MTA objective. Note that a^* also minimizes the functions $\mathrm{MSE}[Y_1^*]$ and $\mathrm{MSE}[Y_2^*]$, separately.

The effect on the risk on the choice of a and the optimal a^* is illustrated in Figure 2.

Analysis of the second derivative shows that this minimizer always holds for $N_1, N_2 \geq 1$.

In the limit case, when the difference in the task means Δ goes to zero (while σ_t^2 stay constant), the optimal task-relatedness a^* goes to infinity, and the weights in (10) on $\bar{Y}_1$ and $\bar{Y}_2$ become $1/2$ each.

4.6 Estimating Task Similarity from Data for $T = 2$ Tasks

The optimal two-task similarity given in (14) requires knowledge of the true means μ_1 and μ_2. These are, in practice, unavailable. What similarity should be used then? A straightforward approach is to use single-task estimates instead:

$$\hat{a}^* = \frac{2}{(\bar{y}_1 - \bar{y}_2)^2},$$

and to use maximum likelihood estimates $\hat{\sigma}_t^2$ to form the matrix $\hat{\Sigma}$. This data-dependent approach is analogous to empirical Bayesian methods in which prior parameters are estimated from data (Casella, 1985).

4.7 Estimating Task Similarity from Data for Arbitrary T Tasks

Based on our analysis in the preceding sections of the optimal A for the two-task case, we propose two methods to estimate A from data for arbitrary $T > 1$. The first method is designed to minimize the approximate risk using a constant similarity matrix. The second method provides a minimax estimator. With both methods one can take advantage of the Sherman-Morrison formula (Sherman and Morrison, 1950) to avoid taking the matrix inverse or solving a set of linear equations in (9) (detailed in Section 4.7.3). For the special case that all task variances are assumed equal (an assumption used in all of our experiments), the computation time is $O(T)$.

4.7.1 MTA Constant

The *risk* of estimator $\hat{Y} = W\bar{Y}$ is

$$R(\mu, W\bar{Y}) = E[(W\bar{Y} - \mu)^\top (W\bar{Y} - \mu)] \tag{15}$$

$$= \mathbf{tr}(W\Sigma W^\top) + \mu^\top (I - W)^\top (I - W)\mu, \tag{16}$$

where (16) uses the fact that $E[\bar{Y}\bar{Y}^\top] = \mu\mu^\top + \Sigma$.

One approach to generalizing the results of Section 4.4 to arbitrary T is to try to find a symmetric, non-negative matrix A such that the (convex, differentiable) risk $R(\mu, W\bar{Y})$ is minimized for $W = \left(I + \frac{\gamma}{T}\Sigma L\right)^{-1}$ (recall L is the graph Laplacian of A). The problem with this approach is two-fold: (i) the solution is not analytically tractable for $T > 2$ and (ii) an arbitrary A has $T(T-1)$ degrees of freedom, which is considerably more than the T means we are trying to estimate in the first place. To avoid these problems, we generalize the two-task results by constraining A to be a scaled constant matrix $A = a\mathbf{1}\mathbf{1}^\top$, and find the optimal a^* that minimizes the risk given by (16). As in Section 4.4, we fix $\gamma = 1$. For analytic tractability, we add the assumption that all the Y_t have the same variance, estimating Σ as $\frac{\mathbf{tr}(\Sigma)}{T} I$. Then minimizing (15) becomes:

$$a^* = \arg\min_a R\left(\mu, \left(I + \frac{1}{T}\frac{\mathbf{tr}(\Sigma)}{T} L(a\mathbf{1}\mathbf{1}^\top)\right)^{-1} \bar{Y}\right),$$

which has the solution

$$a^* = \frac{2}{\frac{1}{T(T-1)} \sum_{r=1}^{T} \sum_{s=1}^{T} (\mu_r - \mu_s)^2}, \tag{17}$$

which does reduce to the optimal two task MTA solution (14) when $T = 2$.

While (17) is theoretically interesting, in practice, one of course does not have $\{\mu_r\}$ as these are precisely the values one is trying to estimate, and thus cannot use (17) directly. Instead, we propose estimating a^* using the sample means $\{\bar{y}_r\}$:

$$\hat{a}^* = \frac{2}{\frac{1}{T(T-1)} \sum_{r=1}^{T} \sum_{s=1}^{T} (\bar{y}_r - \bar{y}_s)^2}. \tag{18}$$

Using the optimal estimated *constant* similarity given in (18) and an estimated covariance matrix $\hat{\Sigma}$ produces what we refer to as the *MTA Constant* estimate

$$Y^* = \left(I + \frac{1}{T}\hat{\Sigma}L(\hat{a}^*\mathbf{1}\mathbf{1}^\top)\right)^{-1}\bar{Y}. \tag{19}$$

Note that we made the assumption that the entries of Σ were the same in order to be able to derive (17), but we do not need nor necessarily suggest that assumption on the $\hat{\Sigma}$ be used in practice with $\hat{a}^*$ in (19).

4.7.2 MTA Minimax

Bock's James-Stein estimator is *minimax* (Lehmann and Casella, 1998)). In this section, we derive a minimax version of MTA for arbitrary T that prescribes less regularization than MTA Constant. Formally, an estimator Y^M of μ is called minimax if it minimizes the maximum risk:

$$\inf_{\tilde{Y}}\sup_{\mu} R(\mu, \tilde{Y}) = \sup_{\mu} R(\mu, Y^M).$$

Let $r(\pi, \hat{Y})$ be the average risk of estimator $\hat{Y}$ w.r.t. a prior $\pi(\mu)$ such that $r(\pi, \hat{Y}) = \int R(\mu, \hat{Y})\pi(\mu)d\mu$. The Bayes estimator Y^π is the estimator that minimizes the average risk, and the Bayes risk $r(\pi, Y^\pi)$ is the average risk of the Bayes estimator. A prior distribution π is called least favorable if $r(\pi, Y^\pi) > r(\pi', Y^{\pi'})$ for all priors π'.

First, we will specify MTA Minimax for the $T = 2$ case. To find a minimax estimator Y^M it is sufficient to show that *(i)* Y^M is a Bayes estimator w.r.t. the least favorable prior (LFP) and *(ii)* it has constant risk (Lehmann and Casella, 1998). To find a LFP, we first need to specify a constraint set for μ_t; we use an interval: $\mu_t \in [b_l, b_u]$, for all t, where $b_l \in \mathbb{R}$ and $b_u \in \mathbb{R}$. With this constraint set the minimax estimator is (see appendix for details):

$$Y^M = \left(I + \frac{2}{T(b_u - b_l)^2}\Sigma L(\mathbf{1}\mathbf{1}^\top)\right)^{-1}\bar{Y},$$

which reduces to (14) when $T = 2$. This minimax analysis is only valid for the case when $T = 2$, but we found that the following extension of MTA Minimax to larger T worked well in simulations and applications for any $T \geq 2$. To estimate b_u and b_l from data we assume the unknown T means are drawn from a uniform distribution and use maximum likelihood estimates of the lower and upper endpoints for the support:

$$\hat{b}_l = \min_t \bar{y}_t \quad \text{and} \quad \hat{b}_u = \max_t \bar{y}_t.$$

Thus, in practice, *MTA Minimax* is

$$Y^M = \left(I + \frac{2}{T(\hat{b}_u - \hat{b}_l)^2}\hat{\Sigma}L(\mathbf{1}\mathbf{1}^\top)\right)^{-1}\bar{Y}.$$

4.7.3 Computational Efficiency of MTA Constant and Minimax

Both MTA Constant and MTA Minimax weight matrices can be written as

$$\begin{aligned}(I + c\Sigma L(\mathbf{1}\mathbf{1}^\top))^{-1} &= (I + c\Sigma(TI - \mathbf{1}\mathbf{1}^\top))^{-1} \\ &= (I + cT\Sigma - c\Sigma\mathbf{1}\mathbf{1}^\top)^{-1} \\ &= (Z - z\mathbf{1}^\top)^{-1},\end{aligned}$$

where c is different for MTA Constant and MTA Minimax, $Z = I + cT\Sigma$, $z = c\Sigma\mathbf{1}$. The Sherman-Morrison formula (Sherman and Morrison, 1950) can be used to find the inverse:

$$(Z - z\mathbf{1}^\top)^{-1} = Z^{-1} + \frac{Z^{-1}z\mathbf{1}^\top Z^{-1}}{1 - \mathbf{1}^\top Z^{-1}z}.$$

Since Z is diagonal, Z^{-1} can be computed in $O(T)$ time, and so can $Z^{-1}z$.

Further the computation becomes $O(T)$ if the covariance matrix Σ is taken to be diagonal with constant component σ^2, an assumption we use in all our experiments. In that case, compute the constant $v = 1/(1 + c\sigma^2)$, and then the MTA estimate reduces to a convex combination of the task sample average and the pooled sample average: $v\bar{Y} + (1-v)\sum_t^T \bar{Y}_t$.

4.8 Generality of MTA

In this section, we use the expression 'matrices of MTA form' to refer to matrices that can be written

$$(I + \Gamma L(A))^{-1}, \tag{20}$$

where A is a matrix with all non-negative entries, and Γ is a diagonal matrix with all non-negative entries. Matrices of the form $(I + \gamma L)^{-1}$ have been used as graph kernels (Fouss et al., 2006; Yajima and Kuo, 2006), and were termed regularized Laplacian kernels (RLKs) by Smola and Kondor (2003). The RLK assumes that A (and L) are symmetric, and thus MTA and (20) strictly generalizes the RLK because ΓL is only symmetric for some special cases such as when Γ is a scaled identity matrix. Thus, one might also refer to matrices of the form (20) as generalized regularized Laplacian kernels, but in this section we focus on their role as estimators and in understanding relationships with the proposed MTA estimator.

Figure 3 is a Venn diagram of the sets of estimators that can be expressed $\hat{Y} = W\bar{Y}$, where W is some $T \times T$ matrix. The first subset (the pink region) is all estimators where W is right-stochastic. The second subset (the green region) is estimators of MTA form as per (20). The innermost subset (the purple region) includes many well-known estimators such as the James-Stein estimator, and estimators that regularize single-task estimates of the mean to the pooled mean or the average of means. In this section we will prove that the innermost purple subset is a *strict* subset of the green MTA subset, such that any innermost estimator can be written in MTA form for specific choices of A, γ, and Σ. Note that the covariance Σ is treated as a "choice" because some classic estimators assume $\Sigma = I$.

Proposition 4 *The set of estimators $W\bar{Y}$ where W is of MTA form as per (20) is strictly larger than the set of estimators that regularize the single-task estimates as follows:*

$$\hat{Y} = \left(\frac{1}{\gamma}I + \mathbf{1}\alpha^\top\right)\bar{Y},$$

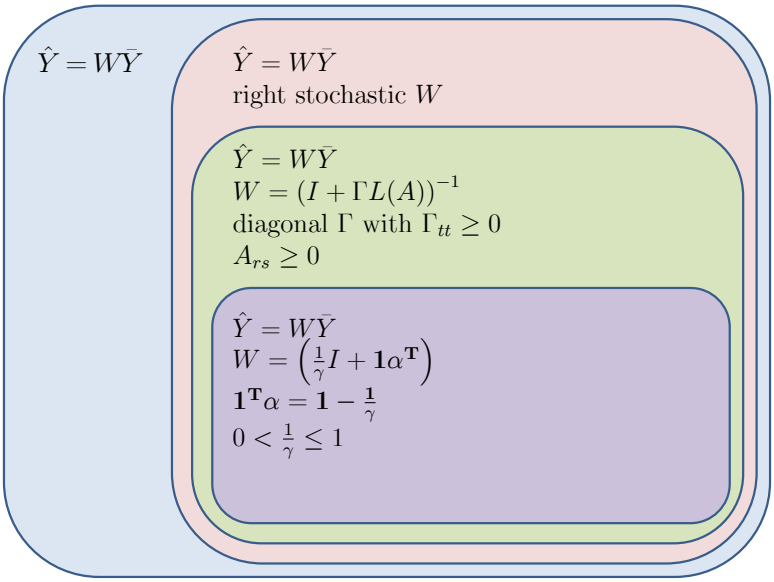


Figure 3: A Venn diagram of the set membership properties of various estimators of the type $\hat{Y} = W\bar{Y}$.

where $\sum_{r=1}^{T} \alpha_r = 1 - \frac{1}{\gamma}$, $\gamma \geq 1$, *and* $\alpha_r \geq 0$, $\forall r$.

Corollary 5 *Estimators that regularize the single task estimate towards the pooled mean such that they can be written*

$$\check{Y}_t = \lambda \bar{Y}_t + \frac{1-\lambda}{\sum_{r=1}^{T} N_r} \sum_{s=1}^{T} \sum_{i=1}^{N_s} Y_{si},$$

for $\lambda \in (0, 1]$ *can also be written in MTA form as*

$$\check{Y} = \left(I + \frac{1-\lambda}{\lambda \mathbf{N}^\top \mathbf{1}} L(\mathbf{1}\mathbf{N}^\top) \right)^{-1} \bar{Y},$$

where $\mathbf{N}$ *is a* T *by* 1 *vector with* N_t *as its* t*th entry since in Proposition 4 we can choose* $\gamma = \frac{1}{\lambda}$ *and* $\alpha = \frac{1-\lambda}{\mathbf{N}^T \mathbf{1}} \mathbf{N}$*, which matches (20) with* $\Gamma = \frac{1-\lambda}{\lambda \mathbf{N}^\top \mathbf{1}} I$ *and* $A = \mathbf{1}\mathbf{N}^\top$.

Corollary 6 *Estimators which regularize the single task estimate towards the average of means such that they can be written*

$$\check{Y}_t = \lambda \bar{Y}_t + \frac{1-\lambda}{T} \sum_{t=1}^{T} \bar{Y}_t,$$

for $\lambda \in (0, 1]$, can also be written in MTA form as

$$\breve{Y} = \left(I + \frac{1-\lambda}{\lambda T} L(\mathbf{1}\mathbf{1}^\top) \right)^{-1} \bar{Y},$$

since in Proposition 4 we can choose $\gamma = \frac{1}{\lambda}$ and $\alpha = \frac{1-\lambda}{T}\mathbf{1}$, which matches (20) with $\Gamma = \frac{1-\lambda}{\lambda T} I$ and $A = \mathbf{1}\mathbf{1}^\top$.

Note that the proof of the proposition in the appendix uses MTA form with *asymmetric* similarity matrix A. The MTA form with asymmetric A arises if you replace the symmetric MTA regularization term in (4) with the following asymmetric regularization term as follows:

$$\frac{1}{2}\sum_{r=1}^{T}\sum_{s=1}^{T} A_{rs}(\tilde{Y}_r - \tilde{Y}_s)^2 + \frac{1}{2}\sum_{r=1}^{T}\left(\sum_{s=1}^{T} A_{rs}\right)\tilde{Y}_r^2 - \frac{1}{2}\sum_{r=1}^{T}\left(\sum_{s=1}^{T} A_{sr}\right)\tilde{Y}_r^2.$$

Lastly, we make a note about the sum of the mean estimates for the different estimators of Figure 3. In general, the sum of the estimates $\hat{Y} = W\bar{Y}$ for right-stochastic W may differ from the sum of the sample means, because $\mathbf{1}^\top W\bar{Y} \neq \mathbf{1}^\top\bar{Y}$ for all right-stochastic W. But in the special case of Bock's positive-part James-Stein estimator the sum is preserved:

Proposition 7

$$\mathbf{1}^\top\hat{Y}^{JS} = \mathbf{1}^\top\bar{Y}, \tag{21}$$

where $\hat{Y}^{JS}$ is given in (7).

We illustrate this property in the Kings' reigns experiments in Table 7.

4.9 Bayesian Interpretation of MTA

The MTA estimates from (4) can be interpreted as jointly maximizing the likelihood of T Gaussian distributions with a joint Gaussian Markov random field (GMRF) prior (Rue and Held, 2005) on the solution. In MTA, the precision matrix (the inverse covariance of the GMRF prior) is L, the graph Laplacian of the similarity matrix, and is thus positive semi-definite (and not strictly positive definite); GMRFs with PSD inverse covariances are called intrinsic GMRFs (IGMRFs).

GMRFs and IGMRFs are commonly used in graphical models, wherein the sparsity structure of the precision matrix (which corresponds to conditional independence between variables) is exploited for computational tractability. Because MTA allows for arbitrary but non-negative similarities between any two tasks, the precision matrix does not (in general) have zeros on the off-diagonal, and it is not obvious how additional sparsity structure of L would be of help computationally.

Additionally, none of the results we show in this paper require a Gaussian assumption nor any other assumption about the parametric form of the underlying distribution.

5. Simulations

As we have shown in the previous section, MTA is a theoretically rich formulation. In the next two sections we test the usefulness of MTA Constant and MTA Minimax given data,

first with simulations, then with real data. In these sections we use lower-case notation to indicate that we are dealing with actual data as opposed to random variables.

In this section, we test estimators using simulations so that comparisons to ground truth can be made. The simulated data was generated from either a Gaussian or uniform hierarchical process with many sources of randomness (detailed below), in an attempt to imitate the uncertainty of real applications, and thereby determine if these are good general-purpose estimators. The reported results demonstrate that MTA works well averaged over many different draws of means, variances, and numbers of samples.

Simulations are run for $T = \{2, 5, 25, 500\}$ tasks, and parameters were set so that the variances of the distribution of the true means are the same in both uniform and Gaussian simulations. Simulation results are reported in Figures 4 and 5 for the Gaussian experiments, and Figures 6 and 7 for the uniform experiments. The Gaussian simulations were run as follows:

1. Fix σ_μ^2, the variance of the distribution from which $\{\mu_t\}$ are drawn.
2. For $t = 1, \ldots, T$:
   (a) Draw the mean of the tth distribution μ_t from a Gaussian with mean 0 and variance σ_μ^2.
   (b) Draw the variance of the tth distribution $\sigma_t^2 \sim \text{Gamma}(0.9, 1.0) + 0.1$, where the 0.1 is added to ensure that variance is never zero.
   (c) Draw the number of samples to be drawn from the tth distribution N_t from an integer uniform distribution in the range of 2 to 100.
   (d) Draw N_t samples $Y_{ti} \sim \mathcal{N}(\mu_t, \sigma_t^2)$.

The uniform simulations were run as follows:

1. Fix σ_μ^2, the variance of the distribution from which $\{\mu_t\}$ are drawn.
2. For $t = 1, \ldots, T$:
   (a) Draw the mean of the tth distribution μ_t from a uniform distribution with mean 0 and variance σ_μ^2.
   (b) Draw the variance of the tth distribution $\sigma_t^2 \sim U(0.1, 2.0)$.
   (c) Draw the number of samples to be drawn from the tth distribution N_t from an integer uniform distribution in the range of 2 to 100.
   (d) Draw N_t samples $Y_{ti} \sim U[\mu_t - \sqrt{3\sigma_t^2}, \mu_t + \sqrt{3\sigma_t^2}]$.

We compared MTA Constant and MTA Minimax to single-task sample averages and to Bock's James-Stein estimator (Bock, 1975) given in (7), with a slight adaptation for better performance. The term $\frac{\mathbf{tr}(\Sigma)}{\lambda_{\max}}$ in (7) is called the *effective dimension* of the estimator. In simulations where we set Σ to be the true covariance matrix and then estimated the effective dimension by estimating the maximum eigenvalue and trace of the sample mean covariance matrix, we found that replacing the effective dimension with the number of tasks T (when Σ is diagonal) resulted in a significant performance boost for Bock's estimator,

due to the high variance of the estimated maximum eigenvalue in the denominator of the effective dimension. Preliminary experiments with real data also showed a performance advantage to using T rather than the effective dimension. Consequently, to present James-Stein estimation in its best light, for all of the experiments in this paper, the James-Stein comparison refers to (7) using T instead of the effective dimension.

James-Stein, MTA Constant and MTA Minimax all self-estimate the amount of regularization to use (for MTA Constant and MTA Minimax the parameter $\gamma = 1$). So we also compared to a 50-50 randomized cross-validated (CV) version of each. For the cross-validated versions, we randomly subsampled $N_t/2$ samples and chose the value of γ for MTA Constant/Minimax or λ for James-Stein that resulted in the lowest average left-out risk compared to the sample mean estimated with *all* N_t samples. In the optimal versions of MTA Constant/Minimax γ was set to 1, as this was the case during derivation. Note that the James-Stein formulation with a cross-validated regularization parameter λ is simply a convex regularization towards the average of the sample means:

$$\lambda \bar{y}_t + (1 - \lambda)\bar{\bar{y}}.$$

We used the following parameters for CV: $\gamma \in \{2^{-5}, 2^{-4}, \ldots, 2^5\}$ for the MTA estimators and for cross-validated James-Stein a comparable set of λ spanning $(0, 1)$ by the transformation $\lambda = \frac{\gamma}{\gamma+1}$. Even when cross-validating the regularization parameter for MTA, an advantage of using the proposed MTA Constant or MTA Minimax is that these estimators provide a data-adaptive scale for γ, where $\gamma = 1$ sets the regularization parameter to be $\frac{a^*}{T}$ or $\frac{1}{T(b_u - b_l)^2}$, respectively.

Some observations from Figures 4-7:

- Further to the right on the x-axis the means are more likely to be further apart, and multi-task approaches help less on average compared to the single-task sample means.

- For $T = 2$, the James-Stein estimator reduces to the single-task estimator. The MTA estimators provide a gain while the means are close with high probability (that is, when $\sigma_\mu^2 < 1$) but deteriorate quickly thereafter.

- For $T = 5$, MTA Constant dominates in the Gaussian case, but in the uniform case does worse than single-task when the means are far apart. For all $T > 2$, MTA Minimax almost always outperforms James-Stein and always outperforms single-task, which is to be expected as it was designed conservatively.

- The $T = 25$ and $T = 500$ cases illustrate that all estimators benefit from an increase in the number of tasks. The difference between $T = 25$ performance and $T = 500$ performance is minor, indicating that benefit from jointly estimating a larger number of tasks together levels off early on.

- For MTA Constant, cross-validation is always worse than the estimated optimal regularization, while the opposite is true for MTA Minimax. This is to be expected, as minimax estimators are not designed to minimizes the average risk, but average risk is the metric optimized during cross-validation and is the metric reported.

- Cross-validating MTA Constant or MTA Minimax should result in similar performance, and this can be seen in the figures where the green and blue dotted lines are superimposed. The performance differs slightly because the discrete set of γ choices multiply different a's for the MTA Constant and MTA Minimax.

In summary, when the tasks are close to each other compared to their variances, MTA Constant is the best estimator to use by a wide margin. When the tasks are farther apart, MTA Minimax provides a win over both James-Stein and sample averages.

5.1 Oracle Performance

To illustrate the best performance we know is possible to achieve with MTA, Figure 8 shows the effect of using the true "oracle" means and variances for the calculation of optimal pairwise similarities for $T > 2$:

$$A_{rs}^{\text{orcl}} = \frac{2}{(\mu_r - \mu_s)^2}. \tag{22}$$

This matrix is the best *pairwise* oracle, but does not generally minimize the risk over all possible A for $T > 2$. However, comparing to it illustrates how well the MTA formulation can do, without the added error due to estimating A from the data.[3]:

Figure 8 reproduces the results from the $T = 5$ Gaussian simulation (excluding cross-validation results), and compares to the performance of oracle pairwise MTA using (22). Oracle MTA is over 30% better than MTA Constant, indicating that practical estimates of the similarity are highly suboptimal compared to the best possible MTA performance, and motivating better estimates of A as a direction for future research.

6. Real Data Experiments

We present four real data experiments,[4] comparing eight estimators on both goals (2) and (3). The first experiment estimates future customer reviews based on past customer reviews. The second experiment estimates final grades based on homework grades. The third experiment forecasts a customer's future order size based on the size of their past orders. The fourth experiment takes a more in-depth look at the estimates produced by these methods for the historical problem of estimating the length of a king's reign.

6.1 Metrics

For all the experiments except estimating final grades, we only have sample data, and so we compare the estimators using a metric that is an empirical approximation to the regression error defined in (3). First, we replace the expectation in (3) with a sum over the samples. Second, we measure the squared error between a sample y_{ti} and an estimator formed without

3. Preliminary experiments (not reported) showed that for $T > 2$ estimating A pairwise as $\hat{A}_{rs} = \frac{2}{(\bar{y}_r - \bar{y}_s)^2}$ was almost always worse than constant MTA.
4. Research-grade Matlab code and the data used in these experiments can be found at `http://mayagupta.org/publications.html`.

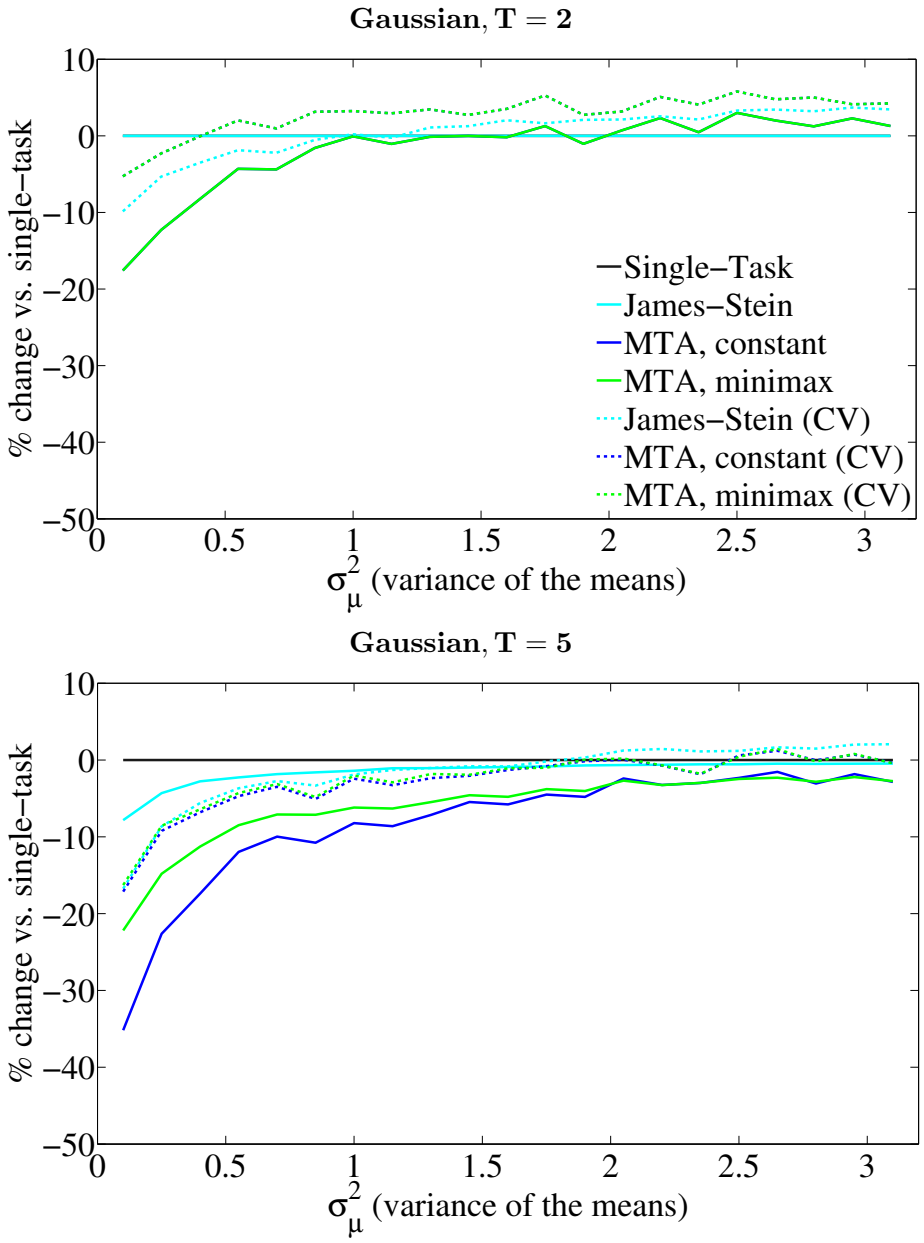


Figure 4: Gaussian experiment results for $T = \{2, 5\}$. The y-axis is average (over 10000 random draws) percent change in risk vs. single-task, such that -50% means the estimator has half the risk of single-task. Note: for $T = 2$ the James-Stein estimator reduces to single-task, and so the cyan and black lines overlap. Similarly, for $T = 2$, MTA Constant and MTA Minimax are identical, and so the blue and green lines overlap.

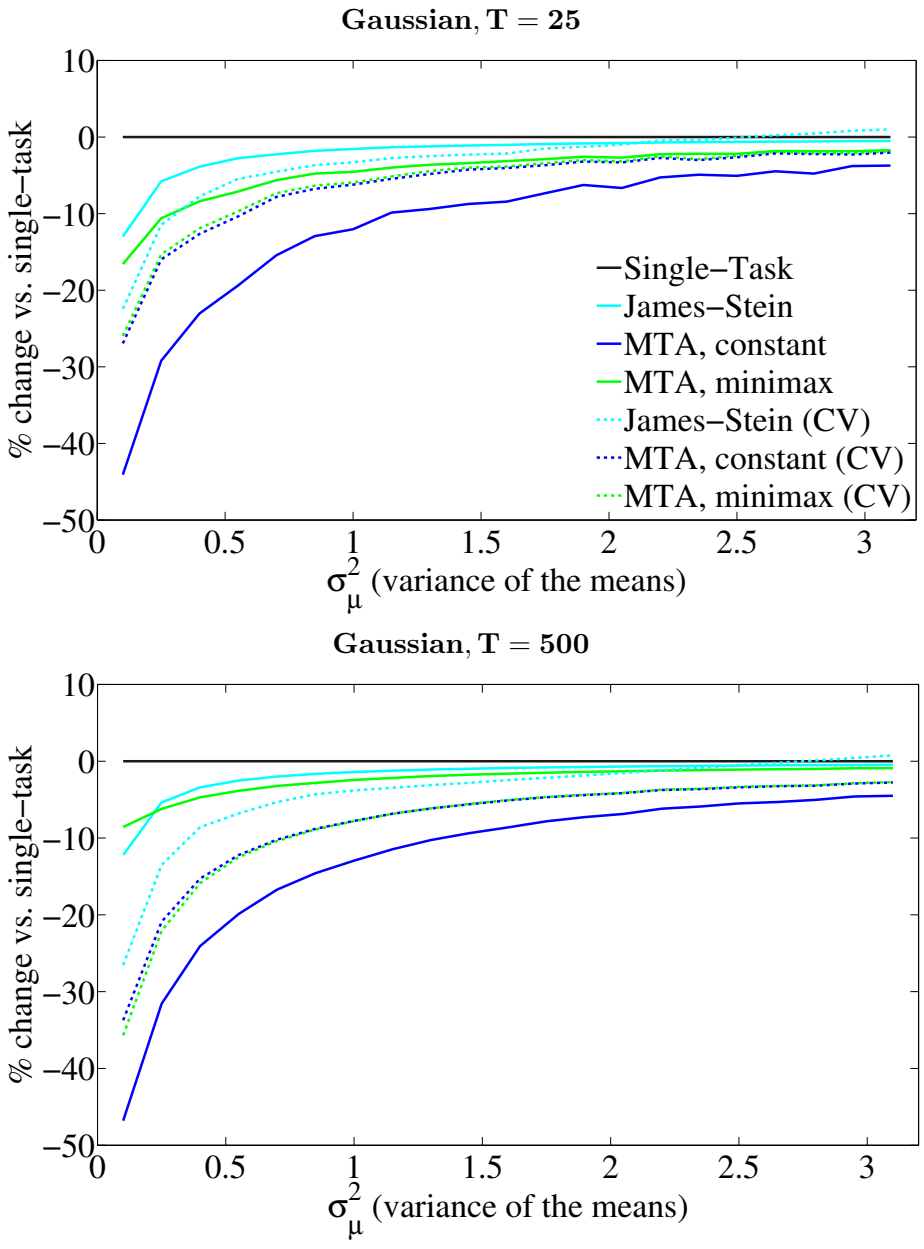


Figure 5: Gaussian experiment results for $T = \{25, 500\}$. The y-axis is average (over 10000 random draws) percent change in risk vs. single-task, such that -50% means the estimator has half the risk of single-task.

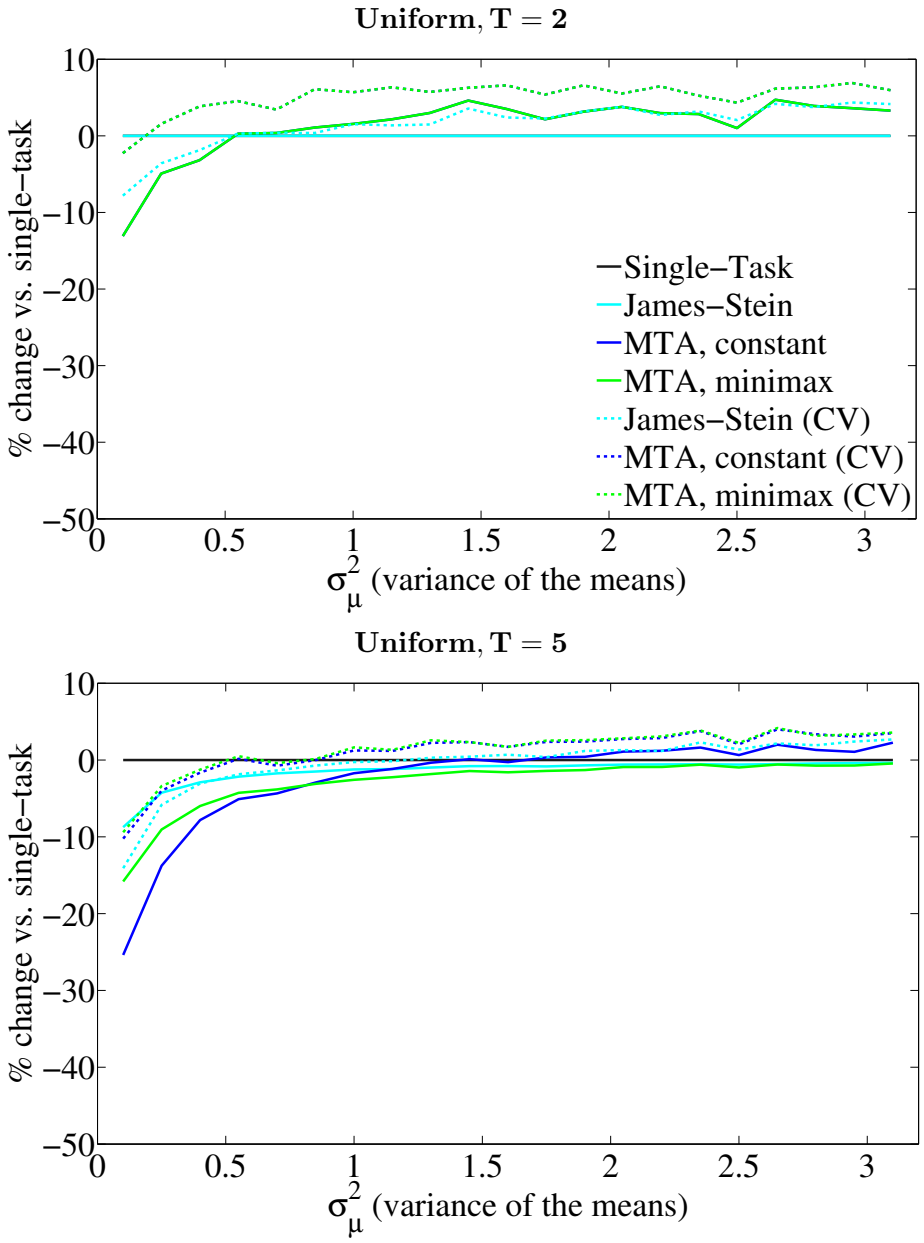


Figure 6: Uniform experiment results for $T = \{2, 5\}$. The y-axis is average (over 10000 random draws) percent change in risk vs. single-task, such that -50% means the estimator has half the risk of single-task. Note: for $T = 2$ the James-Stein estimator reduces to single-task, and so the cyan and black lines overlap. Similarly, for $T = 2$, MTA Constant and MTA Minimax are identical, and so the blue and green lines overlap.

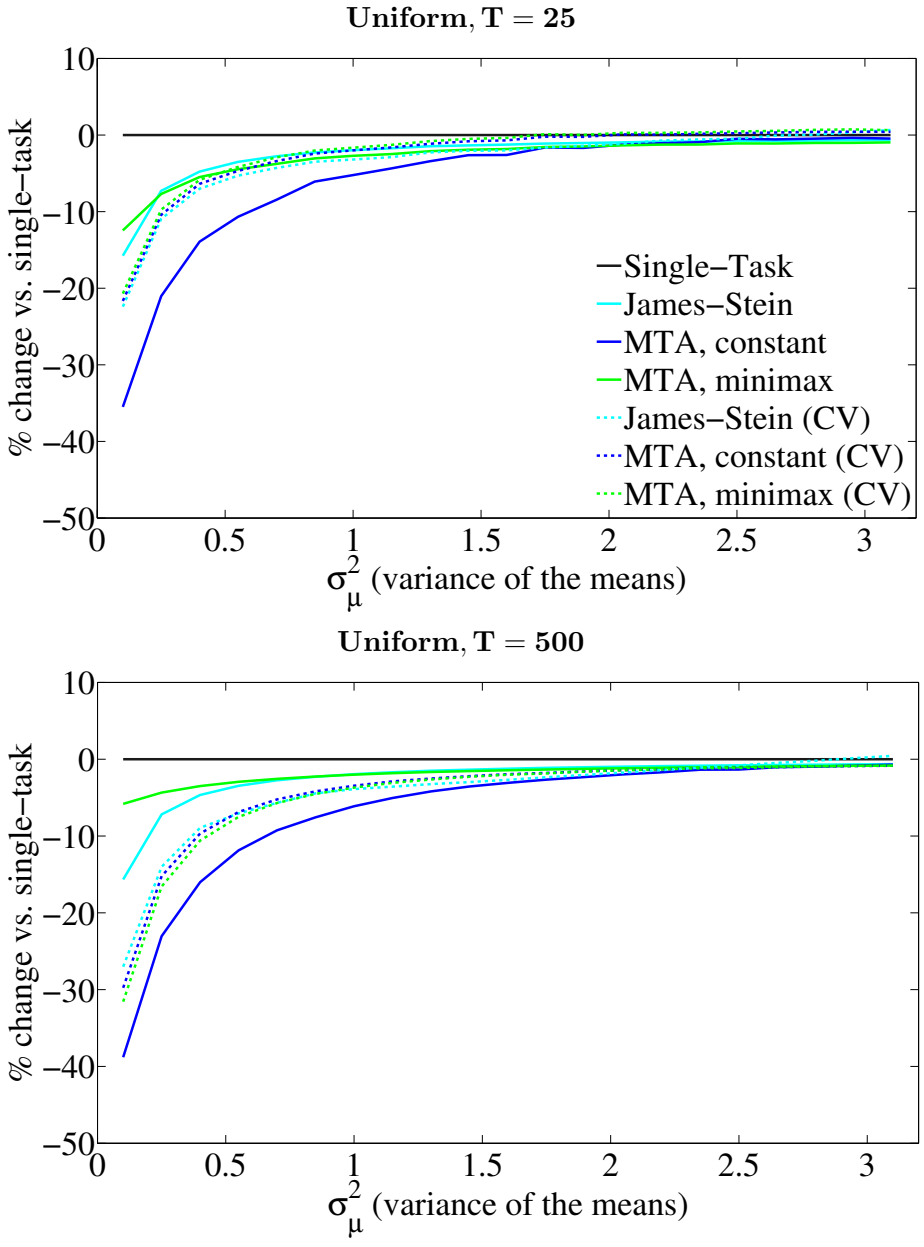


Figure 7: Uniform experiment results for for $T = \{25, 500\}$. The y-axis is average (over 10000 random draws) percent change in risk vs. single-task, such that -50% means the estimator has half the risk of single-task.

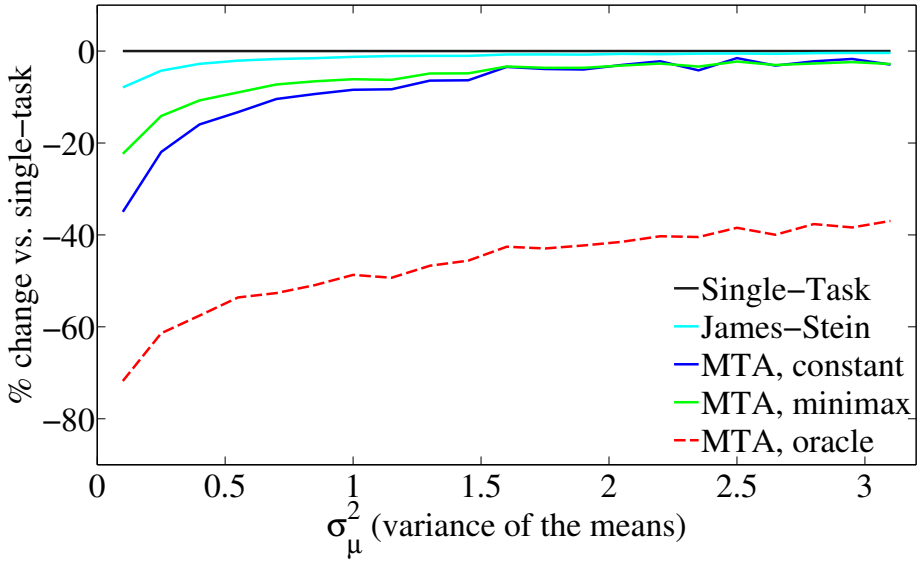


Figure 8: Average (over 10000 random draws) percent change in risk vs. single-task with $T = 5$ for the Gaussian simulation. Oracle MTA uses the true means and variance to specify the weight matrix W.

that sample, $\hat{y}_{t\setminus y_{ti}}$. That is, the empirical risk we measure is:

$$\sum_{t=1}^{T}\left(\frac{1}{N_t}\sum_{i=1}^{N_t}\left[(y_{ti}-\hat{y}_{t\setminus y_{ti}})^2\right]\right). \tag{23}$$

To make the results more comparable across data sets, we present the results as the percent the error given in (23) is reduced compared to the single-task sample mean estimate.

6.2 Experimental Details

For the cross-validation estimators, we cross-validate the regularization parameter from the set $\{2^{-15}, 2^{-14}, \ldots, 2^{14}, 2^{15}\}$. This is a larger range of cross-validation values than used in the simulations, but we found that necessary to achieve good results with cross-validation in the real data experiments. Cross-validation parameters were chosen using double-leave-one-out cross-validation (for each sample left out for test, the remaining N-1 samples undergo leave-one-out cross-validation to optimize (23)). For real-data experiments with more than 50 tasks, to make the double leave-one-out cross-validation fast enough to be feasible, we randomly sub-sampled uniformly and independently for each held-out sample 50 tasks for the estimation of the regularization parameter (but all tasks were used in all cases for the actual estimates).

In addition to James-Stein, MTA, and their variants, we also compare to the completely-regularized baseline, the pooled mean estimator:

$$\hat{y}_t^{\text{pooled}} = \bar{\bar{y}} = \frac{1}{TN} \sum_{s=1}^{T} \sum_{i=1}^{N} y_{si}, \tag{24}$$

which estimates the same value for each task.

For each experiment, a single pooled variance estimate when needed was used for all tasks: $\sigma_t^2 = \sigma^2$, for all t. We found that using a pooled variance estimate improved performance for all the estimators compared.

6.3 Estimating Customer Reviews for Amazon Products

We model amazon.com customer reviews for a product as iid random draws from an unknown distribution. We scraped customer review scores (ranging from 1 to 5) for four different product types from the amazon.com website, as detailed in Table 2. We treat each product as a task, and jointly estimate the mean reviews for all products of the same type. The eight estimators are compared to see how well they predict held-out customer reviews, as per (23); a lower (more negative) score corresponds to greater percent reduction in risk compared to the sample mean estimates.

	# of Products	Mean # of Reviews	Range of # of Reviews
Machine Learning Books	156	7.7	2—80
Blue Suede Shoes	37	16.2	2—143
Espresso Machines	277	47.1	2—1788
Robot Vacuums	59	137.1	3—883

Table 2: Products used in customer reviews experiments, ordered by mean number of reviews (that is, mean sample size).

Table 3 shows the percent risk reduction for each estimator compared to single-task estimates. Some observations:

- MTA Constant (no cross-validation) has the best risk reduction averaged across the products at 11.9% average risk reduction over the single-task estimates, slightly better than the cross-validated forms of MTA.
- The average MTA Constant risk reduction is 34% better than JS (11.9% vs 8.9%), and MTA Constant is better than JS on all the data sets.
- On all data sets, all the joint estimators (not including the pooled mean baseline) do better than the single-task estimates except JS CV on the robot vacuums data set, showing that joint estimation usually helps.
- MTA Minimax consistently provides small gains over single-task, on average reducing risk by 4.0%, with the lowest standard deviation of improvement of 2.1.

- The JS estimator is more sensitive to the quality of the pooled mean estimate than the MTA Constant estimator.
- JS does better on average than its cross-validated counterpart JS CV, and MTA Constant does better on average than its cross-validated counterpart MTA Constant CV.
- The rows in Table 3 are ordered by the average number of reviews (that is, the average number of samples per task). As one would expect from theory, the gains are larger if there are fewer reviews per task.
- Mixing un-related products (the last row of Table 3) still produces substantial gains over single-task estimates.

	Pooled Mean	JS	JS CV	MTA Constant	MTA Constant CV	MTA Minimax	MTA Minimax CV
ML Books	**-24.6**	-23.1	-22.9	**-24.6**	-23.3	-6.5	-23.1
Blue Suede Shoes	-12.4	-11.5	-10.6	**-12.5**	-11.6	-4.8	-11.6
Espresso Machines	2.7	-3.7	-6.3	**-8.4**	-7.8	-3.6	-8.3
Robot Vacuums	8.7	-0.7	7.3	**-2.5**	-2.2	-0.8	-1.8
All Products	-1.9	-5.4	-9.3	**-11.3**	-11.0	-4.3	-10.7
Average	-5.5	-8.9	-8.4	**-11.9**	-11.2	-4.0	-11.1
STD	13.2	8.9	10.8	8.1	7.7	**2.1**	7.7

Table 3: Percent change in risk vs. single-task for customer reviews experiment (lower is better). 'JS' denotes James-Stein, 'CV' denotes cross-validation, and 'STD' denotes standard deviation.

6.4 Estimating Final Grades from Homework Grades

We model homework grades as random samples drawn iid from an unknown distribution where the mean for each student is that student's final class grade. We compare the eight estimators to see how well they predict each student's final grade given only their homework grades. Final class grades are based on the homework, but also on projects, labs, quizzes, exams and sometimes class participation, with the mix varying by class. We collected 22 anonymized data sets from six different instructors at three different universities for undergraduate electrical engineering classes. Further experimental details:

- Each of the 22 data sets is for a different class, and constitutes a single experiment, where each student corresponds to a task.
- We treat the ith homework grade of the tth student as sample y_{ti}.
- For each class and each cross-validation method, cross-validation parameters were chosen independently using leave-one-out cross-validation on the homework grades.

- For each class, the error measurement for estimator $\hat{y}$ is the sum of squared errors across all T students:
$$\sum_{t=1}^{T}(\mu_t - \hat{y}_t)^2,$$
where μ_t is the given tth student's final grade.

Table 4 compares the estimators in terms of the percent change in error compared to the single task estimate $\bar{y}_t$. A lower (more negative) score corresponds to greater percent reduction in risk compared to the single task estimates.

Some observations:

- MTA Constant (no cross-validation) has the best average risk reduction, at 16.6% better on average than the standard single-task estimate. The standard deviation of the win over single task for MTA Constant is 13.7% - also lower than any of the other estimators except MTA Minimax. This shows MTA Constant is consistently providing good error reduction.
- MTA Minimax consistently provides small gains, as designed, with low variance.
- Once again, the higher variance of the James-Stein estimator compared to the others is because of the positive-part aspect of the JS estimator: when the positive-part boundary is triggered, JS reduces to the one-task (average-of-means) estimator, which can have poor performance.
- JS does better on average than its cross-validated counterpart JS CV, and MTA Constant does better on average than its cross-validated counterpart MTA Constant CV.

6.5 Estimating Customer Spending

We collaborated with the wooden jigsaw puzzle company Artifact Puzzles to estimate how much each repeat customer would spend on their next order. We treated each customer as a task; in the time period spanned by the data there are $T = 1355$ unique customers who have each purchased at least twice. We modelled each order by a customer as an iid draw from that customer's unknown spending distribution. The number of orders per customer (that is, samples per task) ranged from 2-23, with a mean of 3.03 orders per customer. The amount spent on a given order had a rather long tail distribution, ranging from \$9-\$2403, with a mean of \$82.16.

Results are shown in Table 5, showing the percentage reduction in (23) compared to the single-task sample means.

Some observations from Table 5:

- MTA Constant performed slightly better than the James-Stein estimator, reducing the empirical risk by 22.4% rather than 21.1%.
- JS does better than its cross-validated counterpart JS CV, and MTA Constant does better than its cross-validated counterpart MTA Constant CV.

Class Size	Pooled Mean	JS	JS CV	MTA Constant	MTA Constant CV	MTA Minimax	MTA Minimax CV
16	26.3	0.7	**-0.0**	0.6	**-0.0**	**-0.0**	**-0.0**
20	71.2	−3.2	**-5.2**	−4.7	−3.4	−1.7	−4.6
25	776.9	−12.2	**-12.3**	−12.2	−12.2	−2.7	−12.1
29	−7.6	−11.6	−31.2	−11.4	**-35.2**	−1.8	−29.6
34	373.6	−4.9	−12.4	−5.0	−12.7	−1.1	**-13.3**
36	**-28.3**	−17.4	−0.0	−16.0	−0.0	−2.8	−0.0
39	42.0	**-5.8**	−0.0	−5.6	−0.0	−0.9	−0.0
44	3.0	−47.6	−64.5	−42.7	−68.0	−7.0	**-69.0**
45	127.6	−3.0	−0.0	**-19.2**	−0.0	−4.6	−0.0
47	**-12.8**	−8.0	−0.0	−7.1	−0.0	−0.7	−0.0
48	**-21.0**	−20.5	−0.0	−18.5	−0.0	−2.5	−0.0
50	63.5	63.5	−0.0	9.3	−0.0	**-4.4**	−0.0
50	3.7	−33.6	−41.5	−29.7	−42.4	−3.2	**-47.4**
57	23.3	**-3.8**	−0.0	−3.6	−0.0	−0.4	−0.0
58	−0.2	**-16.3**	−0.0	−15.6	−0.0	−2.8	−0.0
65	45.0	**-29.4**	−0.0	−26.2	−0.0	−4.2	−0.0
68	−16.9	**-45.5**	−16.5	−39.0	−17.0	−6.1	−19.8
69	−14.7	**-41.0**	−14.7	−39.8	−14.7	−4.5	−14.8
72	34.6	−32.9	−27.3	−29.0	−27.8	−4.0	**-34.8**
73	224.2	−28.1	−41.1	−26.4	−39.6	−2.4	**-41.2**
110	5.7	−14.8	**-25.3**	−13.4	−20.6	−1.2	−22.0
149	**-16.6**	−11.7	−0.0	−10.1	−0.0	−0.8	−0.0
Average	77.4	−14.9	−13.3	**-16.6**	−13.3	−2.7	−14.0
STD	182.0	22.7	18.1	13.7	18.7	**1.9**	19.4

Table 4: Percent change in risk vs. single-task for the grade estimation experiment (lower is better). 'JS' denotes James-Stein, 'CV' denotes cross-validation, and 'STD' denotes standard deviation.

	Pooled Mean	JS	JS CV	MTA Constant	MTA Constant CV	MTA Minimax	MTA Minimax CV
Customer Spending	-10.6	-21.1	-17.6	**-22.4**	-19.7	-0.6	-19.5

Table 5: Percent change in risk vs. single-task for the customer spending experiments (lower is better). 'JS' denotes James-Stein, 'CV' denotes cross-validation.

	Pooled Mean	JS	JS CV	MTA Constant	MTA Constant CV	MTA Minimax	MTA Minimax CV
Kings' Reigns	-8.2	-8.7	-4.7	**-8.9**	-2.9	-3.1	-3.2

Table 6: Percent change in risk vs. single-task for the kings' reigns experiments (lower is better). 'JS' denotes James-Stein, 'CV' denotes cross-validation.

6.6 Estimating the Length of Kings' Reigns

To illustrate the differences between the actual estimates, we re-visit an estimation problem studied by Isaac Newton, "How long does the average king reign?" (Newton, 1728; Stigler, 1999). Newton considered 9 different kingdoms, from the Kings of Judah to more recent French kings. Our data set covers 30 well-known dynasties, listed in Table 7, from ancient to modern times, and spread across the globe. All data was taken from wikipedia.org in August and September 2013 (see the linked data files for the raw data and more historical details).

Results are shown in Table 6, showing the percentage reduction in (23) compared to the single-task sample means. Some observations about these results:

- The pooled mean is 8.2% better than estimating each dynasty's average separately. We found it surprising that pooling across cultures and history forms overall better estimates: the fate of man is apparently the fate of man, regardless of whether it is 1000 BC in Babylon or 19th century Denmark.

- The JS and MTA Constant estimators achieve a slightly bigger reduction in squared error compared to the pooled mean.

- The MTA Constant estimator is very slightly better than the JS estimator, -8.9% vs -8.7%.

- The JS and MTA estimators do better than their cross-validated counterparts.

We also give the actual estimators of the average length of the reign for each kingdom in Table 7. Some observations from Table 7:

Dynasty, # Kings	Avg.	Pooled Mean	JS	JS CV	MTA Const.	MTA Const. CV	MTA MM	MTA MM CV
Larsa, 15	17.7	19.5	19.2	18.5	18.3	18.1	17.8	18.1
Amorite, 11	26.9	19.5	22.3	24.6	24.6	25.5	26.5	25.6
Assyrian, 27	17.3	19.5	19.1	18.2	17.8	17.6	17.4	17.6
Israel, 21	13.4	19.5	17.7	15.6	14.8	14.2	13.6	14.1
Judah, 22	21.5	19.5	20.5	21.0	21.2	21.3	21.5	21.4
Achaemenid, 9	24.3	19.5	21.4	22.9	22.4	23.1	23.9	23.2
Khmer, 33	20.0	19.5	20.0	20.0	20.0	20.0	20.0	20.0
Song, 18	17.7	19.5	19.2	18.5	18.3	18.0	17.8	18.0
Mongol, 4	10.8	19.5	16.8	13.8	16.1	14.5	12.0	14.3
Ming, 17	16.3	19.5	18.7	17.5	17.2	16.8	16.4	16.8
Qing, 12	24.6	19.5	21.6	23.0	23.1	23.7	24.4	23.8
Mamluk, 10	10.1	19.5	16.6	13.4	13.6	12.3	10.7	12.1
Ottoman, 36	17.0	19.5	19.0	18.0	17.4	17.2	17.1	17.2
Normandy, 3	23.0	19.5	21.0	22.0	21.1	21.6	22.5	21.7
Plantagenet, 8	30.8	19.5	23.7	27.2	26.4	28.0	30.0	28.2
Lancaster, 3	20.3	19.5	20.1	20.2	20.1	20.2	20.3	20.2
York, 3	8.0	19.5	15.9	12.0	15.8	13.8	10.1	13.4
Tudor, 5	23.4	19.5	21.1	22.3	21.6	22.2	23.0	22.3
Stuart, 6	16.8	19.5	18.9	17.9	18.4	17.8	17.1	17.8
Hanover, 6	31.0	19.5	23.7	27.3	25.7	27.5	30.0	27.8
Windsor, 3	14.0	19.5	17.9	16.0	17.9	16.9	15.0	16.7
Capet, 15	22.7	19.5	20.9	21.8	21.9	22.3	22.6	22.3
Valois, 7	24.3	19.5	21.4	22.9	22.4	23.1	23.9	23.2
Habsburg, 5	34.4	19.5	24.9	29.6	26.8	29.3	32.8	29.6
Bourbon, 10	21.8	19.5	20.6	21.2	21.2	21.4	21.7	21.4
Oldenburg, 16	25.8	19.5	22.0	23.9	24.3	25.0	25.6	25.0
Mughal, 20	15.7	19.5	18.5	17.1	16.6	16.2	15.8	16.2
Edo, 15	18.6	19.5	19.5	19.1	19.0	18.8	18.7	18.8
Kamehameha, 5	15.4	19.5	18.4	16.9	17.8	17.1	15.9	16.9
Zulu, 4	15.8	19.5	18.5	17.2	18.2	17.5	16.3	17.4
Average Over Dynasties	19.98	19.49	19.98	19.98	20.00	20.03	20.01	20.04

Table 7: Sample average and eight other estimators of the expected length of the reign of a king for each dynasty, ordered chronologically. 'JS' denotes James-Stein, 'CV' denotes cross-validation, 'Const.' denotes Constant, and 'MM' denotes Minimax.

- Table 7 shows that while all the estimators regularize the single task mean (given in column 1) to the pooled mean (given in column 2), the actual estimates can differ quite a bit. For example, MTA Constant and MTA Minimax differ by 5 years in their estimates of the average length of reign of a king from the House of York.

- One sees that the JS estimates are regularized harder towards the pooled mean of 19.5 than the MTA Constant estimates. The MTA Minimax estimates are (as expected) least changed from the task means.

- The last row of Table 7 shows the estimates averaged over the different dynasties. Note that the JS and JS CV estimators have the same average across the tasks (dynasties) as the single-task average, as expected from Proposition 7.

- Based on Tables 5 and 7, we estimate the expected length of a king's reign to be the dynasty-averaged MTA Constant estimate of 20.00 years. Newton's wrote his estimate as "eighteen or twenty years" (Newton, 1728), and the analysis of Stigler (1999) of Newton's data shows that the maximum likelihood estimate from his data was a more pessimistic 19.03 years.

7. Conclusions And Open Questions

We conclude with a summary and then some open questions.

7.1 Summary

We proposed a simple additive regularizer to jointly estimate multiple means using a pairwise task similarity matrix A. Our analysis of the $T = 2$ task case establishes that both MTA estimates are better than the individual sample means when the separation between the true means is small relative to the variance of the samples from each distribution. For the two-task case, we provide a formula for the optimal pairwise task similarity matrix A, that is, one can analytically estimate the optimal amount of regularization without the need to cross-validate or tune a regularization hyper-parameter. We generalized that formula to multiple tasks to form the practical and computationally-efficient MTA Constant mean estimator, as well as a more conservative minimax variant. Simulations and four sets of real data experiments show the MTA Constant estimator can substantially reduce errors over the sample means, and generally performs slightly better than James-Stein estimation (which also does not require cross-validation).

One can also cross-validate the amount of regularization in the MTA formula or in the James-Stein formula. Our results show that both cross-validations work well, though in both simulations and real data experiments, MTA Constant performed slightly better or comparable to the cross-validations.

7.2 Open Questions

Averaging is common, and MTA has potentially broad applicability as a subcomponent to the many algorithms that use means as a subroutine, such as k-means clustering, kernel density estimation, or non-local means denoising.

Most multi-task learning formulations contain an explicit or implicit dependence on the pairwise similarity between tasks. For MTA, this is the A matrix. Even when side information about task similarities is available, it may not be in the optimal numerical form. This paper shows good performance with the assumption that A has constant entries, where that constant is the average of pairwise similarities estimated based on the sample means (MTA Constant). However, the oracle performance plots in Section 5 show that the right choice of A can perform much better. Estimating all $T \times T$ parameters of A optimally may be difficult, but we hypothesize that other structured assumptions, such as low rank A, might perform better than our constant approximation. Martínez-Rego and Pontil (2013) have shown some promising results by clustering tasks in a pre-processing stage.

We focused in this paper on estimating scalar means. The extension to vectors is straightforward (see Section 4.2). However, how well the vector extension works in practice, how to best estimate the block diagonal covariance matrix, and whether different regularization norms would be better remain open questions. A further extension is when the samples themselves are distributions, and the task means to be estimated are expected distributions (Frigyik et al., 2008).

We showed in Section 4 that the matrix inverse needed to compute the MTA Constant and MTA Minimax estimators can be done efficiently. Simulations showed that the achievable gains generally go up slowly with the number of tasks T, with $T = 500$ producing an average risk reduction of 40% in the extreme case that the true means for the 500 tasks were the same. In the real data experiment on customer spending, there were $T = 1355$ tasks that produced a risk reduction of 22.4%. Larger-scale experiments and analysis of the effect of large T on the error would be intriguing.

We focused on squared error loss and the graph Laplacian regularizer because they are standard, generally work well, lead to computationally efficient solutions, and are easy to analyze. But re-considering the MTA objective with other loss functions and regularizers might lead to interesting new perspectives and estimates.

Lastly, we hope that some of the analyses and results in this paper inspire further theoretical analysis of other multi-task learning methods.

Acknowledgments

This work was funded by a United States PECASE Award and by the United States Office of Naval Research. We thank Peter Sadowski for helpful discussions.

Appendix A. MTA Closed-form Solution

When all A_{rs} are non-negative, the differentiable MTA objective is convex, and admits closed-form solution. First, we rewrite the objective in (4) using the graph Laplacian matrix

$L = D - (A + A^\top)/2$:

$$\begin{aligned}
&\frac{1}{T}\sum_{t=1}^{T}\frac{1}{\sigma_t^2}\sum_{i=1}^{N_t}(Y_{ti}-\tilde{Y}_t)^2 + \frac{\gamma}{T^2}\sum_{r=1}^{T}\sum_{s=1}^{T}A_{rs}(\tilde{Y}_r-\tilde{Y}_s)^2 \\
&= \frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{\sigma_t^2}\sum_{i=1}^{N_t}Y_{ti}^2 + \frac{N_t}{\sigma_t^2}\tilde{Y}_t^2 - 2\frac{N_t}{\sigma_t^2}\tilde{Y}_t\bar{Y}_t\right) + \frac{\gamma}{T^2}\tilde{Y}^\top L\tilde{Y} \\
&= \frac{1}{T}\left(\sum_{t=1}^{T}\frac{1}{\sigma_t^2}\sum_{i=1}^{N_t}Y_{ti}^2 + \tilde{Y}^\top\Sigma^{-1}\tilde{Y} - 2\tilde{Y}^\top\Sigma^{-1}\bar{Y}\right) + \frac{\gamma}{T^2}\tilde{Y}^\top L\tilde{Y},
\end{aligned}$$

where, Σ is a diagonal matrix with $\Sigma_{tt} = \frac{\sigma_t^2}{N_t}$, and $\tilde{Y}$ and $\bar{Y}$ are column vectors with tth entries $\tilde{Y}_t$ and $\bar{Y}_t$, respectively.

For simplicity of notation, we assume from now on that A is symmetric. If, in practice, an asymmetric A is provided, it can be symmetrized without loss of generality.

Take the partial derivative of the above objective w.r.t. $\tilde{Y}$ and equate to zero,

$$\begin{aligned}
0 &= \frac{1}{T}\left(2\Sigma^{-1}Y^* - 2\Sigma^{-1}\bar{Y}\right) + 2\frac{\gamma}{T^2}LY^* \\
&= Y^* - \bar{Y} + \frac{\gamma}{T}\Sigma LY^* \\
\bar{Y} &= \left(I + \frac{\gamma}{T}\Sigma L\right)Y^*,
\end{aligned} \tag{25}$$

which yields the following optimal closed-form solution:

$$Y^* = \left(I + \frac{\gamma}{T}\Sigma L\right)^{-1}\bar{Y}, \tag{26}$$

as long as the inverse exists, which we will prove next.

Appendix B. Proof of Lemma 1

Assumptions: $\gamma \geq 0$, $0 \leq A_{rs} < \infty$ for all r, s and $0 < \frac{\sigma_t^2}{N_t} < \infty$ for all t.

Lemma 1 *The MTA solution matrix* $W = \left(I + \frac{\gamma}{T}\Sigma L\right)^{-1}$ exists.

Proof Let $B = W^{-1} = I + \frac{\gamma}{T}\Sigma L$. The (t,s)th entry of B is

$$B_{ts} = \begin{cases} 1 + \frac{\gamma\sigma_t^2}{TN_t}\sum_{s\neq t}A_{ts} & \text{if } t = s \\ -\frac{\gamma\sigma_t^2}{TN_t}A_{ts} & \text{if } t \neq s, \end{cases}$$

The Gershgorin disk (Horn and Johnson, 1990) $\mathcal{D}(B_{tt}, R_t)$ is the closed disk in $\mathbb{C}$ with center B_{tt} and radius

$$R_t = \sum_{s\neq t}|B_{ts}| = \frac{\gamma\sigma_t^2}{TN_t}\sum_{s\neq t}A_{ts} = B_{tt} - 1.$$

One knows that $B_{tt} \geq 1$ for non-negative A and when $\frac{\gamma\sigma_t^2}{TN_t} \geq 0$, as assumed prior to the lemma statement. Also, it is clear that $B_{tt} > R_t$ for all t. Therefore, every Gershgorin disk is contained within the positive half-plane of $\mathbb{C}$, and, by the Gershgorin Circle Theorem (Horn and Johnson, 1990), the real part of every eigenvalue of matrix B is positive. Its determinant is therefore positive, and the matrix B is invertible: $W = B^{-1}$. ■

Appendix C. Proof of Proposition 2

Recall the proposition: *As* $N_t \to \infty \forall\ t$, $Y^* \to \mu$.

Proof First note that the (t,t)th diagonal entry of Σ is $\frac{\sigma_t^2}{N_t}$, which approaches 0 as $N_t \to 0$, implying that all entries of $\frac{\gamma}{T}\Sigma L \to 0$ as $N_t \to 0$ as well. Since matrix inversion is a continuous operation, $\left(I + \frac{\gamma}{T}\Sigma L\right)^{-1} \to I$ in the norm.[5] By the law of large numbers one can conclude that Y^* asymptotically approaches the true mean μ.

Note further that the above proof is only valid for diagonal Σ, but can be easily extended for non-diagonal Σ by noting that $\Sigma_{rs} = \frac{\sigma_r\sigma_s}{\sqrt{N_r N_s}}$ also converges to 0 as $N_r, N_s \to 0$. ■

Appendix D. Proof of Theorem 3

Assumptions: $\gamma \geq 0$, $0 \leq A_{rs} < \infty$ for all r, s and $0 < \frac{\sigma_t^2}{N_t} < \infty$ for all t.

We next state and prove two lemmas that will be used to prove Theorem 3.

Lemma 8 *W has all non-negative entries.*

Proof Because the off-diagonal elements of the graph Laplacian are non-positive, $W^{-1} = \left(I + \frac{\gamma}{T}\Sigma L\right)$ is a *Z-matrix*, defined to be a matrix with non-positive off-diagonal entries (Berman and Plemmons, 1979). If W^{-1} is a Z-matrix, then the following two statements are true and equivalent: "the real part of each eigenvalue of W^{-1} is positive" and "W exists and $W \geq 0$ (elementwise)" (Berman and Plemmons, 1979, Chapter 6, Theorem 2.3, G_{20} and N_{38}). It has already been proven in Lemma 1 that the real part of every eigenvalue of W^{-1} is positive. Therefore, W exists and is element-wise non-negative. ■

Lemma 9 *The rows of W sum to 1.*

5. Any matrix norm will do since the dimensionality is fixed, and on finite dimensional vector spaces all norms are equivalent and therefore generate the same topology.

Proof As proved in Lemma 1, W exists. Therefore, one can write:

$$\begin{aligned} W\mathbf{1} =& \mathbf{1} \\ \mathbf{1} =& W^{-1}\mathbf{1} \\ =& \left(I + \frac{\gamma}{T}\Sigma L\right)\mathbf{1} \\ =& I\mathbf{1} + \frac{\gamma}{T}\Sigma L\mathbf{1} \\ =& \mathbf{1} + \frac{\gamma}{T}\Sigma\mathbf{0} \\ =& \mathbf{1}, \end{aligned}$$

where the the third equality is true because the graph Laplacian has rows that sum to zero. The rows of W therefore sum to 1. ■

Theorem 3 *The MTA solution matrix* $W = \left(I + \frac{\gamma}{T}\Sigma L\right)^{-1}$ *is right-stochastic.*

Proof We know that W exists (from Lemma 1), is entry-wise non-negative (from Lemma 8), and has rows that sum to 1 (from Lemma 9). ■

Appendix E. MTA Constant Derivation

For the case when $T > 2$, analytically specifying a general similarity matrix A that minimizes the risk is intractable. To address this limitation for arbitrary T, we constrain the similarity matrix to be the constant matrix $A = a\mathbf{1}\mathbf{1}^\top$, resulting in the following weight matrix:

$$W^{\text{cnst}} = \left(I + \frac{1}{T}\Sigma L(a\mathbf{1}\mathbf{1}^\top)\right)^{-1}. \tag{27}$$

For tractability, we optimize a using $\mathbf{tr}(\Sigma)I$ rather than the full Σ matrix, such that

$$a^* = \arg\min_a R\left(\mu, \left(I + \frac{1}{T}\frac{\mathbf{tr}(\Sigma)}{T}L(a\mathbf{1}\mathbf{1}^\top)\right)^{-1}\bar{Y}\right), \tag{28}$$

and then plug this a^* into (27) to obtain MTA Constant.

Simplify $\left(I + \frac{1}{T}\frac{\mathbf{tr}(\Sigma)}{T}L(a\mathbf{1}\mathbf{1}^\top)\right)^{-1}$ using the Sherman-Morrison formula,

$$\begin{aligned}
\left(I + \frac{1}{T}\frac{\mathbf{tr}(\Sigma)}{T}L(a\mathbf{1}\mathbf{1}^\top)\right)^{-1} &= \left(I + \frac{a}{T}\frac{\mathbf{tr}(\Sigma)}{T}(TI - \mathbf{1}\mathbf{1}^\top)\right)^{-1} \\
&= \left(I + a\frac{\mathbf{tr}(\Sigma)}{T} - \frac{a}{T}\frac{\mathbf{tr}(\Sigma)}{T}\mathbf{1}\mathbf{1}^\top\right)^{-1} \\
&= \frac{1}{1 + a\frac{\mathbf{tr}(\Sigma)}{T}}I + \frac{\frac{1}{1+a\frac{\mathbf{tr}(\Sigma)}{T}}\frac{a}{T}\frac{\mathbf{tr}(\Sigma)}{T}\mathbf{1}\mathbf{1}^\top\frac{1}{1+a\frac{\mathbf{tr}(\Sigma)}{T}}}{1 - \frac{a}{T}\mathbf{1}^\top\frac{1}{1+a\frac{\mathbf{tr}(\Sigma)}{T}}\frac{\mathbf{tr}(\Sigma)}{T}\mathbf{1}} \\
&= \frac{1}{a\frac{\mathbf{tr}(\Sigma)}{T} + 1}I + \frac{\frac{a\frac{\mathbf{tr}(\Sigma)}{T}}{a\frac{\mathbf{tr}(\Sigma)}{T}+1}\frac{1}{T}\mathbf{1}\mathbf{1}^\top\frac{1}{1+a\frac{\mathbf{tr}(\Sigma)}{T}}}{1 - \frac{a\frac{\mathbf{tr}(\Sigma)}{T}}{1+a\frac{\mathbf{tr}(\Sigma)}{T}}} \\
&= \frac{1}{a\frac{\mathbf{tr}(\Sigma)}{T} + 1}I + \frac{a\frac{\mathbf{tr}(\Sigma)}{T}}{a\frac{\mathbf{tr}(\Sigma)}{T} + 1}\frac{1}{T}\mathbf{1}\mathbf{1}^\top \\
&= \frac{1}{a\frac{\mathbf{tr}(\Sigma)}{T} + 1}\left(I + a\frac{\mathbf{tr}(\Sigma)}{T^2}\mathbf{1}\mathbf{1}^\top\right).
\end{aligned}$$

The risk of $Y^* = \frac{1}{a\frac{\mathbf{tr}(\Sigma)}{T}+1}\left(I + a\frac{\mathbf{tr}(\Sigma)}{T^2}\mathbf{1}\mathbf{1}^\top\right)\bar{Y}$ is

$$\begin{aligned}
&R(\mu, Y^*) \\
&= \mathbf{tr}\left(\frac{1}{a\frac{\mathbf{tr}(\Sigma)}{T} + 1}\left(I + a\frac{\mathbf{tr}(\Sigma)}{T^2}\mathbf{1}\mathbf{1}^\top\right)\Sigma I\frac{1}{a\frac{\mathbf{tr}(\Sigma)}{T} + 1}\left(I + a\frac{\mathbf{tr}(\Sigma)}{T^2}\mathbf{1}\mathbf{1}^\top\right)^\top\right) \\
&\quad + \mu^\top\left(\frac{1}{a\frac{\mathbf{tr}(\Sigma)}{T} + 1}\left(I + a\frac{\mathbf{tr}(\Sigma)}{T^2}\mathbf{1}\mathbf{1}^\top\right) - I\right)^\top\left(\frac{1}{a\frac{\mathbf{tr}(\Sigma)}{T} + 1}\left(I + a\frac{\mathbf{tr}(\Sigma)}{T^2}\mathbf{1}\mathbf{1}^\top\right) - I\right)\mu \\
&= \frac{1}{(a\frac{\mathbf{tr}(\Sigma)}{T} + 1)^2}\mathbf{tr}\left(\left(I + a\frac{\mathbf{tr}(\Sigma)}{T^2}\mathbf{1}\mathbf{1}^\top\right)\Sigma\left(I + a\frac{\mathbf{tr}(\Sigma)}{T^2}\mathbf{1}\mathbf{1}^\top\right)\right) \\
&\quad + \mu^\top\left(\frac{-a\frac{\mathbf{tr}(\Sigma)}{T}}{a\frac{\mathbf{tr}(\Sigma)}{T} + 1}I + \frac{a\frac{\mathbf{tr}(\Sigma)}{T}}{a\frac{\mathbf{tr}(\Sigma)}{T} + 1}\frac{1}{T}\mathbf{1}\mathbf{1}^\top\right)^\top\left(\frac{-a\frac{\mathbf{tr}(\Sigma)}{T}}{a\frac{\mathbf{tr}(\Sigma)}{T} + 1}I + \frac{a\frac{\mathbf{tr}(\Sigma)}{T}}{a\frac{\mathbf{tr}(\Sigma)}{T} + 1}\frac{1}{T}\mathbf{1}\mathbf{1}^\top\right)\mu
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{(a\frac{\mathbf{tr}(\Sigma)}{T}+1)^2}\mathbf{tr}\left(\Sigma + 2a\frac{\mathbf{tr}(\Sigma)}{T^2}\mathbf{1}\mathbf{1}^\top\Sigma + a^2\frac{\mathbf{tr}(\Sigma)^2}{T^4}\mathbf{1}\mathbf{1}^\top\Sigma\mathbf{1}\mathbf{1}^\top\right) \\
&\quad + \frac{(a\frac{\mathbf{tr}(\Sigma)}{T})^2}{(a\frac{\mathbf{tr}(\Sigma)}{T}+1)^2}\mu^\top L\left(\frac{1}{T}\mathbf{1}\mathbf{1}^\top\right)^\top L\left(\frac{1}{T}\mathbf{1}\mathbf{1}^\top\right)\mu \\
&= \frac{\frac{\mathbf{tr}(\Sigma)}{T}}{(a\frac{\mathbf{tr}(\Sigma)}{T}+1)^2}\left(T + 2a\frac{\mathbf{tr}(\Sigma)}{T} + \left(a\frac{\mathbf{tr}(\Sigma)}{T}\right)^2\right) \\
&\quad + \frac{(a\frac{\mathbf{tr}(\Sigma)}{T})^2}{(a\frac{\mathbf{tr}(\Sigma)}{T}+1)^2}\mu^\top L\left(\frac{1}{T}\mathbf{1}\mathbf{1}^\top\right)^\top L\left(\frac{1}{T}\mathbf{1}\mathbf{1}^\top\right)\mu.
\end{aligned}
$$

To find the minimum, we take the partial derivative w.r.t. a and set it equal to zero. Noting that

$$
L\left(\frac{1}{T}\mathbf{1}\mathbf{1}^\top\right)^\top L\left(\frac{1}{T}\mathbf{1}\mathbf{1}^\top\right) = L\left(\frac{1}{T}\mathbf{1}\mathbf{1}^\top\right),
$$

and omitting some tedious algebra,

$$
\begin{aligned}
\frac{\partial}{\partial a^*}R(\mu, Y^*) = 0 &= \frac{2\frac{\mathbf{tr}(\Sigma)}{T}(-T+1+a^*\mu^\top L\left(\frac{1}{T}\mathbf{1}\mathbf{1}^\top\right)\mu)}{(a^*\frac{\mathbf{tr}(\Sigma)}{T}+1)^3} \\
\Leftrightarrow a^* &= \frac{T-1}{\mu^\top L\left(\frac{1}{T}\mathbf{1}\mathbf{1}^\top\right)^\top L\left(\frac{1}{T}\mathbf{1}\mathbf{1}^\top\right)^\top\mu} \\
&= \frac{T-1}{\mu^\top L\left(\frac{1}{T}\mathbf{1}\mathbf{1}^\top\right)\mu} \\
&= \frac{2}{\frac{1}{T(T-1)}\sum_{r=1}^T\sum_{s=1}^T(\mu_r-\mu_s)^2}.
\end{aligned}
$$

Appendix F. MTA Minimax Derivation

Recall Lehmann and Casella (1998, Chapter 5, Theorem 1.4):

Theorem *Suppose that π is a distribution on the space of μ such that*

$$
r(\pi, Y_\pi) = \sup_\mu R(\mu, Y_\pi),
$$

where $r(\pi, Y_\pi) = \int R(\mu, Y_\pi)\pi(\mu)d\mu$ is the Bayes risk. Then:

1. *Y_π is minimax.*
2. *If Y_π is the* unique *Bayes solution w.r.t. π (i.e., if it is the only minimizer of the Bayes risk), then it is the unique minimax estimator.*
3. *The prior π is least favorable.*

Corollary *If a Bayes estimator Y_π has constant risk, then it is minimax.*

The first step in finding a minimax solution for the $T = 2$ case is specifying a constraint set for μ over which a least favorable prior (LFP) can be found. We will use the box constraint set, $\mu_t \in [b_l, b_u]^\top$, where $b_l \in \mathbb{R}$ and $b_u \in \mathbb{R}$. It is straightforward to show that the corresponding LFP is

$$p(\mu) = \begin{cases} \frac{1}{2}, & \text{if } \mu = [b_l, b_u]^\top \\ \frac{1}{2}, & \text{if } \mu = [b_u, b_l]^\top \\ 0, & \text{otherwise.} \end{cases}$$

The next step is to *guess* a minimax weight matrix W^M and show that the estimator $Y^M = W^M \bar{Y}$ (i) has constant risk and (ii) is a Bayes solution. According to the corollary, if both (i) and (ii) hold for the guessed W^M, then $W^M \bar{Y}$ is minimax. For the $T = 2$ case, we guess W^M to be

$$W^M = \left(I + \frac{2}{T(b_l - b_u)^2} \Sigma L(\mathbf{1}\mathbf{1}^\top) \right)^{-1},$$

which is just W^{cnst} with $a = \frac{2}{(b_l - b_u)^2}$. This choice of W is not a function of μ and thus we have shown that (i) the Bayes risk w.r.t the LFP is constant for all μ. What remains to be shown is (ii) W^M is indeed the Bayes solution, i.e., it is minimizer of the Bayes risk:

$$\begin{aligned} &\frac{1}{2} \left([b_l \; b_u](W - I)^\top (W - I) \begin{bmatrix} b_l \\ b_u \end{bmatrix} + \mathbf{tr}(W \Sigma W^\top) \right) \\ +&\frac{1}{2} \left([b_u \; b_l](W - I)^\top (W - I) \begin{bmatrix} b_u \\ b_l \end{bmatrix} + \mathbf{tr}(W \Sigma W^\top) \right). \end{aligned} \tag{29}$$

Note that this expression is the sum of two convex risks. We already know that for $T = 2$ the minimizer of the risk

$$[\mu_1 \; \mu_2](W - I)^\top (W - I) \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} + \mathbf{tr}(W \Sigma W^\top)$$

is $W^* = \left(I + \frac{2}{T(\mu_1 - \mu_2)^2} \Sigma L(\mathbf{1}\mathbf{1}^\top) \right)^{-1}$. Thus, the minimizer of either term in (29) is

$$W^M = \left(I + \frac{2}{T(b_u - b_l)^2} \Sigma L(\mathbf{1}\mathbf{1}^\top) \right)^{-1} \tag{30}$$

as was to be shown. One can conclude that W^M is minimax over all estimators of the form $\left(I + \frac{\gamma}{T} \Sigma L \right)^{-1}$ for $T = 2$ for the box constraint set.

Appendix G. Proof of Proposition 4

Recall the proposition: *The set of estimators $W\bar{Y}$ where W is of MTA form as per (20) is strictly larger than the set of estimators that regularize the single-task estimates as follows:*

$$\hat{Y} = \left(\frac{1}{\gamma} I + \mathbf{1}\alpha^\top\right)\bar{Y},$$

where $\sum_{r=1}^{T} \alpha_r = 1 - \frac{1}{\gamma}$, $\gamma \geq 1$, *and* $\alpha_r \geq 0$, $\forall r$.

Proof Using the Sherman-Morrison formula,

$$\begin{aligned}
\left(\frac{1}{\gamma} I + \mathbf{1}\alpha^\top\right)^{-1} &= \gamma I - \frac{\gamma^2 \mathbf{1}\alpha^\top}{1 + \gamma\alpha^\top \mathbf{1}} \\
&= \gamma I - \gamma \mathbf{1}\alpha^\top \\
&= I + (\gamma - 1) I - \gamma \mathbf{1}\alpha^\top \\
&= I + \gamma\left(1 - \frac{1}{\gamma}\right) I - \gamma \mathbf{1}\alpha^\top \\
&= I + \gamma L(\mathbf{1}\alpha^\top),
\end{aligned}$$

which is a matrix of MTA form for $\Gamma = \gamma I$ and $A = \mathbf{1}\alpha^T$. Thus, estimators $\hat{Y}_t$ can be written in MTA form:

$$\hat{Y} = (I + \gamma L(\mathbf{1}\alpha^\top))^{-1}\bar{Y}. \tag{31}$$

The converse clearly does not hold: not all matrices $(I + \Gamma L(A))^{-1}$ can be written as (31). ■

Appendix H. Proof of Proposition 7

Recall the proposition: $\mathbf{1}^\top \hat{Y}^{JS} = \mathbf{1}^\top \bar{Y}$, where $\hat{Y}^{JS}$ is given in (7).

Proof The tth component of $\hat{Y}^{JS}$ can be written:

$$\hat{Y}_t^{JS} = \frac{1}{T}\sum_{r=1}^{T} \bar{Y}_r + c(\bar{Y}_t - \frac{1}{T}\sum_{r=1}^{T} \bar{Y}_r),$$

for some scalar $c \in [0, 1]$ that does not depend on t. Thus,

$$\hat{Y}^{JS} = \frac{1 - c}{T}\left(\sum_{r=1}^{T} \bar{Y}_r\right)\mathbf{1} + c\bar{Y},$$

and the sum of the estimates is:

$$\begin{aligned}
\mathbf{1}^\top \hat{Y}^{JS} &= \mathbf{1}^\top \left(\frac{1-c}{T} \left(\sum_{r=1}^{T} \bar{Y}_r \right) \mathbf{1} + c\bar{Y} \right) \\
&= \frac{1-c}{T} \left(\sum_{r=1}^{T} \bar{Y}_r \right) \mathbf{1}^\top \mathbf{1} + c\mathbf{1}^\top \bar{Y} \\
&= (1-c) \sum_{r=1}^{T} \bar{Y}_r + c \sum_{r=1}^{T} \bar{Y}_r \\
&= \mathbf{1}^\top \bar{Y}.
\end{aligned}$$

■

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, 2009.

A. Argyriou, C. A. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multi-task structure learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, December 2005.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

A. Berman and R. J. Plemmons. *Nonnegative Matrices Mathematical Sciences*. Academic Press, 1979.

M. E. Bock. Minimax estimators of the mean of a multivariate normal distribution. *Annals of Statistics*, 3(1), 1975.

E. V. Bonilla, K. M. A. Chai, and C. K. I. Williams. Multi-task Gaussian process prediction. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2008.

G. Casella. An introduction to empirical Bayes data analysis. *The American Statistician*, pages 83–87, 1985.

P. Chebotarev and E. Shamis. The matrix-forest theorem and measuring relations in small social groups. *Computing Research Repository*, abs/math/0602070, 2006.

B. Efron and C. N. Morris. Limiting the risk of Bayes and empirical Bayes estimators–part II: The empirical Bayes case. *Journal of the American Statistical Association*, 67(337): 130–139, 1972.

B. Efron and C. N. Morris. Stein's paradox in statistics. *Scientific American*, 236(5): 119–127, 1977.

S. Feldman, M. R. Gupta, and B. A. Frigyik. Multi-task averaging. In *Advances Neural Information Processing Systems (NIPS)*, 2012.

F. Fouss, L. Yen, A. Pirotte, and M. Saerens. An experimental investigation of graph kernels on a collaborative recommendation task. In *ICDM*, pages 863–868, 2006.

B. A. Frigyik, S. Srivastava, and M. R. Gupta. Functional Bregman divergence and Bayesian estimation of distributions. *IEEE Trans. Information Theory*, 54(11):5130–5139, 2008.

C. F. Gauss. *Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections*. Little, Brown, 1857.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990. Corrected reprint of the 1985 original.

L. Jacob, F. Bach, and J.-P. Vert. Clustered multi-task learning: A convex formulation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 745–752, 2008.

W. James and C. Stein. Estimation with quadratic loss. *Proc. Fourth Berkeley Symposium Mathematical Statistics and Probability*, pages 361–379, 1961.

T. Kato, H. Kashima, M. Sugiyama, and K. Asai. Multi-task learning via conic programming. *Advances in Neural Information Processing Systems (NIPS)*, pages 737–744, 2008.

A.-M. Legendre. *Nouvelles Méthodes pour la Détermination des Orbites des Comètes*. Appendix, Paris, 1805.

E. L. Lehmann and G. Casella. *Theory of Point Estimation*. Springer, New York, 1998.

D. Martínez-Rego and M. Pontil. Multi-task averaging via task clustering. In *Proc. SIMBAD*, 2013.

C. A. Micchelli and M. Pontil. Kernels for multi–task learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

I. Newton. *Chronology of Ancient Kingdoms Amended*. Kessinger Publishing, England, 1728.

R. L. Plackett. Studies in the history of probability and statistics: VII. The principle of the arithmetic mean. *Biometrika*, 45(1/2):130–135, 1958.

J. P. Romano and A. F. Siegel. *Counterexamples in Probability and Statistics*. Chapman and Hall, Belmont, CA USA, 1986.

H. Rue and L. Held. *Gaussian Markov Random Fields: Theory and Applications*, volume 104 of *Monographs Statistics and Applied Probability*. Chapman & Hall, London, 2005.

M. Saerens, F. Fouss, L. Yen, and P. Dupont. The principal components analysis of a graph, and its relationships to spectral clustering. In *Proc. Eur. Conf. Machine Learning*, pages 371–383. Springer-Verlag, 2004.

D. Salsburg. *The Lady Tasting Tea*. Holt Paperbacks, New York, NY, 2001.

D. Sheldon. Graphical multi-task learning. In *Advances in Neural Information Processing Systems Workshops*, 2008.

J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *Ann. Math. Stat.*, 21:124–127, 1950.

A. J. Smola and I. R. Kondor. Kernels and regularization on graphs. In *Proc. COLT*, 2003.

C. Stein. Inadmissibility of the usual estimator for the mean of a multivariate distribution. *Proc. Third Berkeley Symposium Mathematical Statistics and Probability*, pages 197–206, 1956.

S. M. Stigler. *Statistics on the Table: The History of Statistical Concepts and Methods*. Harvard University Press, Cambridge, MA, 1999.

U. v. Luxburg. A tutorial on spectral clustering. *Computing Research Repository*, abs/0711.0189, 2007.

Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research*, 8:35–63, 2007.

Y. Yajima and T.-F. Kuo. Efficient formulations for 1-SVM and their application to recommendation tasks. *JCP*, 1(3):27–34, 2006.

Y. Zhang and D.-Y. Yeung. A convex formulation for learning task relationships. In *Proc. of the 26th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010.

X. Zhu and A. B. Goldberg. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2009.

X. Zhu and J. Lafferty. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *In Proc. Int. Conf. Machine Learning*, pages 1052–1059. ACM Press, 2005.

Journal of Machine Learning Research 15 (2014) 3663-3692 Submitted 12/13; Revised 7/14; Published 11/14

Multi-Objective Reinforcement Learning using Sets of Pareto Dominating Policies

Kristof Van Moffaert KVMOFFAE@VUB.AC.BE
Ann Nowé ANOWE@VUB.AC.BE
Department of Computer Science
Vrije Universiteit Brussel
Pleinlaan 2, Brussels, Belgium

Editors: Peter Auer, Marcus Hutter, Laurent Orseau

Abstract

Many real-world problems involve the optimization of multiple, possibly conflicting objectives. Multi-objective reinforcement learning (MORL) is a generalization of standard reinforcement learning where the scalar reward signal is extended to multiple feedback signals, in essence, one for each objective. MORL is the process of learning policies that optimize multiple criteria simultaneously. In this paper, we present a novel temporal difference learning algorithm that integrates the Pareto dominance relation into a reinforcement learning approach. This algorithm is a multi-policy algorithm that learns a set of Pareto dominating policies in a single run. We name this algorithm *Pareto Q-learning* and it is applicable in episodic environments with deterministic as well as stochastic transition functions. A crucial aspect of Pareto Q-learning is the updating mechanism that bootstraps sets of Q-vectors. One of our main contributions in this paper is a mechanism that separates the expected immediate reward vector from the set of expected future discounted reward vectors. This decomposition allows us to update the sets and to exploit the learned policies consistently throughout the state space. To balance exploration and exploitation during learning, we also propose three set evaluation mechanisms. These three mechanisms evaluate the sets of vectors to accommodate for standard action selection strategies, such as ϵ-greedy. More precisely, these mechanisms use multi-objective evaluation principles such as the hypervolume measure, the cardinality indicator and the Pareto dominance relation to select the most promising actions. We experimentally validate the algorithm on multiple environments with two and three objectives and we demonstrate that Pareto Q-learning outperforms current state-of-the-art MORL algorithms with respect to the hypervolume of the obtained policies. We note that (1) Pareto Q-learning is able to learn the entire Pareto front under the usual assumption that each state-action pair is sufficiently sampled, while (2) not being biased by the shape of the Pareto front. Furthermore, (3) the set evaluation mechanisms provide indicative measures for local action selection and (4) the learned policies can be retrieved throughout the state and action space.

Keywords: multiple criteria analysis, multi-objective, reinforcement learning, Pareto sets, hypervolume

1. Introduction

Many real-life problems involve dealing with multiple objectives (Coello et al., 2006; Tesauro et al., 2008; Hernandez-del Olmo et al., 2012). For example, in a wireless sensor network the

criteria consist of energy consumption and latency, which are conflicting objectives (Gorce et al., 2010). When the system engineer wants to optimize more than one objective, it is not always clear a priori which objectives might be correlated and how they influence each other. As the objectives are conflicting, there usually exists no single optimal solution. In those cases, we are interested in a set of trade-off solutions that balance the objectives. More precisely, we want to obtain the set of best trade-off solutions, i.e., the set of solutions that Pareto dominate all the other solutions but are mutually incomparable.

There are two main approaches when dealing with multi-objective problems. The simplest way is to use a *scalarization function* (Miettinen and Mäkelä, 2002) which transforms the multi-objective problem into a standard single-objective problem. However, this transformation may not be valid when the scalarization function is non-linear. This approach is called a *single-policy* algorithm, as each run converges to a single solution. In order to find a variety of trade-off solutions, several parameterized scalarization functions are employed and their results are combined. However, the mapping from weight space to objective space is not guaranteed to be isomorphic (Das and Dennis, 1997). This means that it is not obvious how to define the weights in order to get a good coverage of the Pareto front of policies.

Another class of algorithms are *multi-policy* algorithms. In contrast to focusing only on a single solution at a time, a multi-policy algorithm searches for a set of optimal solutions in a single run. Well-known examples of this class are evolutionary multi-objective algorithms, such as SPEA2 (Zitzler et al., 2002) and NSGA-II (Deb et al., 2002), which evolve a population of multi-objective solutions. These evolutionary multi-objective algorithms are amongst the most powerful techniques for solving multi-objective optimization problems.

In our work, we focus on reinforcement learning for multi-objective problems. Reinforcement learning (Sutton and Barto, 1998) is a machine learning technique that involves an agent operating in an environment and receiving a scalar feedback signal for its behavior. By sampling actions and observing the feedback signal, the agent adjusts its estimate of the quality of its actions. So far, multi-objective reinforcement learning (MORL) has particularly been focusing on single-policy algorithms (Gabor et al., 1998; Mannor and Shimkin, 2004; Van Moffaert et al., 2013b), while only a restricted number of multi-policy MORL algorithms have been proposed so far. For instance, Barrett and Narayanan (2008) propose the *Convex Hull Value Iteration* (CHVI) algorithm. From batch data, CHVI extracts and computes every linear combination of the objectives in order to obtain all deterministic optimal policies. As the algorithm relies on linear combinations, only policies on the convex hull, a subset of the Pareto front, are learned. The most computationally expensive operator is the procedure to compute and combine the convex hulls in the convex-hull version of the Bellman equation. Lizotte et al. (2010) reduce the asymptotic space and time complexity of the bootstrapping rule by learning several value functions corresponding to different weight vectors using a piecewise linear spline representation. Wang and Sebag (2013) propose a multi-objective Monte Carlo Tree Search (MO-MCTS) method to learn a set of solutions. The algorithm performs tree traversals by selecting the most promising actions. The upper confidence bounds of these actions are scalarized by applying the *hypervolume* indicator on the combination of their estimates and the set of Pareto optimal policies computed so far. Hence, a scalarized multi-objective value function is constructed that eases the process of selecting an action with vectorial estimates.

In this paper, we propose a novel MORL algorithm, named *Pareto Q-learning* (PQL). To the best of our knowledge, this is the first temporal difference-based multi-policy MORL algorithm that does not use the linear scalarization function. Thus, Pareto Q-learning is not limited to the convex hull, but it can learn the entire Pareto front of deterministic non-stationary policies, if enough exploration is provided. In contrast to single-policy approaches that only add a scalarization layer on top of single-objective algorithms, we extend the core principles of the learning algorithm to learn a set of non-dominated policies. Our PQL algorithm is particularly suited for on-line use, in other words, when the sampling cost of selecting appropriate actions is important and the performance should gradually increase over time. We also propose three evaluation mechanisms for the sets that provide a basis for on-line action selection strategies. These evaluation mechanisms use multi-objective indicators such as the hypervolume metric, the cardinality indicator and the Pareto dominance relation in order to select the best possible actions throughout the learning process based on the contents of the sets. The Pareto Q-learning algorithm is evaluated on multiple environments with two and three objectives and its performance is compared w.r.t. several single-policy MORL algorithms that use either the linear or Chebyshev scalarization function or the hypervolume indicator.

In Section 2, we introduce notations and concepts of reinforcement learning and current advances in multi-objective reinforcement learning. In Section 3, we present our novel Pareto Q-learning algorithm and discuss its design specifications. Subsequently, in Section 4, we conduct an empirical comparison of our algorithm to other state-of-the-art MORL algorithms. Finally, in Section 5, we draw our conclusions.

2. Background

In this section, we present related work and background concepts such as reinforcement learning and multi-objective reinforcement learning.

2.1 Reinforcement Learning

A reinforcement learning (Sutton and Barto, 1998) environment is typically formalized by means of a Markov decision process (MDP). An MDP can be described as follows. Let $S = \{s_1, \ldots, s_N\}$ be the state space and $A = \{a_1, \ldots, a_r\}$ the action set available to the learning agent. Each combination of current state s, action choice $a \in A$ and next state s' has an associated transition probability $T(s'|s,a)$ and expected immediate reward $R(s,a)$. The goal is to learn a deterministic stationary policy π, which maps each state to an action, such that the value function of a state s, i.e., its expected return received from time step t and onwards, is maximized. The state-dependent value function of a policy π in a state s is then

$$V^{\pi}(s) = E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} \mid s_t = s \right\}, \tag{1}$$

where $\gamma \in [0,1]$ is the discount factor. The value of taking an action in a state under policy π is represented by a $Q^{\pi}(s,a)$-value which stores the expected return starting from state s,

taking action a, and thereafter following π again. The optimal Q^*-values are defined as

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} T(s'|s,a) \max_{a'} Q^*(s',a'). \tag{2}$$

Watkins introduced an algorithm to iteratively approximate Q^*. In the Q-learning algorithm (Watkins, 1989), a Q-table consisting of state-action pairs is stored. Each entry contains a value for $\hat{Q}(s,a)$ which is the learner's current estimate about the actual value of $Q^*(s,a)$. The $\hat{Q}$-values are updated according to the update rule

$$\hat{Q}(s,a) \leftarrow (1-\alpha_t)\hat{Q}(s,a) + \alpha_t(r + \gamma \max_{a'} \hat{Q}(s',a')), \tag{3}$$

where α_t is the learning rate at time step t and r is the reward received for performing action a in state s. Provided that all state-action pairs are visited infinitely often and a suitable evolution for the learning rate is chosen, the estimates, $\hat{Q}$, will converge to the optimal values, Q^* (Tsitsiklis, 1994).

The Q-learning algorithm is listed below in Algorithm 1. In each episode, actions are selected based on a particular action selection strategy, for example ϵ-greedy where a random action is selected with a probability of ϵ, while the greedy action is selected with a probability of $(1-\epsilon)$. Upon applying the action, the environment transitions to a new state s' and the agent receives the corresponding reward r (line 6). At line 7, the $\hat{Q}$-value of the previous state-action pair (s,a) is updated towards the reward r and the maximum $\hat{Q}$-value of the next state s'. This process is repeated until the $\hat{Q}$-values converge or after a predefined number of episodes.

Algorithm 1 Single-objective Q-learning algorithm

1: Initialize $\hat{Q}(s,a)$ arbitrarily
2: **for** each episode t **do**
3: Initialize s
4: **repeat**
5: Choose a from s using a policy derived from the $\hat{Q}$-values, e.g., ϵ-greedy
6: Take action a and observe $s' \in S$, $r \in \mathbb{R}$
7: $\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha_t(r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a))$
8: $s \leftarrow s'$
9: **until** s is terminal
10: **end for**

2.2 Multi-Objective Reinforcement Learning

In multi-objective optimization, the objective space consists of two or more dimensions (Roijers et al., 2013). Therefore, regular MDPs are generalized to multi-objective MDPs or MOMDPs. MOMDPs are MDPs that provide a vector of rewards instead of a scalar reward, i.e.,

$$\mathbf{R}(s,a) = (R_1(s,a), \dots R_m(s,a)), \tag{4}$$

where m represents the number of objectives. In the case of MORL, the state-dependent value function of a state s is vectorial:

$$\mathbf{V}^{\pi}(s) = E_{\pi}\Big\{ \sum_{k=0}^{\infty} \gamma^{k} \mathbf{r}_{t+k+1} \mid s_t = s \Big\}. \tag{5}$$

Since the environment now consists of multiple objectives, different policies can be optimal w.r.t. different objectives. In MORL different optimality criteria are used. For instance, Gabor et al. (1998) employ a lexicographical ordering of the objectives, while Barrett and Narayanan (2008) define linear preferences on the different objectives. Although, in general, the Pareto dominance relation is used as an optimality criterion in multi-objective optimization.

Definition 1 *A policy π_1 is said to strictly dominate another solution π_2, that is $\pi_2 \prec \pi_1$, if each objective in $\mathbf{V}^{\pi_1}$ is not strictly less than the corresponding objective of $\mathbf{V}^{\pi_2}$ and at least one objective is strictly greater. In the case where $\mathbf{V}^{\pi_1}$ strictly improves $\mathbf{V}^{\pi_2}$ on at least one objective and $\mathbf{V}^{\pi_2}$ also strictly improves $\mathbf{V}^{\pi_1}$ on at least one, the two solutions are said to be incomparable. A policy π is Pareto optimal if $\mathbf{V}^{\pi}$ either strictly dominates or is incomparable with the value functions of the other policies. The set of Pareto optimal policies is referred to as the Pareto front.*

2.2.1 Single-Policy MORL

Most approaches of reinforcement learning on multi-objective tasks rely on single-policy algorithms (Gabor et al., 1998; Mannor and Shimkin, 2004) in order to learn Pareto optimal solutions. Single-policy MORL algorithms employ *scalarization* functions (Vamplew et al., 2008) to define a utility over a vector-valued policy and thereby reducing the dimensionality of the underlying multi-objective environment to a single, scalar dimension:

Definition 2 *A scalarization function f is a function that projects a vector $\mathbf{v}$ to a scalar:*

$$v_{\mathbf{w}} = f(\mathbf{v}, \mathbf{w}), \tag{6}$$

where $\boldsymbol{w}$ is a weight vector parameterizing f.

Recently, a general framework for scalarized single-policy MORL algorithms is proposed (Van Moffaert et al., 2013b). In the framework, scalar $\hat{Q}$-values are extended to $\hat{\mathbf{Q}}$-vectors that store a $\hat{Q}$-value for each objective, i.e.,

$$\hat{\mathbf{Q}}(s,a) = (\hat{Q}_1(s,a), \ldots, \hat{Q}_m(s,a)). \tag{7}$$

When selecting an action in a certain state of the environment, a scalarization function f is applied to the $\hat{\mathbf{Q}}$-vector of each action in order to obtain a single, scalar $\widehat{SQ}(s,a)$ estimate (Algorithm 2, line 4). In the following subsection, we will discuss possible instantiations of the scalarization function *f*. At line 5, we store the $\widehat{SQ}(s,a)$ estimates in a list in order to apply traditional action selection strategies, such as, for example, the ϵ-greedy strategy.

Algorithm 2 Scalarized ϵ-greedy strategy, *scal*-ϵ-greedy()

1: $SQList \leftarrow \{\}$
2: **for** each action $a \in A$ **do**
3: $\quad \mathbf{v} \leftarrow \{\hat{Q}_1(s,a), \ldots, \hat{Q}_m(s,a)\}$
4: $\quad \widehat{SQ}(s,a) \leftarrow f(\mathbf{v}, \mathbf{w})$ ▷ Scalarize $\hat{\mathbf{Q}}$-vectors
5: $\quad$ Append $\widehat{SQ}(s,a)$ to $SQList$
6: **end for**
7: **return** ϵ-greedy($SQList$)

The scalarized multi-objective Q-learning algorithm is presented in Algorithm 3. At line 1, the $\hat{Q}$-values for each triplet of states, actions and objectives are initialized. The agent starts each episode in state s (line 3) and chooses an action based on the multi-objective action selection strategy at line 5, e.g, *scal*-ϵ-greedy. Upon taking action a, the environment transitions the agent into the new state s' and provides the vector of sampled rewards $\mathbf{r}$. As the Q-table has been extended to incorporate a separate value for each objective, these values are updated for each objective individually and the single-objective Q-learning update rule is extended for a multi-objective environment at line 9. More precisely, the $\hat{Q}$-values for each triplet of state s, action a and objective o are updated using the corresponding reward for each objective, $\mathbf{r}$, into the direction of the best scalarized action of the next state s'. It is important to note that this framework only adds a scalarization layer on top of the action selection mechanisms of standard reinforcement learning algorithms.

Algorithm 3 Scalarized multi-objective Q-learning algorithm

1: Initialize $\hat{Q}_o(s,a)$ arbitrarily
2: **for** each episode t **do**
3: $\quad$ Initialize state s
4: $\quad$ **repeat**
5: $\quad\quad$ Choose action a from s using the policy derived from $\widehat{SQ}$-values, e.g., *scal*-ϵ-greedy
6: $\quad\quad$ Take action a and observe state $s' \in S$ and reward vector $\mathbf{r} \in \mathbb{R}^m$
7: $\quad\quad$ $a' \leftarrow greedy(s')$ ▷ Call scal. greedy action selection
8: $\quad\quad$ **for** each objective o **do**
9: $\quad\quad\quad$ $\hat{Q}_o(s,a) \leftarrow \hat{Q}_o(s,a) + \alpha_t(\mathbf{r}_o + \gamma\, \hat{Q}_o(s',a') - \hat{Q}_o(s,a))$
10: $\quad\quad$ **end for**
11:
12: $\quad\quad$ $s \leftarrow s'$ ▷ Proceed to next state
13: $\quad$ **until** s is terminal
14: **end for**

A scalarization function can come in many forms and flavors, but the most common function is the *linear scalarization* function. As depicted in Eq. 8, the linear scalarization function calculates a weighted-sum of the $\hat{\mathbf{Q}}$-vector and a non-negative weight vector

$$\widehat{SQ}_{linear}(s,a) = \sum_{o=1}^{m} \mathbf{w}_o \cdot \hat{\mathbf{Q}}_o(s,a). \tag{8}$$

The weight vector itself should satisfy the equation

$$\sum_{o=1}^{m} \mathbf{w}_o = 1. \tag{9}$$

Given these $\widehat{SQ}$-values, the standard action selection strategies can decide on the appropriate action to select. For example, in the greedy case in Eq. 10, the action with the largest $\widehat{SQ}$-value is selected:

$$greedy_{linear}(s) = \arg\max_{a'} \widehat{SQ}_{linear}(s, a'). \tag{10}$$

Because the linear scalarization function computes a convex combination, it has the fundamental limitation that it can only find policies that lie in convex regions of the Pareto front (Vamplew et al., 2008).

An alternative scalarization function is based on the L_p metrics (Dunford et al., 1988). These metrics measure the distance between a point $\mathbf{x}$ in the multi-objective space and a utopian point $\mathbf{z}^*$. This point $\mathbf{z}^*$ serves as a point of attraction to steer the search process to high-quality solutions. The utopian point $\mathbf{z}^*$ is a parameter that is being constantly adjusted during the learning process by recording the best value so far for each objective o, plus a small negative or positive constant τ, depending whether the problem is to be minimized or maximized, respectively. In our setting, we measure the distance between each objective of $\mathbf{x}$ to $\mathbf{z}^*$ with $1 \leq p \leq \infty$:

$$L_p(\mathbf{x}) = \big(\sum_{o=1}^{m} \mathbf{w}_o |\mathbf{x}_o - \mathbf{z}^*_o|^p \big)^{1/p}. \tag{11}$$

In the case of p $= \infty$, the metric results in the weighted L_∞ or the Chebyshev metric and is of the form

$$L_\infty(\mathbf{x}) = \max_{o=1...m} \mathbf{w}_o |\mathbf{x}_o - \mathbf{z}^*_o|. \tag{12}$$

In the case of single-policy MORL, a $\widehat{SQ}_{L_\infty}$-value is obtained by substituting $\mathbf{x}$ for the $\hat{\mathbf{Q}}$-vector of a state-action pair (s, a):

$$\widehat{SQ}_{L_\infty}(s, a) = \max_{o=1...m} \mathbf{w}_o \cdot |\hat{\mathbf{Q}}_o(s, a) - \mathbf{z}^*_o|. \tag{13}$$

L_p metrics entail that the action corresponding to the *minimal* $\widehat{SQ}_{L_p}$-value is considered the greedy action in state s. Hence, for the Chebyshev metric that is $greedy_{L_\infty}(s)$:

$$greedy_{L_\infty}(s) = \arg\min_{a'} \widehat{SQ}_{L_\infty}(s, a'). \tag{14}$$

Although the Chebyshev metric is a common and established function in evolutionary algorithms, its application in reinforcement learning lacks theoretical guarantees. More precisely, there exist examples which indicate that the Chebyshev metric, being a non-linear function, does not guarantee the scalarized returns to be additive. As a result, the Bellman equation no longer holds and the learning algorithm is not proven to converge to

the optimal policy (Perny and Weng, 2010; Roijers et al., 2013). Nevertheless, even without this theoretical guarantee, a Chebyshev scalarized MORL algorithm can obtain high-quality solutions (Van Moffaert et al., 2013b).

Quality indicators are functions that assign a real value to a set of vectors and are usually employed to evaluate the results of multi-objective algorithms. Yet, particular multi-objective algorithms also use indicators in their internal workings to steer the search process (Beume et al., 2007; Igel et al., 2007). This class of algorithms are called *indicator-based* algorithms. Many quality indicators exist, but the one that is the most interesting for our context is the hypervolume (Zitzler et al., 2003) indicator. The hypervolume measure is a quality indicator that evaluates a particular set of vectorial solutions by calculating the volume with respect to its elements and a reference point (Figure 1). As the goal is to maximize the hypervolume, this reference point is usually defined by determining the lower limit of each objective in the environment.

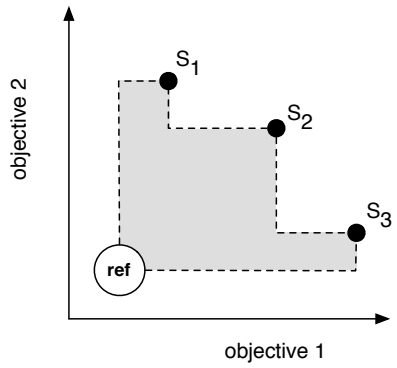


Figure 1: Illustration of the hypervolume calculator. It calculates the area of a set of non-dominated policies, i.e., S_1, S_2 and S_3, in the objective space with a given reference point *ref*.

The hypervolume indicator is of particular interest in multi-objective optimization as it is the only quality measure known to be strictly increasing with regard to Pareto dominance. Recently, the *hypervolume-based* MORL algorithm (HB-MORL) is proposed (Van Moffaert et al., 2013a). HB-MORL is a specific multi-objective algorithm that uses an archive of $\hat{\mathbf{Q}}$-vectors of previously visited states and actions. The innovative part of HB-MORL lies in the action selection mechanism, i.e., the action that maximizes its contribution to the archive in terms of the hypervolume measure is selected.

2.2.2 Multi-Policy MORL

In contrast to single-policy MORL, multi-policy algorithms do not reduce the dimensionality of the objective space but aim to learn a set of optimal solutions at once. White (1982) proposed a dynamic programming (DP) algorithm that computes a set of Pareto dominating policies. Dynamic programming differs from reinforcement learning in the fact that it assumes a model of the environment while reinforcement learning does not need any a

priori knowledge about the environment but is able to work *model-free*. The DP function is

$$\hat{Q}_{set}(s,a) = \mathbf{R}(s,a) \oplus \gamma \sum_{s' \in S} T(s'|s,a)\, V^{ND}(s'), \tag{15}$$

where $\mathbf{R}(s,a)$ is the expected reward vector observed after taking action a in state s and $T(s'|s,a)$ is the corresponding transition probability of reaching state s' from (s,a). We refer to $V^{ND}(s')$ as the set of non-dominated vectors of the $\hat{Q}_{set}$'s of each action in s', as denoted in Eq. 16. The ND operator is a function that removes all Pareto dominated elements of the input set and returns the set of non-dominated elements:

$$V^{ND}(s') = ND(\cup_{a'} \hat{Q}_{set}(s',a')). \tag{16}$$

The $\oplus$ operator performs a vector-sum between a vector $\mathbf{v}$ and a set of vectors V. Summing two vectors can be performed simply by adding the corresponding components of the vectors:

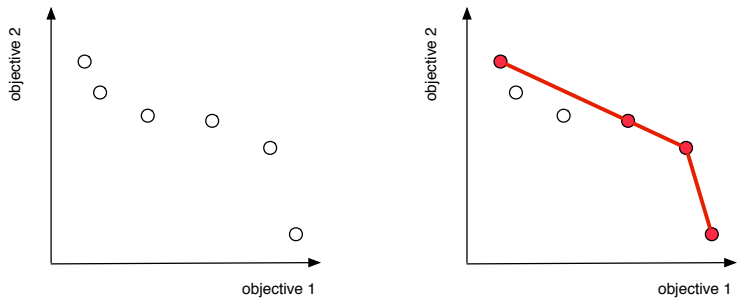
$$\mathbf{v} \oplus V = \bigcup_{\mathbf{v}' \in V} (\mathbf{v} + \mathbf{v}'). \tag{17}$$

The idea is that, after the discounted Pareto dominating rewards are propagated and the $\hat{Q}_{set}$'s converge to a set of Pareto dominating policies, the user can traverse the tree of $\hat{Q}_{set}$'s by applying a preference function. As highlighted in Section 2.1, a deterministic stationary policy suffices for single-objective reinforcement learning. In the case of MORL, White (1982) showed that deterministic non-stationary policies, i.e., policies that do not only condition on the current state but usually also on the time step t, can Pareto dominate the best deterministic stationary policies. As a result, in infinite horizon problems with large values for the discount factor, the number of non-stationary policies increases exponentially and therefore it can lead to an *explosion* of the sets. In order to make the algorithm practically applicable, Wiering and de Jong (2007) proposed the CON-MODP algorithm which solves the problem of non-stationary policies by introducing a consistency operator, but their work is limited to deterministic transition functions.

Several multi-policy algorithms were inspired by the work of White. For instance, Barrett and Narayanan (2008) proposed the *convex hull value-iteration* (CHVI) algorithm which computes the deterministic stationary policies that are on the convex hull of the Pareto front. The convex hull is a set of policies for which the linear combination of the value of policy π, $\mathbf{V}^{\pi}$, and some weight vector $\mathbf{w}$ is maximal (Roijers et al., 2013). In Figure 2 (a), white dots denote the Pareto front of a bi-objective problem and in Figure 2 (b) the red line represents the corresponding convex hull. The 4 deterministic policies denoted by red dots are the ones that CHVI would learn. CHVI bootstraps by calculating the convex hull of the union over all actions in s', that is $\bigcup_{a'} \mathbf{Q}(s',a')$. The most computationally expensive operator is the procedure of combining convex hulls in the bootstrapping rule. Lizotte et al. (2010) reduce the asymptotic space and time complexity of the bootstrapping rule by simultaneously learning several value functions corresponding to different weights and by calculating their piecewise linear spline representation. Recently, they validated their work on clinical trial data for three objectives, although the practical possibilities for higher dimensional spaces are not straightforward (Lizotte et al., 2012).

To conclude, it is important to note that (1) these methods are batch algorithms that assume a model of the environment is known and that (2), in general, only policies that lie on a subset of the Pareto front, i.e., the convex hull, are obtained.

While the aforementioned multi-policy algorithms learn only a finite set of deterministic policies, it might be interesting to employ probabilistic combinations of these policies. Such a stochastic combination of two policies is called a *mixture policy* (Vamplew et al., 2009) and can be explained with the following example. Take for instance a very easy multi-



(a) (b)

Figure 2: (a) A Pareto front of bi-objective policies represented by white dots. (b) The convex hull of the same Pareto front is represented by a red line. The 4 red dots denote policies that CHVI and Lizotte's method would learn.

objective problem where the agent can only follow two deterministic policies π_1 and π_2 with $\mathbf{V}^{\pi_1}(s_0) = (1, 0)$ and $\mathbf{V}^{\pi_2}(s_0) = (0, 1)$, where s_0 denotes the start state. If one would follow policy π_1 with probability p and policy π_2 with probability $(1 - p)$, the average reward vector would be $(p, 1 - p)$. Thus, although there are only two deterministic policies for the original problem, a mixture policy implicates that we can sample the entire convex hull of policies by combining the deterministic policies with a certain probability. Hence, stochastic combinations of the policies of the Pareto front in Figure 2 (a) can represent every solution on the red line in Figure 2 (b).

However, mixture policies might not be appropriate in all situations, as highlighted by Roijers et al. (2013). For instance, in the setting of Lizotte et al. (2010), clinical data is analyzed to propose a treatment to patients based on a trade-off between the effectiveness of the drugs and severity of the side effects. Consider the case where only two policies exist that either maximize the effectiveness and the severity of the side effects and vice versa. While the average performance of the mixture policy of these two basic policies might yield good performance across a number of episodes, the policy itself might be unacceptable in each episode individually, i.e., for each patient independently.

3. Pareto Q-learning

In this section, we will propose a novel on-line TD-based multi-objective learning algorithm, named Pareto Q-learning or PQL, which uses the algorithm of White (1982) as a starting point. As a result, PQL also learns deterministic non-stationary policies. Before we present

the details of our algorithm, we first describe the assumption that we make. We currently only focus on episodic problems, i.e., environments with terminal states that end the episode. In Section 5, we analyze the challenges to extend PQL to ergodic environments.

The Pareto Q-learning algorithm does not assume a given model, i.e., it works *model-free*. Therefore, we present three mechanisms that allow action selection based on the content of the sets of $\hat{\mathbf{Q}}$-vectors. We name them *set evaluation* mechanisms as they provide a scalar evaluation of the sets. These scalar evaluations can then be used to guide the standard exploration strategies, such as ϵ-greedy. The details of the evaluation mechanisms are presented in Section 3.2. First, we elaborate on how we can learn and update sets of vectors in Section 3.1.

3.1 Set-Based Bootstrapping

The single-objective Q-learning bootstrapping rule updates an estimate of an (s, a)-pair based on the reward and an estimate of the next state (Watkins, 1989). The update rule guarantees that the $\hat{Q}$-values converge to their expected future discounted reward, even when the environment has a stochastic transition function. In this section, we analyze the problem of bootstrapping sets of vectors. We first present a naive approach whereupon we present our novel Pareto Q-learning algorithm.

3.1.1 Naive Approach

The set-based bootstrapping problem boils down to the general problem of updating the set of vectors of the current state-action (s, a)-pair with an observed reward vector $\mathbf{r}$ and a set of non-dominated vectors of the next state, $ND(\cup_{a'}\hat{Q}_{set}(s', a'))$ over time. The difficulty in this process arises from the lack of *correspondence* between the vectors in the two sets, i.e., it is not clear which vector of the set of the current (s, a)-pair to update with which vector in s'. This correspondence is needed to perform a pairwise update of each vector in $\hat{Q}_{set}(s, a)$ with the corresponding vector (if any) in the other set (see Figure 3).

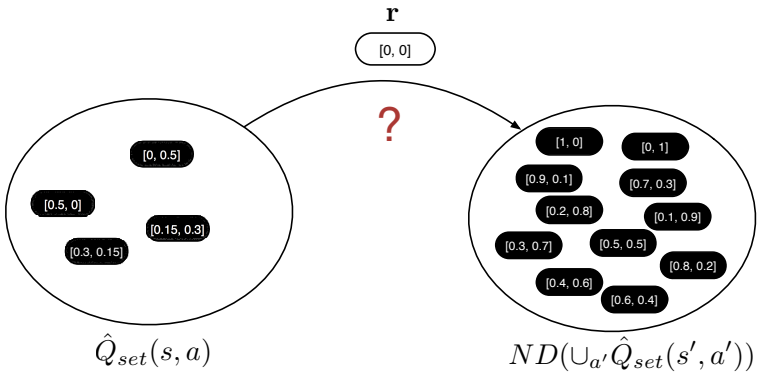


Figure 3: Set-based bootstrapping: the problem of updating over time the set of vectors of the current state-action pair with the observed reward vector and the optimal vectors of the next state. There is no explicit correspondence between the elements in both sets, so as to perform a pairwise update.

A possible solution is to make this correspondence explicit by labelling or tagging the vectors in the two sets. When vectors in the sets of (s, a) and s' are tagged with the same label or color, the bootstrapping process knows that these vectors can be updated in a pairwise manner. More precisely, the process would be as follows: when sampling an action a in s for the first time, the vectors in the set of the next state $ND(\cup_{a'}\hat{Q}_{set}(s', a'))$ are labeled with a unique tag. Next, the bootstrapping process can continue for each vector in s' individually and the tag is copied to the set of (s, a). This process is illustrated in Figure 4 (a) for a bi-objective environment. Subsequently, when action a is sampled in future time steps, we actually have a correspondence between the vectors in the two sets and we can perform a pairwise update for each objective of each vector with the same label (Figure 4 (b)). However, the main problem with this naive solution is that these sets are not *stable* but can change over time. We highlight two main cases that can occur in a temporal difference setting:

- It is possible that the set of (s, a) was updated with vectors from s' at time step t, while actions in s' were sampled at time step $t + 1$, that were previously unexplored. Possibly, new non-dominated vectors then appear in $ND(\cup_{a'}\hat{Q}_{set}(s', a'))$. When, in future episodes, the set of (s, a) is to be updated again, there are elements in s' that were not bootstrapped before and the correspondence between the sets is incomplete (Figure 4 (c)).

- As estimates are being updated over time, it is very likely that vectors in s' that were non-dominated at time step t, become dominated by other vectors at time step $t+1$. In Figure 4 (d), we see that in that case the correspondence no longer holds, i.e., different labels appear in the two sets. As a consequence, learning would have to begin from scratch again for those vectors. Especially in early learning cycles, the vectorial estimates can repeatedly switch between being non-dominated and dominated. Hence, this naive updating process would waste a lot of samples before the vectors mature.

It is clear that such a naive updating procedure would become even more cumbersome and complex in environments with stochastic transitions. As a result, it would not be generally applicable to a wide range of problem domains.

3.1.2 Our Approach: Learning Immediate And Future Reward Separately

In the presentation of our updating principle, we first limit ourselves to environments with deterministic transition functions. We then proceed to highlight the minimal extensions to the algorithm to also cover stochastic transitions.

In standard, single-objective Q-learning (Eq. 3), $\hat{Q}$-values store the sum of the estimated value of the immediate reward and the future discounted reward. Our idea consists of storing this information separately. We use $\overline{\mathfrak{R}}(s, a)$ to denote the average observed immediate reward vector of (s, a) and $ND_t(s, a)$ the set of non-dominated vectors in the next state of s that is reached through action a at time step t. The next state of s is determined by observing the transitions during learning. By storing $\overline{\mathfrak{R}}(s, a)$ and $ND_t(s, a)$ separately, we allow them to converge separately as well. This way, no explicit correspondence between the two sets is required and the current set of non-dominating policies at time step t, $ND_t(s, a)$ is allowed to evolve over time. The $\hat{Q}_{set}$ of (s, a) can be calculated *at run time* by performing

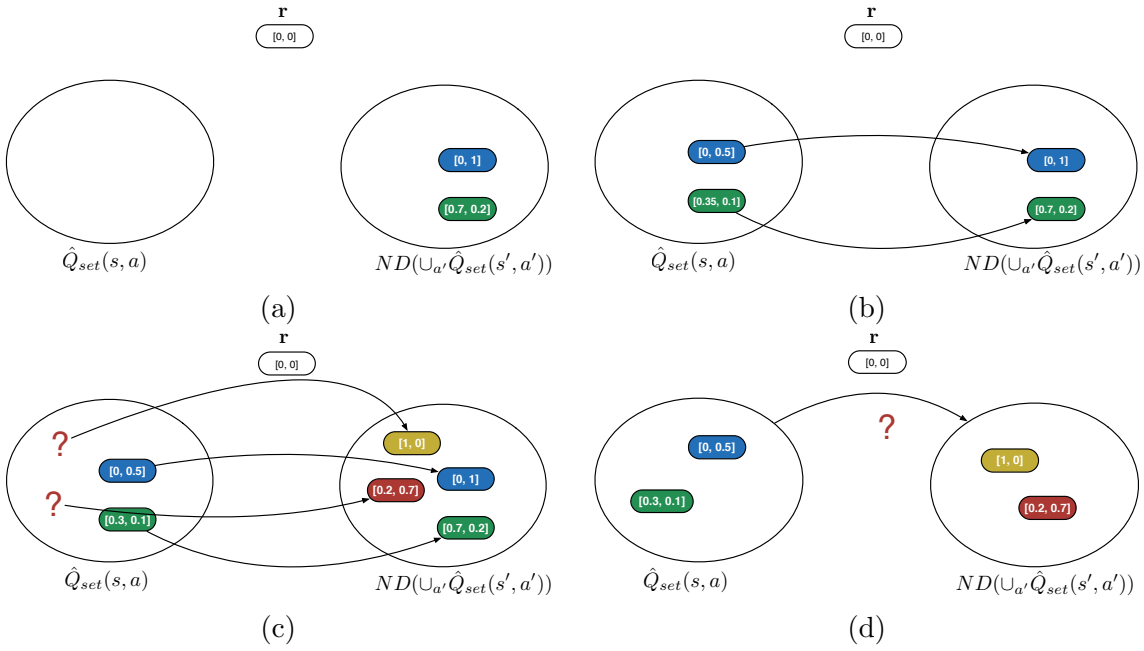


Figure 4: Several situations can occur when updating a set with another set over time. In (a), we would naively label the vectors of s' with a certain color when sampling action a in s for the first time. In (b), we note that the labeled and colored vectors of s' are now bootstrapped and present in (s,a). As the colors are also copied in (s,a), the correspondence between the vectors in (s,a) and s' is explicit and in future time steps the vectors can be updated in a pairwise manner. (c) and (d) highlight the different situations one should account for as the sets are not stable but can change over time. For instance, new vectors can appear in s' (c) or estimates that were non-dominated can become dominated (d). We refer to Section 3.1 for more details.

a vector-sum over the average immediate reward vector and the set of discounted Pareto dominating future rewards:

$$\hat{Q}_{set}(s,a) \leftarrow \overline{\mathfrak{R}}(s,a) \oplus \gamma ND_t(s,a). \tag{18}$$

Whenever the action a in s is selected, the average immediate reward vector $\overline{\mathfrak{R}}(s,a)$ is updated and the $ND_t(s,a)$ list is updated using the non-dominated $\hat{\mathbf{Q}}$-vectors in the $\hat{Q}_{set}$ of every action a' in s', i.e., $ND(\cup_{a'}\hat{Q}_{set}(s',a'))$.

We present an algorithmic outline of the Pareto Q-learning algorithm in Algorithm 4. The algorithm starts by initializing the $\hat{Q}_{set}$'s as empty sets. In each episode, an action is selected using a particular action selection strategy (line 5). How we actually perform the action selection based on the $\hat{Q}_{set}$'s will be presented in the subsequent section. Afterwards, the environment transfers the agent to state s' and provides the reward vector $\mathbf{r}$. In state s', the non-dominated $\hat{\mathbf{Q}}$-vectors for each action are retrieved at line 8 and are discounted. At line 9, the average immediate reward for each objective, $\overline{\mathfrak{R}}(s,a)$, is iteratively updated

given the new reward $\mathbf{r}$ and the number of times that action a was sampled, denoted by $n(s,a)$. The algorithm proceeds until the $\hat{Q}_{set}$'s converge or after a predefined number of episodes.

Algorithm 4 Pareto Q-learning algorithm

Initialize $\hat{Q}_{set}(s,a)$'s as empty sets
for each episode t **do**
  Initialize state s
  repeat
    Choose action a from s using a policy derived from the $\hat{Q}_{set}$'s
    Take action a and observe state $s' \in S$ and reward vector $\mathbf{r} \in \mathbb{R}^m$

    $ND_t(s,a) \leftarrow ND(\cup_{a'} \hat{Q}_{set}(s',a'))$ ▷ Update ND policies of s' in s
    $\overline{\mathfrak{R}}(s,a) \leftarrow \overline{\mathfrak{R}}(s,a) + \frac{\mathbf{r} - \overline{\mathfrak{R}}(s,a)}{n(s,a)}$ ▷ Update average immediate rewards
    $s \leftarrow s'$ ▷ Proceed to next state
  until s is terminal
end for

Although we do not provide a formal proof on the convergence of PQL, its convergence can be argued by the observation that the procedure of repeatedly calculating the set of non-dominated vectors, as was applied in White's algorithm, is guaranteed to converge and the fact that the convergence of the $\overline{\mathfrak{R}}(s,a)$ is trivial.

The updating principle can also be extended to stochastic environments, where the transition probability $T(s'|s,a) \neq 1$ for some next state s', given state s and action a. In the case of stochastic transition functions, we store the expected immediate and future non-dominated rewards per (s,a,s')-tuple that was observed during sampling, i.e., $\overline{\mathfrak{R}}(s,a,s')$ and $ND_t(s,a,s')$, respectively. By also considering the observed frequencies of the occurrence of next state s' per (s,a)-pair, i.e., $F_{s,a}^{s'}$, we estimate $T(s'|s,a)$ for each (s,a). Hence, we learn a small model of the transition probabilities in the environment, similar to Dyna-Q (Sutton and Barto, 1998), which we use to calculate a weighted pairwise combination between the sets. To combine a vector from one set with a vector from the other set, we propose the $\mathcal{C}$-operator, which simply weighs them according to the observed transition frequencies:

$$\mathcal{C}(\hat{\mathbf{Q}}(s,a,s'), \hat{\mathbf{Q}}(s,a,s'')) = \frac{F_{s,a}^{s'}}{\sum_{s''' \in S} F_{s,a}^{s'''}} \hat{\mathbf{Q}}(s,a,s') + \frac{F_{s,a}^{s''}}{\sum_{s''' \in S} F_{s,a}^{s'''}} \hat{\mathbf{Q}}(s,a,s''). \tag{19}$$

3.2 Set Evaluation Mechanisms

In reinforcement learning, the on-line performance is crucial. Therefore, it is interesting to see how the standard exploration strategies, such as ϵ-greedy, can be applied on the $\hat{Q}_{set}$'s during learning. In this section, we propose three evaluation mechanisms that obtain a scalar indication of the quality of a $\hat{Q}_{set}$. These scalar evaluations are used in action selection strategies to balance the exploration and the exploitation. We name these techniques *set evaluation* mechanisms.

3.2.1 Hypervolume Set Evaluation

The first set evaluation mechanism we propose uses the hypervolume measure to evaluate the $\hat{Q}_{set}$'s. The hypervolume indicator is well-suited for two reasons: (1) it is the only quality indicator to be strictly monotonic with the Pareto dominance relation and (2) it provides a scalar measure of the quality of a set of vectors. An outline of the algorithm is given in Algorithm 5. First, we initialize the list where the evaluations of each action of s will be stored. At line 4, we calculate the $\hat{Q}_{set}$ for each action and we compute its hypervolume which we append to the list. The list of evaluations can then be used in an action selection strategy, similar to the single-objective case. For instance, when selecting an action greedily, the action corresponding to the $\hat{Q}_{set}$ with the largest hypervolume is selected. When the $\hat{Q}_{set}$'s are empty, the hypervolume of each action is 0 and an action is selected uniformly at random.[1] This set evaluation mechanism, in combination with Pareto Q-learning, is referred to as *HV-PQL*. A crucial parameter of the hypervolume calculation

Algorithm 5 Hypervolume Q_{set} evaluation

1: Retrieve current state s
2: evaluations = {}
3: **for** each action a **do**
4: $\quad hv_a \leftarrow HV(\hat{Q}_{set}(s,a))$
5: $\quad$ Append hv_a to evaluations ▷ Store hypervolume of the $\hat{Q}_{set}(s,a)$
6: **end for**
7: **return** evaluations

is the reference point. This parameter is problem-specific and should be chosen with great care. A good practice is to define it pessimistically by considering the worst possible value for each objective of every possible policy in the environment.

3.2.2 Cardinality Set Evaluation

An alternative to the previous evaluation mechanism is to consider the number of Pareto dominating $\hat{\mathbf{Q}}$-vectors of the $\hat{Q}_{set}$ of each action. This evaluation mechanism closely relates to the cardinality indicator in multi-objective optimization, hence the abbreviation *C-PQL*.

The rationale behind this evaluation mechanism is that it can heuristically guide the search process by providing a degree of domination one action has over other actions, locally in a state. It is expected that these actions then have a larger probability to lead to global Pareto dominating solutions. Especially when estimates are not yet mature, it might be interesting to bias the action selection to actions with a large number of non-dominated solutions. An outline of the algorithm is given in Algorithm 6. At line 2, we initialize a list where we store the individual $\hat{\mathbf{Q}}$-vectors of the $\hat{Q}_{set}$, together with a reference to its corresponding action a (line 5). At line 8, we remove all dominated $\hat{\mathbf{Q}}$-vectors using the ND operator, such that only the non-dominated $\hat{\mathbf{Q}}$-vectors remain in the $NDQs$ list. Using this list, the underlying action selection strategy can simply count the number of times each action a of s remains in the list of Pareto dominating $\hat{\mathbf{Q}}$-vectors, i.e., the $NDQs$ list, and eventually perform the action selection. Thus, when selecting an action greedily, the action

1. This is also the case for the other set evaluation mechanisms below.

that relates to the largest number of Pareto dominating $\hat{\mathbf{Q}}$-vectors over all actions in s is selected.

Algorithm 6 Cardinality Q_{set} evaluation

```
Retrieve current state s
allQs = {}
for each action a in s do
    for each Q̂ in Q̂_set(s, a) do
        Append [a, Q̂] to allQs          ▷ Store for each Q̂-vector a reference to a
    end for
end for
NDQs ← ND(allQs)                        ▷ Keep only the non-dominating solutions
return NDQs
```

3.2.3 Pareto Set Evaluation

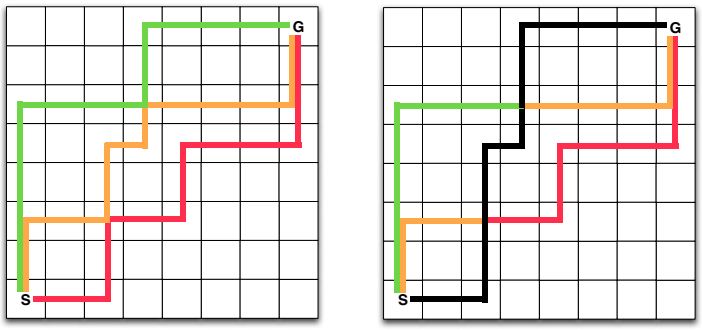
The third evaluation mechanism is a simplified version of the cardinality metric. Instead of considering the number of non-dominated elements in the $\hat{Q}_{set}$ of each action in s, we simply consider if action a has a non-dominated vector across every other action a' or not. The approach eliminates any actions which are dominated, and then randomly selects amongst the non-dominated actions. Hence, this mechanism only relies on the Pareto relation and is therefore called *PO-PQL*. PO-PQL removes the bias that the cardinality indicator in *C-PQL* might have for actions with a large number of non-dominated vectors over actions with just a few. The rationale behind this mechanism is to have a more relaxed evaluation of the actions of a particular state and to treat every non-dominated solution equally.

3.3 Consistently Tracking a Policy

The set evaluation mechanisms in Section 3.2 provide the necessary tools to perform action selection during learning, i.e., balancing the exploration towards uncharted areas of the state space and the exploitation of non-dominated actions. However, at any moment in time, it might be necessary to apply the learned policies. In single-objective reinforcement learning, the learned policy can be easily tracked by applying the arg max-operator over all actions in each state, i.e., applying greedy action selection. In the case of a multi-policy problem, we are learning multiple policies at the same time which requires an adapted definition of a greedy policy in MORL.

Because of the nature of multi-policy setting, one needs to select actions *consistently* in order to retrieve a desired policy based on the $\hat{\mathbf{Q}}$-vectors. If one would select actions based on *local* information about the 'local' Pareto front attainable from each action, then there is no guarantee that the cumulative reward vectors obtained throughout the episode will be *globally* Pareto optimal. This process is highlighted in Figure 5 (a) where the state space is an 8×8 grid and three global Pareto optimal policies exist, each given a different color. In Figure 5 (b), we select actions that are locally non-dominated, i.e, non-dominated within the current state. The black policy is a sequence of locally optimal actions, as it always overlaps with one of the colored lines, however, the resulting policy is not globally Pareto

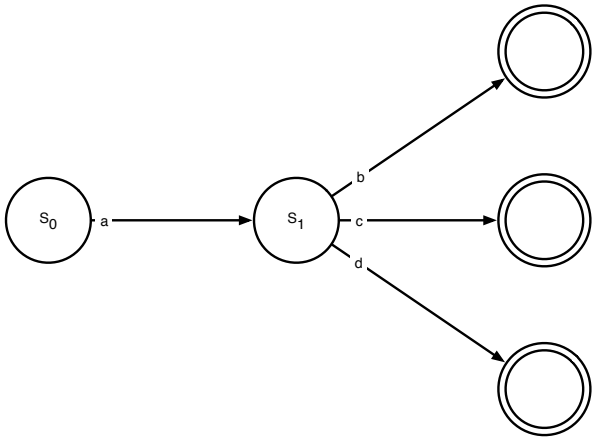
optimal. To conclude, when in a state where multiple actions are considered non-dominated and therefore are incomparable, one can not randomly select between these actions when exploiting a chosen balance between criteria but actions need to be selected consistently.



(a) (b)

Figure 5: (a) In this environment, there is a green, yellow and red Pareto optimal action sequence that is globally optimal. (b) Selecting actions that are locally non-dominated within the current state does not guarantee that the entire policy is globally Pareto optimal. Hence, the information about the global Pareto front has been lost in the local Pareto front.

In order to solve the problem of locally optimal actions that are globally dominated, we define a globally greedy policy as a policy π that consistently *follows* or *tracks* a given expected return vector $\mathbf{V}^{\pi}(s)$ from a state s so that its return equals $\mathbf{V}^{\pi}(s)$ in expectation. Therefore, we need to retrieve π, i.e., which actions to select from a start state to a terminal state. However, due to the stochastic behavior of the environment, it is not trivial to select the necessary actions so as to track the desired return vectors. Let us consider the small bi-objective MDP with deterministic transitions in Figure 6. When the agent reaches a terminal state, denoted by a double circle, the episode is finished. Assume that the discount factor γ is set to 1 for simplicity reasons. Once the Q_{set}'s have converged separately, we can identify three Pareto optimal policies in the start state s_0. These policies have corresponding expected reward vectors $(1.1, 0.5)$, $(2.2, 0.4)$ and $(0.2, 0.6)$. When one is for instance interested in following the policy with an expected reward vector of $(2.2, 0.4)$, the agent should select action a as the vector $(2.2, 0.4)$ is an element of the $\hat{Q}_{set}$ of action a (and there is no other option). But, once in the next state, the next action to select is not clear when one only stores the converged Q_{set}'s. Hence, should the agent select action b, c or d to acquire a return of $(2.2, 0.4)$ at the end of the episode? The approach we propose to solve this issue is based on the separation of the average immediate and future rewards, i.e., we can simply subtract the average immediate reward from the expected return we are targeting, in order to retrieve the next action to select. This way, we can consistently follow the expected return from state s, $\mathbf{V}^{\pi}(s)$, throughout the entire state space. In the example, the agent should select the action that contains $(2.2, 0.4) - (0.2, 0) = (2.0, 0.4)$ in its Q_{set}, i.e., action c. The pseudo-code of the tracking algorithm for environments with deterministic transitions is listed in Algorithm 7. The agent starts in a starting state s of



Action	$\overline{\mathfrak{R}}(s,a)$	$ND_t(s,a)$	$\hat{Q}_{set}(s,a)$
a	(0.2, 0.0)	((0.9, 0.5), (2.0, 0.4), (0.0, 0.6))	((1.1, 0.5), (2.2, 0.4) , (0.2, 0.6))
b	(0.9, 0.5)	()	(0.9, 0.5)
c	(2.0, 0.4)	()	(2.0, 0.4)
d	(0.0, 0.6)	()	(0.0, 0.6)

Figure 6: A small multi-objective MDP and its corresponding Q_{set}'s. As we store the expected immediate and future non-dominated vectors separately, we can consistently follow expected return vectors from start to end state.

the environment and has to follow a particular policy so as to obtain the expected value of the policy from that state, i.e., $\mathbf{V}^{\pi}(s)$, at the end of the episode. For each action of the action set A, we retrieve both the averaged immediate reward $\overline{\mathfrak{R}}(s,a)$ and $ND_t(s,a)$, which we discount. If the sum of these two components equals the target vector to follow, we select the corresponding action and proceed to the next state. The return *target* to follow in the next state s' is then assigned to $\mathbf{Q}$ and the process continues until a terminal state is reached. When the vectors have not entirely converged yet or the transition scheme is stochastic, the equality operator at line 7 should be relaxed. In this case, the action is to be selected that minimizes the difference between the left and the right term. In our experiments, we select the action that minimizes the Manhattan distance between these terms.

4. Results and Discussion

Before we analyze the experiments, we first discuss the general challenges in assessing the performance of on-line multi-policy MORL algorithms in Section 4.1. In Section 4.2, we evaluate and discuss the performance of the Pareto Q-learning algorithm in combination with each of the set evaluation mechanisms on two test problems. In the subsequent section, we perform an empirical comparison of the Pareto Q-learning algorithm to several single-policy MORL algorithms that are described in Section 2.2.1, such as the scalarized MORL

Algorithm 7 Track policy π given the expected reward vector $\mathbf{V}^{\pi}(s)$ from state s

```
target ← V^π(s)
repeat
    for each a in A do
        Retrieve R̄(s, a)
        Retrieve ND_t(s, a)
        for each Q in ND_t(s, a) do
            if γQ + R̄(s, a) = target then
                s ← s' : T(s'|s, a) = 1
                target ← Q
            end if
        end for
    end for
until s is not terminal
```

framework in combination with the linear and Chebyshev scalarization function and the HB-MORL algorithm.

4.1 Performance Assessment of Multi-Policy Algorithms

In single-objective reinforcement learning, an algorithm is usually evaluated by its average reward accumulated over time. The curve of the graph then indicates both the speed of learning and the final performance of the converged policy. In multi-objective reinforcement learning, the performance assessment is more complex because of two main reasons: (1) the reward signal is vectorial and not scalar and (2) there exists no total ordering of the policies but there is a set of *incomparable* optimal policies.

For scalarized MORL algorithms that converge to a single policy, Vamplew et al. (2010) propose to employ the hypervolume indicator on the *approximation* set of policies, i.e., the policies that are obtained after applying a greedy policy for a range of experiments with varying parameters in the scalarization functions. The hypervolume of the approximation set can then be compared to the hypervolume of the *true* Pareto front, i.e., the set of Pareto optimal policies of the environment. Each experiment then represents an individual run of a scalarized MORL algorithm with a specific weight vector $\mathbf{w}$. In the case of tracking globally greedy policies, we can adopt the mechanism by Vamplew et al. (2010) and calculate the hypervolume of the cumulative reward vectors obtained by the tracking algorithm of Section 3.3 for each of the non-dominated vectors in the $ND(\cup_a \hat{Q}_{set}(s_0, a))$, where s_0 is the start state. The hypervolume of each of these vectors should then approach the hypervolume of the Pareto front. It is important to note that in this way, we will evaluate and track as many policies as there exist non-dominated $\hat{\mathbf{Q}}$-vectors in the start state for all actions.

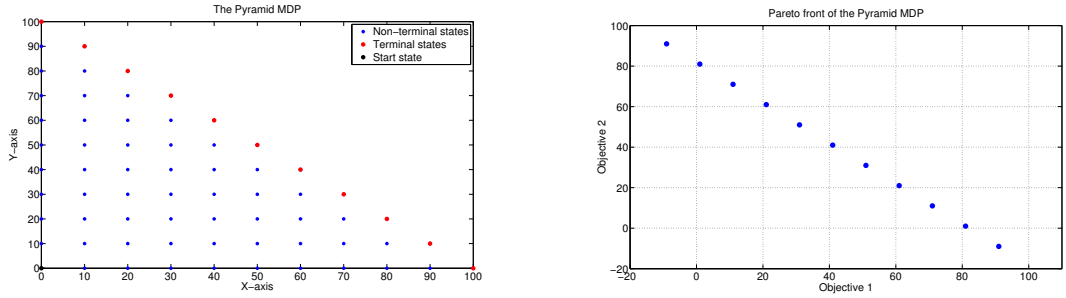
4.2 Benchmarking Pareto Q-learning

In this section, we analyze the performance of the Pareto Q-learning algorithm for each of the three set evaluation mechanisms, i.e., with either the hypervolume, cardinality or Pareto evaluation mechanism. The algorithms are tested on three benchmark environments

with a linear, convex and non-convex Pareto front. All the experiments are averaged over 50 runs and their 95% confidence interval is depicted at regular intervals.

4.2.1 The Pyramid MDP

The Pyramid MDP is a new and simple multi-objective benchmark, which we introduce in this paper. A visual representation of the world is depicted in Figure 7 (a). The agent starts in the down-left position, denoted by a black dot at $(0, 0)$, and it can choose any of the four cardinal directions (up, down, left and right). The transition function is stochastic so that with a probability of 0.95 the selected action is performed and with a probability of 0.05 a random transition is executed to a neighboring state. The red dots represent terminal states. The reward scheme is bi-objective and returns a reward drawn from a Normal distribution with $\mu = -1$ and $\sigma = 0.01$ for both objectives, unless a terminal state is reached. In that case, the x and y position of the terminal state is returned for the first and second objective, respectively. The Pareto front is therefore linear as depicted in Figure 7 (b).



(a) (b)

Figure 7: The Pyramid MDP: the agent starts in the down-left position and can select actions until a terminal state is reached, denoted by a red dot. In (b), we represent the corresponding linear Pareto front.

As we are learning multiple policies simultaneously, which potentially may involve different parts of the state space, we found it beneficial to employ a *train* and *test* setting, where in the *train* mode, we learn with an ϵ-greedy action selection strategy with decreasing epsilon.[2] In the *test* mode of the algorithm, we perform multiple greedy policies using Algorithm 7 for every element in $ND(\cup_a \hat{Q}_{set}(s_0, a))$ of the start state s_0 and we average the accumulated returns along the paths. Each iteration, these average returns are collected and the hypervolume is calculated.

In Figure 8, we present the results of learning and sampling Pareto optimal policies in the Pyramid MDP environment for the train and test phases, respectively. In Figure 8 (a), we depict the hypervolume over time of the estimates in the start state s_0.[3] Hence,

2. At episode *eps*, we assigned ϵ to be 0.997^{eps} to allow for significant amounts of exploration in early runs while maximizing exploitation in later runs of the experiment.
3. In the Pyramid MDP, the reference point for the hypervolume calculation in both *HV-PQL* and the performance assessment was specified to $(-20, -20)$ after observing the reward scheme.

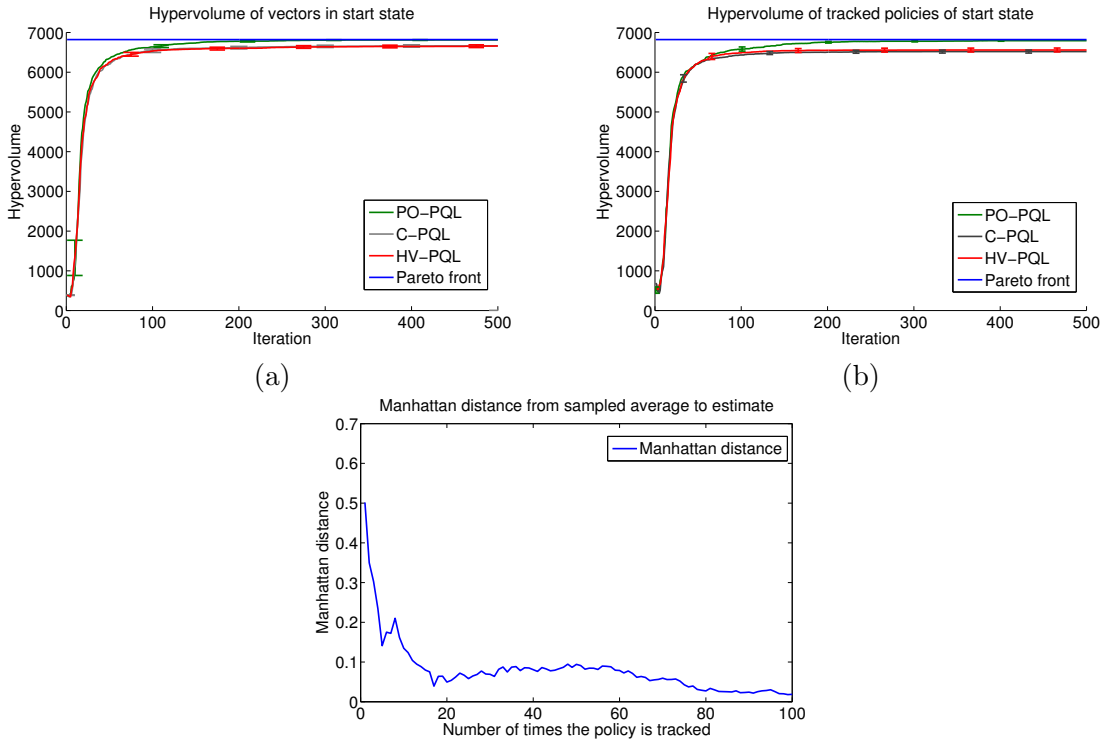


Figure 8: In (a), we depict the hypervolume of the estimates in the start state of the stochastic Pyramid MDP and we note that the entire Pareto front is learned very quickly during the learning phase. In (b), we track these estimates through the state space and denote that the hypervolume of their average returns approaches the Pareto front as well. In (c), we denote the performance of the tracking algorithm for a specific estimate. We see that the Manhattan distance of the running average of the return vector approaches the tracked estimate over time.

for each iteration, we calculate $HV(\cup_a \hat{Q}_{set}(s_0, a))$. We see that each of the set evaluation mechanisms guide the Pareto Q-learning algorithm very well as the hypervolume of the learned estimates approaches the hypervolume of the Pareto front (γ is set to 1). Based on the graph, we see that each of the set evaluation mechanisms has very similar performance in early stages of the learning process. After around hundred iterations, however, we note a small distinction in performance between C-PQL and HV-PQL on the one hand and PO-PQL on the other hand. Closer investigation of the results taught us that this difference is caused by the fact that, once the estimates become stable, C-PQL and HV-PQL still create a total order out of the set of Pareto optimal estimates, even though they are incomparable. That is why, in later iterations of the learning phase, C-PQL and HV-PQL provide a (too) large bias towards particular areas of the state and action space and therefore some estimates are no longer updated. Hence, the very close, but not coinciding curves of their learning graphs. PO-PQL does not provide a total order, but keeps the partial order that the multi-

objective nature of the problem entails. Therefore, it treats every Pareto optimal solution equally and the estimates are updated much more consistently.

In Figure 8 (b), we depict the results of the tracking algorithm of Section 3.3 that globally follows every element of the start state in Figure 8 (a). We see that the hypervolume of the average returns of each the estimated $\hat{\mathbf{Q}}$-vectors in $ND(\cup_a \hat{Q}_{set}(s,a))$ is very similar to the learned estimates themselves. We see that tracking the estimates obtained by PO-PQL allows to sample the entire Pareto front over time.

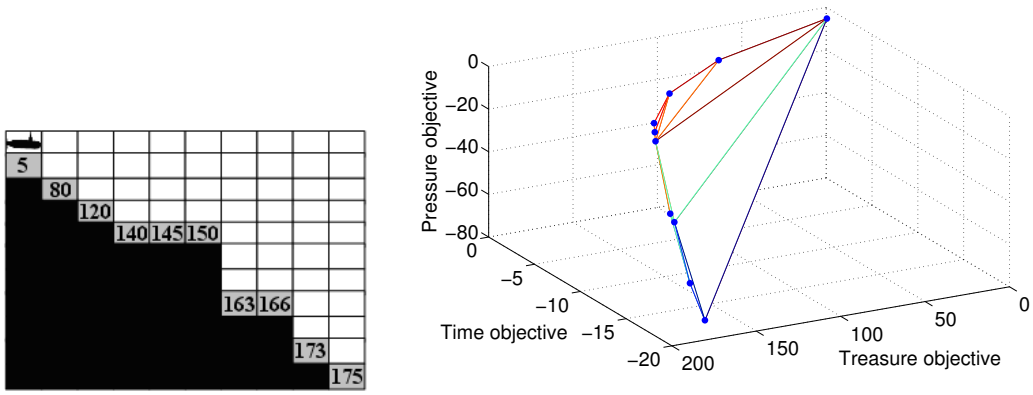
In Figure 8 (c), we see the tracking algorithm at work to retrieve the policy of a specific vectorial estimate from the start state. In the figure, we denote the Manhattan distance of the running average return to the estimate after learning. We see that averaging the return of the policy obtained by the tracking algorithm over time approaches the estimate predicted at the start state, i.e., the distance becomes zero in the limit.

4.2.2 The Pressurized Bountiful Sea Treasure Environment

In order to evaluate the Pareto Q-learning algorithm on an environment with a larger number of objectives, we propose the Pressurized Bountiful Sea Treasure (PBST) environment, which is inspired by the Deep Sea Treasure (DST) environment (Vamplew et al., 2010). Similar to the DST environment, the Pressurized Bountiful Sea Treasure environment concerns a deterministic episodic task where an agent controls a submarine, searching for undersea treasures. The world consists of a 10×11 grid where 10 treasures are located, with larger values as the distance from the starting location increases. A visualization of the environment is depicted in Figure 9 (a). At each time step, the agent can move into one of the cardinal directions. The goal of the agent is to minimize the time needed to reach the treasure, while maximizing the treasure value and to minimize the water pressure.[4] The pressure objective is a novel objective that was not included in the DST environment. It is defined as the agent's y-coordinate. In contrast to the DST, the values of the treasures are altered to create a convex Pareto front. In the PBST environment, a Pareto optimal policy is a path to a treasure that minimizes the Manhattan distance while staying at the surface as long as possible before making the descent to retrieve a treasure. As a result, there are 10 Pareto optimal policies as shown in Figure 9 (b).

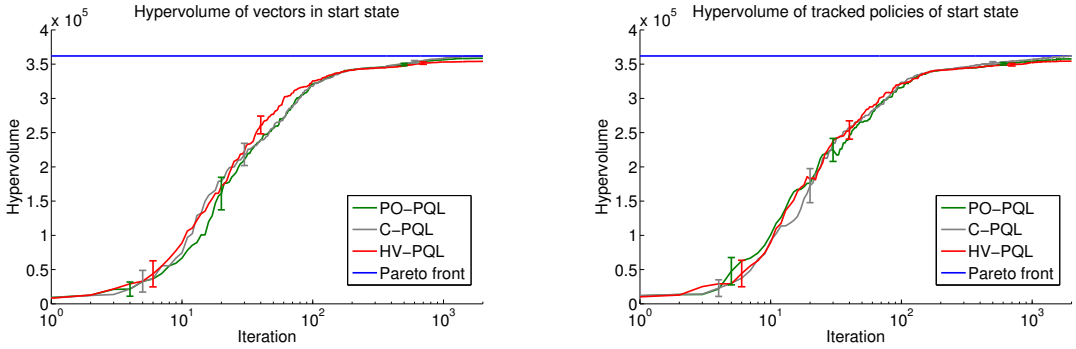
The results on the train and test phases are depicted in Figure 10 (a) and (b), respectively.[5] In Figure 10 (b), we depict the hypervolume of the tracked policies of the start state by applying the greedy policies. As the environment has both deterministic reward and transition schemes, the performance of the different set evaluation mechanisms is almost identical. The tracking algorithm performs very well and the graph is almost identical to Figure 10 (a) as in the previous environment.

4. Traditionally, single-objective reinforcement learning solves a maximization problem. If the problem at hand concerns a minimization of one of the objectives, negative rewards are used for that objective to transform it also into a maximization problem.
5. In the PBST environment, the reference point for the hypervolume calculation in both *HV-PQL* and the performance assessment was specified to $(-25, 0, -120)$ after observing the reward scheme.



(a) (b)

Figure 9: The Pressurized Bountiful Sea Treasure environment (a) and its Pareto front (b).



(a) (b)

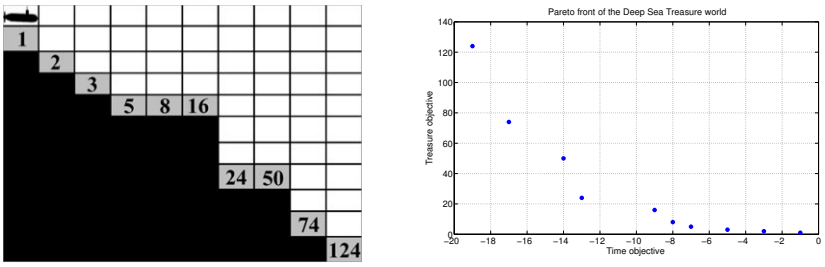
Figure 10: The results on the PBST environment. In (a), we depict the hypervolume over time of the learned estimates in the start state. In (b), we see that the hypervolume of the tracked policies is very similar to the hypervolume of the learned policies, which means that the learned policies are also retrievable.

4.3 Comparison to Single-Policy MORL Algorithms

In the previous section, we analyzed the performance of Pareto Q-learning in combination with the three set evaluation mechanisms . In this section, we conduct an empirical comparison of the algorithms to several single-policy MORL algorithms.

4.3.1 The Deep Sea Treasure Environment

The Deep Sea Treasure (DST) is proposed by Vamplew et al. (2010) and is a standard MORL benchmark instance. A brief description of the environment can be found in Section 4.2.2.



(a) (b)

Figure 11: Deep Sea Treasure environment (a) and its Pareto front (b).

The DST environment and its non-convex Pareto front are depicted in Figure 11 (a) and (b), respectively.

In Figure 12, we denote the hypervolume during the test phase of the algorithm with Pareto, cardinality and hypervolume set evaluations, i.e., PO-PQL, C-PQL and HV-PQL, respectively.[6] Furthermore, we also evaluate two single-policy algorithms that employ the linear and Chebyshev scalarization functions and HB-MORL, the indicator-based MORL algorithm of Section 2.2.1. These single-policy algorithms are evaluated using the configuration specified in Section 4.1. Below, we highlight the performance of each algorithm individually.

The linear scalarized MORL algorithm is run with 10 uniformly distributed weight vectors, i.e., the continuous range of $[0, 1]$ is uniformly discretized with steps of $\frac{1}{10-1}$ while satisfying $\sum_{o=1}^{m} \mathbf{w}_o = 1$. Each of these weights is then used in an individual execution of the scalarization algorithm and its results are collected in order to obtain a set of sampled policies in the test phase. We note that, although we have a uniform spread of weights, the algorithm only manages to retrieve a hypervolume of 768. When we take a closer look at the results obtained, we see that the algorithm learns fast but from iteration 200, only the optimal policies with return $(-1, 1)$ and $(-19, 124)$ for the time and treasure objectives, respectively, are sampled. This is shown by the 95% confidence intervals that become zero after iteration 200. This is to be expected as the Pareto front is non-convex and, hence, a linear combination of the objectives then can only differentiate between the *extreme* policies of the Pareto front. Therefore, the linear scalarized MORL algorithm converges to either of the optimal policies with return $(-1, 1)$ or $(-19, 124)$.

The Chebyshev scalarized MORL algorithm is equipped with the same set of weight vectors as the linear scalarization function. While the Chebyshev scalarization function has proven its effectiveness in multi-objective optimization, it is not guaranteed to converge to a Pareto optimal policy in a value-iteration approach (Perny and Weng, 2010). Nevertheless, we see that the algorithm performs acceptably in practice as it learns almost at the same speed as the linear scalarized MORL algorithm and attains a bigger hypervolume of 957. Other experiments will have to investigate whether this performance in practice is consistent over multiple environments, despite the lack of theoretical guarantees. A first initiative has been given in Van Moffaert et al. (2013b).

6. The reference point for the hypervolume calculation was specified to $(-25, 0)$.

Although the indicator-based HB-MORL algorithm does not rely on any weighting parameters, we also run the experiment 10 times in parallel to obtain a set of policies which we can compare at every time step. As there are no parameters to steer the search process, each individual experiment is not guaranteed to converge to a particular solution of the Pareto front. That is why the graph is increasing in the beginning of the learning process but it is slightly decreasing near the end as there is no mechanism, like for instance weights in the standard scalarization algorithms, to specify which experiment should focus on which part of the objective space. In the end, we see that the performance of HB-MORL drops slightly under the curve of the linear scalarized MORL algorithm.

So far, the single-policy algorithms did not manage to sample the entire Pareto front. This was a result of either the shape of the Pareto front, i.e., being non-convex, or the assignment of the weight vectors. Pareto Q-learning is not biased by any of these aspects, but treats every Pareto optimal solution equally in the bootstrapping process. As the environment consist of 10 Pareto optimal policies, the set of non-dominated vectors of the start state, i.e., $ND(\cup_a \hat{Q}_set(s_0, a))$, exactly contains 10 elements. Therefore, we evaluate as many policies as the scalarization algorithms in this comparison. In early learning phases, we see that for each of the set evaluation mechanisms, Pareto Q-learning learns slower than the scalarized MORL algorithms. This is because the weights of the scalarization algorithms guide each individual experiment to explore specific parts of the objective space, while the guidance mechanisms of the Pareto Q-learning algorithm, i.e., the set evaluation mechanisms, are less explicit. In the end, regardless of the set evaluation mechanisms used, Pareto Q-learning surpasses the performance of the single-policy algorithms. In this experiment, the Pareto set evaluation mechanism starts out the worst, but, in the end, it performs a bit better than the other set evaluation mechanisms and samples every element of the Pareto front.

5. Conclusions

In this paper, we have presented and discussed multi-objective optimization and reinforcement learning approaches for learning policies in multi-objective environments. We have highlighted that single-policy MORL algorithms rely on scalarization functions and weight vectors to translate the original multi-objective problem into a single-objective problem. Although these algorithms are very common in practice, they suffer from two main shortcomings: their performance depends heavily on (1) the shape of the Pareto front and on (2) an appropriate choice of the weight vectors, which are hard to specify a priori.

The main contribution of this paper is the novel Pareto Q-learner algorithm that learns deterministic non-stationary non-dominated multi-objective policies for episodic environments with a deterministic as well as stochastic transition function. To the best of our knowledge, PQL is the first multi-policy TD algorithm that allows to learn the entire Pareto front, and not just a subset. The core principle of our work consists of keeping track of the immediate reward vectors and the future discounted Pareto dominating vectors separately. This mechanism provides a neat and clean solution to update sets of vectors over time.

In a reinforcement learning algorithm, the exploration and exploitation trade-off is crucial. Therefore, we developed three evaluation mechanisms that use the $\hat{Q}_{set}$'s as a basis for action selection purposes during learning. We name them set evaluation mechanisms.

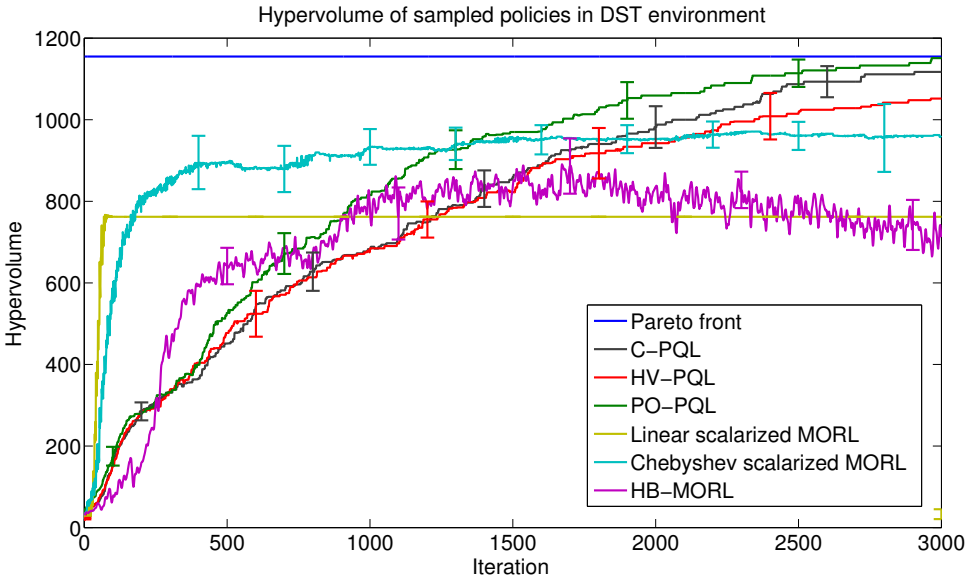


Figure 12: The results on the Deep Sea Treasure environment. We compare three single-policy MORL algorithms that use a linear and Chebyshev scalarization function and HB-MORL to Pareto Q-learning with the three set evaluation mechanisms. We note that, regardless of the set evaluation mechanisms, PQL obtained better performance than the single-policy MORL algorithms and in the end PO-PQL sampled the entire Pareto front. For a more in-depth analysis of the results, we refer to Section 4.3.1.

The current set evaluation mechanisms rely on basic multi-objective indicators to translate the quality of a set of vectors into a scalar value. Based on these indications, local action selection is possible during the learning phase. Currently, we have combined PQL with a cardinality, hypervolume and Pareto indicator. We have seen that the Pareto indicator performed the best on average as it treats every Pareto optimal solution equally. The cardinality and hypervolume set evaluation measures rate the actions also on additional criteria than the Pareto relation to provide a total order. We have seen that in more complex environments, these set evaluation mechanisms bias the action selection too much in order to learn the entire Pareto front. Nevertheless, it could be that the user is not interested in sampling the entire Pareto front but is looking for particular policies that satisfy certain criteria. For instance, other quality indicators such as the spread indicator (Van Veldhuizen and Lamont, 1998) could be used to sample policies that are both Pareto optimal and well-spread in the objective space. This can straightforwardly be incorporated in our framework.

In our experiments, we have tested the Pareto Q-learning algorithm on environments with two and three objectives. However, as the algorithm is based on the Pareto relation, it is without problem applicable to environments with a larger number of objectives. Additionally, we also conducted empirical evaluations on a benchmark instance and we compared

Pareto Q-learning's performance to several single-policy MORL algorithms. We have seen that selecting actions that are locally dominating does not guarantee that the overall combination of selected actions in each state, i.e., the policy, is globally Pareto optimal. As a solution, we proposed a mechanism that *tracks* a given return vector, i.e., we can follow a selected expected return consistently from the start state to a terminal state in order to collect the predicted rewards.

In Pareto Q-learning, the Q_{set}'s grow according to the size of the Pareto front. Therefore, PQL is primarily designed for episodic environments with a finite number of Pareto optimal policies. To make the algorithm practically applicable for infinite horizon problems with a large value for the discount factor, we have to consider that all states can be revisited during the execution of an optimal policy. Upon revisiting a state, a different action that is optimal w.r.t. other criteria can be chosen. As explained by Mannor and Shimkin (2002), this offers a possibility to steer the average reward vector towards a target set using *approaching* policies. Alternatively, we could reduce the number of learned policies by using a consistency operator to select the same action during each revisit of some state, similar to the work of Wiering and de Jong (2007) for multi-criteria DP.

Currently, PQL is limited to a tabular representation where each state-action pair stores a Q_{set}. In order to make PQL applicable to real-life problems or ergodic environments, these sets should also be represented through function approximation. A possible idea is to fit the elements in each set through a geometric approach, such as for instance ordinal least-squares. If the environment would consist of two or three objectives, we would be fitting the vectors on a curve or a plane, respectively. In that case, we would be learning the shape of the Pareto front through local interactions that each update parts of this geometric shape.

To summarize, we note that PQL (1) can learn the entire Pareto front under the assumption that each state-action pair is sufficiently sampled, (2) while not being biased by the shape of the Pareto front or a specific weight vector. Furthermore, we have seen that (3) the set evaluation mechanisms provide indicative measures to explore the objective space based on local action selections and (4) the learned policies can be tracked throughout the state and action space.

Acknowledgements

We would like to thank Tim Brys for his collaboration on the PBST environment and Diederik M. Roijers for his valuable insights and comments. This work has been carried out within the framework of the Perpetual project (grant nr. 110041) of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). Together with the Multi-Criteria Reinforcement Learning project (grant G.0878.14N) of the Fonds Wetenschappelijk Onderzoek - Vlaanderen (FWO).

References

L. Barrett and S. Narayanan. Learning all optimal policies with multiple criteria. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 41–47, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390162.

N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, September 2007.

C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387332545.

I. Das and J. E. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Structural and Multidisciplinary Optimization*, 14:63–69, 1997. ISSN 1615-147X.

K. D. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2): 182–197, April 2002. ISSN 1089778X. doi: 10.1109/4235.996017.

N. Dunford, J. T. Schwartz, W. G. Bade, and R. G. Bartle. *Linear Operators: General theory. Part. I.* Linear Operators. Interscience Publishers, 1988. ISBN 9780470226056.

Z. Gabor, Z. Kalmar, and C. Szepesvari. Multi-criteria reinforcement learning. In *Proceedings of the 15th International Conference on Machine Learning*, ICML '98, Madison, WI, 1998.

J.-M. Gorce, R. Zhang, K. Jaffrès-Runser, and C. Goursaud. Energy, latency and capacity trade-offs in wireless multi-hop networks. In *Proceedings of the IEEE 21st International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2010*, pages 2757–2762. IEEE, 2010.

F. Hernandez-del Olmo, F. H. Llanes, and E. Gaudioso. An emergent approach for the control of wastewater treatment plants by means of reinforcement learning techniques. *Expert Systems with Applications*, 39(3):2355–2360, 2012.

C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28, 2007.

D. J. Lizotte, M. Bowling, and S. A. Murphy. Efficient reinforcement learning with multiple reward functions for randomized controlled trial analysis. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, ICML '10, pages 695–702, 2010.

D. J. Lizotte, M. Bowling, and S. A. Murphy. Linear fitted-q iteration with multiple reward functions. *Journal of Machine Learning Research*, 13:3253–3295, 2012.

S Mannor and N. Shimkin. The steering approach for multi-criteria reinforcement learning. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1563–1570. MIT Press, 2002.

S. Mannor and N. Shimkin. A geometric approach to multi-criterion reinforcement learning. *Journal of Machine Learning Research*, 5:325–360, 2004.

K. Miettinen and M. M. Mäkelä. On scalarizing functions in multiobjective optimization. *OR Spectrum*, 24:193–213, 2002. ISSN 0171-6468.

P. Perny and P. Weng. On finding compromise solutions in multiobjective Markov decision processes. In *Proceedings of the 19th European Conference on Artificial Intelligence*, ECAI '10, pages 969–970, Amsterdam, The Netherlands, 2010. IOS Press. ISBN 978-1-60750-605-8.

D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. Mit Press, 1998. ISBN 9780262193986.

G. Tesauro, R. Das, H. Chan, J. O. Kephart, D. W. Levine, F. Rawson, and C. Lefurgy. Managing power consumption and performance of computing systems using reinforcement learning. In J.C. Platt, D. Koller, Y. Singer, and S.T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1497–1504. Curran Associates, Inc., 2008.

J. N. Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Journal of Machine Learning*, 16(3):185–202, 1994.

P. Vamplew, J. Yearwood, R. Dazeley, and A. Berry. On the limitations of scalarisation for multi-objective reinforcement learning of Pareto fronts. In *Proceedings of the 21st Australasian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*, AI '08, pages 372–378, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-89377-6.

P. Vamplew, R. Dazeley, E. Barker, and A. Kelarev. Constructing stochastic mixture policies for episodic multiobjective reinforcement learning tasks. In A. E. Nicholson and X. Li, editors, *Australasian Conference on Artificial Intelligence*, volume 5866 of *Lecture Notes in Computer Science*, pages 340–349. Springer, 2009. ISBN 978-3-642-10438-1.

P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, 84 (1-2):51–80, 2010.

K. Van Moffaert, M. M. Drugan, and A. Nowé. Hypervolume-based multi-objective reinforcement learning. In R. Purshouse, P. Fleming, C. Fonseca, S. Greco, and J. Shaw, editors, *Evolutionary Multi-Criterion Optimization*, volume 7811 of *Lecture Notes in Computer Science*, pages 352–366. Springer Berlin Heidelberg, 2013a. ISBN 978-3-642-37139-4.

K. Van Moffaert, M. M Drugan, and A. Nowé. Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE International Symposium on Adaptive Dynamic Programming And Reinforcement Learning (ADPRL)*, pages 191–199. IEEE, 2013b.

D. A. Van Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithm research: a history and analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1998.

W. Wang and M. Sebag. Hypervolume indicator and dominance reward based multi-objective Monte-Carlo tree search. *Machine Learning*, 92(2-3):403–429, 2013. ISSN 0885-6125. doi: 10.1007/s10994-013-5369-0.

C. Watkins. *Learning from Delayed Rewards*. PhD thesis, University of Cambridge, England, 1989.

D. J. White. Multi-objective infinite-horizon discounted Markov decision processes. *Journal of Mathematical Analysis and Applications*, 89(2), 1982.

M. A. Wiering and E. D. de Jong. Computing optimal stationary policies for multi-objective Markov decision processes. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 158–165. IEEE, April 2007. ISBN 1-4244-0706-0.

E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In K.C. Giannakoglou et al., editors, *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, pages 95–100. International Center for Numerical Methods in Engineering (CIMNE), 2002.

E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

Journal of Machine Learning Research 15 (2014) 3693-3720 Submitted 1/14; Revised 7/14; Published 11/14

Seeded Graph Matching for Correlated Erdős-Rényi Graphs

Vince Lyzinski VLYZINS1@JHU.EDU
Human Language Technology Center of Excellence
Johns Hopkins University
Baltimore, MD, 21218, USA

Donniell E. Fishkind DEF@JHU.EDU
Carey E. Priebe CEP@JHU.EDU
Department of Applied Mathematics and Statistics
Johns Hopkins University
Baltimore, MD, 21218, USA

Editor: Edoardo Airoldi

Abstract

Graph matching is an important problem in machine learning and pattern recognition. Herein, we present theoretical and practical results on the consistency of graph matching for estimating a latent alignment function between the vertex sets of two graphs, as well as subsequent algorithmic implications when the latent alignment is partially observed. In the correlated Erdős-Rényi graph setting, we prove that graph matching provides a strongly consistent estimate of the latent alignment in the presence of even modest correlation. We then investigate a tractable, restricted-focus version of graph matching, which is only concerned with adjacency involving vertices in a partial observation of the latent alignment; we prove that a logarithmic number of vertices whose alignment is known is sufficient for this restricted-focus version of graph matching to yield a strongly consistent estimate of the latent alignment of the remaining vertices. We show how Frank-Wolfe methodology for approximate graph matching, when there is a partially observed latent alignment, inherently incorporates this restricted-focus graph matching. Lastly, we illustrate the relationship between seeded graph matching and restricted-focus graph matching by means of an illuminating example from human connectomics.

Keywords: graph matching, Erdős-Rényi graph, consistency, estimation, seeded vertices, Frank-Wolfe, assignment problem

1. Background and Overview

The graph matching problem (GMP)—i.e., finding the alignment between the vertices of two graphs which best preserves the structure of the graphs—has a rich and active place in the literature. Graph matching has applications in a wide variety of disciplines, including machine learning (Cour et al., 2007; Liu and Qiao, 2012; Fiori et al., 2013), computer vision (Cho et al., 2009; Cho and Lee, 2012; Zhou and De la Torre, 2012), pattern recognition (Berg et al., 2005; Caelli and Kosinov, 2004), manifold and embedded graph alignment (Robles-Kelly and Hancock, 2007; Xiao et al., 2009), shape matching and object recognition (Huet et al., 1999), and MAP inference (Leordeanu et al., 2009), to name a few.

There are no efficient algorithms known for solving graph matching exactly. Even the easier problem of just deciding if two graphs are isomorphic is notoriously of unknown complexity (Garey and Johnson, 1979; Read and Corneil, 1977). Indeed, graph matching is a special case of the NP-hard quadratic assignment problem and, if the graphs are allowed to be directed, loopy, and weighted, then graph matching is actually equivalent to the quadratic assignment problem. Because of its practical applicability, there is a vast amount of literature devoted to approximate graph matching algorithms; for an interesting survey of the literature, see e.g., "Thirty Years of Graph Matching in Pattern Recognition" by Conte et al. (2004).

In the presence of a latent alignment function between the vertex sets of two graphs, it is natural to ask how well graph matching would mirror this underlying alignment. In Section 2.2 we describe the correlated Erdős-Rényi random graph, which provides us with a useful and natural setting to explore this question. The correlated Erdős-Rényi random graph consists of two Erdős-Rényi random graphs which share a common vertex set and a common Bernoulli-trial probability parameter; for each pair of vertices, there is a given correlation between the two vertices' adjacency in one graph and the two vertices' adjacency in the other graph. In this manner, there is a natural latent alignment between the two graphs, and we can then explore whether or not graph matching the two graphs will consistently estimate this alignment.

If $\Phi : V(G_1) \mapsto V(G_2)$ is the latent alignment function between the vertex sets of two graphs, we define a vertex $v \in V(G_1)$ to be *mismatched* by graph matching if there exists a solution ψ to the graph matching problem such that $\Phi(v) \neq \psi(v)$. The graph matching problem provides a *consistent* estimate of Φ if the number of mismatched vertices goes to zero in probability as $|V(G_1)|$ tends to infinity, and provides a *strongly consistent* estimate of Φ if the number of mismatched vertices converges to zero almost surely as $|V(G_1)|$ tends to infinity.

The first of our main results is Theorem 1, stated in Section 2.2 and proven in Appendix A. For correlated Erdős-Rényi random graphs, under mild assumptions, Theorem 1.i establishes that even very modest correlation is sufficient for graph matching to yield a strongly consistent estimate of the latent alignment; this expands and strengthens the important results in Pedarsani and Grossglauser (2011). Theorem 1.ii provides a partial converse; for very weakly correlated graphs, we prove that the expected number of permutations that align the graph more effectively (i.e., with fewer induced edge disagreements) than the latent alignment goes to infinity as the number of vertices tends to infinity. Unfortunately, since there is no known efficient algorithm for graph matching, Theorem 1.i doesn't in-of-itself provide a means of efficient graph alignment. However, it does suggest that efficient approximate graph matching algorithms may be successful in graph alignment when there is correlation between the graphs above the threshold given in Theorem 1.i.

Next, in Section 2.3, we discuss the seeded graph matching problem. This is a graph matching problem for which part of the bijection between the two graphs' vertices is prespecified and fixed, and we seek to complete the bijection so as to minimize the number of edge disagreements between the graphs; in our correlated Erdős-Rényi graph setting, the seeds are taken from the existing latent alignment. Also in Section 2.3, we describe a restricted-focus version of the graph matching problem in the context of seeding; this is a problem wherein we seek the bijection between the two seeded graphs' vertices that mini-

mizes *only the number of seeded vertex to nonseeded vertex edge disagreements* between the two seeded graphs. Restricting the focus of graph matching in this particular fashion enables this restricted-focus graph matching problem to be efficiently solved as a linear assignment problem, in contrast to the algorithmic difficulty of (unrestricted) graph matching.

Our second main result is Theorem 2, which we state in Section 2.4 and prove in Appendix A. For correlated Erdős-Rényi graphs, under mild assumptions, Theorem 2.i asserts that a logarithmic number of seeds is sufficient for restricted-focus graph matching to yield a strongly consistent estimate of the latent alignment function. Theorem 2.ii again provides a partial converse; for very weakly correlated graphs, we prove that the expected number of permutations that align the unseeded vertices more effectively (i.e., with fewer induced seeded vertex to nonseeded vertex edge disagreements) than the latent alignment goes to infinity as the number of vertices tends to infinity. Now, what should we do if we want to perform graph alignment and there are seeds, but the number of seeds is below this logarithmic threshold? The remainder of this paper deals with that situation.

Back in the setting where there are no seeds, an important class of approximate graph matching algorithms utilize a Frank-Wolfe approach; the idea is more formally described later in Section 3. To briefly describe here, such methods relax an integer programming formulation of graph matching to obtain a continuous problem, then perform an iterative procedure in which a linearization about the current iterate is optimized, and the next iterate comes from a line search between the current iterate and the linearization optimum. At the conclusion of the iterative procedure, the final iterate is projected to the nearest integer-valued point which is feasible as a graph match, and this is taken as the approximate graph matching solution. It turns out that the linear optimization done in each iteration can be formulated as a linear assignment problem, which can be solved efficiently, and this makes the Frank-Wolfe approach an appealing method in terms of speed. The Frank-Wolfe approach can also be a very accurate method for approximate graph matching as well; see Brixius and Anstreicher (2001); Vogelstein et al. (2011); Zaslavskiy et al. (2009) for Frank-Wolfe methodology and variants.

As done in Fishkind et al. (2012), we describe in Section 3.2 how this Frank-Wolfe methodology for approximate graph matching is naturally and seamlessly extended to the setting of seeded graph matching so as to perform approximate seeded graph matching. In analyzing Frank-Wolfe methodology for approximate seeded graph matching, we observe in Section 3.3 that each Frank-Wolfe iteration involves optimizing a sum of two terms. Restricting this optimization to just the first of these two terms turns out to be precisely solving the aforementioned restricted-focus graph matching problem, and restricting this optimization to just the second of these two terms turns out to be precisely the Frank-Wolfe methodology step if the seeds are completely ignored.

We conclude this paper with simulations and a real-data example from human connectomics. These simulations and experiments illuminate the relationship between seeded graph matching via Frank-Wolfe and restricted-focus graph matching via the Hungarian algorithm. We demonstrate that Frank-Wolfe methodology is often superior to restricted-focus graph matching, an unsurprising result as the Frank-Wolfe methodology merges restricted-focus graph matching with seedless Frank-Wolfe methodology. Perhaps more surprising, we also demonstrate the capacity for restricted-focus graph matching to outper-

form the full Frank-Wolfe methodology; in these cases, the noise in the unseeded adjacency can actually degrade overall performance!

2. Graph Matching, Random Graph Setting, Main Results

In this paper, all graphs will be simple graphs; in particular, edges are undirected, there are no edges with a common vertex for both endpoints, and there are no multiple edges between any pair of vertices. We will define $\mathcal{G}_n$ to be the set of simple graphs on n vertices. If $G \in \mathcal{G}_n$, we will denote the vertex set of G as $V(G)$ and the edge set of G via $E(G)$. For any $v, v' \in V(G)$, if v and v' are adjacent in G then this will be denoted $\{v, v'\} \in E(G)$, and if v and v' are not adjacent in G then this will be denoted $\{v, v'\} \notin E(G)$. For any finite set V, the symbol $\binom{V}{2}$ will denote all of the $\binom{n}{2}$ unordered pairs of distinct elements from V.

2.1 The Graph Matching Problem

We now describe the graph matching problem. Suppose G_1 and G_2 are graphs with the same number of vertices. Let $\mathit{\Pi}$ denote the set of bijections $V(G_1) \to V(G_2)$. For any $\psi \in \mathit{\Pi}$, the *number of adjacency disagreements induced by* ψ, which will be denoted $\Delta(\psi)$, is the number of vertex pairs $\{v, v'\} \in \binom{V(G_1)}{2}$ such that $[\{v, v'\} \in E(G_1)$ and $\{\psi(v), \psi(v')\} \notin E(G_2)]$ or $[\{v, v'\} \notin E(G_1)$ and $\{\psi(v), \psi(v')\} \in E(G_2)]$. The *graph matching problem* is to find a bijection in $\mathit{\Pi}$ that minimizes the number of induced edge disagreements; we will denote the set of solutions $\Psi := \arg\min_{\psi \in \mathit{\Pi}} \Delta(\psi)$. Equivalently stated, if $n := |V(G_1)| = |V(G_2)|$, and if $A, B \in \{0,1\}^{n \times n}$ are the respectively the adjacency matrices for G_1 and G_2, then the graph matching problem is to minimize $\|A - PBP^T\|_F$ over all n-by-n permutation matrices P, where $\|\cdot\|_F$ is the Frobenius matrix norm.

There are no efficient algorithms known for graph matching. Even the easier problem of just deciding if G_1 is isomorphic to G_2 (i.e., deciding if there is a bijection $V(G_1) \to V(G_2)$ which does not induce any edge disagreements) is of unknown complexity (Garey and Johnson, 1979; Read and Corneil, 1977), and is a candidate for being in an intermediate class strictly between P and NP-complete (if P$\neq$NP). Also, the problem of minimizing $\|A - PBP^T\|_F$ over all n-by-n permutation matrices P, where A and B are any real-valued matrices, is equivalent to the NP-hard quadratic assignment problem. There are numerous approximate graph matching algorithms in the literature; in Section 3 we will discuss Frank-Wolfe methodology.

2.2 Correlated Erdős-Rényi Random Graphs

Presently, we describe the correlated Erdős-Rényi random graph; this will provide a theoretical framework within which we will prove our main theorems, Theorem 1 and Theorem 2.

The parameters are a positive integer n, a real number p in the interval $(0, 1)$, and a real number ϱ in the interval $[0, 1]$; these parameters completely specify the distribution. There is an underlying vertex set V of cardinality n which is common to two graphs; call these graphs G_1 and G_2. For each $i = 1, 2$ and each pair of vertices $\{v, v'\} \in \binom{V}{2}$, let $\mathbb{1}\{\{v, v'\} \in E(G_i)\}$ denote the indicator random variable for the event $\{v, v'\} \in E(G_i)$. For each $i = 1, 2$ and each pair of vertices $\{v, v'\} \in \binom{V}{2}$, the random variable $\mathbb{1}\{\{v, v'\} \in E(G_i)\}$

is Bernoulli(p) distributed, and they are all collectively independent except that, for each pair of vertices $\{v, v'\} \in \binom{V}{2}$, the variables $\mathbb{1}\{\{v, v'\} \in E(G_1)\}$ and $\mathbb{1}\{\{v, v'\} \in E(G_2)\}$ have Pearson product-moment correlation coefficient ϱ. At one extreme, if ϱ is 1, then G_1 and G_2 are equal, almost surely, and at the other extreme, if ϱ is 0, then G_1 and G_2 are independent. After G_1 and G_2 are thus realized, their vertices are (separately) arbitrarily relabeled, so that we don't directly observe the *latent alignment function (bijection)* $\Phi : V(G_1) \to V(G_2)$ wherein, for all $v \in V(G_1)$, the vertices v and $\Phi(v)$ were corresponding vertices across the graphs before the relabeling (i.e., the same element of V).

If G_1 is graph matched to G_2, to what extent will the graph match provide a consistent estimate of the latent alignment function? The following Theorem is our first main result. We will be considering a sequence of random correlated Erdős-Rényi graphs with $n = 1$, then $n = 2$, then $n = 3 \ldots$, and the parameters p and ϱ are each functions of n; i.e., $p := p(n)$ and $\varrho := \varrho(n)$. In this paper, when we say a sequence of events holds *almost always*, we mean that, with probability 1, all but a finite number of the events hold.

Theorem 1 *Suppose there exists a fixed real number $\xi_1 < 1$ such that $p \leq \xi_1$. Then there exists fixed positive real numbers c_1, c_2, c_3, c_4 (depending only on the value of ξ_1) such that:*
i) If $\varrho \geq c_1\sqrt{\frac{\log n}{np}}$ and $p \geq c_2\frac{\log n}{n}$ then almost always $\Psi = \{\Phi\}$, and
ii) If $\varrho \leq c_3\sqrt{\frac{\log n}{n}}$ and $p \geq c_4\frac{\log n}{n}$ then $\lim_{n\to\infty} \mathbb{E}|\{\psi \in \Pi : \Delta(\psi) < \Delta(\Phi)\}| = \infty$.

For proof of Theorem 1, see Appendix A.

Note that Theorem 1.i establishes the strong consistency of the graph matching estimate of the latent alignment function in the presence of even modest correlation between G_1 and G_2. This theorem is a strengthening and an extension of the pioneering work on de-anonymizing networks in Pedarsani and Grossglauser (2011), wherein the authors proved a weaker version of Theorem 1.i in a sparse setting (in particular they require both p and ϱ to converge to 0 at rate $p\varrho^3 = O(log(n)/n)$). Note that range of values of p for which Theorem 1.i applies includes both the sparse and the dense regimes.

Because there is no known efficient algorithm for graph matching, Theorem 1.i does not directly provide a practical means of computing the latent alignment function. But it does hold out the hope that a good graph matching heuristic might be effective in approximating the latent alignment function for various classes of graphs.

When proving Theorems 1 and 2, it will be useful for us to observe an equivalent way to formulate correlated Erdős-Rényi graphs. For all pairs of vertices $\{v, v'\} \in \binom{V}{2}$, the indicator random variables $\mathbb{1}\{\{v, v'\} \in E(G_1)\}$ are independently distributed Bernoulli(p) and then (independently for the different pairs v, v'), conditioning on $\mathbb{1}\{\{v, v'\} \in E(G_1)\} = 1$, we let $\mathbb{1}\{\{v, v'\} \in E(G_2)\}$ be distributed Bernoulli($p + \varrho(1 - p)$) and, conditioning on $\mathbb{1}\{\{v, v'\} \in E(G_1)\} = 0$, we let $\mathbb{1}\{\{v, v'\} \in E(G_2)\}$ be distributed Bernoulli($p(1 - \varrho)$). It is an easy exercise to verify that as such, for each $\{v, v'\} \in \binom{V}{2}$, it holds that $\mathbb{1}\{\{v, v'\} \in E(G_2)\}$ is distributed Bernoulli(p), and that the correlation of $\mathbb{1}\{\{v, v'\} \in E(G_1)\}$ and $\mathbb{1}\{\{v, v'\} \in E(G_2)\}$ is ϱ, as desired.

2.3 Seeded Graph Matching, Restricted-focus Graph Matching

Continuing with the setting from Section 2.1, suppose that we are also given a subset $U_1 \subseteq V(G_1)$ of *seeds* and an injective *seeding function* $\phi : U_1 \to V(G_2)$, say that $U_2 \subseteq V(G_2)$ is the image of ϕ. Let Π_ϕ denote the set of bijections $\psi : V(G_1) \to V(G_2)$ such that for all $u \in U_1$ it holds that $\psi(u) = \phi(u)$. As before, for any bijection $\psi \in \Pi_\phi$, the *number of adjacency disagreements induced by* ψ, which will be denoted $\Delta(\psi)$, is the number of vertex pairs $\{v, v'\} \in \binom{V(G_1)}{2}$ such that $[\{v, v'\} \in E(G_1)$ and $\{\psi(v), \psi(v')\} \notin E(G_2)]$ or $[\{v, v'\} \notin E(G_1)$ and $\{\psi(v), \psi(v')\} \in E(G_2)]$. The *seeded graph matching problem* is to find a bijection in Π_ϕ that minimizes the number of induced edge disagreements; as before, we will denote the set of solutions $\Psi := \arg\min_{\psi \in \Pi_\phi} \Delta(\psi)$. Equivalently stated, suppose without loss of generality that $U_1 = U_2 = \{v_1, v_2, \ldots, v_s\}$, and that for all $j = 1, 2, \ldots, s$, $\phi(v_j) = v_j$; with A and B denoting the adjacency matrices for G_1 and G_2 respectively, the seeded graph matching problem is to minimize $\|A - (I \oplus P)B(I \oplus P)^T\|_F$ over all m-by-m permutation matrices P, where $m := |V(G_1)| - s$, and $\oplus$ is the direct sum, and I is the s-by-s identity matrix.

Like graph matching, there are no efficient algorithms known for seeded graph matching; in fact, seeded graph matching is at least as difficult as graph matching. In Section 3.2 we discuss how Frank-Wolfe methodology extends to provide efficient approximate seeded graph matching.

We now present a restricted version of seeded graph matching which is efficiently solvable, in contrast to graph matching and seeded graph matching. Let $W_1 := V(G_1) \backslash U_1$ denote the nonseeds in $V(G_1)$. For any $\psi \in \Pi_\phi$, let $\Delta_R(\psi)$ denote the number of pairs $(w, u) \in W_1 \times U_1$ such that $[\{w, u\} \in E(G_1)$ and $\{\psi(w), \psi(u)\} \notin E(G_2)]$ or $[\{w, u\} \notin E(G_1)$ and $\{\psi(w), \psi(u)\} \in E(G_2)]$. The *restricted-focus seeded graph matching problem* (RGM) is to find a bijection in Π_ϕ which minimizes such seed-nonseed adjacency disagreements; denote the set of solutions $\Psi_R := \arg\min_{\psi \in \Pi_\phi} \Delta_R(\psi)$. Equivalently stated, if the adjacency matrices for G_1 and G_2 are respectively partitioned as

$$A = \begin{pmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{pmatrix}, \text{ and } B = \begin{pmatrix} B_{11} & B_{21}^T \\ B_{21} & B_{22} \end{pmatrix}$$

where $A_{21}, B_{21} \in \mathbb{R}^{|W_1| \times |U_1|}$ each represent the adjacencies between the nonseed vertices and the seed vertices (and the seed vertices are ordered in A_{11} conformally to B_{11}), then finding a member of Ψ_R is accomplished by minimizing $\|A_{21} - PB_{21}\|_F$ over all $|W_1| \times |W_1|$ permutation matrices P. Expanding,

$$\begin{aligned} \|A_{21} - PB_{21}\|_F^2 &= \text{trace}(A_{21} - PB_{21})^T(A_{21} - PB_{21}) \\ &= \text{trace}A_{21}^T A_{21} - \text{trace}A_{21}^T PB_{21} - \text{trace}B_{21}^T P^T A_{21} + \text{trace}B_{21}^T P^T PB_{21} \\ &= \|A_{21}\|_F^2 + \|B_{21}\|_F^2 - 2 \cdot \text{trace}\left(P^T(A_{21}B_{21}^T)\right) \quad , \end{aligned} \tag{1}$$

thus finding a member of Ψ_R is accomplished by maximizing $\text{trace}P^T A_{21}B_{21}^T$ over all $|W_1| \times |W_1|$ permutation matrices P. This is a linear assignment problem and can be exactly solved in $O(|W_1|^3)$ time with the Hungarian Algorithm (Edmonds and Karp, 1972; Kuhn, 2006). So, whereas finding a member of Ψ is intractable, finding a member of Ψ_R

can done efficiently. An important question is how well Ψ_R approximates Ψ. Slightly abusing notation, we shall refer to both the restricted-focus graph matching problem and the associated algorithm for exactly solving it by RGM.

2.4 Seeded, Correlated Erdős-Rényi Graphs

Seeded, correlated Erdős-Rényi graphs are correlated Erdős-Rényi graphs G_1 and G_2 where part of the latent alignment function is observed; specifically, there is a subset of seeds $U_1 \subseteq V(G_1)$ such that Φ is known on U_1. If we take ϕ to be the restriction of Φ to U_1 and we run RGM, we may hope that $\Psi_R = \{\Phi\}$; if this hope is true then we are provided an efficient means of computing the latent alignment function.

The next theorem is another of our main results. We will be considering a sequence of random correlated Erdős-Rényi graphs where the number of nonseed vertices is $m = 1$, then $m = 2$, then $m = 3 \ldots$, and the number of seeds s is a function of m.

Theorem 2 *Suppose there exists a fixed real number $\xi_2 > 0$ such that $\xi_2 \leq p \leq 1 - \xi_2$ and $\xi_2 \leq \varrho \leq 1 - \xi_2$. Then there exists fixed real numbers $c_5, c_6 > 0$ (depending only on ξ_2) such that:*
i) If $s \geq c_5 \log m$ then almost always $\Psi_R = \{\Phi\}$, and
ii) If $s \leq c_6 \log m$ then $\lim_{m\to\infty} \mathbb{E}|\{\psi \in \Pi_\phi : \Delta_R(\psi) < \Delta_R(\Phi)\}| = \infty$.

For proof of Theorem 2, see Appendix A.

Note that Theorem 2.i establishes that RGM provides a strongly consistent estimate of the latent alignment in the presence of a logarithmic number of seeds. As noted, a member of Ψ_R is efficiently computable, and thus Theorem 2 (unlike Theorem 1) directly provides a means to efficiently recover the latent alignment bijection Φ, if there are enough seeds.

3. The SGM Algorithm: Extending Frank-Wolfe Methodology for Approximate Graph Matching to Include Seeds

In the setting with no seeds, there are numerous approximate graph matching algorithms in the literature. One such algorithm is the FAQ algorithm of Vogelstein et al. (2011), which is an efficient, state-of-the-art approximate graph matching algorithm based on Frank-Wolfe methodology. The algorithm's performance is empirically shown to be state-of-the-art on many benchmark problems, and when a fixed constant number of Frank-Wolfe iterations are performed, the running time of FAQ is $O(n^3)$, where n is the number of vertices to be matched. Moreover, if $100 \leq |V(G_1)|$ and G_1 is selected with a discrete-uniform distribution (i.e., all possible graphs on $V(G_1)$ are equally likely) and G_2 is an isomorphic copy of G_1 with $V(G_2)$ being a discrete-uniform random permutation of $V(G_1)$, then the probability that FAQ (with, say, 20 Frank-Wolfe iterations allowed) yields the correct isomorphism is empirically observed to be very nearly 1. We choose to focus on the FAQ algorithm here because of its amenability to seeding and because it is the simplest algorithm utilizing the Frank-Wolfe methodology while also achieving excellent performance on many of the QAP benchmark problems; see Vogelstein et al. (2011).

In Section 3.2, we describe the SGM algorithm from Fishkind et al. (2012), which extends the Frank-Wolfe methodology to incorporate utilization of seeds in approximate

seeded graph matching. In Section 3.3 we point out that each Frank-Wolfe iteration in SGM involves optimizing a sum of two terms. Restricting this optimization to just the first of these two terms turns out to be precisely the optimization of RGM from Section 2.3, and restricting this optimization to just the second of these two terms turns out to be precisely the corresponding optimization step of FAQ (i.e., the seeds are completely ignored).

We conclude with simulations and real data experiments that illuminate the relationship between SGM and RGM. SGM can be superior to RGM matching, unsurprising in that SGM makes use of the unseeded adjacency information while RGM does not. Perhaps more surprisingly, we also demonstrate the capacity for RGM to outperform SGM in the presence of very informative seeds; in these case the unseeded connectivity is detrimental to overall algorithmic performance!

3.1 The Frank-Wolfe Algorithm and Frank-Wolfe Methodology

First, a brief review of the Frank-Wolfe algorithm: The general optimization problem that the Frank-Wolfe algorithm is applied to is maximize $f(x)$ such that $x \in S$, where S is a polyhedral set in a Euclidean space, and the function $f : S \to \mathbb{R}$ is continuously differentiable. The Frank-Wolfe algorithm is an iterative procedure. A starting point $x^{(1)} \in S$ is chosen in some fashion, perhaps arbitrarily. For $i = 1, 2, 3, \ldots$, a Frank-Wolfe iteration consists of maximizing the first order (i.e., linear) approximation to f about $x^{(i)}$, that is maximize $f(x^{(i)}) + \nabla f(x^{(i)})^T(x - x^{(i)})$ over $x \in S$, call the solution $y^{(i)}$ (of course, this is equivalent to maximizing $\nabla f(x^{(i)})^T x$ over $x \in S$), then $x^{(i+1)}$ is defined to be the solution to maximize $f(x)$ over x on the line segment from $x^{(i)}$ to $y^{(i)}$. Terminate the Frank-Wolfe algorithm when the the sequence of iterates $x^{(1)}, x^{(2)}, \ldots$ (or their respective objective function values) stops changing much.

Of course, the seeded graph matching problem is a combinatorial optimization problem and, as such, the Frank-Wolfe algorithm cannot be directly applied. The term *Frank-Wolfe methodology* will refer to the approach in which the integer constraints are relaxed so that the domain is a polyhedral set and the Frank-Wolfe algorithm can be directly applied to the relaxation and, at the termination of the Frank-Wolfe algorithm, the fractional solution is projected to the nearest feasible integer point. It is this projected-to point that is adopted as an approximate solution to the original combinatorial optimization problem. We next describe the SGM algorithm, which applies Frank-Wolfe methodology to the Seeded Graph Matching Problem.

3.2 The SGM Algorithm

We now describe the SGM algorithm for approximate seeded graph matching.

Suppose G_1 and G_2 are graphs, say $V(G_1) = \{v_1, v_2, \ldots, v_n\}$ and $V(G_2) = \{v'_1, v'_2, \ldots, v'_n\}$, and let A and B be the respective adjacency matrices of G_1 and G_2. Suppose without loss of generality that $U_1 = \{v_1, v_2, \ldots, v_s\}$ are seeds, and the seeding function $\phi : U_1 \to V(G_2)$ is given by $\phi(v_i) = v'_i$ for all $i = 1, 2, \ldots, s$. Denote the number of nonseed vertices $m := n - s$. Let A and B be partitioned

$$A = \begin{bmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{bmatrix} B = \begin{bmatrix} B_{11} & B_{21}^T \\ B_{21} & B_{22} \end{bmatrix}$$

where A_{11}, $B_{11} \in \{0,1\}^{s\times s}$, A_{22}, $B_{22} \in \{0,1\}^{m\times m}$, and $A_{21}, B_{21} \in \{0,1\}^{m\times s}$.

As mentioned in Section 2.3, the seeded graph matching problem is precisely to minimize $\|A - (I \oplus P)B(I \oplus P)^T\|_F^2 = \|A\|_F^2 + \|B\|_F^2 - 2 \cdot \text{trace}A^T(I \oplus P)B(I \oplus P)^T$ over all m-by-m permutation matrices P. Clearly, the seeded graph matching problem is equivalent to maximizing the quadratic function $\text{trace}A^T(I \oplus P)B(I \oplus P)^T$ over all m-by-m permutation matrices P.

Relax this maximization of $\text{trace}A^T(I \oplus P)B(I \oplus P)^T$ over all m-by-m permutation matrices P to the maximization of $\text{trace}A^T(I \oplus P)B(I \oplus P)^T$ over all m-by-m *doubly stochastic* matrices P (which form a polyhedral set), and then the Frank-Wolfe algorithm can be applied directly to the relaxation. Simplification yields the objective function

$$\begin{aligned} f(P) &= \text{trace}A_{11}B_{11} + \text{trace}A_{21}^T P B_{21} + \text{trace}A_{21}B_{21}^T P^T + \text{trace}A_{22}PB_{22}P^T \\ &= \text{trace}A_{11}B_{11} + 2 \cdot \text{trace}P^T A_{21}B_{21}^T + \text{trace}A_{22}PB_{22}P^T \end{aligned} \quad (2)$$

which has gradient

$$\nabla(P) = 2 \cdot A_{21}B_{21}^T + 2 \cdot A_{22}PB_{22}.$$

We start the Frank-Wolfe algorithm at an arbitrarily selected doubly stochastic m-by-m matrix $P^{(1)}$; for convenience we use the "barycenter" matrix $P^{(1)}$ with all entries equal to $\frac{1}{m}$. Then, for successive $i = 1, 2, \ldots$, the Frank-Wolfe iteration is to maximize the inner product of P with the gradient of f at $P^{(i)}$ over all m-by-m doubly stochastic matrices matrices P; this maximization problem is (ignoring a benign factor of 2) maximizing trace $P^T(A_{21}B_{21}^T + A_{22}P^{(i)}B_{22})$ over m-by-m doubly stochastic matrices. This is a linear assignment problem since the optimal P in this subproblem must be a permutation matrix (by the Birkhoff-von Neumann Theorem which states that the m-by-m doubly stochastic matrices are precisely the convex hull of the m-by-m permutation matrices), and this linear assignment problem can be solved efficiently with the Hungarian Algorithm in $O(m^3)$ time. Say the optimal value of P in this subproblem is $Y^{(i)}$; then, the function f on the line segment from $P^{(i)}$ to $Y^{(i)}$ is a quadratic that is easily maximized exactly, with $P^{(i+1)}$ defined as the doubly stochastic matrix attaining this maximum.

When the Frank-Wolfe iterates $P^{(1)}, P^{(2)}, P^{(3)}, \ldots$ stop changing much (or a constant maximum of iterations are performed—we allowed 20 iterations), then the Frank-Wolfe algorithm terminates; let the resultant approximate solution to the relaxed problem is the doubly stochastic matrix Q. The final step is to project Q to the nearest m-by-m permutation matrix. Minimizing $\|P - Q\|_F$ over permutation matrices P is again a linear assignment problem solvable in $O(m^3)$ time; indeed, minimizing

$$\|P - Q\|_F^2 = \|P\|_F^2 - 2\text{trace}P^T Q + \|Q\|_F^2$$

is equivalent to maximizing trace $P^T Q$ over permutation matrices P. This optimal permutation matrix P is adopted as the approximate solution to the seeded graph matching problem. Specifically, the algorithm output is the bijection $\psi : V(G_1) \to V(G_2)$ where, for $i = 1, 2, \ldots, s$, $\psi(v_i) = v'_i$ and, for each $i = 1, 2, \ldots, m$, $\psi(v_{s+i}) = v'_{s+j}$ for the j such that $P_{ij} = 1$. This Frank-Wolfe Methodology approach described above is called the *SGM algorithm*.

When there are no seeds, the SGM algorithm is exactly the FAQ algorithm of Vogelstein et al. (2011); the above development is a seamless extension of the Frank-Wolfe methodology for approximate graph matching when there are no seeds to Frank-Wolfe methodology for approximate seeded graph matching.

The running time for the SGM algorithm, like for the FAQ algorithm, is $O(n^3)$. This is because of the linear assignment problem formulation and the use of the Hungarian algorithm in each Frank-Wolfe iteration, and is a huge savings over using the simplex method or an interior point method for solving the linearizations in each Frank-Wolfe iteration. This trick has made Frank-Wolfe methodology a very potent weapon for efficient approximate graph matching.

3.3 Frank-Wolfe Methodology for Approximate Seeded Graph Matching Inherently Includes RGM

In each Frank-Wolfe iteration (described in Section 3.2), the linearization which is solved is maximize (trace $P^T A_{21} B_{21}^T$ +trace$P^T A_{22} P^{(i)} B_{22}$) over all m-by-m permutation matrices P. Observe that if this maximization were just over the first term trace $P^T A_{21} B_{21}^T$ then it would be precisely solving RGM from Section 2.3, as per Equation (1) there. Also observe that if the maximization were just over the second term trace$P^T A_{22} P^{(i)} B_{22}$, then it would be exactly the FAQ algorithm (ignoring all of the seeds). In this manner, the SGM algorithm can be seen as leveraging a combination of the information gleaned from the nonseed-seed relationships (the "restricted-focus term") and the nonseed-nonseed relationships (the "FAQ term").

Although performing RGM is much simpler than performing SGM, and although RGM almost always produces the correct graph alignment if there are enough seeds, nonetheless SGM may perform substantially better when there aren't enough seeds. Indeed, as noted, SGM merges RGM with FAQ, and thus utilizes the information contained in the unseeded adjacency structure. While FAQ alone is often unable to extract out this information (see Figure 1 below), the RGM term can steer the FAQ term in SGM, allowing it to extract the relevant signal in the nonseed–to–nonseed adjacency structure.

The utility of this nonseeded term depends on the amount of information captured in the seed–to–nonseed adjacency. With less informative seeds, the SGM algorithm often significantly outperforms RGM alone, as there is important signal in the unseeded adjacency which RGM discards. However, in the presence of well chosen seeds, the seed–to–nonseed adjacency structure may contain all the relevant signal about the unknown alignment, and the unseeded adjacency information can be a nuisance (see Figure 4). As the RGM algorithm is exactly and efficiently solvable, this points to the centrality of both selecting and quantifying "good" seeds. This is a direction of future research, as we do not address the problem of intelligent seed selection at present.

As we will see in Figure 1, for weakly correlated graphs, RGM can outperform SGM. Even with poorly-chosen seeds, the noise in the nonseed–to–nonseed adjacency structure can outweigh the relevant signal, and the performance of SGM is harmed by including this extra nuisance information. This further highlights the utility of RGM in real data applications, where the correlation between graphs can be low.

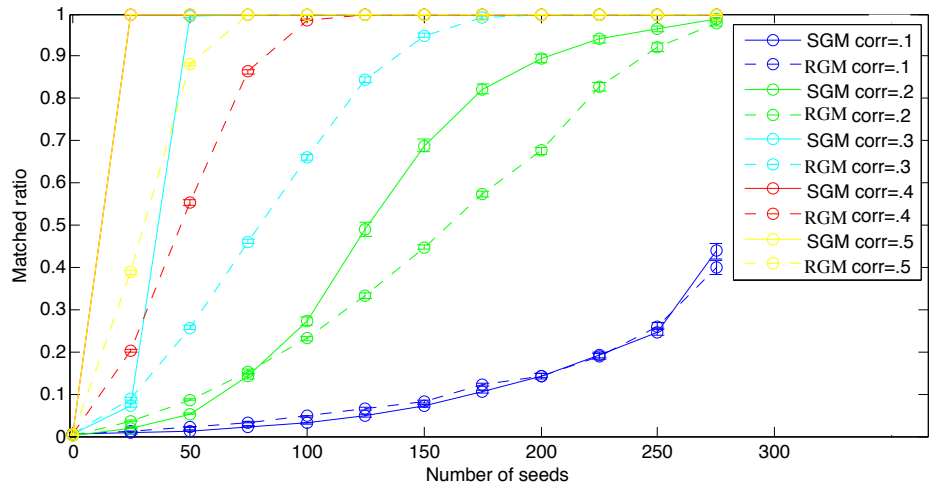
We explore the above further in Figure 1. There we compare the performance of SGM against solving RGM for correlated Erdős-Rényi graphs with $n = 300$ vertices, $p = 0.5$, seeding levels ranging from $s = 0$ to 275, and correlation ranging from $\varrho = 0.1$ to 1. For each value of ϱ and s, we ran 100 simulations and plotted the fraction of nonseeded vertices correctly matched across the graphs, with corresponding error bars of ± 2 s.e. In all cases (except $\rho = 0.1$), RGM needed more seeds to perform comparably to SGM. Indeed, with sufficiently many seeds, all available information about the unknown alignment is captured in the seed–to–nonseed connectivity, and the (exactly solvable) RGM algorithm alone is enough to properly align the graphs.

Also note the following from Figure 1. When there are no seeds, we see FAQ (which is SGM in the absence of seeds) working perfectly at capturing the latent alignment function when the two graphs are isomorphic (it bears noting that we have also observed FAQ perfectly matching when the two graphs are not isomorphic but rather *very* highly correlated), but FAQ does a surprisingly poor job (indeed, comparable to chance) when the correlation is even modestly less than one. However, with seeds, SGM quickly does a very substantially better job; indeed, the "restricted-focus" term is steering the SGM algorithm in the proper direction!

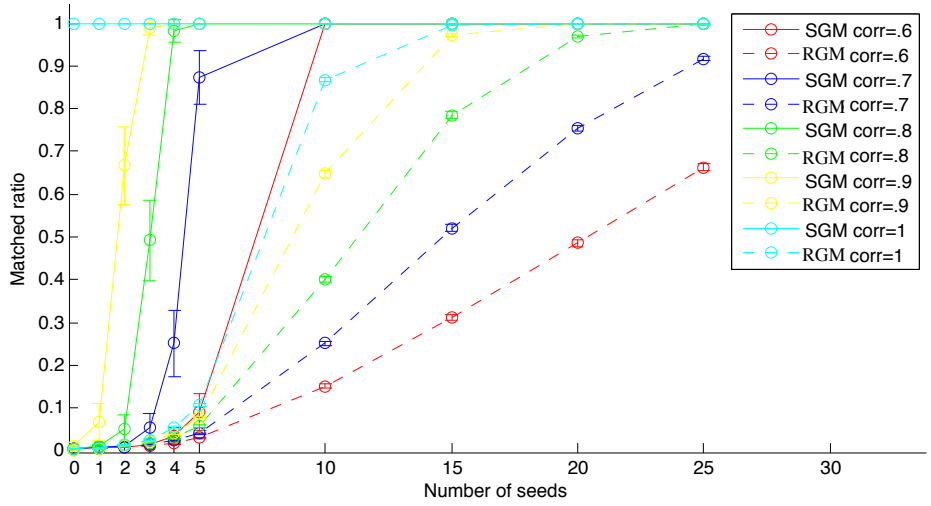
4. Matching Human Connectomes

We further illuminate the relationship between SGM and RGM through a real data experiment, which will serve to highlight both the utility of RGM and the effect of SGM's further incorporation of the unseeded adjacency information. Our data set consists of 45 graphs, each on 70 vertices, these graphs constructed respectively from diffusion tensor (DT) MRI scans of 45 distinct healthy patients. We have 21 scans from the Kennedy Krieger Institute (KKI), with raw data available at `http://www.nitrc.org/projects/multimodal/`, and 24 scans from the Nathan Kline Institute (NKI), with a description of the raw data available at `http://fcon_1000.projects.nitrc.org/indi/pro/eNKI_RS_TRT/FrontPage.html`. All raw scans were registered to a common template and identically processed with the MIGRAINE pipeline of Gray et al. (2012), each yielding a weighted, symmetric graph on 70 vertices. (All graphs can be found at `http://openconnecto.me/data/public/.MR/MIGRAINE/`). Vertices in the graphs correspond to regions in the Desikan brain atlas, with edge weights counting the number of neural fiber bundles connecting the regions (note that although the theory and algorithms presented earlier were for simple graphs, they are easily modified to handle edge weights). In addition to shedding light on the relationship between SGM and RGM, we also explore the batch effect induced by the different medical centers and demonstrate the capacity for seeding to potentially ameliorate this batch effect.

The pipeline which processes the scans into graphs first registers each of the graphs to a common template. As a result, there is a canonical alignment between the vertex sets of these graphs (vertices corresponding to respective regions in the Desikan brain atlas). How well is this alignment preserved *across* medical centers by the adjacency structure of the graphs alone? Figure 2 explores this question, and presents strong evidence for the existence of a batch effect (in both adjacency and geometric structure) induced by the different medical centers. In the figure, the heat map labeled "KKI matched to KKI" represents a 70×70 matrix, whose i, jth entry measures the relative number of times vertex



(a)



(b)

Figure 1: Fraction of vertices correctly matched for the SGM algorithm and for RGM, plotted versus the number of seeds utilized, for $n = 300$, $p = 1/2$ and correlation ϱ varying from 0.1 to 1. For each value of ϱ and s, we ran 100 simulations and plotted the fraction of nonseeded vertices correctly matched across the graphs, with corresponding error bars of ± 2 s.e.

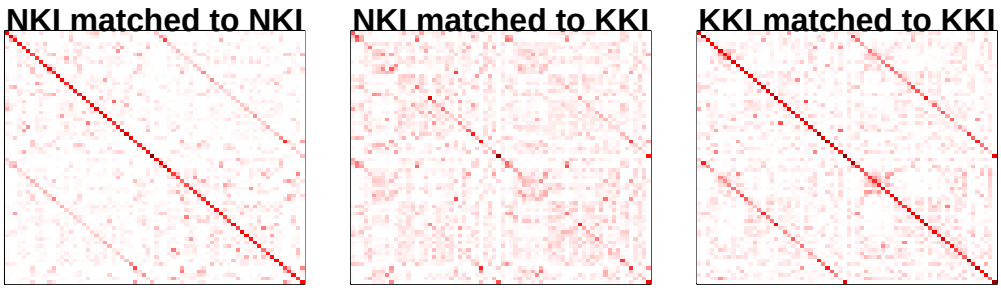


Figure 2: Left: NKI to NKI matching. Center: NKI to KKI matching. Right: KKI to KKI matching. Each plot is a 70×70 heat map with the color intensity (from white to red) representing the relative number of times vertex i was match with vertex j across the experiments (white denoting no matches, dark red denoting many matches). The dark red diagonal in the left and right heat maps (as compared to the center map), indicates presence of a substantial batch effect, i.e., the correct alignment was recovered significantly better matching within medical center versus across medical center. Vertices 1–35 and 36–70 (as ordered) correspond to the respective brain hemispheres.

i was mapped to vertex j when we ran the FAQ algorithm (i.e., no seeds) over the $\binom{21}{2}$ pairs of graphs from the KKI data set. Similarly, the "NKI to KKI" heat map counts the relative number of times vertices were matched to each other when running the FAQ algorithm over the $21 \cdot 24$ pairs of graphs, with one graph from each of the KKI and NKI data sets. The "NKI matched to NKI" heat map is defined similarly. The chromatic intensity of the pixel in the i, jth entry of each heat map represents the relative frequency in which vertex i was matched to vertex j across the experiments, with darker red implying more frequent and lighter red implying less frequent. White pixels represent vertex pairs that were never matched.

Figure 2 demonstrates the existence of significant signal in the adjacency structure alone (without the associated brain geometry and without seeding) for recovering the latent alignment in all three experiments. When matching KKI to KKI, 32.8% of the vertices are correctly matched on average; when matching NKI to NKI, 37.4% of the vertices were correctly matched on average; when matching NKI to KKI, 9.8% of the vertices were correctly matched on average (whereas chance would have matched $\approx 1.4\%$ on average). We note that the dramatic performance difference when matching within versus across medical centers is strong evidence of the presence of a batch effect induced by the different medical centers. Whether this batch effect is an artifact of experimental differences across medical centers (different MRI machines, different technicians, etc.) or the registration pipeline, it must be addressed before the data sets can be aggregated for use in further inference.

Also note that while much of the within medical center matching error was due to mismatching brain hemispheres (vertices 1–35 representing one hemisphere, and vertices 36–70 the other), the mismatch across medical centers appears significantly less structured.

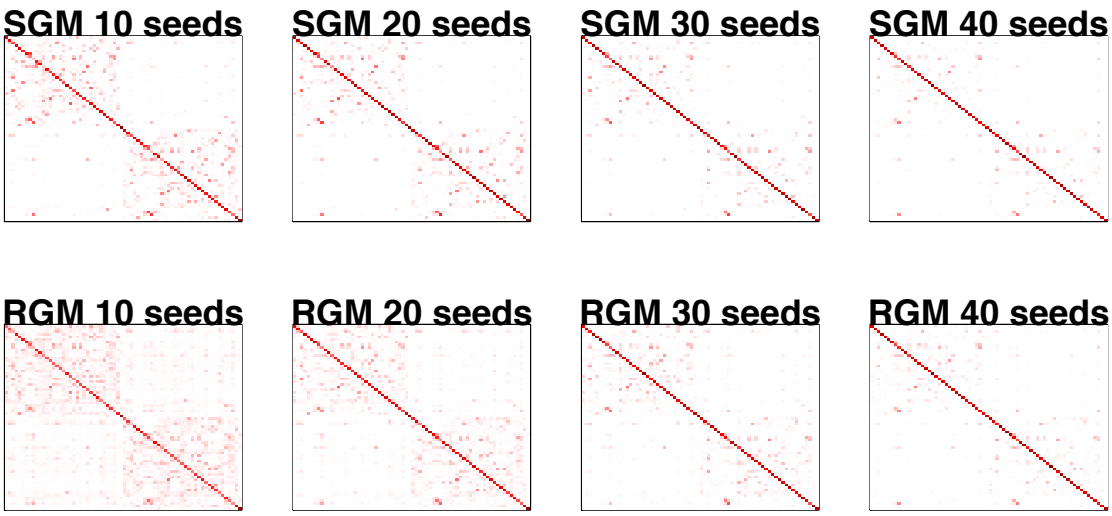


Figure 3: Clockwise from top left: SGM matching the $21 \cdot 24$ pairs of brains, one each from the NKI and KKI data sets, using 10, 20, 30, 40 seeds; RGM matching the same set of graphs using 40, 30, 20, 10 seeds. For each seed level, and each method we ran 100 paired MC replicates. Each plot is a 70×70 heat map with the color intensity (from white to red) representing the relative number of times vertex i was matched with vertex j across experiments (white denoting no matches, dark red denoting many matches). We do not count seeded vertices as being correctly matched to each other, which would have artificially inflated the diagonal.

Can we use seeding to ameliorate this batch effect? In Figure 1, we established the capacity of seeded vertices to unearth significant signal in the adjacency structure for recovering the latent alignment function, signal which was not found without seeds. Figure 3 further demonstrates this phenomenon in our present real data setting. We plot heat maps showing the $21 \cdot 24$ matchings of pairs of graphs, one each from the NKI and KKI data sets, for various seed levels. For each number of seeds$= 10, 20, 30, 40$, we ran 100 Monte Carlo replicates (for each of SGM and RGM) for each pair of matched graphs, with each seed set chosen uniformly at random from the 70 vertices. Clockwise, from the top left, we plot the performance of SGM with 10, 20, 30, and 40 seeds and then the performance of RGM with 40, 30, 20, and 10 seeds. The chromatic intensity of the pixel in the i, jth entry of each heat map represents the relative frequency in which vertex i was matched to vertex j across the experiments (seeded vertices are not counted as correctly matched here), with darker red implying more frequent and lighter red implying less frequent.

The figure conclusively demonstrates that seeding extracts statistically significant signal in the adjacency structure alone for correctly aligning graphs across medical center, signal that was effectively obfuscated in the absence of seeds. While unseeded FAQ correctly matched 9.8% of the vertices on average across medical centers, with $10, 20, 30, 40$ seeds,

SGM (RGM) correctly matches 49.9%,68.4%,78.8%, 85.1% (29.9%, 53.8%, 70.7%, 80.9%) of the unmatched vertices on average across medical centers. We also see that SGM outperforms RGM across all seed levels, with RGM requiring more seeds to achieve the same performance as SGM. This is not surprising, as RGM is not utilizing any of the adjacency information amongst the unseeded vertices.

We also see that seeding teases out additional information on the neural geometry inherent to the graphs. For instance, with only 10 seeds, 4.3% (15.1%) vertices on average are mismatched across hemispheres by SGM (RGM). In contrast, 43.8% vertices on average were mismatched across hemispheres without seeds. Interestingly, some vertex pairs are consistently mismatched across all seed levels. For example, vertex 57 is matched by SGM to vertex 47 across medical centers 23.8%, 20.8%, 23.1%, 23.5% of the time with $10, 20, 30, 40$ seeds, whereas, with no seeds, vertex 57 is matched to vertex 47 on average 10.9% of the time when matching among the NKI data set and 10% of the time when matching amongst the KKI graphs. Indeed, these persistent artifacts are indicative of substantive differences across (and within) data sets and demand further investigation.

We have noted that, on average, SGM outperformed RGM across all seed levels. How much of this performance gap is a function of the particular seeds chosen? We explore this further in Figure 4. For a pair of graphs, one each from the NKI and KKI data sets (we randomly chose graph 2 in the NKI data set and graph 7 in the KKI set—note that we see similar patterns across all tested graph pairs), we ran 200 Monte Carlo replicates of SGM and RGM seeded with the same randomly selected seeds. For each of seeds= $10, 20, 30, 40, 50$ (chosen uniformly at random from the vertices), the associated histogram plots the 200 values of the number of vertices correctly matched by SGM minus the number of vertices correctly matched by RGM.

The RGM algorithm ignores all the adjacency information amongst the unseeded vertices. If, in Figure 4, SGM performed uniformly better than RGM at each seed level, then there is consistently relevant signal in the unseeded adjacency structure, and we should never use RGM when SGM is feasibly run. However, we see that there are choices of seeds (at every level) for which RGM outperforms SGM. The unseeded adjacency information is a nuisance in these cases. As RGM is efficiently exactly solvable, this dramatically highlights the importance of intelligent seeding. Indeed, "good" seeds (and hence the RGM algorithm) have the potential to capture all of the relevant adjacency structure in the graph. While we do not pursue the question of *how* to select "good" seeds here, the figure points to the centrality of this question, and we plan on pursuing active seed selection in future work.

For higher seed levels, we note that there is significantly less difference (and less variability) in the performance of SGM and RGM. More seeds capture more information in their neighborhood structure, and the effect of the unseeded adjacency on algorithm performance is dampened. Also, at higher seed levels the particular choice of seeds is less important, as any selection of a large number of seeds will probably contain enough "good" seeds to strongly align the graphs.

5. Discussion

Estimating the latent alignment between the vertices of two graphs is an important problem in many disciplines, and our results have both theoretical and practical implications

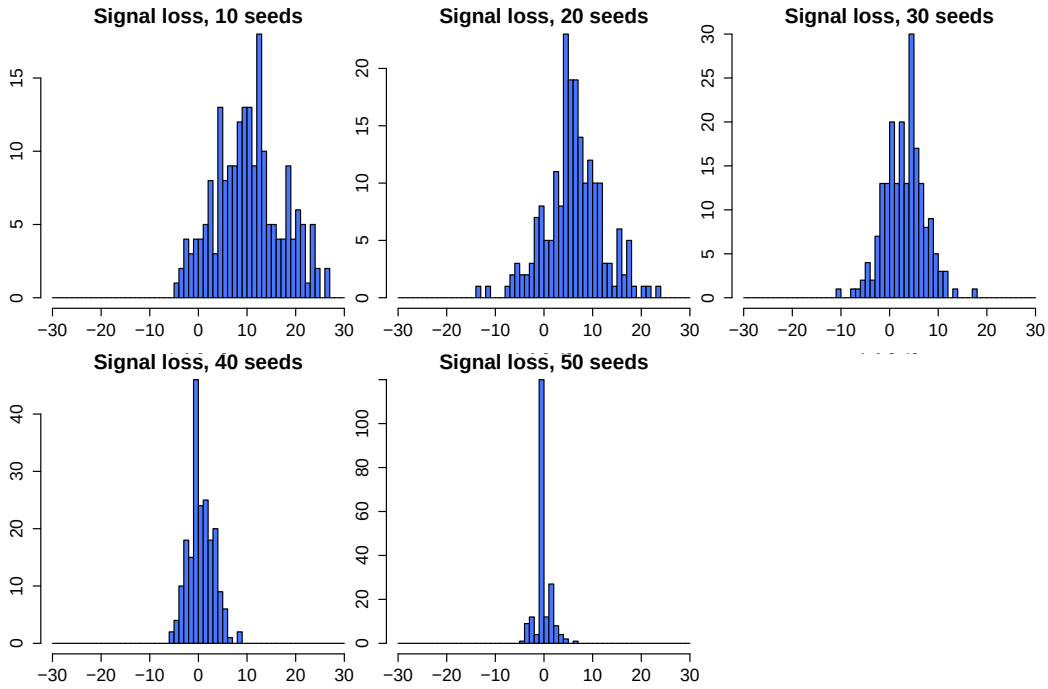


Figure 4: RGM versus SGM when matching one graph each from the NKI (graph 2) and KKI (graph 7) data sets over differing seed levels. For each of seeds= $10, 20, 30, 40, 50$ (chosen uniformly at random from the vertices), each histogram above plots 200 values of the number of vertices correctly matched by SGM minus the number of vertices correctly matched by RGM utilizing the same random seeds.

for this problem. Indeed, under mild assumptions, we proved the strong consistency of the graph matching problem—and its restricted focus subproblem—for estimating the latent alignment function between the vertex sets of two correlated Erdős-Rényi graphs. Although seeded graph matching is computationally hard, this result gives hope that efficient approximation algorithms will be effective in recovering the latent alignment across a broad array of graphs.

Embedded in the hard seeded graph matching problem is the tractable restricted-focus graph matching problem. This problem is exactly solvable and also provides a strongly consistent estimator of the latent alignment. While full seeded graph matching often out performs this restricted focus variant, we demonstrated the capacity for the restricted-focus subproblem to also outperform the full matching. The relation between the two approaches hinges on the information contained in the seeded vertices. If the seeds capture the adjacency structure of the graph, then the restricted-focus subproblem can benefit by *not* including the unseeded adjacency information, and we demonstrate this phenomenon in both real and simulation data. This points to the primacy of intelligently seeding in graph matching, and we are working on active seeding algorithms for choosing good seeded vertices.

Even when outperformed by the full matching problem, we can still use the restricted-focus problem to extract signal in the graphs that was obfuscated without seeding. In very large, complex problems, when it may be infeasible to run the full seeded graph matching algorithm, the restricted-focus approach could be run to provide a baseline matching between the graphs. We are presently investigating this further, as scalability of these approaches is an increasingly important demand of modern big-data.

Acknowledgments

This work is partially supported by a National Security Science and Engineering Faculty Fellowship (NSSEFF), Johns Hopkins University Human Language Technology Center of Excellence (JHU HLT COE), and the XDATA program of the Defense Advanced Research Projects Agency (DARPA) administered through Air Force Research Laboratory contract FA8750-12-2-0303. The line of research here was suggested by Dr. Richard Cox, director of the JHU HLT COE. We would like to thank Joshua Vogelstein and William Gray Roncal for providing us with the data for Section 5, and Daniel Sussman for his insightful suggestions and comments throughout. Lastly, we would like to thank the anonymous referees whose comments greatly improved the present draft.

Appendix A. Proofs of Theorems 1 and 2

Theorem 1 is proved in Sections A.2, A.3, and A.4, and these three subsections are a continuation one of the other. Theorem 2 is proved in Sections A.5, A.6, and A.7 and these three subsections are a continuation one of the other. Interestingly, the underlying methodology for proving Theorem 1 is very similar (but with notable differences) to the methodology for proving Theorem 2. We begin with some results that will subsequently be used in the proof of Theorems 1 and 2.

A.1 Supporting Results

The next result, Theorem 3, is from Alon et al. (1997), in the form found in Kim et al. (2002).

Theorem 3 *Suppose random variable X is a function of η independent Bernoulli(q) random variables such that changing the value of any one of the Bernoulli random variables changes the value of X by at most 2. For any $t : 0 \leq t < \sqrt{\eta q(1-q)}$, we have* $\mathbb{P}\left[|X - \mathbb{E}X| > 4t\sqrt{\eta q(1-q)}\right] \leq 2e^{-t^2}$.

The next result, Theorem 4, is a Chernoff-Hoeffding bound which is Theorem 3.2 in Chung and Lu (2006).

Theorem 4 *Suppose X has a Binomial(η, q) distribution. Then for all $t \geq 0$ it holds that*

$$\mathbb{P}\left[X - \mathbb{E}X \geq t\right] \leq exp\left\{\frac{-t^2}{2\eta q + 2t/3}\right\}.$$

For any $r, q \in (0,1)$, define $H(r,q) := r \log\left(\frac{r}{q}\right) + (1-r)\log\left(\frac{1-r}{1-q}\right)$. This is the Kullback-Leibler divergence between binomial random variables with respective success probabilities r and q. We will later use the following rough lower bound estimate of a binomial tail probability:

Proposition 5 *Suppose X has a Binomial(η, q) distribution, and suppose that $0 < q < r < 1 - \frac{1}{\eta}$ for a real number r. Then*

$$\mathbb{P}(X \geq \eta r) \geq \frac{\sqrt{\pi}}{e^3} \cdot \sqrt{\frac{(1-r)}{r}} \eta^{-1/2} q \cdot e^{-\eta H(r,q)}.$$

Proof We compute and bound

$$\begin{aligned}
\mathbb{P}(X \geq \eta r) &\geq \mathbb{P}(X = \lceil \eta r \rceil) = \binom{\eta}{\lceil \eta r \rceil} q^{\lceil \eta r \rceil} (1-q)^{\eta - \lceil \eta r \rceil} \\
&\geq \frac{\sqrt{2\pi}}{e^2} q^{\lceil \eta r \rceil} (1-q)^{\eta - \lceil \eta r \rceil} \frac{\eta^{\eta + 0.5}}{\lceil \eta r \rceil^{\lceil \eta r \rceil + 0.5} (\eta - \lceil \eta r \rceil)^{\eta - \lceil \eta r \rceil + 0.5}} \\
&= \frac{\sqrt{2\pi}}{e^2} q^{\lceil \eta r \rceil} (1-q)^{\eta - \lceil \eta r \rceil} \frac{\eta^{\eta + 0.5}}{(\eta r)^{\eta r + 0.5} (\eta - \eta r)^{\eta - \eta r + 0.5}} \cdot \frac{(\eta r)^{\eta r + 0.5} (\eta - \eta r)^{\eta - \eta r + 0.5}}{\lceil \eta r \rceil^{\lceil \eta r \rceil + 0.5} (\eta - \lceil \eta r \rceil)^{\eta - \lceil \eta r \rceil + 0.5}},
\end{aligned}$$

where the inequality in the second display line follows from Stirling's formula. Now,

$$\begin{aligned}
\frac{(\eta r)^{\eta r + 0.5} (\eta - \eta r)^{\eta - \eta r + 0.5}}{\lceil \eta r \rceil^{\lceil \eta r \rceil + 0.5} (\eta - \lceil \eta r \rceil)^{\eta - \lceil \eta r \rceil + 0.5}} &= \frac{(r)^{\eta r + 0.5} (1-r)^{\eta - \eta r + 0.5}}{\left(\frac{\lceil \eta r \rceil}{\eta}\right)^{\lceil \eta r \rceil + 0.5} \left(1 - \frac{\lceil \eta r \rceil}{\eta}\right)^{\eta - \lceil \eta r \rceil + 0.5}} \\
&\geq \left(\frac{1}{\frac{\lceil \eta r \rceil}{\eta r}}\right)^{\eta r + 0.5} (1-r)^{\lceil \eta r \rceil - \eta r} \\
&\geq \left(\frac{1}{1 + \frac{1}{\eta r}}\right)^{\eta r + 0.5} (1-r) \quad \geq \quad \frac{1}{e\sqrt{2}}(1-r).
\end{aligned}$$

Combining the above, we obtain

$$\begin{aligned}
\mathbb{P}(X \geq \eta r) &\geq \frac{\sqrt{\pi}}{e^3} \cdot \frac{1-r}{r^{1/2}(1-r)^{1/2}} \eta^{-1/2} q^{\eta r + 1} (1-q)^{\eta - \eta r} \frac{\eta^{\eta}}{(\eta r)^{\eta r} (\eta - \eta r)^{\eta - \eta r}} \\
&= \frac{\sqrt{\pi}}{e^3} \cdot \sqrt{\frac{1-r}{r}} \eta^{-1/2} q \cdot e^{-\eta H(r,q)},
\end{aligned}$$

as desired. ∎

A.2 Overall Argument of the Proof for Theorem 1, Part i

It is notationally convenient to assume without loss of generality that the correlated Erdős-Rényi graphs G_1 and G_2 are on the same set of n vertices V and we do **not** relabel the

vertices. Let Π denote the set of bijections $V \to V$; here, the identity function $e \in \Pi$ is the latent alignment bijection Φ. For any $\psi \in \Pi$,

$$\Delta^+(G_1, G_2, \psi) := |\left\{ \{v, v'\} \in \binom{V}{2} \text{ s.t. } \{v, v'\} \notin E(G_1) \text{ and } \{\psi(v), \psi(v')\} \in E(G_2) \right\}|,$$

$$\Delta^-(G_1, G_2, \psi) := |\left\{ \{v, v'\} \in \binom{V}{2} \text{ s.t. } \{v, v'\} \in E(G_1) \text{ and } \{\psi(v), \psi(v')\} \notin E(G_2) \right\}|,$$

$$\Delta^{0+}(G_1, G_2, \psi) := |\left\{ \{v, v'\} \in \binom{V}{2} \text{ s.t. } \{v, v'\} \notin E(G_1),\ \{\psi(v), \psi(v')\} \in E(G_1), \right.$$
$$\left. \{\psi(v), \psi(v')\} \notin E(G_2) \right\}|,$$

$$\Delta^{0-}(G_1, G_2, \psi) := |\left\{ \{v, v'\} \in \binom{V}{2} \text{ s.t. } \{v, v'\} \in E(G_1),\ \{\psi(v), \psi(v')\} \notin E(G_1), \right.$$
$$\left. \{\psi(v), \psi(v')\} \in E(G_2) \right\}|,$$

$$\Delta(G_1, G_2, \psi) := \Delta^+(G_1, G_2, \psi) + \Delta^-(G_1, G_2, \psi).$$

First, note that

$$\Delta^+(G_1, G_1, \psi) = \Delta^-(G_1, G_1, \psi) = \frac{1}{2}\Delta(G_1, G_1, \psi) \ ; \tag{3}$$

this is because the number of edges in G_1 isn't changed when its vertices are permuted by ψ.

Next, note that

$$\Delta(G_1, G_2, \psi) - \Delta(G_1, G_2, e) = \Delta(G_1, G_1, \psi) - 2 \cdot \Delta^{0+}(G_1, G_2, \psi) - 2 \cdot \Delta^{0-}(G_1, G_2, \psi) \ ; \tag{4}$$

this is easily verified by replacing "G_2" in (4) by "G", and observing the truth of (4) as G, starting out with $G = G_1$, is changed one edge-flip at a time until $G = G_2$.

Now, consider the event, which we shall call Υ, that for all $\psi \in \Pi \backslash \{e\}$,

$$\Delta^{0+}(G_1, G_2, \psi) \ < \ \Delta^+(G_1, G_1, \psi) \cdot \left((1-p)(1-\varrho) + \frac{\varrho}{2} \right) \text{ and also} \tag{5}$$

$$\Delta^{0-}(G_1, G_2, \psi) \ < \ \Delta^-(G_1, G_1, \psi) \cdot \left(p(1-\varrho) + \frac{\varrho}{2} \right). \tag{6}$$

We will next show in Section A.3 that, under the hypotheses of the first part of Theorem 1, Υ almost always happens (in other words, with probability 1, Υ happens for all but a finite numbers of n's). Then, adding (5) to (6) and using (3), we then obtain that almost always $\Delta^{0+}(G_1, G_2, \psi) + \Delta^{0-}(G_1, G_2, \psi) < \frac{1}{2} \cdot \Delta(G_1, G_1, \psi)$ for all $\psi \in \Pi \backslash \{e\}$. Substituting this into (4) yields that almost always $\Delta(G_1, G_2, \psi) > \Delta(G_1, G_2, e)$ for all $\psi \in \Pi \backslash \{e\}$, and the first part of Theorem 1 is then proven.

A.3 Under Hypotheses of Theorem 1, Part i, Υ Occurs Almost Always

For any $k \in \{1, 2, \ldots, n\}$, let $\Pi(k)$ denote the set of bijections in Π such that the number of non-fixed-points of the bijection is exactly k; that is, $\Pi(k) := \{\psi \in \Pi : |\{v \in V : \psi(v) \neq$

$v\}| = k\}$. A simple upper bound for $|\Pi(k)|$ is $|\Pi(k)| \leq \binom{n}{k}k! = n(n-1)(n-2)\cdots(n-k+1) \leq n^k$.

Just for now, let $k \in \{1, 2, \ldots, n\}$ be chosen, and let $\psi \in \Pi(k)$ be chosen. Denoting $T(\psi) := \left\{\{v, v'\} \in \binom{V}{2} \text{ such that } v = \psi(v'),\ v' = \psi(v)\}\right\}$, we have that the random variable $\Delta(G_1, G_1, \psi)$ is a function of the $\eta := \binom{k}{2} + (n-k)k - |T(\psi)|$ independent Bernoulli(p) random variables

$$\{\mathbb{1}\{\{v, v'\} \in E(G_1)\}\}_{\{v,v'\}\in\binom{V}{2}\setminus T(\psi)\ :\ \psi(v)\neq v \text{ or } \psi(v')\neq v'}$$

and note that the hypotheses of Theorem 3 are satisfied, hence for the choice of $t = \frac{1}{20}\sqrt{\eta p(1-p)}$ in Theorem 3 we obtain that

$$\mathbb{P}\left[|\Delta(G_1, G_1, \psi) - \mathbb{E}\Delta(G_1, G_1, \psi)| > \frac{1}{5}\eta p(1-p)\right] \leq 2e^{-\eta p(1-p)/400}. \tag{7}$$

Also note that

$$\Delta(G_1, G_1, \psi) = \sum_{\substack{\{v,v'\}\in\binom{V}{2}\setminus T(\psi)\\ \text{s.t. } \psi(v)\neq v \text{ or } \psi(v')\neq v'}} \mathbb{1}\left\{\mathbb{1}\{\{v, v'\} \in E(G_1)\} \neq \mathbb{1}\{\{\psi(v), \psi(v')\} \in E(G_1)\}\right\}$$

is the sum of η Bernoulli($2p(1-p)$) random variables hence

$$\mathbb{E}\Delta(G_1, G_1, \psi) = 2\eta p(1-p). \tag{8}$$

Because $|T(\psi)| \leq \frac{k}{2}$, we have by elementary algebra that $\frac{(n-2)k}{2} \leq \eta \leq nk$. Thus, by (7) and (8) we obtain that (for large enough n; in the following our constants are very conservatively chosen)

$$\mathbb{P}\left(\frac{\Delta(G_1, G_1, \psi)}{nkp(1-p)} \notin [1/2,\ 5/2]\right) \leq 2e^{\frac{-1}{1000}nkp(1-p)} \leq 2e^{\frac{-(1-\xi_1)}{1000}nkp}. \tag{9}$$

Conditioning on G_1, random variable $\Delta^{0+}(G_1, G_2, \psi)$ has a

$$\text{Binomial}\left(\Delta^{+}(G_1, G_1, \psi), (1-p)(1-\varrho)\right)$$

distribution, and random variable $\Delta^{0-}(G_1, G_2, \psi)$ has a

$$\text{Binomial}\left(\Delta^{-}(G_1, G_1, \psi), p(1-\varrho)\right)$$

distribution. Conditioning also on the event that $\frac{\Delta(G_1,G_1,\psi)}{nkp(1-p)} \in [1/2,\ 5/2]$, we apply Theorem 4 with the value $t = \frac{\varrho}{2}\cdot\Delta^{+}(G_1, G_1, \psi)$, and we use (3) to show

$$\mathbb{P}\left[\Delta^{0+}(G_1, G_2, \psi) \geq \Delta^{+}(G_1, G_1, \psi)\cdot\left((1-p)(1-\varrho) + \frac{\varrho}{2}\right)\right] \leq e^{\frac{-(1-\xi_1)}{40}nkp\varrho^2}, \tag{10}$$

$$\mathbb{P}\left[\Delta^{0-}(G_1, G_2, \psi) \geq \Delta^{-}(G_1, G_1, \psi)\cdot\left(p(1-\varrho) + \frac{\varrho}{2}\right)\right] \leq e^{\frac{-(1-\xi_1)}{40}nkp\varrho^2}. \tag{11}$$

Finally, applying (9), (10) and (11), the probability of Υ^C can be bounded using sub-additivity:

$$\begin{aligned}\mathbb{P}(\Upsilon^C) &\leq \sum_{k=1}^{n}\sum_{\psi\in\Pi(k)}\left(2e^{\frac{-(1-\xi_1)}{1000}nkp}+e^{\frac{-(1-\xi_1)}{40}nkp\varrho^2}+e^{\frac{-(1-\xi_1)}{40}nkp\varrho^2}\right)\\ &\leq \sum_{k=1}^{n}n^k\left(2e^{\frac{-(1-\xi_1)}{1000}nkp}+2e^{\frac{-(1-\xi_1)}{40}nkp\varrho^2}\right)\\ &\leq \sum_{k=1}^{n}\left(2e^{\frac{-(1-\xi_1)}{1000}nkp+k\log n}+2e^{\frac{-(1-\xi_1)}{40}nkp\varrho^2+k\log n}\right)\leq n\cdot\frac{4}{n^3},\end{aligned}$$

the last inequality holding if $p\geq c_2\frac{\log n}{n}$ and $\varrho\geq c_1\sqrt{\frac{\log n}{np}}$ for sufficiently large, for fixed constants c_1,c_2. Because $\sum_{n=1}^{\infty}\frac{4}{n^2}<\infty$, we have by the Borel-Cantelli Lemma that Υ almost always happens. As mentioned in Section A.2, this completes the proof of the first part of Theorem 1.

Remark 6 Note that we could tighten the constants c_1 and c_2 appearing above. Here we choose not to, instead focusing on the orders of magnitude of ϱ, and do not pursue exact constants further.

A.4 Proof of Theorem 1, Part ii

We now prove the second part of Theorem 1.

Just for now, let $\psi\in\Pi(n)$ be chosen (i.e., ψ is a derangement), and condition on $\Delta^+(G_1,G_1,\psi)=\Delta$, where $\frac{1}{4}n^2p(1-p)\leq\Delta\leq\frac{5}{4}n^2p(1-p)$. The random variables $\Delta^{0+}(G_1,G_2,\psi)$ and $\Delta^{0-}(G_1,G_2,\psi)$ are independent, and have distributions Binomial(Δ,q_1) and Binomial(Δ,q_2), respectively, where $q_1:=(1-p)(1-\varrho)$ and $q_2:=p(1-\varrho)$.

Denoting $r_1:=q_1+\frac{\varrho}{2}$ and $r_2:=q_2+\frac{\varrho}{2}$, and observing that, under the hypotheses of Theorem 2, part ii, it holds that $r_1<1-\frac{1}{\Delta}$ and $r_2<1-\frac{1}{\Delta}$ we thus have by Proposition 5 that (as $\frac{\pi}{e^6}>\frac{1}{200}$)

$$\begin{aligned}&\mathbb{P}\left(\Delta^{0+}(G_1,G_2,\psi)\geq\Delta\cdot r_1\ \text{ and }\ \Delta^{0-}(G_1,G_2,\psi)\geq\Delta\cdot r_2\right)\\ &\qquad\geq\frac{q_1q_2}{200\Delta}\sqrt{\frac{(1-r_1)(1-r_2)}{r_1r_2}}e^{-\Delta\cdot H(r_1,q_1)-\Delta\cdot H(r_2,q_2)}\end{aligned}$$

Note that we can change the inequalities "$\geq$" in the expression $\mathbb{P}(\quad)$ above into strict inequalities "$>$" with a harmless tweak. An elementary calculus argument yields that $H(x+y,y)\leq x^2/(y-y^2)$ for all $0<y<1$ and $x\geq 0$ such that $y+2x<1$. Indeed, fixing any value for y, the function value and the derivative of $H(x+y,y)$ with respect to x are both 0 at $x=0$, the function value and the derivative of $x^2/(y-y^2)$ with respect to x are both 0 at $x=0$, and the second derivative of $H(x+y,y)$ is less than the second derivative of $x^2/(y-y^2)$ for all relevant x. This, together with the fact that $1-r_1=r_2$, $1-r_2=r_1$

and assuming that ϱ is bounded away from 1 (which, indeed, will turn out to be assumed), we have that there exists a real number $c > 0$ such that

$$\begin{aligned}\mathbb{P}\Bigg(\Delta^{0+}(G_1, G_2, \psi) > \Delta \cdot r_1 \text{ and } \Delta^{0-}(G_1, G_2, \psi) > \Delta \cdot r_2\Bigg) \\ &\geq \frac{q_1 q_2}{200\Delta} e^{-\varrho^2 \Delta\left(\frac{1}{4q_1(1-q_1)} + \frac{1}{4q_2(1-q_2)}\right)} \\ &\geq \frac{c}{n^2} \cdot e^{-\varrho^2 n^2 p \cdot \left(\frac{1}{c \cdot p}\right)} \\ &= \frac{c}{n^2} \cdot e^{\frac{-\varrho^2 n^2}{c}} \quad . \end{aligned} \tag{12}$$

From (3) and (9) we have that there exists a fixed constant c_4 such that if $p \geq c_4 \frac{\log n}{n}$ then, with probability $> \frac{1}{2}$ (for n large enough) it holds that $\frac{1}{4} n^2 p(1-p) \leq \Delta^+(G_1, G_1, \psi) \leq \frac{5}{4} n^2 p(1-p)$. Thus, by (12), noting again that $r_1 + r_2 = 1$ and that $\Delta^+(G_1, G_1, \psi) = \frac{1}{2}\Delta(G_1, G_1, \psi)$, we have unconditionally

$$\mathbb{P}\left(\Delta^{0+}(G_1, G_2, \psi) + \Delta^{0-}(G_1, G_2, \psi) > \frac{1}{2} \cdot \Delta(G_1, G_1, \psi)\right) \geq \frac{c}{2n^2} \cdot e^{\frac{-\varrho^2 n^2}{c}} \tag{13}$$

Next, the number of derangements $|\Pi(n)|$ satisfies $\lim_{n\to\infty} \frac{|\Pi(n)|}{n!} = \frac{1}{e}$, thus with Stirling's formula we have that for n large enough it will hold that $|\Pi(n)| \geq \left(\frac{n}{e}\right)^n$. Thus, for n large enough, by (4) and (13),

$$\begin{aligned}\mathbb{E}\left|\left\{\psi \in \Pi : \Delta(G_1, G_2, \psi) < \Delta(G_1, G_2, e)\right\}\right| &= \sum_{\psi \in \Pi} \mathbb{P}\left(\Delta(G_1, G_2, \psi) < \Delta(G_1, G_2, e)\right) \\ &\geq \sum_{\psi \in \Pi(n)} \mathbb{P}\left(\Delta(G_1, G_2, \psi) < \Delta(G_1, G_2, e)\right) \\ &\geq \left(\frac{n}{e}\right)^n \frac{c}{2n^2} \cdot e^{\frac{-\varrho^2 n^2}{c}} \\ &= \frac{c}{2n^2} \cdot e^{\frac{-\varrho^2 n^2}{c} + n\log n - n},\end{aligned}$$

so that there exists a fixed real number $c_3 > 0$ such that if $\varrho \leq c_3 \sqrt{\frac{\log n}{n}}$ then it holds that $\mathbb{E}\left|\left\{\psi \in \Pi(n) : \Delta(G_1, G_2, \psi) < \Delta(G_1, G_2, e)\right\}\right| \to \infty$ as $n \to \infty$, and the second part of Theorem 1 is proven.

Remark 7 Note that we could tighten the constants c_3 and c_4 appearing above. Here we choose not to, instead focusing on the orders of magnitude of ϱ, and do not pursue exact constants further.

A.5 Overall Argument of the Proof for Theorem 2, part i

The proof of Theorem 2 is very similar in structure to the proof of Theorem 1. For simplicity of notation, suppose without loss of generality that the correlated Erdős-Rényi graphs G_1

and G_2 are on the same set of n vertices V, and we do **not** relabel the vertices. Let Π denote the set of bijections $V \to V$; here the identity function $e \in \Pi$ is the latent alignment bijection. Further suppose that V is partitioned into s seed vertices U, and m nonseed vertices W. Let $\phi : U \to U$ be the identity function, and let $\Pi_\phi := \{\psi \in \Pi : \forall u \in U\ \psi(u) = u\}$. For any $\psi \in \Pi_\phi$, define

$$\begin{aligned}
\Delta_R^+(G_1, G_2, \psi) &:= |\{(w,u) \in W \times U : \{w,u\} \notin E(G_1) \text{ and } \{\psi(w),u\} \in E(G_2)\}|,\\
\Delta_R^-(G_1, G_2, \psi) &:= |\{(w,u) \in W \times U : \{w,u\} \in E(G_1) \text{ and } \{\psi(w),u\} \notin E(G_2)\}|,\\
\Delta_R^{0+}(G_1, G_2, \psi) &:= \big|\{(w,u) \in W \times U : \{w,u\} \notin E(G_1),\ \{\psi(w),u\} \in E(G_1),\\
&\qquad\qquad \{\psi(w),u\} \notin E(G_2)\}\big|,\\
\Delta_R^{0-}(G_1, G_2, \psi) &:= \big|\{(w,u) \in W \times U : \{w,u\} \in E(G_1),\ \{\psi(w),u\} \notin E(G_1),\\
&\qquad\qquad \{\psi(w),u\} \in E(G_2)\}\big|,\\
\Delta_R(G_1, G_2, \psi) &:= \Delta_R^+(G_1, G_2, \psi) + \Delta_R^-(G_1, G_2, \psi).
\end{aligned}$$

First note that

$$\Delta_R^+(G_1, G_1, \psi) = \Delta_R^-(G_1, G_1, \psi) = \frac{1}{2}\Delta_R(G_1, G_1, \psi) \quad ; \tag{14}$$

this can be easily verified by considering, for each $u \in U$ and for each cycle C of the permutation ψ, the changes of status in adjacency-to-u of the vertices as the vertices of C are considered in their cyclic order. (Specifically, the number of changes along C from adjacency-to-u to nonadjacency-to-u are equal to the number of changes along C from nonadjacency-to-u to adjacency-to-u.)

Next, note that

$$\begin{aligned}
\Delta_R(G_1, G_2, \psi) - \Delta_R(G_1, G_2, e) = \Delta_R(G_1, G_1, \psi) &- 2 \cdot \Delta_R^{0+}(G_1, G_2, \psi)\\
&- 2 \cdot \Delta_R^{0-}(G_1, G_2, \psi);
\end{aligned} \tag{15}$$

this is easily verified by replacing "G_2" in (15) with "G", and observing the truth of (15) as G, starting out with $G = G_1$, is changed one edge-flip at a time until $G = G_2$.

Now, consider the event Υ_R defined as it holding that, for all $\psi \in \Pi_\phi$ besides e,

$$\Delta_R^{0+}(G_1, G_2, \psi) \;<\; \Delta_R^+(G_1, G_1, \psi) \cdot \left((1-p)(1-\varrho) + \frac{\varrho}{2}\right) \text{ and also} \tag{16}$$

$$\Delta_R^{0-}(G_1, G_2, \psi) \;<\; \Delta_R^-(G_1, G_1, \psi) \cdot \left(p(1-\varrho) + \frac{\varrho}{2}\right). \tag{17}$$

We will show in Section A.6 that, under the hypotheses of the first part of Theorem 2, Υ_R almost always happens. Then, adding (16) to (17) and using (14), we then obtain that almost always $\Delta_R^{0+}(G_1, G_2, \psi) + \Delta_R^{0-}(G_1, G_2, \psi) < \frac{1}{2} \cdot \Delta_R(G_1, G_1, \psi)$ for all $\psi \in \Pi_\phi \backslash \{e\}$. Substituting this into (15) yields that almost always $\Delta_R(G_1, G_2, \psi) > \Delta_R(G_1, G_2, e)$ for all $\psi \in \Pi_\phi \backslash \{e\}$, and the first part of Theorem 2 will then be proven.

A.6 Under Hypotheses of Theorem 2, Part i, Υ_R Occurs Almost Always

For any $k \in \{1, 2, \ldots, m\}$, denote $\Pi_\phi(k) := \{\psi \in \Pi_\phi : |\{v \in V : \psi(v) \neq v\}| = k\}$. Just for now, let $k \in \{1, 2, \ldots, m\}$ be chosen, and let $\psi \in \Pi_\phi(k)$ be chosen. The random variable

$\Delta_R(G_1, G_1, \psi)$ is a function of the $\eta' := ks$ independent Bernoulli(p) random variables

$$\{\mathbb{1}\{\{w,u\} \in E(G_1)\}\}_{(w,u)\in W\times U:\psi(w)\neq w},$$

and note that the hypotheses of Theorem 3 are satisfied, hence for the choice of $t = \frac{1}{20}\sqrt{\eta' p(1-p)}$ in Theorem 3 we obtain that

$$\mathbb{P}\left[|\Delta_R(G_1, G_1, \psi) - \mathbb{E}\Delta_R(G_1, G_1, \psi)| > \frac{1}{5}\eta' p(1-p)\right] \leq 2e^{-\eta' p(1-p)/400}. \tag{18}$$

Also note that

$$\Delta_R(G_1, G_1, \psi) = \sum_{\substack{(w,u)\in W\times U\\ \text{s.t.}\psi(w)\neq w}} \mathbb{1}\Big\{\mathbb{1}\{\{w,u\} \in E(G_1)\} \neq \mathbb{1}\{\{\psi(w),u\} \in E(G_1)\}\Big\}$$

is the sum of η' Bernoulli($2p(1-p)$) random variables hence

$$\mathbb{E}\Delta_R(G_1, G_1, \psi) = 2\eta' p(1-p). \tag{19}$$

Thus, by (18) and (19) we obtain that

$$\mathbb{P}\left(\frac{\Delta_R(G_1, G_1, \psi)}{ksp(1-p)} \notin [9/5,\ 11/5]\right) \leq 2e^{\frac{-1}{400}ksp(1-p)} \leq 2e^{\frac{-\xi_2^2}{400}ks}. \tag{20}$$

Conditioning on G_1, random variable $\Delta_R^{0+}(G_1, G_2, \psi)$ has a

$$\text{Binomial}\left(\Delta_R^{+}(G_1, G_1, \psi), (1-p)(1-\varrho)\right)$$

distribution, and random variable $\Delta_R^{0-}(G_1, G_2, \psi)$ has a

$$\text{Binomial}\left(\Delta_R^{-}(G_1, G_1, \psi), p(1-\varrho)\right)$$

distribution. Conditioning also on the event that $\frac{\Delta_R(G_1,G_1,\psi)}{ksp(1-p)} \in [9/5,\ 11/5]$, applying Theorem 4 with the value $t = \frac{\varrho}{2}\cdot\Delta_R^{+}(G_1, G_1, \psi)$, and using (14), we have that

$$\mathbb{P}\left[\Delta_R^{0+}(G_1, G_2, \psi) \geq \Delta_R^{+}(G_1, G_1, \psi)\cdot\left((1-p)(1-\varrho) + \frac{\varrho}{2}\right)\right] \leq e^{\frac{-\xi_2^4}{20}\cdot ks}, \tag{21}$$

$$\mathbb{P}\left[\Delta_R^{0-}(G_1, G_2, \psi) \geq \Delta_R^{-}(G_1, G_1, \psi)\cdot\left(p(1-\varrho) + \frac{\varrho}{2}\right)\right] \leq e^{\frac{-\xi_2^4}{20}\cdot ks}. \tag{22}$$

Finally, applying (20), (21) and (22), the probability of Υ_R^C can be bounded using subadditivity:

$$\begin{aligned}
\mathbb{P}(\Upsilon_R^C) &\leq \sum_{k=1}^{m}\sum_{\psi\in\Pi_\phi(k)}\left(2e^{\frac{-\xi_2^2}{400}ks} + e^{\frac{-\xi_2^4}{20}\cdot ks} + e^{\frac{-\xi_2^4}{20}\cdot ks}\right)\\
&\leq \sum_{k=1}^{m} m^k\left(2e^{\frac{-\xi_2^2}{400}ks} + 2e^{\frac{-\xi_2^4}{20}\cdot ks}\right)\\
&\leq \sum_{k=1}^{m}\left(2e^{\frac{-\xi_2^2}{400}ks + k\log m} + 2e^{\frac{-\xi_2^4}{20}\cdot ks + k\log m}\right) \leq m\cdot\frac{4}{m^3},
\end{aligned}$$

the last inequality holding if $s \geq c_5 \log m$ for sufficiently large, fixed constant c_5. Because $\sum_{m=1}^{\infty} \frac{4}{n^2} < \infty$ we have by the Borel-Cantelli Lemma that Υ_R almost always happens. As mentioned in Section A.5, this completes the proof of the first part of Theorem 2.

Remark 8 We do not chase the exact constant c_5 here, focusing on the order of magnitude of s instead. Also, if we allow p and ρ to vary with m, then a minor alteration of the above proof (and a tighter Chernoff-Hoeffding bound) yields the same conclusion as in Theorem 2.i if for (an arbitrary but) fixed $0 < \epsilon < 2$ and $q_1 := (1-p)(1-\varrho)$ and $q_2 := p(1-\varrho)$

$$\begin{aligned} c_5 &:= c_5(p, \varrho) \\ &> \max\left\{ \frac{2}{H(q_1 + \frac{\varrho}{2}, q_1) \cdot p(1-p)(2-\epsilon)}, \frac{2}{H(q_2 + \frac{\varrho}{2}, q_2) \cdot p(1-p)(2-\epsilon)}, \frac{16}{\epsilon^2 p(1-p)} \right\}. \end{aligned}$$

Details are left to the reader.

A.7 Proof of the Theorem 2, Part ii

We now prove the second part of Theorem 2.

Just for now, let $\psi \in \Pi(m)$ be chosen (i.e., none of the nonseeds are fixed points for ψ), and condition on $\Delta_R^+(G_1, G_1, \psi) = L$, where $\frac{9}{10} smp(1-p) \leq L \leq \frac{11}{10} smp(1-p)$. The random variables $\Delta_R^{0+}(G_1, G_2, \psi)$ and $\Delta_R^{0-}(G_1, G_2, \psi)$ are independent, and have distributions Binomial(L, q_1) and Binomial(L, q_2), respectively, where $q_1 := (1-p)(1-\varrho)$ and $q_2 := p(1-\varrho)$.

Denoting $r_1 := q_1 + \frac{\rho}{2}$ and $r_2 := q_2 + \frac{\varrho}{2}$, we have by Proposition 5 that

$$\begin{aligned} &\mathbb{P}\left(\Delta_R^{0+}(G_1, G_2, \psi) > L \cdot r_1 \text{ and } \Delta_R^{0-}(G_1, G_2, \psi) > L \cdot r_2 \right) \\ &\qquad \geq \frac{q_1 q_2}{200L} \sqrt{\frac{(1-r_1)(1-r_2)}{r_1 r_2}} e^{-L \cdot H(r_1, q_1) - L \cdot H(r_2, q_2)} \end{aligned}$$

Considering the bound on $H(x+y, y)$ described in Section A.4, we have that $H(r_1, q_1)$ and $H(r_2, q_2)$ are both bounded above by a constant. With the fact that $1 - r_1 = r_2$ and $1 - r_2 = r_1$, from the above we obtain that there is a positive real number c such that

$$\begin{aligned} \mathbb{P}\left(\Delta_R^{0+}(G_1, G_2, \psi) > L \cdot r_1 \text{ and } \Delta_R^{0-}(G_1, G_2, \psi) > L \cdot r_2 \right) &\geq \frac{c}{sm} \cdot e^{-\frac{sm}{c}} \\ &\geq \frac{c}{m \log m} \cdot e^{-\frac{sm}{c}} \end{aligned} \tag{23}$$

under the hypotheses of the second part of Theorem 2.

Next, $|\Pi_\phi(m)|$ is the number of derangements of an m element set, and it satisfies $\lim_{m\to\infty} \frac{|\Pi_\phi(m)|}{m!} = \frac{1}{e}$, thus with Stirling's formula we have that for m large enough it will

hold that $|\Pi(m)| \geq \left(\frac{m}{e}\right)^m$. Thus, for m large enough, by (15) and (23),

$$\begin{aligned}\mathbb{E}|\{\psi \in \Pi_\phi : \Delta_R(G_1, G_2, \psi) < \Delta_R(G_1, G_2, e)\}| &= \sum_{\psi \in \Pi_\phi} \mathbb{P}\Big(\Delta_R(G_1, G_2, \psi) < \Delta_R(G_1, G_2, e)\Big) \\ &\geq \sum_{\psi \in \Pi_\phi(m)} \mathbb{P}\Big(\Delta_R(G_1, G_2, \psi) < \Delta_R(G_1, G_2, e)\Big) \\ &\geq \left(\frac{m}{e}\right)^m \frac{c}{m \log m} \cdot e^{-\frac{sm}{c}} \\ &= \frac{c}{m \log m} \cdot e^{-\frac{sm}{c} + m \log m - m},\end{aligned}$$

so that there exists a fixed real number $c_6 > 0$ such that if $s \leq c_6 \log m$ then it follows that $\mathbb{E}|\{\psi \in \Pi_\phi : \Delta_R(G_1, G_2, \psi) < \Delta_R(G_1, G_2, e)\}| \to \infty$ as $m \to \infty$, and Theorem 2 part ii is proven.

Remark 9 We could tighten the constant c_6 here, but choose instead to focus on the order of magnitude of s. If we allow p and ϱ to be functions of m, then a simple alteration of the above proof yields the same results of Theorem 2.ii if

$$c_6 := c_6(p, \varrho) < \frac{1}{4\big[H(q_1 + \frac{\varrho}{2}, q_1) + H(q_2 + \frac{\varrho}{2}, q_2)\big]p(1-p)};$$

again details are left to the reader.

References

N. Alon, J. Kim, and J. Spencer. Nearly perfect matchings in regular simple hypergraphs. *Israel Journal of Mathematics*, 100(1):171–187, 1997.

A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *2005 IEEE Conference on Computer Vision and Pattern Recognition*, pages 26–33, 2005.

N.W. Brixius and K. M. Anstreicher. Solving quadratic assignment problems using convex quadratic programming relaxations. *Optimization Methods and Software*, 16:49–68, 2001.

T. Caelli and S. Kosinov. An eigenspace projection clustering method for inexact graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(4):515–519, 2004.

M. Cho and K. M. Lee. Progressive graph matching: Making a move of graphs via probabilistic voting. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 398–405, 2012.

M. Cho, J. Lee, and K. M. Lee. Feature correspondence and deformable object matching via agglomerative correspondence clustering. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1280–1287, 2009.

F. Chung and L. Lu. Concentration inequalities and martingale inequalities: A survey. *Internet Mathematics*, 3(1):79–127, 2006.

D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18 (03):265–298, 2004.

T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *Advances in Neural Information Processing Systems*, volume 19, pages 313–320. MIT; 1998, 2007.

J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19:248–264, 1972.

M. Fiori, P. Sprechmann, J. T. Vogelstein, P. Musé, and G. Sapiro. Robust multimodal graph matching: Sparse coding meets graph matching. In *Advances in Neural Information Processing Systems*, pages 127–135, 2013.

D. E. Fishkind, S. Adali, and C. E. Priebe. Seeded graph matching. *arXiv preprint arXiv:1209.0367*, 2012.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman, 1979.

W. R. Gray, J. A. Bogovic, J. T. Vogelstein, B. A. Landman, J. L. Prince, and R. J. Vogelstein. Magnetic resonance connectome automated pipeline: An overview. *IEEE Pulse*, 3(2):42–48, 2012.

B. Huet, A. D. J. Cross, and E. R. Hancock. Graph matching for shape retrieval. In *Advances in Neural Information Processing Systems*, pages 896–902, 1999.

J. H. Kim, B. Sudakov, and V. H. Vu. On the asymmetry of random regular graphs and random graphs. *Random Structures & Algorithms*, 21(3-4):216–224, 2002.

H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 2006.

M. Leordeanu, M. Hebert, and R. Sukthankar. An integer projected fixed point method for graph matching and map inference. In *Advances in Neural Information Processing Systems*, pages 1114–1122, 2009.

Z. Liu and H. Qiao. A convex-concave relaxation procedure based subgraph matching algorithm. In *ACML*, pages 237–252, 2012.

P. Pedarsani and M. Grossglauser. On the privacy of anonymized networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1243, 2011.

R. C. Read and D. G. Corneil. The graph isomorphism disease. *Journal of Graph Theory*, 1(4):339–363, 1977.

A. Robles-Kelly and E. R. Hancock. A Riemannian approach to graph embedding. *Pattern Recognition*, 40(3):1042–1056, 2007.

J. T. Vogelstein, J. M. Conroy, L. J. Podrazik, S. G. Kratzer, E. T. Harley, D. E. Fishkind, R. J. Vogelstein, and C. E. Priebe. Large (brain) graph matching via fast approximate quadratic programming. *arXiv preprint arXiv:1112.5507*, 2011.

B. Xiao, E. R. Hancock, and R. C. Wilson. A generative model for graph matching and embedding. *Computer Vision and Image Understanding*, 113(7):777–789, 2009.

M. Zaslavskiy, F. Bach, and J. P. Vert. A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2227–2242, 2009.

F. Zhou and F. De la Torre. Factorized graph matching. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 127–134, 2012.

Journal of Machine Learning Research 15 (2014) 3721-3742 Submitted 4/12; Revised 2/14; Published 10/14

Asymptotic Accuracy of Distribution-Based Estimation of Latent Variables

Keisuke Yamazaki K-YAM@MATH.DIS.TITECH.AC.JP
Department of Computational Intelligence and Systems Science,
Tokyo Institute of Technology
G5-19 4259 Nagatsuta, Midori-ku, Yokohama, Japan

Editor: Athanasios Kottas

Abstract

Hierarchical statistical models are widely employed in information science and data engineering. The models consist of two types of variables: observable variables that represent the given data and latent variables for the unobservable labels. An asymptotic analysis of the models plays an important role in evaluating the learning process; the result of the analysis is applied not only to theoretical but also to practical situations, such as optimal model selection and active learning. There are many studies of generalization errors, which measure the prediction accuracy of the observable variables. However, the accuracy of estimating the latent variables has not yet been elucidated. For a quantitative evaluation of this, the present paper formulates distribution-based functions for the errors in the estimation of the latent variables. The asymptotic behavior is analyzed for both the maximum likelihood and the Bayes methods.

Keywords: unsupervised learning, hierarchical parametric models, latent variable, maximum likelihood method, Bayes method

1. Introduction

Hierarchical probabilistic models, such as mixture models, are mainly employed in unsupervised learning. The models have two types of variables: observable and latent. The observable variables represent the given data, and the latent ones describe the hidden data-generation process. For example, in mixture models that are employed for clustering tasks, observable variables are the attributes of the given data and the latent ones are the unobservable labels.

One of the main concerns in unsupervised learning is the analysis of the hidden processes, such as how to assign clustering labels based on the observations. Hierarchical models have an appropriate structure for this analysis, because it is straightforward to estimate the latent variables from the observable ones. Even within the limits of the clustering problem, there are a great variety of ways to detect unobservable labels, both probabilistically and deterministically, and many criteria have been proposed to evaluate the results (Dubes and Jain, 1979). For parametric models, the focus of the present paper, learning algorithms such as the expectation-maximization (EM) algorithm and the variational Bayes (VB) method (Attias, 1999; Ghahramani and Beal, 2000; Smidl and Quinn, 2005; Beal, 2003) have been

Estimation Target \Model Case	Regular Case	Singular Case
Observable Variable	Reg-OV estimation	Sing-OV estimation
Latent Variable	Reg-LV estimation	Sing-LV estimation

Table 1: Estimation classification according to the target variable and the model case

developed for estimating the latent variables. These algorithms must estimate both the parameter and the variables, since the parameter is also unknown in the general case.

Theoretical analysis of the models plays an important role in evaluating the learning results. There are many studies on predicting performance in situations where both training and test data are described by the observable variables. The results of asymptotic analysis have been used for practical applications, such as model selection and active learning (Akaike, 1974; Takeuchi, 1976; Fedorov, 1972). The simplest case of the analysis is when the learning model contains the true model, which generates the data. Recently, it has been pointed out that when there is the redundant range/dimension of the latent variables in the learning model, singularities exist in the parameter space and the conventional statistical analysis is not valid (Amari and Ozeki, 2001). To tackle this issue, a theoretical analysis of the Bayes method was established using algebraic geometry (Watanabe, 2009). The generalization performance was then derived for various models (Yamazaki and Watanabe, 2003a,b; Rusakov and Geiger, 2005; Aoyagi, 2010; Zwiernik, 2011). Based on this analysis of the singularities, some criteria for model selection have been proposed (Watanabe, 2010; Yamazaki et al., 2005, 2006).

Although validity of the learning algorithms is necessary for unsupervised tasks, statistical properties of the accuracy of the estimation of the latent variables have not been studied sufficiently. Table 1 summarizes the classification according to the target variable of estimation and the model case. We will use the abbreviations shown in the table to specify the target variable and the model case; for example, Reg-OV estimation stands for estimation of the observable variable in the regular case. As mentioned above, theoretical analysis have been conducted in both the Reg-OV and the Sing-OV estimations. On the other hand, there is no statistical approach to measure the accuracy of the Reg-LV or the Sing-LV estimation.

The goal of the present paper is to provide an error function for measuring the accuracy, which is suitable for the unsupervised learning with hierarchical models, and to derive its asymptotic form. For the first step, we consider the simplest case, in which the attributes, such as the range and dimension, of the latent variables are known; there is no singularity in the parameter space. This corresponds to the Reg-OV estimation in the table. Since the mathematical structure of the parameter is much more complicated in the singular case, we leave the analysis of the Sing-LV estimation for Yamazaki (2012). The main contributions of the present paper are the following three items: (1) estimation for the latent variables falls into three types as shown in Figure 1 and their error functions are formulated in a distribution-based manner; (2) the asymptotic forms of the error functions are derived on the maximum likelihood and the Bayes methods in Type I and variants of Types II and III shown in Figure 2; (3) it is determined that the Bayes method is more accurate than the maximum likelihood method in the asymptotic situation.

The rest of this paper is organized as follows: In Section 2 we explain the estimation of latent variables by comparing it with the prediction of observable variables. In Section 3 we provide the formal definitions of the estimation methods and the error functions. Section 4 then presents the main results for the asymptotic forms and the proofs. Discussions and conclusions are stated in Sections 5 and 6, respectively.

2. Estimations of Variables

This section distinguishes between the estimation of latent variables and the prediction of observable variables. There are variations on the estimation of latent variables due to the estimated targets.

Assume that the observable data and unobservable labels are represented by the observable variables x and the latent variables y, respectively. Let us define that $x \in R^M$ and $y \in \{1, 2, \ldots, K\}$. In the case of a discrete x such as $x \in \{1, 2, \ldots, M\}$, all the results in this paper hold if $\int dx$ is replaced with $\sum_{x=1}^{M}$. A set of n independent data pairs is expressed as $(X^n, Y^n) = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $X^n = \{x_1, \ldots, x_n\}$ and $Y^n = \{y_1, \ldots, y_n\}$. More precisely, there is no dependency between x_i and x_j or between y_i and y_j for $i \neq j$.

Figure 1 shows a variety of estimations of variables: prediction of an observable variable and three types of estimations of latent variables. Solid and dotted nodes are the observable and latent variables, respectively. A data pair is depicted by a connection between two nodes. The gray nodes are the target items of the estimations. We consider a stochastic approach, where the probability distribution of the target(s) is estimated from the training data X^n.

The top-left panel shows the prediction of unseen observable data. Based on X^n, the next observation $x = x_{n+1}$ is predicted. The top-right panel shows the estimation of Y^n, which is referred to as Type I. In the stochastic approach, the joint probability of Y^n is estimated. The bottom-left panel shows marginal estimation, referred to as Type II. The marginal probability of y_i (y_1 is the example in the figure) is estimated; the rest of the latent variables in the probability are marginalized out. Note that there is no unseen/future data in either of Types I or II. The bottom-right panel shows estimation of y in the unseen data, which is referred to as Type III. The difference between this and Type II is the training data; the corresponding observable part of the target is included in the training set in Type II, but it is not included in Type III. In the present paper we use a distribution-based approach to analyze the theoretical accuracy of a Type-I estimation, but we also consider connections to the other types.

3. Formal Definitions of Estimation Methods and Accuracy Evaluations

This section presents the maximum likelihood and Bayes methods for estimating latent variables and the corresponding error functions. Here, we consider only the Type-I estimation problem for the joint probability of the hidden part. The other types will be defined and discussed in Section 5.

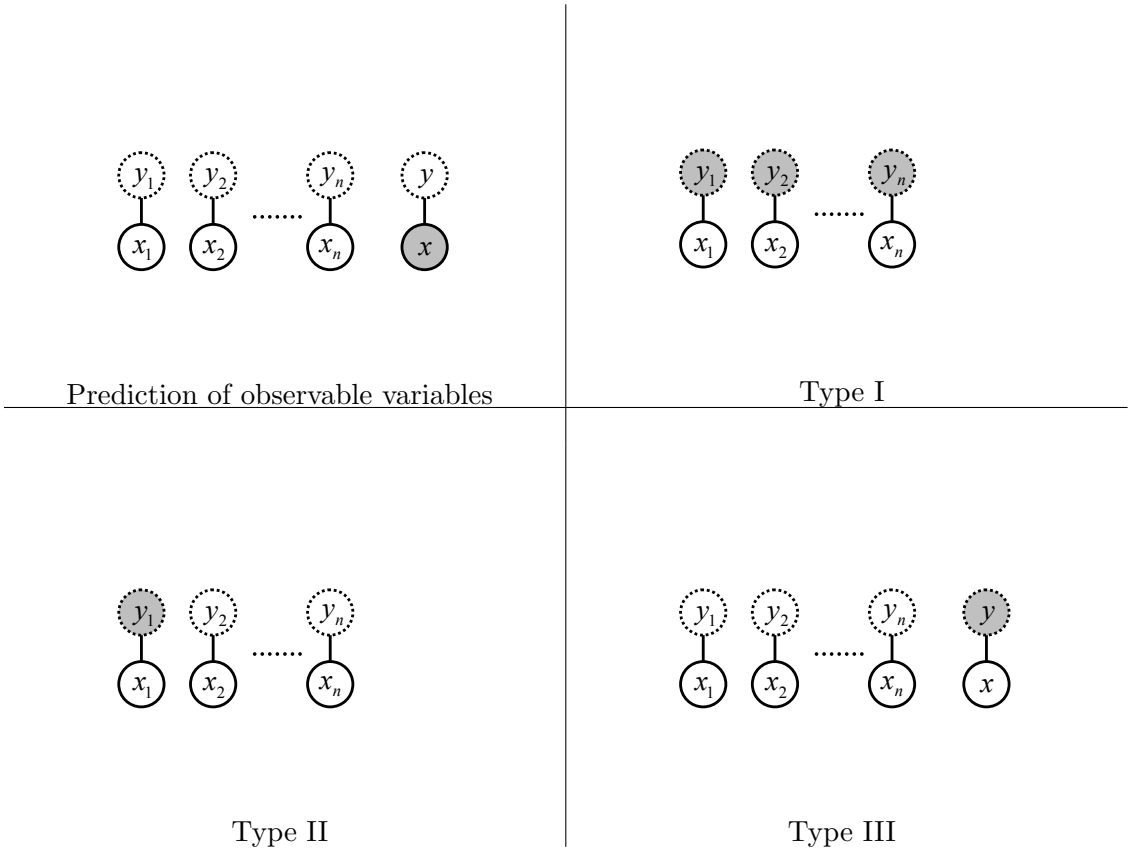


Figure 1: Prediction of observable variables and estimations of latent variables. The observable data are $\{x_1, \ldots, x_n\}$. Solid and dotted nodes are observable and unobservable, respectively. Gray nodes are estimation targets.

Let $p(x, y|w) = p(y|w)p(x|y, w)$ be a learning model, where $w \in W \subset R^d$ is the parameter. The probability of the observable data is expressed as

$$p(x|w) = \sum_{y=1}^{K} p(y|w)p(x|y, w).$$

Assume that the true model generating the data (X^n, Y^n) is expressed as $q(x, y) = p(y|w^*)p(x|y, w^*)$, where w^* is the true parameter, and that the following Fisher information matrices exist and are positive definite;

$$\{I_{XY}(w^*)\}_{ij} = E\left[\frac{\partial \ln p(x, y|w^*)}{\partial w_i} \frac{\partial \ln p(x, y|w^*)}{\partial w_j}\right],$$
$$\{I_X(w^*)\}_{ij} = E\left[\frac{\partial \ln p(x|w^*)}{\partial w_i} \frac{\partial \ln p(x|w^*)}{\partial w_j}\right],$$

where the expectation is

$$E[f(x,y)] = \int \sum_{y=1}^{K} f(x,y)p(x,y|w^*)dx.$$

This condition requires the identifiability of the true model, i.e., $q(y) > 0$ for all y and $i \neq j \Rightarrow q(x|y=i) \neq q(x|y=j)$. The joint probability distribution of (X^n, Y^n) is denoted by $q(X^n, Y^n) = \prod_{i=1}^n q(x_i, y_i)$.

We introduce two ways to construct a probability distribution of Y^n based on the observable X^n. First, we define an estimation method based on the maximum likelihood estimator. The likelihood is defined by

$$L_X(w) = \prod_{i=1}^{n} p(x_i|w).$$

The maximum likelihood estimator $\hat{w}_X$ is given by

$$\hat{w}_X = \arg\max L_X(w).$$

Definition 1 (The maximum likelihood method) *In the maximum likelihood estimation, the estimated distribution of the latent variables is defined by*

$$\begin{aligned} p(Y^n|X^n) =& \frac{p(X^n, Y^n|\hat{w}_X)}{\sum_{Y^n} p(X^n, Y^n|\hat{w}_X)} \\ =& \prod_{i=1}^{n} \frac{p(x_i, y_i|\hat{w}_X)}{\sum_{y_i} p(x_i, y_i|\hat{w}_X)} = \prod_{i=1}^{n} p(y_i|x_i, \hat{w}_X). \end{aligned} \tag{1}$$

The notation $p(Y^n|X^n, \hat{w}_X)$ is used when the method is emphasized.

Next, we define the Bayesian estimation. Let the likelihood of the joint probability distribution be

$$L_{XY}(w) = \prod_{i=1}^{n} p(x_i, y_i|w).$$

The marginal likelihood functions are given by

$$\begin{aligned} Z(X^n, Y^n) =& \int L_{XY}(w)\varphi(w;\eta)dw, \\ Z(X^n) =& \sum_{Y^n} Z(X^n, Y^n) = \int L_X(w)\varphi(w;\eta)dw, \end{aligned}$$

where $\varphi(w;\eta)$ is a prior with the hyperparameter η. We assume that the support of the prior includes w^*.

Definition 2 (The Bayes method) *In the Bayes estimation, the estimated distribution of Y^n is expressed as*

$$p(Y^n|X^n) = \frac{Z(X^n, Y^n)}{Z(X^n)}. \tag{2}$$

Based on the posterior distribution defined by

$$p(w|X^n) = \frac{1}{Z(X^n)} L_X(w)\varphi(w;\eta),$$

the estimated distribution has another equivalent form

$$p(Y^n|X^n) = \int \prod_{i=1}^{n} p(y_i|x_i, w)p(w|X^n)dw. \tag{3}$$

Comparing Equation 3 with Equation 1 reveals that the Bayes estimation is based on the expectation over the posterior instead of the plug-in parameter $\hat{w}_X$.

The distribution of Y^n in the true model is uniquely expressed as

$$q(Y^n|X^n) = \prod_{i=1}^{n} q(y_i|x_i) = \prod_{i=1}^{n} \frac{q(x_i, y_i)}{q(x_i)},$$

where $q(x_i) = \sum_{y_i=1}^{K} q(x_i, y_i)$. Accuracy of the latent variable estimation is measured by the difference between the true distribution $q(Y^n|X^n)$ and the estimated one $p(Y^n|X^n)$. For the present paper, we define the error function as the average Kullback-Leibler divergence,

$$D(n) = \frac{1}{n} E_{X^n}\left[\sum_{Y^n} q(Y^n|X^n) \ln \frac{q(Y^n|X^n)}{p(Y^n|X^n)}\right], \tag{4}$$

where the expectation is

$$E_{X^n}[f(X^n)] = \int f(X^n)q(X^n)dX^n.$$

Note that this function is available for any construction of $p(Y^n|X^n)$ when we consider the cases of the maximum likelihood and the Bayes methods below.

4. Asymptotic Analysis of the Error Function

In this section we present the main theorems for the asymptotic forms of the error function.

4.1 Asymptotic Errors of the Two Methods

In the unsupervised learning, there is label switching, which makes interpretation of the estimation result difficult. For example, define the parameter w_s^* as $p(x, y = 1|w_s^*) = p(x, y = 2|w^*)$, $p(x, y = 2|w_s^*) = p(x, y = 1|w^*)$, and $p(x, y = k|w_s^*) = p(x, y = k|w^*)$ for $k > 2$. In this parameter, the label $y = 1$ and $y = 2$ are switched compared with w^*. It holds that $p(x|w_s^*) = p(x|w^*)$ whereas $p(x, y|w_s^*) \neq p(x, y|w^*)$. Therefore, the estimation methods

can search for w_s^* as the true parameter instead of w^* since there is no information of the true labels. In the present paper, we focus on the best performance, where we successfully estimate the true parameter. In other words, we define the true parameter according to the estimated label assignment. Under the best performance situation, the maximum likelihood estimator $\hat{w}_X$ converges to w^* in probability, and the posterior distribution of the Bayes method converges to the normal distribution, the mean of which is $\hat{w}_X$, in law. Then, it is obvious that the error function $D(n)$ goes to zero at $n \to \infty$.

The following theorems show the speed of decrease of the error function;

Theorem 3 (The asymptotic error of the maximum likelihood method) *In the latent variable estimation given by Equation 1, the error function Equation 4 has the following asymptotic form:*

$$D(n) = \frac{1}{2n}\mathrm{Tr}[\{I_{XY}(w^*) - I_X(w^*)\}I_X^{-1}(w^*)] + o\left(\frac{1}{n}\right).$$

Theorem 4 (The asymptotic error of the Bayes method) *In the latent variable estimation given by Equation 2, the error function Equation 4 has the following asymptotic form:*

$$D(n) = \frac{1}{2n}\ln\det\left[I_{XY}(w^*)I_X^{-1}(w^*)\right] + o\left(\frac{1}{n}\right).$$

The proofs are in the appendix. The dominant order is $1/n$ in both methods, and its coefficient depends on the Fisher information matrices. It is not an unaccountable result that the error value depends on the position of w^*. For example, let us consider cluster analysis and assume that distances among the clusters are large. Since we can easily distinguish the clusters, there is not much additional information on the label y. Then, $I_{XY}(w^*)$ is close to $I_X(w^*)$, which makes $D(n)$ small in both methods. The true parameter generally determines difficulty of tasks in the unsupervised learning, and the theorems reflect this fact. We will present a more detailed discussion on the coefficient in Section 5.

The following corollary shows the advantage of the Bayes estimation.

Corollary 5 *Let the error functions for the maximum likelihood and the Bayes methods be denoted by $D^{ML}(n)$ and $D^{Bayes}(n)$, respectively. Assume that $I_{XY}(w^*) \neq I_X(w^*)$. For any true parameter w^*, there exists a positive constant c such that*

$$D^{ML}(n) - D^{Bayes}(n) \geq \frac{c}{n} + o\left(\frac{1}{n}\right).$$

The proof is in the appendix. This result shows that $D^{\mathrm{ML}}(n) > D^{\mathrm{Bayes}}(n)$ for a sufficiently large data size n.

5. Discussion

In this section, we first discuss relations to other error functions such as the generalization error and the error functions on Types II and III. Next, we consider variants of Types II and III, and show the asymptotic forms of their error functions. Last, we summarize comparison between the maximum likelihood and the Bayes methods.

5.1 Relation to Other Error Functions

We now formulate the predictions of observable data and the remaining estimations for Types II and III, and we consider the relations of their error functions to that of Type I.

First, we compare the Reg-LV estimation with the Reg-OV estimation. In the observable-variable estimation, the error function is referred to as the generalization error, which measures the prediction performance on unseen observable data. The generalization error is defined as

$$D_x(n) = E_{X^n}\left[\int q(x)\ln\frac{q(x)}{p(x|X^n)}dx\right],$$

where x is independent of X^n in the data-generating process of $q(x)$. The predictive distribution $p(x|X^n)$ is constructed by

$$p(x|X^n) = p(x|\hat{w}_X)$$

for the maximum likelihood method and

$$p(x|X^n) = \int p(x|w)p(w|X^n)dw$$

for the Bayes method. Both methods estimation have the same dominant terms in their asymptotic forms,

$$D_x(n) = \frac{d}{2n} + o\left(\frac{1}{n}\right).$$

The coefficient of the asymptotic generalization error depends only on the dimension of the parameter for any model, but that of $D(n)$ is determined by both the model expression and the true parameter w^*. This dependency appears when the learning model does not contain the true model in the Reg-OV estimation, and $\hat{w}_X$ is used for approximation of the error function for model selection (Takeuchi, 1976) and active learning (Fedorov, 1972). In the same way, by replacing w^* with $\hat{w}_X$, Theorems 3 and 4 enable us to calculate the error function in the Reg-LV estimation.

In the observable-variable estimation, the error $D_x(n)$ is approximated by the cross-validation and bootstrap methods since unseen data x_{n+1} are interchangeable with one of the given observable data. On the other hand, there is no substitution for the latent variable, which means that any numerical approximation does not exist for $D(n)$ in principle. The theoretical results in the present paper are thus far the only way to estimate the accuracy.

Next, we discuss Type-II estimation; we focus on the value y_i from Y^n and its estimation accuracy. Based on the joint probability, the estimation of y_i is defined by

$$p(y_i|X^n) = \sum_{Y^n\setminus y_i} p(Y^n|X^n),$$

where the summation is taken over Y^n except for y_i. Thus the error function depends on which y_i we exclude. In order to measure the average effect of the exclusions, we define the error as follows:

$$D_{y|X^n}(n) = E_{X^n}\left[\frac{1}{n}\sum_{i=1}^{n}\sum_{y_i} q(y_i|x_i)\ln\frac{q(y_i|x_i)}{p(y_i|X^n)}\right].$$

The maximum likelihood method has the following estimation,

$$\begin{aligned}p(y_i|X^n) =& \sum_{Y^n\setminus y_i}\prod_{i=1}^n \frac{p(x_i,y_i|\hat{w}_X)}{p(x_i|\hat{w}_X)}\\ =&\frac{p(x_1|\hat{w}_X)\cdots p(x_{i-1}|\hat{w}_X)p(x_i,y_i|\hat{w}_X)p(x_{i+1}|\hat{w}_X)\cdots p(x_n|\hat{w}_X)}{\prod_{i=1}^n p(x_i|\hat{w}_X)}\\ =&\frac{p(x_i,y_i|\hat{w}_X)}{p(x_i|\hat{w}_X)} = p(y_i|x_i,\hat{w}_X).\end{aligned}$$

We can easily find that

$$\begin{aligned}D_{y|X^n}(n) =&E_{X^n}\left[\frac{1}{n}\sum_{i=1}^n\sum_{y_i=1}^K q(y_i|x_i)\ln\frac{q(y_i|x_i)}{p(y_i|x_i,\hat{w}_X)}\right]\\ =&\frac{1}{n}E_{X^n}\left[\sum_{Y^n} q(Y^n|X^n)\ln\frac{q(Y^n|X^n)}{p(Y^n|X^n,\hat{w}_X)}\right].\end{aligned}$$

Therefore, it holds that $D_{y|X^n}(n) = D(n)$ in the maximum likelihood method. However, the Bayes method has the estimation,

$$p(y_i|X^n) =\frac{\int p(x_1|w)\cdots p(x_{i-1}|w)p(x_i,y_i|w)p(x_{i+1}|w)\cdots p(x_n|w)\varphi(w;\eta)dw}{Z(X^n)},$$

which indicates $D_{y|X^n}(n) \neq D(n)$. A sufficient condition for $D_{y|X^n}(n) = D(n)$ is to satisfy $p(Y^n|X^n) = \prod_{i=1}^n p(y_i|X^n)$.

Finally, we consider the Type-III estimation. The error is defined by

$$D_{y|x}(n) =E_{X^n}\left[\int q(x)\sum_{y=1}^K q(y|x)\ln\frac{q(y|x)}{p(y|x,X^n)}dx\right].$$

Note that the new observation x is not used for estimation of y, or $D_{y|x}(n)$ will be equivalent to the Type-II error $D_{y|X^{n+1}}(n+1)$. The maximum likelihood estimation $p(y|x,X^n)$ is given by

$$p(y|x,X^n) =\frac{p(x,y|\hat{w}_X)}{p(x|\hat{w}_X)},$$

and for the Bayes method it is

$$p(y|x,X^n) = \int \frac{p(x,y|w)}{p(x|w)}p(w|X^n)dw. \tag{5}$$

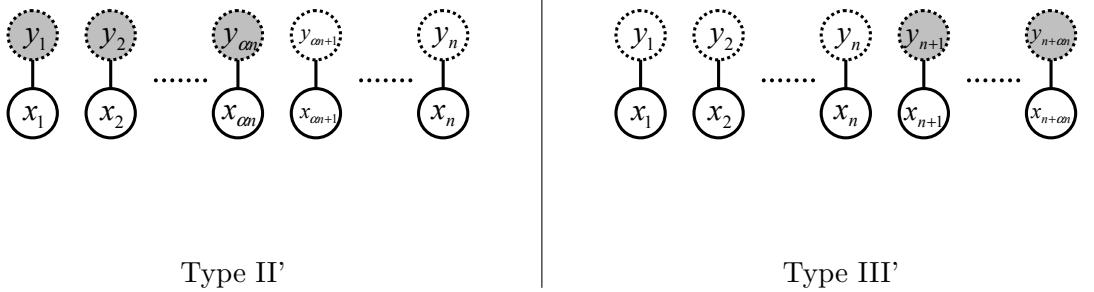
Using the result in Shimodaira (1993) for a variant Akaike information criterion (AIC) from partially observed data, we immediately obtain the asymptotic form of $D_{y|x}(n)$ as

$$D_{y|x}(n) =\frac{1}{2n}\mathrm{Tr}\left[\left\{I_{XY}(w^*) - I_X(w^*)\right\}I_X(w^*)^{-1}\right] + o\left(\frac{1}{n}\right).$$

We thus conclude that all estimation types have the same accuracy in the maximum likelihood method. The difference of the training data between Types II and III does not asymptotically affect the estimation results. The analysis of the Type-III estimate in the Bayes method is left for future study.

	Prediction	Type I	Type II	Type III
ML	$d/2$	$\mathrm{Tr}[\{I_{XY} - I_X\}I_X^{-1}]/2$	$\mathrm{Tr}[\{I_{XY} - I_X\}I_X^{-1}]/2$	$\mathrm{Tr}[\{I_{XY} - I_X\}I_X^{-1}]/2$
Bayes	$d/2$	$\ln\det[I_{XY}I_X^{-1}]/2$	unknown	unknown

Table 2: Coefficients of the dominant order $1/n$ in the error functions



Type II' Type III'

Figure 2: (Left) Partial marginal estimation for $y_1, \ldots, y_{\alpha n}$. (Right) Estimation for future data $y_{n+1}, \ldots, y_{n+\alpha n}$.

5.2 Variants of Types II and III

Table 2 summarizes the results in the previous subsection. The rows indicate the maximum likelihood (ML) and the Bayes methods, respectively. The Fisher information matrices $I_{XY}(w^*)$ and $I_X(w^*)$ are abbreviated in a form that does not include the true parameter, i.e., I_{XY} and I_X. The error functions of Types II and III in the Bayes method are still unknown. The analysis is not straightforward when there is a single target of estimation, because the asymptotic expansion is not available when the number of target nodes is constant with respect to the training data size n.

Consider the variants of Types II and III depicted in Figure 2. Assume that $0 < \alpha \leq 1$ is a constant rational number and that n gets large enough to satisfy that αn is an integer. The left panel shows the partial marginal estimation referred to as Type II'. We will consider the joint probability of $y_1, \ldots, y_{\alpha n}$, where the remaining variables $y_{\alpha n+1}, \ldots, y_n$ have been marginalized out. Type II' is equivalent to Type I when $\alpha = 1$. Note that the order in which the target nodes are determined does not change the average accuracy for i.i.d. data. The right panel indicates the estimations for future data $y_{n+1}, \ldots, y_{n+\alpha n}$. We refer to it as Type III' and construct the joint probability on these variables. In the variant types, the targets are changed from a single node to αn nodes, which enables us to analyze the asymptotic behavior.

We will use the following notation:

$$X_1 = \{x_1, \ldots, x_{\alpha n}\},$$
$$Y_1 = \{y_1, \ldots, y_{\alpha n}\}$$

	Pred.	Type I	Type II'	Type III'
ML	$d/2$	$\mathrm{Tr}[\{I_{XY} - I_X\}I_X^{-1}]/2$	$\mathrm{Tr}[\{I_{XY} - I_X\}I_X^{-1}]/2$	$\mathrm{Tr}[\{I_{XY} - I_X\}I_X^{-1}]/2$
Bayes	$d/2$	$\ln\det[I_{XY}I_X^{-1}]/2$	$\ln\det[K_{XY}I_X^{-1}]/(2\alpha)$	$\ln\det[K_{XY}I_X^{-1}]/(2\alpha)$

Table 3: Coefficients of the dominant order $1/n$ in the error functions

for Type II' and

$$X_2 = \{x_{n+1}, \ldots, x_{n+\alpha n}\},$$
$$Y_2 = \{y_{n+1}, \ldots, y_{n+\alpha n}\}$$

for Type III'. The Bayes estimations are given by

$$p(Y_1|X^n) = \frac{\int \prod_{j=1}^{\alpha n} p(x_j, y_j|w) \prod_{i=\alpha n+1}^{n} p(x_i|w)\varphi(w;\eta)dw}{\int \prod_{i=1}^{n} p(x_i|w)\varphi(w;\eta)dw},$$
$$p(Y_2|X_2, X^n) = \int \prod_{i=n+1}^{n+\alpha n} \frac{p(x_i, y_i|w)}{p(x_i|w)} p(w|X^n)dw$$

for Type II' and Type III', respectively. The respective error functions are defined by

$$D_{Y_1|X^n}(n) = \frac{1}{\alpha n} E_{X^n}\left[\sum_{Y_1} q(Y_1|X^n) \ln \frac{q(Y_1|X^n)}{p(Y_1|X^n)}\right],$$
$$D_{Y_2|X_2}(n) = \frac{1}{\alpha n} E_{X^n, X_2}\left[\sum_{Y_2} q(Y_2|X_2) \ln \frac{q(Y_2|X_2)}{p(Y_2|X_2, X^n)}\right].$$

In ways similar to the proofs of Theorems 3 and 4, the asymptotic forms are derived as follows.

Theorem 6 *In Type II', the error function has the following asymptotic form:*

$$D_{Y_1|X^n}(n) = \frac{1}{2\alpha n} \ln\det[K_{XY}(w^*)I_X(w^*)^{-1}] + o\left(\frac{1}{n}\right),$$

where $K_{XY}(w) = \alpha I_{XY}(w) + (1-\alpha)I_X(w)$.

The proof is in the appendix.

Theorem 7 *In Type III', the error function has the following asymptotic form:*

$$D_{Y_2|X_2}(n) = \frac{1}{2\alpha n} \ln\det[K_{XY}(w^*)I_X^{-1}(w^*)] + o\left(\frac{1}{n}\right).$$

This proof is also in the appendix. These theorems show that when Types II' and III' have the same α, they asymptotically have the same accuracy. This implies the asymptotic equivalency of Types II and III by combining the results of the maximum likelihood method.

Table 3 summarizes the results. Based on the definitions, the results for the maximum likelihood method are also available for Types II' and III'. Using the asymptotic forms, we can compare the relation of the magnitudes for the maximum likelihood method.

Corollary 8 *Assume that $I_{XY}(w) \neq I_X(w)$. For $0 < \alpha \leq 1$, there exists a positive constant c_1 such that*

$$\mathrm{Tr}[\{I_{XY}(w) - I_X(w)\}I_X^{-1}(w)] - \frac{1}{\alpha}\ln\det[K_{XY}(w)I_X^{-1}(w)] \geq \frac{c_1}{n} + o\left(\frac{1}{n}\right).$$

The proof is in the appendix. We immediately obtain the following relation, which shows the advantage of the Bayes estimation in the asymptotic case:

$$\begin{aligned} D^{\mathrm{Bayes}}_{Y_1|X^n}(n) <& D^{\mathrm{ML}}_{Y_1|X^n}(n) \\ D^{\mathrm{Bayes}}_{Y_2|X_2}(n) <& D^{\mathrm{ML}}_{Y_2|X_2}(n) \end{aligned}$$

for respective α's.

By comparing the errors of Types I and II' in the Bayes method, we can obtain the effect of supplementary observable data. Let us consider the Type-II' case in which the estimation target is Y_1 and the training data is only X_1. This corresponds to the estimation in Type I with αn training data, which we emphasize by calling it Type I'. The difference between Type I' and Type II' is the addition of supplementary data $X^n \setminus X_1$.

Corollary 9 *Assume that the minimum eigenvalue of $I_{XY}(w^*)I_X^{-1}(w^*)$ is not less than one, i.e., $\lambda_d \geq 1$. The error difference is asymptotically described as*

$$\begin{aligned} D(\alpha n) - D_{Y_1|X^n}(n) =& \frac{1}{2\alpha n}\ln\det[I_{XY}(w^*)K_{XY}^{-1}(w^*)] + o\left(\frac{1}{n}\right) \\ \geq& \frac{c_2}{n} + o\left(\frac{1}{n}\right), \end{aligned}$$

where c_2 is a positive constant. This shows that Type II' has a smaller error than Type I' in the asymptotic situation; the supplementary data make the estimation more accurate.

The proof is in the appendix.

5.3 Comparison Between the Two Methods

Corollaries 5 and 8 show that the Bayes method is more accurate than the maximum likelihood method for Types I, II', and III'. There have been many data-based comparisons of the predicting performances of these two methods (e.g., Akaike, 1980; Mackay, 1992; Browne and Draper, 2006). We will now discuss the computational costs of the two methods for the estimation of latent variables. We note there will be a trade-off between cost and accuracy.

We will assume that the estimated distribution is to be calculated for a practical purpose. For example, the value of $p(Y^n|X^n)$ in Type I is used for sampling label assignments and for searching for the optimal assignment $\arg\max_{Y^n} p(Y^n|X^n)$. The maximum likelihood method requires the determination of $\hat{w}_X$ for all Types I, II, and III. The computation is not expensive once $\hat{w}_X$ is successfully found, but the global maximum point of the likelihood function is not easily obtained. The EM algorithm is commonly used for searching for the maximum likelihood estimator in models with latent variables, but it is often trapped in

one of the local maxima. The results of the steepest descent method also depend on the initial point and the step size of the iteration.

The Bayes method is generally expensive. In the estimated distribution $p(Y^n|X^n)$ of Type I, the numerator $Z(X^n, Y^n)$ contains integrals that depend on Y^n. Sampling y_i in Type II requires the same computation as for Type I: we can obtain y_i by ignoring the other elements $Y^n \setminus y_i$, which realizes the marginalization $\sum_{Y^n \setminus y_i} p(Y^n|X^n)$. A conjugate prior allows us to have a tractable form of $Z(X^n, Y^n)$ (Dawid and Lauritzen, 1993; Heckerman, 1999), which reduces the computational cost. In Type III, Equation 5 shows that there is no direct sampling method for y. In this case, expensive sampling from the posterior $p(w|X^n)$ is necessary.

The VB method is an approximation that allows the direct computation of $P(Y^n|X^n)$ and $p(w|X^n)$, which have tractable forms and reduced computational costs. However, the assumption that $P(Y^n|X^n)$ and $p(w|X^n)$ are independent does not hold in many cases. We conjecture that the $P(Y^n|X^n)$ of the VB method will be less accurate than that of the original Bayes method.

6. Conclusions

In the present paper we formalized the estimation from the observable data of the distribution of the latent variables, and we measured its accuracy by using the Kullback-Leibler divergence. We succeeded in deriving the asymptotic error functions for both the maximum likelihood and the Bayes methods. These results allow us to mathematically compare the estimation methods: we determined that the Bayes method is more accurate than the maximum likelihood method in most cases, while their prediction accuracies are equivalent. The generalization error has been approximated from the given observable data, such as by using the cross-validation and bootstrap methods, but there is no approximation technique for the error of the estimation of the latent variables, because the latent data can not be obtained. Therefore, these asymptotic forms are thus far the only way we have to estimate their accuracy.

Acknowledgments

This research was partially supported by the Kayamori Foundation of Informational Science Advancement and KAKENHI 23500172.

Appendix A. Proofs

In this section, we prove the theorems and the corollaries.

A.1 Proof of Theorem 3

Proof First, let us define another Fisher information matrix:

$$\{I_{Y|X}(w)\}_{ij} = E\left[\frac{\partial \ln p(y|x,w)}{\partial w_i}\frac{\partial \ln p(y|x,w)}{\partial w_j}\right].$$

Based on $p(y|x,w) = p(x,y|w)/p(x|w)$,

$$I_{Y|X}(w) = I_{XY}(w) + I_X(w) - J_{XY}(w) - J_{XY}^{\top}(w),$$

where

$$\{J_{XY}(w)\}_{ij} = E\left[\frac{\partial \ln p(x,y|w)}{\partial w_i}\frac{\partial \ln p(x|w)}{\partial w_j}\right].$$

According to the definition, we obtain

$$\begin{aligned}\{J_{XY}(w)\}_{ij} =& E\left[\frac{1}{p(x,y|w)}\frac{\partial p(x,y|w)}{\partial w_i}\frac{\partial \ln p(x|w)}{\partial w_j}\right] \\ =& \int \sum_y \frac{\partial p(x,y|w)}{\partial w_i}\frac{\partial \ln p(x|w)}{\partial w_j}dx \\ =& \int \frac{\partial p(x|w)}{\partial w_i}\frac{\partial \ln p(x|w)}{\partial w_j}dx \\ =& \int \frac{\partial \ln p(x|w)}{\partial w_i}\frac{\partial \ln p(x|w)}{\partial w_j}p(x|w)dx = \{I_X(w)\}_{ij}.\end{aligned}$$

Thus, it holds that

$$I_{Y|X}(w) = I_{XY}(w) - I_X(w). \tag{6}$$

Next, let us divide the error function into three parts:

$$\begin{aligned}D(n) =& D_1(n) - D_2(n) - D_3(n), \\ D_1(n) =& \frac{1}{n}E_{X^nY^n}\left[\ln q(X^n,Y^n)\right], \\ D_2(n) =& \frac{1}{n}E_{X^nY^n}\left[\ln p(X^n,Y^n|\hat{w}_X)\right], \\ D_3(n) =& \frac{1}{n}E_{X^n}\left[\ln \frac{q(X^n)}{p(X^n|\hat{w}_X)}\right],\end{aligned} \tag{7}$$

where the expectation is

$$E_{X^nY^n}[f(X^n,Y^n)] = \int \sum_{Y^n} f(X^n,Y^n)q(X^n,Y^n)dX^n.$$

Because $D_3(n)$ is the training error on $p(x|\hat{w}_X)$, the asymptotic form is known (Akaike, 1974):

$$D_3(n) = -\frac{d}{2n} + o\left(\frac{1}{n}\right).$$

Let another estimator be defined by

$$\hat{w}_{XY} = \arg\max L_{XY}(w).$$

According to the Taylor expansion, $D_2(n)$ can be rewritten as

$$\begin{aligned}
D_2(n) =& \frac{1}{n} E_{X^n Y^n}\left[\sum_{i=1}^{n} \ln p(X_i, Y_i|\hat{w}_{XY})\right] \\
&+ \frac{1}{n} E_{X^n Y^n}\left[\delta w^\top \sum_{i=1}^{n} \frac{\partial \ln p(X_i, Y_i|\hat{w}_{XY})}{\partial w}\right] \\
&+ \frac{1}{2n} E_{X^n Y^n}\left[\delta w^\top \sum_{i=1}^{n} \frac{\partial^2 \ln p(X_i, Y_i|\hat{w}_{XY})}{\partial w^2} \delta w + R_1(\delta w)\right] \\
=& \frac{1}{n} E_{X^n Y^n}\left[\sum_{i=1}^{n} \ln p(X_i, Y_i|\hat{w}_{XY})\right] \\
&- \frac{1}{2} E_{X^n Y^n}\left[\delta w^\top I_{XY}(w^*) \delta w\right] + o\left(\frac{1}{n}\right),
\end{aligned}$$

where $\delta w = \hat{w}_X - \hat{w}_{XY}$, and $R_1(\delta w)$ is the remainder term. The matrix $\sum_{i=1}^{n} \frac{\partial^2 \ln p(X_i, Y_i|\hat{w}_{XY})}{\partial w^2}$ was replaced with $I_{XY}(w^*)$ on the basis of the law of large numbers. As for the first term of D_2,

$$\begin{aligned}
&D_1(n) - \frac{1}{n} E_{X^n Y^n}\left[\sum_{i=1}^{n} \ln p(X_i, Y_i|\hat{w}_{XY})\right] \\
&= -\frac{d}{2n} + o\left(\frac{1}{n}\right)
\end{aligned}$$

because it is the training error on $p(x, y|\hat{w}_{XY})$. The factor in the second term of D_2 can be rewritten as

$$\begin{aligned}
&E_{X^n Y^n}\left[\delta w^\top I_{XY}(w^*) \delta w\right] \\
&= E_{X^n Y^n}\left[(\hat{w}_X - w^*)^\top I_{XY}(w^*)(\hat{w}_X - w^*)\right] \\
&- E_{X^n Y^n}\left[(\hat{w}_{XY} - w^*)^\top I_{XY}(w^*)(\hat{w}_X - w^*)\right] \\
&- E_{X^n Y^n}\left[(\hat{w}_X - w^*)^\top I_{XY}(w^*)(\hat{w}_{XY} - w^*)\right] \\
&+ E_{X^n Y^n}\left[(\hat{w}_{XY} - w^*)^\top I_{XY}(w^*)(\hat{w}_{XY} - w^*)\right].
\end{aligned} \tag{8}$$

Let us define an extended likelihood function,

$$L_2(w_{12}) = \sum_{i=1}^{n} \ln p(X_i, Y_i|w_1) + \sum_{i=1}^{n} \ln p(X_i|w_2),$$

where $w_{12} = (w_1^\top, w_2^\top)^\top$, $\hat{w}_{12} = (\hat{w}_{XY}^\top, \hat{w}_X^\top)^\top$, and $w^{**} = (w^{*\top}, w^{*\top})^\top$ are extended vectors. According to the Taylor expansion,

$$\begin{aligned}\frac{\partial L_2(w_{12})}{\partial w_{12}} =& \left(\frac{\partial \sum \ln p(X_i, Y_i|w^*)}{\partial w_1}^\top, \frac{\partial \sum \ln p(X_i|w^*)}{\partial w_2}^\top\right)^\top \\ & - M\delta w_{12}, \\ \delta w_{12} =& w_{12} - w^{**} \\ M =& \begin{bmatrix} -\frac{\partial^2 \sum \ln p(X_i, Y_i|w^*)}{\partial w_1^2} & 0 \\ 0 & -\frac{\partial^2 \sum \ln p(X_i|w^*)}{\partial w_2^2} \end{bmatrix}.\end{aligned}$$

According to $\frac{\partial L_2(\hat{w}_{12})}{\partial w_{12}} = 0$, $\delta\hat{w}_{12} = \hat{w}_{12} - w^{**}$ can be written as

$$\delta\hat{w}_{12} = M^{-1}\left(\frac{\partial \sum \ln p(X_i, Y_i|w^*)}{\partial w_1}^\top, \frac{\partial \sum \ln p(X_i|w^*)}{\partial w_2}^\top\right)^\top.$$

Based on the central limit theorem, $\delta\hat{w}_{12}$ is distributed from $\mathcal{N}(0, nM^{-1}\Sigma^{-1}M^{-1})$, where

$$\Sigma^{-1} = \begin{bmatrix} I_{XY}(w^*) & J_{XY}(w^*) \\ J_{XY}^\top(w^*) & I_X(w^*) \end{bmatrix}.$$

The covariance $nM^{-1}\Sigma^{-1}M^{-1}$ of $\delta\hat{w}_{12}$ directly shows the covariance of the estimators $\hat{w}_X$ and $\hat{w}_{XY}$ in Equation 8. Thus it holds that

$$\begin{aligned}& E_{X^nY^n}\left[\delta w^\top I_{XY}(w^*)\delta w\right] \\ &= \frac{1}{n}\mathrm{Tr}\left[I_{XY}(w^*)I_X^{-1}(w^*)\right] - \frac{1}{n}\mathrm{Tr}\left[J_{XY}(w^*)I_X^{-1}(w^*)\right] \\ &- \frac{1}{n}\mathrm{Tr}\left[J_{XY}^\top(w^*)I_X^{-1}(w^*)\right] + \frac{1}{n}\mathrm{Tr}\left[I_X(w^*)I_X^{-1}(w^*)\right] + o\left(\frac{1}{n}\right).\end{aligned}$$

Considering the relation Equation 7, we obtain that

$$D(n) = \frac{1}{2n}\mathrm{Tr}[I_{Y|X}(w^*)I_X^{-1}(w^*)] + o\left(\frac{1}{n}\right).$$

Based on Equation 6, the theorem is proved. ■

A.2 Proof of Theorem 4

Proof Let us define the following entropy functions:

$$\begin{aligned}S_{XY} &= -\sum_{y=1}^{K^*}\int q(x, y)\ln q(x, y)dx, \\ S_X &= -\int q(x)\ln q(x)dx.\end{aligned}$$

According to the definition, the error function Equation 4 with the Bayes estimation can be rewritten as

$$D(n) = \frac{1}{n}\Big\{ F_{XY}(n) - F_X(n) \Big\},$$

where

$$F_{XY}(n) = -nS_{XY} - E_{X^nY^n}\Big[\ln Z(X^n, Y^n) \Big],$$
$$F_X(n) = -nS_X - E_{X^n}\Big[\ln Z(X^n) \Big].$$

Based on the Taylor expansion at $w = \hat{w}_X$,

$$\begin{aligned} F_X(n) =& -nS_X - E_{X^n}\Big[\ln \int \exp\Big\{ \ln p(X^n|\hat{w}_X) \\ &+ \frac{1}{2}(w-\hat{w}_X)^\top \frac{\partial^2 \ln p(X^n|\hat{w}_X)}{\partial w^2}(w-\hat{w}_X) + r_1(w) \Big\} \varphi(w;\eta) dw \Big] \\ =& -nS_X - E_{X^n}[\ln p(X^n|\hat{w}_X] - E_{X^n}\Big[\ln \int e^{r_1(w)} \varphi(w;\eta) \mathcal{N}(\hat{w}_X, \Sigma_1/n) dw \Big], \end{aligned}$$

where $r_1(w)$ is the remainder term and

$$\Sigma_1^{-1} = -\frac{1}{n}\frac{\partial^2 \ln p(X^n|\hat{w}_X)}{\partial w^2},$$

which converges to $I_X(w^*)$ based on the law of large numbers. Again, applying the expansion at $w = w^*$ to $e^{r_1(w)}\varphi(w;\eta)$, we obtain

$$\begin{aligned} F_X(n) =& E_{X^n}\Big[\ln \frac{q(X^n)}{p(X^n|\hat{w}_X)} \Big] - \ln \sqrt{2\pi}^d \sqrt{\det\{nI_X(w^*)\}^{-1}} \\ &- E_{X^n}\Big[\ln \int \Big\{ e^{r_1(w^*)} \varphi(w^* : \eta) \\ &+ (w-w^*)^\top \frac{\partial e^{r_1(w^*)} \varphi(w^*;\eta)}{\partial w} + r_2(w) \Big\} \mathcal{N}\big(\hat{w}_X, \{nI_X(w^*)\}^{-1}\big) dw \Big] + o(1), \end{aligned}$$

where $r_2(w)$ is the remainder term. The first term is the training error on $p(x|\hat{w}_X)$. According to Akaike (1974), it holds that

$$E_{X^n}\Big[\ln \frac{q(X^n)}{p(X^n|\hat{w}_X)} \Big] = -\frac{d}{2} + o(1).$$

Then, we obtain

$$F_X(n) = \frac{d}{2}\ln\frac{n}{2\pi e} + \ln \frac{\sqrt{\det I_X(w^*)}}{\varphi(w^*;\eta)} + o(1),$$

which is consistent with the result of Clarke and Barron (1990). By replacing X^n with (X^n, Y^n),

$$F_{XY}(n) = \frac{d}{2} \ln \frac{n}{2\pi e} + \ln \frac{\sqrt{\det I_{XY}(w^*)}}{\varphi(w^*; \eta)} + o(1).$$

Therefore,

$$D(n) = \frac{1}{2n} \Big\{ \ln \det I_{XY}(w^*) - \ln \det I_X(w^*) \Big\} + o\Big(\frac{1}{n}\Big),$$

which proves the theorem. ■

A.3 Proof of Corollary 5

Proof Because $I_{XY}(w)$ is symmetric positive definite, we have a decomposition $I_{XY}(w) = LL^\top$, where L is a lower triangular matrix. The other Fisher information matrix $I_X(w)$ is also symmetric positive definite. Thus, $L^T I_X^{-1}(w) L$ is positive definite. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d > 0$ be the eigenvalues of $L^\top I_X^{-1}(w) L$. According to the assumption, at least one eigenvalue is different from the others. Then, we obtain

$$\begin{aligned}
2n\{D^{\mathrm{ML}}(n) - D^{\mathrm{Bayes}}(n)\} =& \mathrm{Tr}[I_{XY}(w) I_X^{-1}(w)] - d - \ln \det[I_{XY}(w) I_X^{-1}(w)] + o(1) \\
=& \mathrm{Tr}[L^\top I_X^{-1}(w) L] - d - \ln \det[L^\top I_X^{-1}(w) L] + o(1) \\
=& \sum_{i=1}^{d} \{\lambda_i - 1\} - \ln \prod_{i=1}^{d} \lambda_i + o(1) \\
=& \sum_{i=1}^{d} \{\lambda_i - 1 - \ln \lambda_i\} + o(1).
\end{aligned}$$

The first term in the last expression is positive, which proves the corollary. ■

A.4 Proof of Theorem 6

Proof The error function is rewritten as

$$\begin{aligned}
D_{Y_1|X^n}(n) =& \frac{1}{\alpha n} \Big\{ F_{XY}^{(1)}(n) - F_X(n) \Big\}, \\
F_{XY}^{(1)}(n) =& -\alpha n S_{XY} - (1-\alpha) n S_X - E_{X^n, Y_1} \Big[\ln \int L_{XY}^{(1)}(w) \varphi(w; \eta) dw \Big], \\
L_{XY}^{(1)}(w) =& \prod_{j=1}^{\alpha n} p(x_j, y_j | w) \prod_{i=\alpha n+1}^{n} p(x_i | w).
\end{aligned}$$

Based on the Taylor expansion at $w = \hat{w}^{(1)}$, where $\hat{w}^{(1)} = \arg\max L^{(1)}(w)$,

$$F^{(1)}_{XY}(n) = E_{X^n,Y_1}\Big[\sum_{j=1}^{\alpha n} \ln \frac{q(x_j,y_j)}{p(x_j,y_j|\hat{w}^{(1)})} + \sum_{i=\alpha n+1}^{n} \ln \frac{q(x_i)}{p(x_i|\hat{w}^{(1)})} + \ln \int \exp\Big\{ - n(w-\hat{w}^{(1)})^\top G^{(1)}(X^n,Y_1)(w-\hat{w}^{(1)}) + r_3(w)\Big\}\varphi(w;\eta)dw\Big],$$

where $r_3(w)$ is the remainder term and

$$G^{(1)}(X^n,Y_1) = -\frac{1}{n}\frac{\partial^2}{\partial w^2}\Big(\sum_{j=1}^{\alpha n} \ln p(x_j,y_j|\hat{w}^{(1)}) + \sum_{i=\alpha n+1}^{n} \ln p(x_i|\hat{w}^{(1)})\Big).$$

The first and the second terms of $F^{(1)}_{XY}(n)$ correspond to the training error. Following the same method as we used in the proof of Theorem 4 and noting that

$$G^{(1)}(X^n,Y_1) \to K_{XY}(w^*),$$

we obtain

$$F^{(1)}_{XY}(n) = \frac{d}{2}\ln\frac{n}{2\pi e} + \ln\frac{\sqrt{\det K_{XY}(w^*)}}{\varphi(w^*;\eta)} + o(1),$$

which completes the proof. ■

A.5 Proof of Theorem 7

Proof The error function is rewritten as

$$D_{Y_2|X_2}(n) = \frac{1}{\alpha n}\Big\{F^{(2)}_{XY}(n) - F_X(n)\Big\},$$

$$F^{(2)}_{XY}(n) = -\alpha n S_{XY} - nS_X - E_{X^n,X_2,Y_2}\Big[\ln\int L^{(2)}_{XY}(w)\varphi(w;\eta)dw\Big],$$

$$L^{(2)}_{XY}(w) = \prod_{j=n+1}^{n+\alpha n} p(y_j|x_j,w)\prod_{i=1}^{n} p(x_i|w).$$

Based on the Taylor expansion at $w = \hat{w}^{(2)}$, where $\hat{w}^{(2)} = \arg\max L^{(2)}(w)$,

$$F^{(2)}_{XY}(n) = E_{X^n,X_2,Y_2}\Big[\sum_{j=n+1}^{\alpha n} \ln \frac{q(y_j|x_j)}{p(y_j|x_j,\hat{w}^{(2)})} + \sum_{i=1}^{n} \ln \frac{q(x_i)}{p(x_i|\hat{w}^{(2)})} + \ln \int \exp\Big\{ - n(w-\hat{w}^{(2)})^\top G^{(2)}(X^n,X_2,Y_2)(w-\hat{w}^{(2)}) + r_4(w)\Big\}\varphi(w;\eta)dw\Big],$$

where $r_4(w)$ is the remainder term and

$$G^{(2)}(X^n,X_2,Y_2) = -\frac{1}{n}\frac{\partial^2}{\partial w^2}\Big(\sum_{j=n+1}^{\alpha n} \ln p(y_j|x_j,\hat{w}^{(2)}) + \sum_{i=1}^{n} \ln p(x_i|\hat{w}^{(2)})\Big).$$

The first and the second terms of $F^{(1)}_{XY}(n)$ correspond to the training error, which are stated as

$$E_{X^n,X_2,Y_2}\Big[\sum_{j=n+1}^{\alpha n}\ln\frac{q(y_j|x_j)}{p(y_j|x_j,\hat{w}^{(2)})}+\sum_{i=1}^{n}\ln\frac{q(x_i)}{p(x_i|\hat{w}^{(2)})}\Big]$$
$$=-\mathrm{Tr}\Big[\big\{\alpha I_{Y|X}(w^*)+I_X(w^*)\big\}K_{XY}(w^*)^{-1}\Big]+o(1).$$

Following the same method we used in the proof of Theorem 4 and noting that

$$G^{(2)}(X^n,X_2,Y_2)\to K_{XY}(w^*),$$

we obtain

$$\begin{aligned}F^{(1)}_{XY}(n)=&-\mathrm{Tr}\Big[\big\{\alpha I_{Y|X}(w^*)+I_X(w^*)\big\}K_{XY}(w^*)^{-1}\Big]\\&+\frac{d}{2}\ln\frac{n}{2\pi}+\ln\frac{\sqrt{\det K_{XY}(w^*)}}{\varphi(w^*;\eta)}+o(1)\\=&\frac{d}{2}\ln\frac{n}{2\pi e}+\ln\frac{\sqrt{\det K_{XY}(w^*)}}{\varphi(w^*;\eta)}+o(1),\end{aligned}$$

which completes the proof. ■

A.6 Proof of Corollary 8

Proof It holds that

$$\frac{1}{\alpha}\ln\det[K_{XY}(w)I_X^{-1}(w)]=\frac{1}{\alpha}\ln\det[\alpha\{I_{XY}(w)-I_X(w)\}I_X^{-1}(w)+E_d],$$

where E_d is the $d\times d$ unit matrix. On the other hand,

$$\mathrm{Tr}[\{I_{XY}(w)-I_X(w)\}I_X^{-1}(w)]=\frac{1}{\alpha}\Big\{\mathrm{Tr}[\alpha\{I_{XY}(w)-I_X(w)\}I_X^{-1}(w)+E_d]-d\Big\}.$$

It is easy to confirm that $\alpha L_1^\top I_X^{-1}(w)L_1+E_d$ is positive definite, where $L_1^\top L_1=I_{XY}(w)-I_X(w)$. Considering the eigenvalues $\mu_1\geq\mu_2\geq\cdots\geq\mu_d>0$, we can obtain the following relation in the same way as we did in the proof of Corollary 5:

$$\mathrm{Tr}[\{I_{XY}(w)-I_X(w)\}I_X^{-1}(w)]-\frac{1}{\alpha}\ln\det[K_{XY}(w)I_X^{-1}(w)]=\frac{1}{\alpha}\sum_{i=1}^{d}\Big\{\mu_i-1-\ln\mu_i\Big\}.$$

It is easy to confirm that the right-hand side is positive, which completes the proof. ■

A.7 Proof of Corollary 9

Proof Based on the eigenvalues of $I_{XY}(w^*)I_X^{-1}(w^*)$, it holds that

$$\begin{aligned}\ln\det[I_{XY}(w^*)K_{XY}^{-1}(w^*)] &= \ln\det[I_{XY}(w^*)I_X^{-1}(w^*)] - \ln\det[\alpha I_{XY}(w^*)I_X^{-1}(w^*) + (1-\alpha)E_d] \\ &= \sum_{i=1}^{d}\ln\lambda_i - \sum_{i=1}^{d}\ln\{\alpha\lambda_i + (1-\alpha)\} \geq 0,\end{aligned}$$

which completes the proof. ■

References

Hirotsugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

Hirotsugu Akaike. Likelihood and Bayes procedure. In *J. M. Bernald, Bayesian statistics*, pages 143–166, Valencia, Italy, 1980. University Press.

Shun-ichi Amari and Tomoko Ozeki. Differential and algebraic geometry of multilayer perceptrons. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E84-A 1:31–38, 2001.

Miki Aoyagi. Stochastic complexity and generalization error of a restricted Boltzmann machine in Bayesian estimation. *Journal of Machine Learning Research*, 11:1243–1272, 2010.

Hagai Attias. Inferring parameters and structure of latent variable models by variational Bayes. In *Proceedings of Uncertainty in Artificial Intelligence*, 1999.

Matthew J. Beal. Variational algorithms for approximate Bayesian inference. Technical report, 2003.

William J. Browne and David Draper. A comparison of Bayesian and likelihood-based methods for fitting multilevel models. *Bayesian Analysis*, 1(3):473–514, 2006.

Bertrand Clarke and Andrew R. Barron. Information-theoretic asymptotics of Bayes methods. *IEEE Transactions on Information Theory*, 36:453–471, 1990.

A. Philip Dawid and Steffen L. Lauritzen. Hyper-Markov laws in the statistical analysis of decomposable graphical models. *Annals of Statistics*, 21(3):1272–1317, 1993.

Richard Dubes and Anil K Jain. Validity studies in clustering methodologies. *Pattern Recognition*, 11:235–254, 1979.

Valerii V. Fedorov. *Theory of Optimal Experiments*. Academic Press, New York, 1972.

Zoubin Ghahramani and Matthew J. Beal. Graphical models and variational methods. In *Advanced Mean Field Methods - Theory and Practice*. MIT Press, 2000.

David Heckerman. A tutorial on learning with Bayesian networks. In Michael I. Jordan, editor, *Learning in Graphical Models*, pages 301–354. MIT Press, Cambridge, MA, USA, 1999.

David. J. C. Mackay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

Dmitry Rusakov and Dan Geiger. Asymptotic model selection for naive Bayesian networks. *Journal of Machine Learning Research*, 6:1–35, 2005.

Hidetoshi Shimodaira. A new criterion for selecting models from partially observed data. In P. Cheeseman and R.W. Oldford, editors, *Selecting Models from Data*, volume 89 of *Lecture Notes in Statistics*, pages 381–386. Springer-Verlag, 1993.

Václav Smidl and Anthony Quinn. *The Variational Bayes Method in Signal Processing (Signals and Communication Technology)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. ISBN 3540288198.

K. Takeuchi. Distribution of informational statistics and a criterion of model fitting. *Suri-Kagaku* (Mathematica Sciences), 153:12–18, 1976. (in Japanese).

Sumio Watanabe. *Algebraic Geometry and Statistical Learning Theory*. Cambridge University Press, New York, NY, USA, 2009. ISBN 0521864674, 9780521864671.

Sumio Watanabe. Equations of states in singular statistical estimation. *Neural Networks*, 23(1):20–34, 2010.

Keisuke Yamazaki. Asymptotic accuracy of Bayes estimation for latent variables with redundancy. arXiv:1205.3234, 2012.

Keisuke Yamazaki and Sumio Watanabe. Singularities in mixture models and upper bounds of stochastic complexity. *International Journal of Neural Networks*, 16:1029–1038, 2003a.

Keisuke Yamazaki and Sumio Watanabe. Stochastic complexity of Bayesian networks. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 592–599, 2003b.

Keisuke Yamazaki, Kenji Nagata, and Sumio Watanabe. A new method of model selection based on learning coefficient. In *Proceedings of International Symposium on Nonlinear Theory and its Applications*, pages 389–392, 2005.

Keisuke Yamazaki, Kenji Nagata, Sumio Watanabe, and Klaus-Robert Müller. A model selection method based on bound of learning coefficient. In *LNCS*, volume 4132, pages 371–380. Springer, 2006.

Piotr Zwiernik. An asymptotic behaviour of the marginal likelihood for general Markov models. *Journal of Machine Learning Research*, 999888:3283–3310, November 2011.

Journal of Machine Learning Research 15 (2014) 3743-3773 Submitted 6/13; Published 11/14

What Regularized Auto-Encoders Learn from the Data-Generating Distribution

Guillaume Alain GUILLAUME.ALAIN@UMONTREAL.CA
Yoshua Bengio YOSHUA.BENGIO@UMONTREAL.CA
Department of Computer Science and Operations Research
University of Montreal
Montreal, H3C 3J7, Quebec, Canada

Editors: Aaron Courville, Rob Fergus, and Christopher Manning

Abstract

What do auto-encoders learn about the underlying data-generating distribution? Recent work suggests that some auto-encoder variants do a good job of capturing the local manifold structure of data. This paper clarifies some of these previous observations by showing that minimizing a particular form of regularized reconstruction error yields a reconstruction function that locally characterizes the shape of the data-generating density. We show that the auto-encoder captures the score (derivative of the log-density with respect to the input). It contradicts previous interpretations of reconstruction error as an energy function. Unlike previous results, the theorems provided here are completely generic and do not depend on the parameterization of the auto-encoder: they show what the auto-encoder would tend to if given enough capacity and examples. These results are for a contractive training criterion we show to be similar to the denoising auto-encoder training criterion with small corruption noise, but with contraction applied on the whole reconstruction function rather than just encoder. Similarly to score matching, one can consider the proposed training criterion as a convenient alternative to maximum likelihood because it does not involve a partition function. Finally, we show how an approximate Metropolis-Hastings MCMC can be setup to recover samples from the estimated distribution, and this is confirmed in sampling experiments.

Keywords: auto-encoders, denoising auto-encoders, score matching, unsupervised representation learning, manifold learning, Markov chains, generative models

1. Introduction

Machine learning is about capturing aspects of the unknown distribution from which the observed data are sampled (the *data-generating distribution*). For many learning algorithms and in particular in *manifold learning*, the focus is on identifying the regions (sets of points) in the space of examples where this distribution concentrates, i.e., which configurations of the observed variables are plausible.

Unsupervised *representation-learning* algorithms try to characterize the data-generating distribution through the discovery of a set of features or latent variables whose variations capture most of the structure of the data-generating distribution. In recent years, a number of unsupervised feature learning algorithms have been proposed that are based on minimizing some form of *reconstruction error*, such as auto-encoder and sparse coding variants (Ol-

shausen and Field, 1997; Bengio et al., 2007; Ranzato et al., 2007; Jain and Seung, 2008; Ranzato et al., 2008; Vincent et al., 2008; Kavukcuoglu et al., 2009; Rifai et al., 2011b,a; Gregor et al., 2011). An auto-encoder reconstructs the input through two stages, an encoder function f, which outputs a learned representation $h = f(x)$ of an example x, and a decoder function g, such that $g(f(x)) \approx x$ for most x sampled from the data-generating distribution. These feature learning algorithms can be *stacked* to form deeper and more abstract representations. *Deep learning* algorithms learn multiple levels of representation, where the number of levels is data-dependent. There are theoretical arguments and much empirical evidence to suggest that when they are well-trained, deep learning algorithms (Hinton et al., 2006; Bengio, 2009; Lee et al., 2009; Salakhutdinov and Hinton, 2009; Bengio and Delalleau, 2011; Bengio et al., 2013b) can perform better than their shallow counterparts, both in terms of learning features for the purpose of classification tasks and for generating higher-quality samples.

Here we restrict ourselves to the case of continuous inputs $x \in \mathbb{R}^d$ with the data-generating distribution being associated with an unknown *target density* function, denoted p. Manifold learning algorithms assume that p is concentrated in regions of lower dimension (Cayton, 2005; Narayanan and Mitter, 2010), i.e., the training examples are by definition located very close to these high-density manifolds. In that context, the core objective of manifold learning algorithms is to identify where the density concentrates.

Some important questions remain concerning many of feature learning algorithms based on reconstruction error. Most importantly, *what is their training criterion learning about the input density*? Do these algorithms implicitly learn about the whole density or only some aspect? If they capture the essence of the target density, then can we formalize that link and in particular exploit it to *sample from the model*? The answers may help to establish that these algorithms actually learn *implicit density* models, which only define a density indirectly, e.g., through the estimation of statistics or through a generative procedure. These are the questions to which this paper contributes.

The paper is divided in two main sections, along with detailed appendices with proofs of the theorems. Section 2 makes a direct link between denoising auto-encoders (Vincent et al., 2008) and contractive auto-encoders (Rifai et al., 2011b), justifying the interest in the contractive training criterion studied in the rest of the paper. Section 3 is the main contribution and regards the following question: when minimizing that criterion, *what does an auto-encoder learn about the data-generating density*? The main answer is that it estimates the *score* (first derivative of the log-density), i.e., the direction in which density is increasing the most, which also corresponds to the *local mean*, which is the expected value in a small ball around the current location. It also estimates the Hessian (second derivative of the log-density).

Finally, Section 4 shows how having access to an estimator of the score can be exploited to estimate energy differences, and thus perform approximate MCMC sampling. This is achieved using a Metropolis-Hastings MCMC in which the energy differences between the proposal and the current state are approximated using the denoising auto-encoder. Experiments on artificial data sets show that a denoising auto-encoder can recover a good estimator of the data-generating distribution, when we compare the samples generated by the model with the training samples, projected into various 2-D views for visualization.

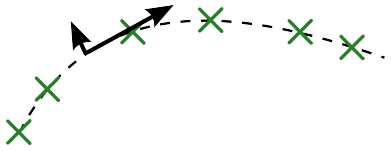


Figure 1: Regularization forces the auto-encoder to become less sensitive to the input, but minimizing reconstruction error forces it to remain sensitive to variations along the manifold of high density. Hence the representation and reconstruction end up capturing well variations on the manifold while mostly ignoring variations orthogonal to it.

2. Contractive and Denoising Auto-Encoders

Regularized auto-encoders (see Bengio et al. 2012b for a review and a longer exposition) capture the structure of the training distribution thanks to the productive opposition between reconstruction error and a regularizer. An auto-encoder maps inputs x to an internal representation (or code) $f(x)$ through the encoder function f, and then maps back $f(x)$ to the input space through a decoding function g. The composition of f and g is called the reconstruction function r, with $r(x) = g(f(x))$, and a reconstruction loss function ℓ penalizes the error made, with $r(x)$ viewed as a prediction of x. When the auto-encoder is regularized, e.g., via a sparsity regularizer, a contractive regularizer (detailed below), or a denoising form of regularization (that we find below to be very similar to a contractive regularizer), the regularizer basically attempts to make r (or f) as simple as possible, i.e., as constant as possible, as unresponsive to x as possible. It means that f has to throw away some information present in x, or at least represent it with less precision. On the other hand, to make reconstruction error small on the training set, examples that are neighbors on a high-density manifold must be represented with sufficiently different values of $f(x)$ or $r(x)$. Otherwise, it would not be possible to distinguish and hence correctly reconstruct these examples. It means that the derivatives of $f(x)$ or $r(x)$ in the x-directions along the manifold must remain large, while the derivatives (of f or r) in the x-directions orthogonal to the manifold can be made very small. This is illustrated in Figure 1. In the case of principal components analysis, one constrains the derivative to be exactly 0 in the directions orthogonal to the chosen projection directions, and around 1 in the chosen projection directions. In regularized auto-encoders, f is non-linear, meaning that it is allowed to choose different principal directions (those that are well represented, i.e., ideally the manifold tangent directions) at different x's, and this allows a regularized auto-encoder with non-linear encoder to capture non-linear manifolds. Figure 2 illustrates the extreme case when the regularization is very strong (r wants to be nearly constant where density is high) in the special case where the distribution is highly concentrated at three points (three training examples). It shows the compromise between obtaining the identity function at the training examples and having a flat r near the training examples, yielding a vector field $r(x) - x$ that points towards the high density points.

Here we show that the denoising auto-encoder (Vincent et al., 2008) with very small Gaussian corruption and squared error loss is actually a particular kind of contractive auto-encoder (Rifai et al., 2011b), contracting the whole auto-encoder reconstruction function

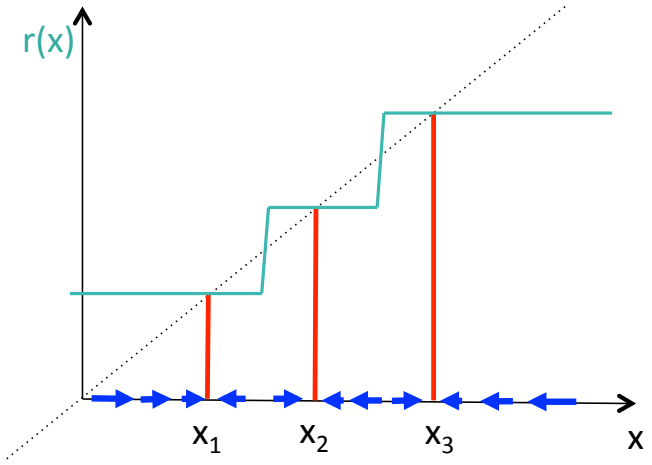


Figure 2: The reconstruction function $r(x)$ (in turquoise) which would be learned by a high-capacity auto-encoder on a 1-dimensional input, i.e., minimizing reconstruction error *at the training examples* x_i (with $r(x_i)$ in red) while trying to be as constant as possible otherwise. The figure is used to exaggerate and illustrate the effect of the regularizer (corresponding to a large σ^2 in the loss function $\mathcal{L}$ later described by Equation 6. The dotted line is the identity reconstruction (which might be obtained without the regularizer). The blue arrows shows the vector field of $r(x)-x$ pointing towards high density peaks as estimated by the model, and estimating the score (log-density derivative), as shown in this paper.

rather than just the encoder, whose contraction penalty coefficient is the magnitude of the perturbation. This was first suggested in Rifai et al. (2011b).

The contractive auto-encoder, or CAE (Rifai et al., 2011b), is a particular form of regularized auto-encoder which is trained to minimize the following regularized reconstruction error:

$$\mathcal{L}_{CAE} = \mathbb{E}\left[\ell(x, r(x)) + \lambda \left\| \frac{\partial f(x)}{\partial x} \right\|_F^2\right] \tag{1}$$

where $r(x) = g(f(x))$ and $||A||_F^2$ is the sum of the squares of the elements of A. Both the squared loss $\ell(x, r) = ||x-r||^2$ and the cross-entropy loss $\ell(x, r) = -x \log r - (1-x) \log(1-r)$ have been used, but here we focus our analysis on the squared loss because of the easier mathematical treatment it allows. Note that success in minimizing the CAE criterion strongly depends on the parameterization of f and g and in particular on the tied weights constraint used, with $f(x) = \text{sigmoid}(Wx + b)$ and $g(h) = \text{sigmoid}(W^T h + c)$. The above regularizing term forces f (as well as g, because of the tied weights) to be contractive, i.e., to have singular values less than 1.[1] Larger values of λ yield more contraction (smaller singular values) where it hurts reconstruction error the least, i.e., in the local directions where there are only little or no variations in the data. These typically are the directions orthogonal to the manifold of high density concentration, as illustrated in Figure 2.

1. Note that an auto-encoder without any regularization would tend to find many leading singular values near 1 in order to minimize reconstruction error, i.e., preserve input norm in all the directions of variation present in the data.

The denoising auto-encoder, or DAE (Vincent et al., 2008), is trained to minimize the following denoising criterion:

$$\mathcal{L}_{DAE} = \mathbb{E}\left[\ell(x, r(N(x)))\right] \tag{2}$$

where $N(x)$ is a stochastic corruption of x and the expectation is over the training distribution and the corruption noise source. Here we consider mostly the squared loss and Gaussian noise corruption, again because it is easier to handle them mathematically. In many cases, the exact same proofs can be applied to any kind of additive noise, but Gaussian noise serves as a good frame of reference.

Theorem 1 *Let p be the probability density function of the data. If we train a DAE using the expected quadratic loss and corruption noise $N(x) = x + \epsilon$ with*

$$\epsilon \sim \mathcal{N}\left(0, \sigma^2 I\right),$$

then the optimal reconstruction function $r^*(x)$ will be given by

$$r^*(x) = \frac{\mathbb{E}_\epsilon\left[p(x-\epsilon)(x-\epsilon)\right]}{\mathbb{E}_\epsilon\left[p(x-\epsilon)\right]} \tag{3}$$

for values of x where $p(x) \neq 0$.

Moreover, if we consider how the optimal reconstruction function $r^*_\sigma(x)$ behaves asymptotically as $\sigma \to 0$, we get that

$$r^*_\sigma(x) = x + \sigma^2 \frac{\partial \log p(x)}{\partial x} + o(\sigma^2) \quad as \quad \sigma \to 0. \tag{4}$$

The proof of this result is found in the Appendix. We make use of the small o notation throughout this paper and assume that the reader is familiar with asymptotic notation. In the context of Theorem 1, it has to be understood that all the other quantities except for σ are fixed when we study the effect of $\sigma \to 0$.

Equation 3 reveals that the optimal DAE reconstruction function at every point x is given by a kind of convolution involving the density function p, or weighted average from the points in the neighbourhood of x, depending on how we would like to view it. A higher noise level σ means that a larger neighbourhood of x is taken into account. Note that the total quantity of "mass" being included in the weighted average of the numerator of (3) is found again at the denominator.

Gaussian noise is a simple case in the sense that it is additive and symmetrical, so it avoids the complications that would occur when trying to integrate over the density of preimages x' such that $N(x') = x$ for a given x. The ratio of those quantities that we have in Equation 3, however, depends strongly on the decision that we made to minimize the expected square error.

When we look at the asymptotic behavior with Equation 4, the first thing to observe is that the leading term in the expansion of $r^*_\sigma(x)$ is x, and then the remainder goes to 0 as $\sigma \to 0$. When there is no noise left at all, it should be clear that the best reconstruction target for any value x would be that x itself.

We get something even more interesting if we look at the second term of Equation 4 because it gives us an estimator of the score from

$$\frac{\partial \log p(x)}{\partial x} = \left(r^*_\sigma(x) - x\right)/\sigma^2 + o(1) \quad \text{as} \quad \sigma \to 0. \tag{5}$$

This result is at the core of our paper. It is what allows us to start from a trained DAE, and then recover properties of the training density $p(x)$ that can be used to sample from $p(x)$.

Most of the asymptotic properties that we get by considering the limit as the Gaussian noise level σ goes to 0 could be derived from a family of noise distribution that approaches a point mass distribution in a relatively "nice" way.

An interesting connection with contractive auto-encoders can also be observed by using a Taylor expansion of the denoising auto-encoder loss and assuming only that $r_\sigma(x) = x + o(1)$ as $\sigma \to 0$. In that case we get the following proposition.

Proposition 1 *Let p be the probability density function of the data. Consider a DAE using the expected quadratic loss and corruption noise $N(x) = x + \epsilon$, with $\epsilon \sim \mathcal{N}\left(0, \sigma^2 I\right)$. If we assume that the non-parametric solutions $r_\sigma(x)$ satisfies*

$$r_\sigma(x) = x + o(1) \quad \textit{as} \quad \sigma \to 0,$$

then we can rewrite the loss as

$$\mathcal{L}_{DAE} = \mathbb{E}\left[\|r(x) - x\|_2^2 + \sigma^2 \left\| \frac{\partial r(x)}{\partial x} \right\|_F^2\right] + o(\sigma^2) \quad \textit{as} \quad \sigma \to 0.$$

The proof is in Appendix and uses a simple Taylor expansion around x.

Proposition 1 shows that *the DAE with small corruption of variance σ^2 is similar to a contractive auto-encoder with penalty coefficient $\lambda = \sigma^2$* but where the contraction is imposed explicitly on the whole reconstruction function $r(\cdot) = g(f(\cdot))$ rather than on $f(\cdot)$ alone.[2]

This analysis motivates the definition of the *reconstruction contractive auto-encoder* (RCAE), a variation of the CAE where loss function is instead the squared reconstruction loss plus contractive penalty on the reconstruction:

$$\mathcal{L}_{\text{RCAE}} = \mathbb{E}\left[\|r(x) - x\|_2^2 + \sigma^2 \left\| \frac{\partial r(x)}{\partial x} \right\|_F^2\right]. \tag{6}$$

This is an analytic version of the denoising criterion with small noise σ^2, and also corresponds to a contractive auto-encoder with contraction on both f and g, i.e., on r.

Because of the similarity between DAE and RCAE when taking $\lambda = \sigma^2$ and because the semantics of σ^2 is clearer (as a squared distance in input space), we will denote σ^2 for the penalty term coefficient in situations involving RCAE. For example, in the statement of

2. In the CAE there is a also a contractive effect on $g(\cdot)$ as a side effect of the parameterization with weights tied between $f(\cdot)$ and $g(\cdot)$.

Theorem 2, this σ^2 is just a positive constant; there is no notion of additive Gaussian noise, i.e., σ^2 does not explicitly refer to a variance, but using the notation σ^2 makes it easier to intuitively see the connection to the DAE setting.

The connection between DAE and RCAE established in Proposition 1 also serves as the basis for an alternative demonstration to Theorem 1 in which we study the asymptotic behavior of the RCAE solution. This result is contained in the following theorem.

Theorem 2 *Let p be a probability density function that is continuously differentiable once and with support $\mathbb{R}^d$ (i.e., $\forall x \in \mathbb{R}^d$ we have $p(x) \neq 0$). Let $\mathcal{L}_{\sigma^2}$ be the loss function defined by*

$$\mathcal{L}_{\sigma^2}(r) = \int_{\mathbb{R}^d} p(x) \left[\|r(x) - x\|_2^2 + \sigma^2 \left\| \frac{\partial r(x)}{\partial x} \right\|_F^2 \right] dx \tag{7}$$

for $r : \mathbb{R}^d \to \mathbb{R}^d$ assumed to be differentiable twice, and $0 \leq \sigma^2 \in \mathbb{R}$ used as factor to the penalty term.

Let $r^*_{\sigma^2}(x)$ denote the optimal function that minimizes $\mathcal{L}_{\sigma^2}$. Then we have that

$$r^*_{\sigma^2}(x) = x + \sigma^2 \frac{\partial \log p(x)}{\partial x} + o(\sigma^2) \quad as \quad \sigma^2 \to 0. \tag{8}$$

Moreover, we also have the following expression for the derivative

$$\frac{\partial r^*_{\sigma^2}(x)}{\partial x} = I + \sigma^2 \frac{\partial^2 \log p(x)}{\partial x^2} + o(\sigma^2) \quad as \quad \sigma^2 \to 0. \tag{9}$$

Both these asymptotic expansions are to be understood in a context where we consider $\left\{r^*_{\sigma^2}(x)\right\}_{\sigma^2 \geq 0}$ to be a family of optimal functions minimizing $\mathcal{L}_{\sigma^2}$ for their corresponding value of σ^2. The asymptotic expansions are applicable point-wise in x, that is, with any fixed x we look at the behavior as $\sigma^2 \to 0$.

The proof is given in the appendix and uses the Euler-Lagrange equations from the calculus of variations.

3. Minimizing the Loss to Recover Local Features of $p(\cdot)$

One of the central ideas of this paper is that in a non-parametric setting (without parametric constraints on r), we have an asymptotic formula (as the noise level $\sigma \to 0$) for the optimal reconstruction function for the DAE and RCAE that allows us to recover the score $\frac{\partial \log p(x)}{\partial x}$.

A DAE is trained with a method that knows nothing about p, except through the use of training samples to minimize a loss function, so it comes as a surprise that we can compute the score of p at any point x.

In the following subsections we explore the consequences and the practical aspect of this.

3.1 Empirical Loss

In an experimental setting, the expected loss (7) is replaced by the empirical loss

$$\hat{\mathcal{L}} = \frac{1}{N}\sum_{n=1}^{N}\left(\left\|r(x^{(n)}) - x^{(n)}\right\|_2^2 + \sigma^2 \left\| \left.\frac{\partial r(x)}{\partial x}\right|_{x=x^{(n)}} \right\|_F^2\right)$$

based on a sample $\left\{x^{(n)}\right\}_{n=1}^{N}$ drawn from $p(x)$.

Alternatively, the auto-encoder is trained online (by stochastic gradient updates) with a stream of examples $x^{(n)}$, which corresponds to performing stochastic gradient descent on the expected loss (7). In both cases we obtain an auto-encoder that approximately minimizes the expected loss.

An interesting question is the following: what can we infer from the data-generating density when given an auto-encoder reconstruction function $r(x)$?

The premise is that this auto-encoder $r(x)$ was trained to approximately minimize a loss function that has exactly the form of (7) for some $\sigma^2 > 0$. This is assumed to have been done through minimizing the empirical loss and the distribution p was only available indirectly through the samples $\left\{x^{(n)}\right\}_{n=1}^{N}$. We do not have access to p or to the samples. We have only $r(x)$ and maybe σ^2.

We will now discuss the usefulness of $r(x)$ based on different conditions such as the model capacity and the value of σ^2.

3.2 Perfect World Scenario

As a starting point, we will assume that we are in a perfect situation, i.e., with no constraint on r (non-parametric setting), an infinite amount of training data, and a perfect minimization. We will see what can be done to recover information about p in that ideal case. Afterwards, we will drop certain assumptions one by one and discuss the possible paths to getting back some information about p.

We use notation $r_{\sigma^2}(x)$ when we want to emphasize the fact that the value of $r(x)$ came from minimizing the loss with a certain fixed σ^2.

Suppose that $r_{\sigma^2}(x)$ was trained with an infinite sample drawn from p. Suppose also that it had infinite (or sufficient) model capacity and that it is capable of achieving the minimum of the loss function (7) while satisfying the requirement that $r(x)$ be twice differentiable. Suppose that we know the value of σ^2 and that we are working in a computing environment of arbitrary precision (i.e., no rounding errors).

As shown by Theorem 1 and Theorem 2, we would be able to get numerically the values of $\frac{\partial \log p(x)}{\partial x}$ at any point $x \in \mathbb{R}^d$ by simply evaluating

$$\frac{r_{\sigma^2}(x) - x}{\sigma^2} \rightarrow \frac{\partial \log p(x)}{\partial x} \quad \text{as} \quad \sigma^2 \rightarrow 0. \tag{10}$$

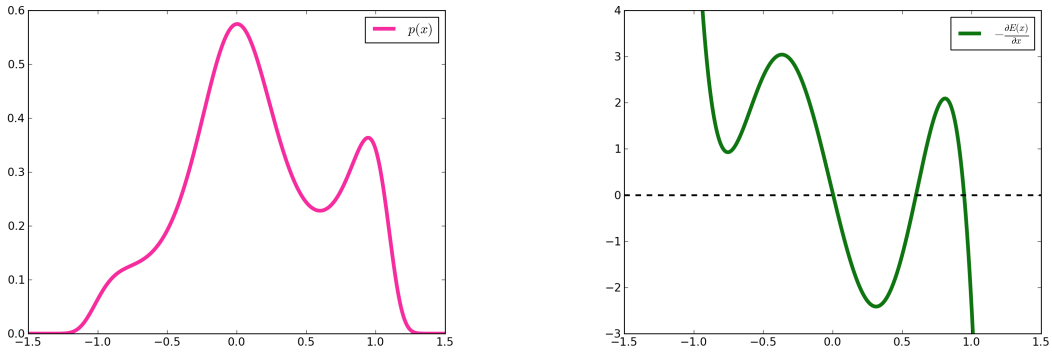
In the setup described, we do not get to pick values of σ^2 so as to take the limit $\sigma^2 \rightarrow 0$. However, it is assumed that σ^2 is already sufficiently small that the above quantity is close to $\frac{\partial \log p(x)}{\partial x}$ for all intents and purposes.

3.3 Simple Numerical Example

To give an example of this in one dimension, we will show what happens when we train a non-parametric model $\hat{r}(x)$ to minimize numerically the loss relative to $p(x)$. We train

both a DAE and an RCAE in this fashion by minimizing a discretized version of their losses defined by equations (2) and (6). The goal here is to show that, for either a DAE or RCAE, the approximation of the score that we get through Equation 5 gets arbitrarily close to the actual score $\frac{\partial}{\partial x} \log p(x)$ as $\sigma \to 0$.

The distribution $p(x)$ studied is shown in Figure 3 (left) and it was created to be simple enough to illustrate the mechanics. We plot $p(x)$ in Figure 3 (left) along with the score of $p(x)$ (right).



(a) $p(x) = \frac{1}{Z} \exp(-E(x))$ (b) $\frac{\partial}{\partial x} \log p(x) = -\frac{\partial}{\partial x} E(x)$

Figure 3: The density $p(x)$ and its score for a simple one-dimensional example.

The model $\hat{r}(x)$ is fitted by dividing the interval $[-1.5, 1.5]$ into $M = 1000$ partition points $x_1, \ldots, x_M$ evenly separated by a distance Δ. The discretized version of the RCAE loss function is

$$\sum_{i=1}^{M} p(x_i)\Delta \left(\hat{r}(x_i) - x_i\right)^2 + \sigma^2 \sum_{i=1}^{M-1} p(x_i)\Delta \left(\frac{\hat{r}(x_{i+1}) - \hat{r}(x_i)}{\Delta}\right)^2. \tag{11}$$

Every value $\hat{r}(x_i)$ for $i = 1, \ldots, M$ is treated as a free parameter. Setting to 0 the derivative with respect to the $\hat{r}(x_i)$ yields a system of linear equations in M unknowns that we can solve exactly. From that RCAE solution $\hat{r}$ we get an approximation of the score of p at each point x_i. A similar thing can be done for the DAE by using a discrete version of the exact solution (3) from Theorem 1. We now have two ways of approximating the score of p.

In Figure 4 we compare the approximations to the actual score of p for decreasingly small values of $\sigma \in \{1.00, 0.31, 0.16, 0.06\}$.

3.4 Vector Field Around a Manifold

We extend the experimentation of Section 3.3 to a 1-dimensional manifold in 2-D space, in which one can visualize $r(x) - x$ as a vector field, and we go from the non-parametric estimator of the previous section to an actual auto-encoder trained by numerically minimizing the regularized reconstruction error.

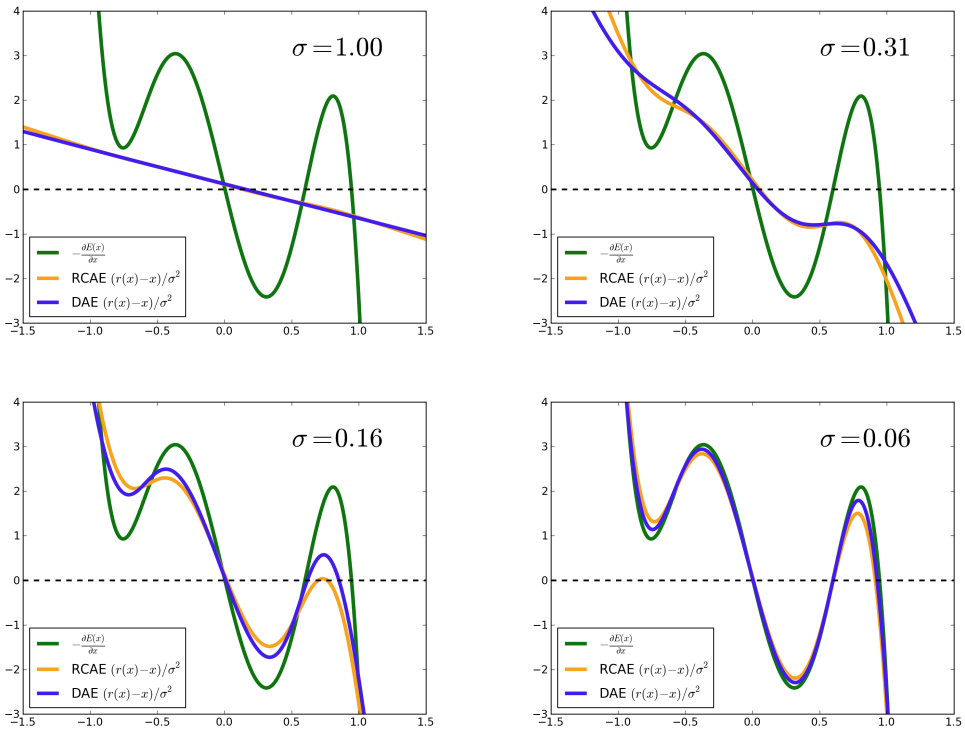


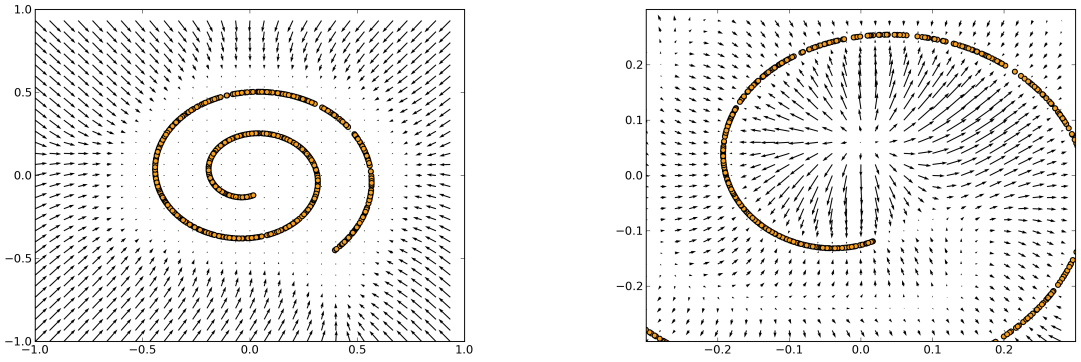
Figure 4: Comparing the approximation of the score of p given by discrete versions of optimally trained auto-encoders with infinite capacity. The approximations given by the RCAE are in orange while the approximations given by the DAE are in purple. The results are shown for decreasing values of $\sigma \in \{1.00, 0.31, 0.16, 0.06\}$ that have been selected for their visual appeal.
As expected, we see in that the RCAE (orange) and DAE (purple) approximations of the score are close to each other as predicted by Proposition 1. Moreover, they are also converging to the true score (green) as predicted by Theorem 1 and Theorem 2.

Two-dimensional data points (x, y) were generated along a spiral according to the following equations:

$$x = 0.04 \sin(t), \quad y = 0.04 \cos(t), \quad t \sim \text{Uniform}\,(3, 12)\,.$$

A denoising auto-encoder was trained with Gaussian corruption noise $\sigma = 0.01$. The encoder is $f(x) = \tanh(b + Wx)$ and the decoder is $g(h) = c + Vh$. The parameters (b, c, V, W) are optimized by BFGS to minimize the average squared error, using a fixed training set of 10 000 samples (i.e., the same corruption noises were sampled once and for all). We found better results with untied weights, and BFGS gave more accurate models than stochastic gradient descent. We used 1000 hidden units and ran BFGS for 1000 iterations.

The non-convexity of the problem makes it such that the solution found depends on the initialization parameters. The random corruption noise used can also influence the final outcome. Moreover, the fact that we are using a finite training sample size with reasonably small noise may allow for undesirable behavior of r in regions far away from the training samples. For those reasons, we trained the model multiple times and selected two of the most visually appealing outcomes. These are found in Figure 5 which features a more global perspective along with a close-up view.



(a) $r(x) - x$ vector field, acting as sink, zoomed out

(b) $r(x) - x$ vector field, close-up

Figure 5: The original 2-D data from the data-generating density $p(x)$ is plotted along with the vector field defined by the values of $r(x) - x$ for trained auto-encoders (corresponding to the estimation of the score $\frac{\partial \log p(x)}{\partial x}$).

Figure 5 shows the data along with the learned score function (shown as a vector field). We see that that the vector field points towards the nearest high-density point on the data manifold. The vector field is close to zero near the manifold (i.e., the reconstruction error is close to zero), also corresponding to peaks of the implicitly estimated density. The points on the manifolds play the role of sinks for the vector field. *Other places where reconstruction error may be low, but where the implicit density is not high, are sources of the vector field.* In Figure 5(b) we can see that we have that kind of behavior halfway between two sections of the manifold. This shows that reconstruction error plays a very different role as what was previously hypothesized: whereas in Ranzato et al. (2008) the reconstruction error was viewed as an *energy function*, our analysis suggests that in regularized auto-encoders, it is the norm of an approximate score, i.e., the derivative of the energy w.r.t. input. Note that the norm of the score should be small near training examples (corresponding to local maxima of density) but it could also be small at other places corresponding to *local minima of density*. This is indeed what happens in the spiral example shown. It may happen whenever there are high-density regions separated by a low-density region: tracing paths from one high-density region to another should cross a "median" lower-dimensional region (a manifold) where the density has a local maximum along the path direction. The reason such a median region is needed is because at these points the vectors $r(x) - x$ must change

sign: on one side of the median they point to one of the high-density regions while on the other side they point to the other, as clearly visible in Figure 5(b) between the arms of the spiral.

We believe that this analysis is valid not just for contractive and denoising auto-encoders, but for regularized auto-encoders in general. The intuition behind this statement can be firmed up by analyzing Figure 2: the score-like behavior of $r(x) - x$ arises simply out of the opposing forces of (a) trying to make $r(x) = x$ at the training examples and (b) trying to make $r(x)$ as regularized as possible (as close to a constant as possible).

Note that previous work (Rifai et al., 2012; Bengio et al., 2013b) has already shown that contractive auto-encoders (especially when they are stacked in a way similar to RBMs in a deep belief net) learn good models of high-dimensional data (such as images), and that these models can be used not just to obtain good representations for classification tasks but that good quality samples can be obtained from the model, by a random walk near the manifold of high-density. This was achieved by essentially following the vector field and adding noise along the way.

3.5 Missing σ^2

When we are in the same setting as in Section 3.2 but the value of σ^2 is unknown, we can modify (10) a bit and avoid dividing by σ^2. That is, for a trained reconstruction function $r(x)$ given to us we just take the quantity $r(x) - x$ and it should be approximately the score *up to a multiplicative constant*. We get that

$$r(x) - x \propto \frac{\partial \log p(x)}{\partial x}.$$

Equivalently, if one estimates the density via an energy function (minus the unnormalized log density), then $x - r(x)$ estimates the derivative of the energy function.

We still have to assume that σ^2 is small. Otherwise, if the unknown σ^2 is too large we might get a poor estimation of the score.

3.6 Limited Parameterization

We should also be concerned about the fact that $r(x) - x$ is trying to approximate $-\frac{\partial E(x)}{\partial x}$ as $\sigma^2 \to 0$ but we have not made any assumptions about the space of functions that r can represent when we are dealing with a specific implementation.

When using a certain parameterization of r such as the one from Section 3.3, there is no guarantee that the family of functions in which we select r each represent a conservative vector field (i.e., the gradient of a potential function). Even if we start from a density $p(x) \propto \exp(-E(x))$ and we have that $r(x) - x$ is very close to $-\frac{\partial}{\partial x}E(x)$ in terms of some given norm, there is not guarantee that there exists an associated function $E_0(x)$ for which $r(x) - x \propto -\frac{\partial}{\partial x}E_0(x)$ and $E_0(x) \approx E(x)$.

In fact, in many cases we can trivially show the non-existence of such a $E_0(x)$ by computing the curl of $r(x)$. The curl has to be equal to 0 everywhere if $r(x)$ is indeed the derivative of a potential function. We can omit the x terms from the computations because we can easily find its antiderivative by looking at $x = \frac{\partial}{\partial x} \|x\|_2^2$.

Conceptually, another way to see this is to argue that if such a function $E_0(x)$ existed, its second-order mixed derivatives should be equal. That is, we should have that

$$\frac{\partial^2 E_0(x)}{\partial x_i \partial x_j} = \frac{\partial^2 E_0(x)}{\partial x_j \partial x_i} \quad \forall i, j,$$

which is equivalent to

$$\frac{\partial r_i(x)}{\partial x_j} = \frac{\partial r_j(x)}{\partial x_i} \quad \forall i, j.$$

Again in the context of Section 3.3, with the parameterization used for that particular kind of denoising auto-encoder, this would yield the constraint that $V^T = W$. That is, unless we are using tied weights, we know that no such potential $E_0(x)$ exists, and yet when running the experiments from Section 3.3 we obtained much better results with untied weights. To make things worse, it can also be demonstrated that the energy function that we get from tied weights leads to a distribution that is not normalizable (it has a divergent integral over $\mathbb{R}^d$). In that sense, this suggests that we should not worry too much about the exact parameterization of the denoising auto-encoder as long as it has the required flexibility to approximate the optimal reconstruction function sufficiently well.

3.7 Relation to Denoising Score Matching

There is a connection between our results and previous research involving score matching for denoising auto-encoders. We will summarize here the existing results from Vincent (2011) and show that, whereas they have shown that denoising auto-encoders with a particular form estimated the score, our results extend this to a very large family of estimators (including the non-parametric case). This will provide some reassurance given some of the potential issues mentioned in Section 3.6.

Motivated by the analysis of denoising auto-encoders, the authors of Vincent (2011) are concerned with the case where we explicitly parameterize an energy function $\mathcal{E}(x)$, yielding an associated score function $\psi(x) = -\frac{\partial \mathcal{E}(x)}{\partial x}$ and we stochastically corrupt the original samples $x \sim p$ to obtain noisy samples $\tilde{x} \sim q_\sigma(\tilde{x}|x)$. In particular, the article analyzes the case where q_σ adds Gaussian noise of variance σ^2 to x. The main result is that minimizing the expected square difference between $\psi(\tilde{x})$ and the score of $q_\sigma(\tilde{x}|x)$,

$$E_{x,\tilde{x}}\left[\left\|\psi(\tilde{x}) - \frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}}\right\|^2\right],$$

is equivalent to performing *score matching* (Hyvärinen, 2005) with estimator $\psi(\tilde{x})$ and target density $q_\sigma(\tilde{x}) = \int q_\sigma(\tilde{x}|x)p(x)dx$, where $p(x)$ generates the training samples x. Note that when a finite training set is used, $q_\sigma(\tilde{x})$ is simply a smooth of the empirical distribution (e.g., the Parzen density with Gaussian kernel of width σ). When the corruption noise is Gaussian, $\frac{q_\sigma(\tilde{x}|x)}{\partial \tilde{x}} = \frac{x-\tilde{x}}{\sigma^2}$, from which we can deduce that if we define a reconstruction function

$$r(\tilde{x}) = \tilde{x} + \sigma^2 \psi(\tilde{x}), \tag{12}$$

then the above expectation is equivalent to

$$E_{x,\tilde{x}}\left[\left\|\frac{r(\tilde{x}) - \tilde{x}}{\sigma^2} - \frac{x - \tilde{x}}{\sigma^2}\right\|^2\right] = \frac{1}{\sigma^2} E_{x,\tilde{x}}\left[\|r(\tilde{x}) - x\|^2\right]$$

which is the denoising criterion. This says that when the reconstruction function r is parameterized so as to correspond to the score ψ of a model density (as per Equation 12, and where ψ is a derivative of some log-density), the denoising criterion on r with Gaussian corruption noise is equivalent to score matching with respect to a smooth of the data-generating density, i.e., a regularized form of score matching. Note that this regularization appears desirable, because matching the score of the empirical distribution (or an insufficiently smoothed version of it) could yield undesirable results when the training set is finite. Since score matching has been shown to be a consistent induction principle (Hyvärinen, 2005), it means that this *denoising score matching* (Vincent, 2011; Kingma and LeCun, 2010; Swersky et al., 2011) criterion recovers the underlying density, up to the smoothing induced by the noise of variance σ^2. By making σ^2 small, we can make the estimator arbitrarily good (and we would expect to want to do that as the amount of training data increases). Note the correspondence of this conclusion with the results presented here, which show (1) the equivalence between the RCAE's regularization coefficient and the DAE's noise variance σ^2, and (2) that minimizing the equivalent analytic criterion (based on a contraction penalty) estimates the score when σ^2 is small. The difference is that our result holds even when r is not parameterized as per Equation 12, i.e., is not forced to correspond with the score function of a density.

3.8 Estimating the Hessian

Since we have $\frac{r(x)-x}{\sigma^2}$ as an estimator of the score, we readily obtain that the Hessian of the log-density, can be estimated by the Jacobian of the reconstruction function minus the identity matrix:

$$\frac{\partial^2 \log p(x)}{\partial x^2} \approx (\frac{\partial r(x)}{\partial x} - I)/\sigma^2$$

as shown by Equation 9 of Theorem 2.

In spite of its simplicity, this result is interesting because it relates the derivative of the reconstruction function, i.e., a Jacobian matrix, with the second derivative of the log-density (or of the energy). This provides insights into the geometric interpretation of the reconstruction function when the density is concentrated near a manifold. In that case, near the manifold the score is nearly 0 because we are near a ridge of density, and the density's second derivative matrix tells us in which directions the first density remains close to zero or increases. The ridge directions correspond to staying on the manifold and along these directions we expect the second derivative to be close to 0. In the orthogonal directions, the log-density should decrease sharply while its first and second derivatives would be large in magnitude and negative in directions away from the manifold.

Returning to the above equation, keep in mind that in these derivations σ^2 is near 0 and $r(x)$ is near x, so that $\frac{\partial r(x)}{\partial x}$ is close to the identity. In particular, in the ridge (manifold) directions, we should expect $\frac{\partial r(x)}{\partial x}$ to be closer to the identity, which means that the reconstruction remains faithful ($r(x) = x$) when we move on the manifold, and this corresponds to the eigenvalues of $\frac{\partial r(x)}{\partial x}$ that are near 1, making the corresponding eigenvalues of $\frac{\partial^2 \log p(x)}{\partial x^2}$ near 0. On the other hand, in the directions orthogonal to the manifold, $\frac{\partial r(x)}{\partial x}$ should be smaller than 1, making the corresponding eigenvalues of $\frac{\partial^2 \log p(x)}{\partial x^2}$ negative.

Besides first and second derivatives of the density, other local properties of the density are its local mean and local covariance, discussed in the Appendix, Section D.

4. Sampling with Metropolis-Hastings

In this section we show how a technique to generate samples from a given denoising auto-encoder by using Metropolis-Hastings.

We start by explaining how to compute differences in energies Section 4.1, and then we use this in Section 4.2 to generate samples. We provide an experimental example and we discuss the potential problems that this method has.

This section serves as a demonstration that the main result of this paper, i.e., the connection between denoising auto-encoders and the score $\frac{\partial \log p(x)}{\partial x}$, is more than just an observation : it can have practical uses.

4.1 Estimating Energy Differences

One of the immediate consequences of Theorem 2 and Equation 10 is that, while we cannot easily recover the energy $E(x)$ itself, it is possible to approximate the energy difference $E(x^*) - E(x)$ between two states x and x^*. This can be done by using a first-order Taylor approximation

$$E(x^*) - E(x) = \frac{\partial E(x)}{\partial x}^T (x^* - x) + o(\|x^* - x\|).$$

To get a more accurate approximation, we can also use a path integral from x to x^* that we can discretize in sufficiently many steps. With a smooth path $\gamma(t) : [0, 1] \to \mathbb{R}^d$, assuming that γ stays in a region where our DAE/RCAE can be used to approximate $\frac{\partial E}{\partial x}$ well enough, we have that

$$E(x^*) - E(x) = \int_0^1 \left[\frac{\partial E}{\partial x}(\gamma(t))\right]^T \gamma'(t) dt. \tag{13}$$

The simplest way to discretize this path integral is to pick points $\{x_i\}_{i=1}^n$ spread at even distances on a straight line from $x_1 = x$ to $x_n = x^*$. We approximate (13) by

$$E(x^*) - E(x) \approx \frac{1}{n} \sum_{i=1}^{n} \left[\frac{\partial E}{\partial x}(x_i)\right]^T (x^* - x) \tag{14}$$

4.2 Sampling

With Equation 13 from Section 4.1 we can perform approximate sampling from the estimated distribution, using the score estimator to approximate energy differences which are needed in the Metropolis-Hastings accept/reject decision. Using a symmetric proposal $q(x^*|x)$, the acceptance ratio is

$$\alpha = \frac{p(x^*)}{p(x)} = \exp(-E(x^*) + E(x))$$

which can be computed with (13) or approximated with (14) as long as we trust that our DAE/RCAE was trained properly and has enough capacity to be a sufficiently good

estimator of $\frac{\partial E}{\partial x}$. An example of this process is shown in Figure 6 in which we sample from a density concentrated around a 1-d manifold embedded in a space of dimension 10. For this particular task, we have trained only DAEs and we are leaving RCAEs out of this exercise. Given that the data is roughly contained in the range $[-1.5, 1.5]$ along all dimensions, we selected a training noise level $\sigma_{\text{train}} = 0.1$ so that the noise would have an appreciable effect while still being relatively small. As required by Theorem 1, we have used isotropic Gaussian noise of variance σ^2_{train}.

The Metropolis-Hastings proposal $q(x^*|x) = \mathcal{N}(0, \sigma^2_{\text{MH}} I)$ has a noise parameter σ_{MH} that needs to be set. In the situation shown in Figure 6, we used $\sigma_{\text{MH}} = 0.1$. After some hyperparameter tweaking and exploring various scales for $\sigma_{\text{train}}, \sigma_{\text{MH}}$, we found that setting both to be 0.1 worked well.

When σ_{train} is too large, the DAE trained learns a "blurry" version of the density that fails to represent the details that we are interested in. The samples shown in Figure 6 are very convincing in terms of being drawn from a distribution that models well the original density. We have to keep in mind that Theorem 2 describes the behavior as $\sigma_{\text{train}} \to 0$ so we would expect that the estimator becomes worse when σ_{train} is taking on larger values. In this particular case with $\sigma_{\text{train}} = 0.1$, it seems that we are instead modeling something like the original density to which isotropic Gaussian noise of variance σ^2_{train} has been added.

In the other extreme, when σ_{train} is too small, the DAE is not exposed to any training example farther away from the density manifold. This can lead to various kinds of strange behaviors when the sampling algorithm falls into those regions and then has no idea what to do there and how to get back to the high-density regions. We come back to that topic in Section 4.3.

It would certainly be possible to pick both a very small value for $\sigma_{\text{train}} = \sigma_{\text{MH}} = 0.01$ to avoid the spurious maxima problem illustrated in Section 4.3. However, this leads to the same kind of mixing problems that any kind of MCMC algorithm has. Smaller values of σ_{MH} lead to higher acceptance ratios but worse mixing properties.

4.3 Spurious Maxima

There are two very real concerns with the sampling method discussed in Section 4.2. The first problem is with the mixing properties of MCMC and it is discussed in that section. The second issue is with spurious probability maxima resulting from inadequate training of the DAE. It happens when an auto-encoder lacks the capacity to model the density with enough precision, or when the training procedure ends up in a bad local minimum (in terms of the DAE parameters).

This is illustrated in Figure 7 where we show an example of a vector field $r(x) - x$ for a DAE that failed to properly learn the desired behavior in regions away from the spiral-shaped density.

5. Conclusion

Whereas auto-encoders have long been suspected of capturing information about the data-generating density, this work has clarified what some of them are actually doing, showing that they can actually implicitly recover the data-generating density altogether. We have shown that regularized auto-encoders such as the denoising auto-encoder and a form of con-

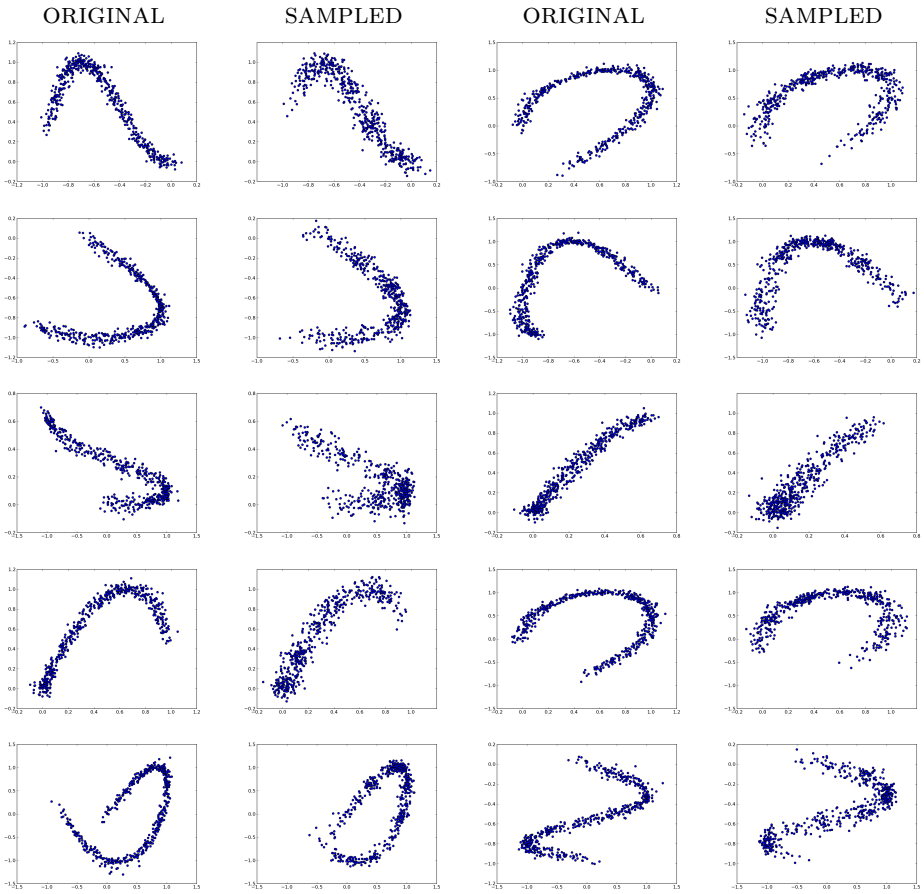
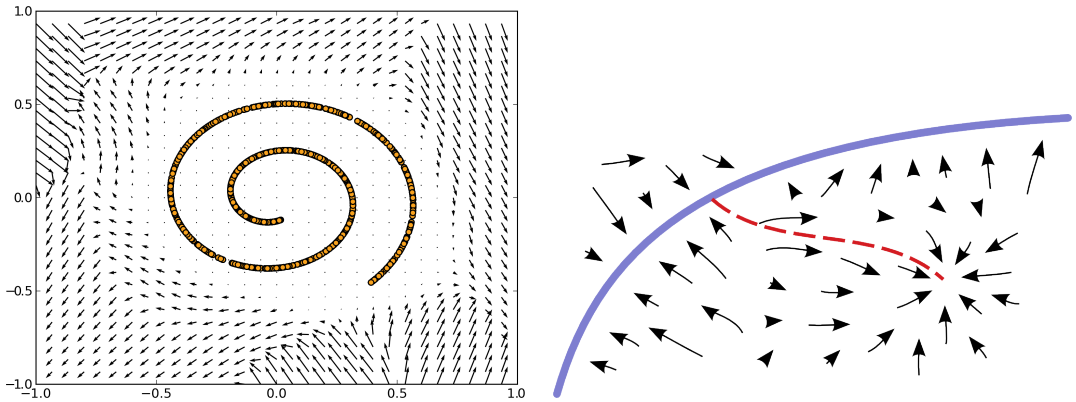


Figure 6: Samples drawn from the estimate of $\frac{\partial E}{\partial x}$ given by a DAE by the Metropolis-Hastings method presented in Section 4. By design, the data density distribution is concentrated along a 1-d manifold embedded in a space of dimension 10. This data can be visualized in the plots above by plotting pairs of dimensions $(x_0, x_1), \ldots, (x_8, x_9), (x_9, x_0)$, going in reading order from left to right and then line by line. For each pair of dimensions, we show side by side the original data (left) with the samples drawn (right).

tractive auto-encoder are closely related to each other and estimate local properties of the data-generating density: the first derivative (score) and second derivative of the log-density, as well as the local mean. This contradicts the previous interpretation of reconstruction error as being an energy function (Ranzato et al., 2008) but is consistent with our experimental findings. Our results do not require the reconstruction function to correspond to the derivative of an energy function as in Vincent (2011), but hold simply by virtue of minimizing the regularized reconstruction error training criterion. This suggests that minimizing a regularized reconstruction error may be an alternative to maximum likelihood for unsupervised learning, avoiding the need for MCMC in the inner loop of training, as in RBMs and deep Boltzmann machines, analogously to score matching (Hyvärinen, 2005;



(a) DAE misbehaving when away from manifold (b) sampling getting trapped into bad attractor

Figure 7: (a) On the left we show a $r(x) - x$ vector field similar to that of the earlier Figure 5. The density is concentrated along a spiral manifold and we should have the reconstruction function r bringing us back towards the density. In this case, it works well in the region close to the spiral (the magnitude of the vectors is so small that the arrows are shown as dots). However, things are out of control in the regions outside. This is because the level of noise used during training was so small that not enough of the training examples were found in those regions.
(b) On the right we sketch what may happen when we follow a sampling procedure as described in Section 4.2. We start in a region of high density (in purple) and we illustrate in red the trajectory that our samples may take. In that situation, the DAE/RCAE was not trained properly. The resulting vector field does not reflect the density accurately because it should not have this attractor (i.e., stable fixed point) outside of the manifold on which the density is concentrated. Conceptually, the sampling procedure visits that spurious attractor because it assumes that it corresponds to a region of high probability. In some cases, this effect is regrettable but not catastrophic, but in other situations we may end up with completely unusable samples. In the experiments, training with enough of the examples involving sufficiently large corruption noise typically eliminates that problem.

Vincent, 2011). Toy experiments have confirmed that a good estimator of the density can be obtained when this criterion is non-parametrically minimized. The experiments have also confirmed that an MCMC could be setup that approximately samples from the estimated model, by estimating energy differences to first order (which only requires the score) to perform approximate Metropolis-Hastings MCMC.

Many questions remain open and deserve further study. A big question is how to generalize these ideas to discrete data, since we have heavily relied on the notions of scores, i.e., of derivatives with respect to x. A natural extension of the notion of score that could be applied to discrete data is the notion of *relative energy*, or energy difference between a point x and a perturbation $\tilde{x}$ of x. This notion has already been successfully applied to obtain the equivalent of score matching for discrete models, namely ratio matching (Hyvärinen, 2007).

More generally, we would like to generalize to any form of reconstruction error (for example many implementations of auto-encoders use a Bernoulli cross-entropy as reconstruction loss function) and any (reasonable) form of corruption noise (many implementations use masking or salt-and-pepper noise, not just Gaussian noise). More fundamentally, the need to rely on $\sigma \to 0$ is troubling, and getting rid of this limitation would also be very useful. A possible solution to this limitation, as well as adding the ability to handle both discrete and continuous variables, has recently been proposed while this article was under review (Bengio et al., 2013a).

It would also be interesting to generalize the results presented here to other regularized auto-encoders besides the denoising and contractive types. In particular, the commonly used sparse auto-encoders seem to fit the qualitative pattern illustrated in 2 where a score-like vector field arises out of the opposing forces of minimizing reconstruction error and regularizing the auto-encoder.

We have mostly considered the harder case where the auto-encoder parameterization does not guarantee the existence of an analytic formulation of an energy function. It would be interesting to compare experimentally and study mathematically these two formulations to assess how much is lost (because the score function may be somehow inconsistent) or gained (because of the less constrained parameterization).

Acknowledgments

The authors thank Salah Rifai Max Welling, Yutian Chen and Pascal Vincent for fruitful discussions, and acknowledge the funding support from NSERC, Canada Research Chairs and CIFAR.

Appendix A. Exact Solution for DAE

There is a way to get an exact solution to the DAE loss (2) without assuming that $\sigma \to 0$ or that the noise is Gaussian (but still using the quadratic loss).

We let p be the density function of the data such that $\forall x \in \mathbb{R}^d, p(x) > 0$, and we use additive isotropic Gaussian noise of variance σ^2. We are in the non-parametric setting in which we are minimizing

$$\mathcal{L}_{\text{DAE}} = \int_{\mathbb{R}^d} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,\sigma^2 I)} \left[p(x) \left\| r(x+\epsilon) - x \right\|_2^2 \right] dx \tag{15}$$

with respect to the function $r : \mathbb{R}^d \to \mathbb{R}^d$.

By using an auxiliary variable $\tilde{x} = x + \epsilon$, we can rewrite this loss in a way that puts the quantity $r(\tilde{x})$ in focus and allows us to perform the minimization with respect to each choice of $r(\tilde{x})$ independently. That is, we have that

$$\mathcal{L}_{\text{DAE}} = \int_{\mathbb{R}^d} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,\sigma^2 I)} \left[p(\tilde{x} - \epsilon) \left\| r(\tilde{x}) - \tilde{x} + \epsilon \right\|_2^2 \right] d\tilde{x} \tag{16}$$

which can be differentiated with respect to the quantity $r(\tilde{x})$ and set to be equal to 0. Denoting the optimum by $r^*(\tilde{x})$, we get

$$0 = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,\sigma^2 I)} \left[p(\tilde{x} - \epsilon) r^*(\tilde{x}) - \tilde{x} + \epsilon \right] \tag{17}$$

$$\mathbb{E}_{\epsilon \sim \mathcal{N}(0,\sigma^2 I)} \left[p(\tilde{x} - \epsilon) r^*(\tilde{x}) \right] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,\sigma^2 I)} \left[p(\tilde{x} - \epsilon)(\tilde{x} - \epsilon) \right] \tag{18}$$

$$r^*(\tilde{x}) \quad = \quad \frac{\mathbb{E}_{\epsilon \sim \mathcal{N}(0,\sigma^2 I)} \left[p(\tilde{x} - \epsilon)(\tilde{x} - \epsilon) \right]}{\mathbb{E}_{\epsilon \sim \mathcal{N}(0,\sigma^2 I)} \left[p(\tilde{x} - \epsilon) \right]} \tag{19}$$

Conceptually, this means that the optimal DAE reconstruction function at every point $\tilde{x} \in \mathbb{R}^d$ is given by a kind of convolution involving the density function p, or weighted average from the points in the neighbourhood of $\tilde{x}$, depending on how we would like to view it. A higher noise level σ means that a larger neighbourhood of $\tilde{x}$ is taken into account. Note that the total quantity of "mass" being included in the weighted average of the numerator of (19) is found again at the denominator.

Appendix B. Relationship between Contractive Penalty and Denoising Criterion

Theorem 1 *When using corruption noise $N(x) = x + \epsilon$ with*

$$\epsilon \sim \mathcal{N}\left(0, \sigma^2 I\right),$$

the objective function $\mathcal{L}_{DAE}$ is

$$\mathcal{L}_{DAE} = \left(\mathbb{E}\left[\|x - r(x)\|^2 \right] + \sigma^2 \mathbb{E}\left[\left\| \frac{\partial r(x)}{\partial x} \right\|_F^2 \right] \right) + o(\sigma^2)$$

as $\sigma \to 0$.

Proof With a Taylor expansion around x we have that

$$r(x + \epsilon) = r(x) + \frac{\partial r(x)}{\partial x} \epsilon + o(\sigma^2).$$

Substituting this into $\mathcal{L}_{DAE}$ we have that

$$\begin{aligned}
\mathcal{L}_{DAE} &= \mathbb{E}\left[\tfrac{1}{2} \left\| x - \left(r(x) + \tfrac{\partial r(x)}{\partial x} \epsilon + o(\sigma^2) \right) \right\|^2 \right] \\
&= \left(\mathbb{E}\left[\|x - r(x)\|^2 \right] - 2E[\epsilon]^T \mathbb{E}\left[\tfrac{\partial r(x)}{\partial x}^T (x - r(x)) \right] \right) \\
&\quad + Tr\left(\mathbb{E}\left[\epsilon \epsilon^T \right] \mathbb{E}\left[\tfrac{\partial r(x)}{\partial x}^T \tfrac{\partial r(x)}{\partial x} \right] \right) + o(\sigma^2) \\
&= \tfrac{1}{2} \left(\mathbb{E}\left[\|x - r(x)\|^2 \right] + \sigma^2 \mathbb{E}\left[\left\| \tfrac{\partial r(x)}{\partial x} \right\|_F^2 \right] \right) + o(\sigma^2)
\end{aligned} \tag{20}$$

where in the second line we used the independence of the noise from x and properties of the trace, while in the last line we used $\mathbb{E}\left[\epsilon \epsilon^T \right] = \sigma^2 I$ and $\mathbb{E}[\epsilon] = 0$ by definition of ϵ. ■

Appendix C. Calculus of Variations

Theorem 2 *Let p be a probability density function that is continuously differentiable once and with support $\mathbb{R}^d$ (i.e., $\forall x \in \mathbb{R}^d$ we have $p(x) \neq 0$). Let $\mathcal{L}_{\sigma^2}$ be the loss function defined by*

$$\mathcal{L}_{\sigma^2}(r) = \int_{\mathbb{R}^d} p(x) \left[\|r(x) - x\|_2^2 + \sigma^2 \left\| \frac{\partial r(x)}{\partial x} \right\|_F^2 \right] dx$$

for $r : \mathbb{R}^d \to \mathbb{R}^d$ assumed to be differentiable twice, and $0 \leq \sigma^2 \in \mathbb{R}$ used as factor to the penalty term.

Let $r^*_{\sigma^2}(x)$ denote the optimal function that minimizes $\mathcal{L}_{\sigma^2}$. Then we have that

$$r^*_{\sigma^2}(x) = x + \sigma^2 \frac{\partial \log p(x)}{\partial x} + o(\sigma^2) \quad as \quad \sigma^2 \to 0.$$

Moreover, we also have the following expression for the derivative

$$\frac{\partial r^*_{\sigma^2}(x)}{\partial x} = I + \sigma^2 \frac{\partial^2 \log p(x)}{\partial x^2} + o(\sigma^2) \quad as \quad \sigma^2 \to 0.$$

Both these asymptotic expansions are to be understood in a context where we consider $\left\{r^*_{\sigma^2}(x)\right\}_{\sigma^2 \geq 0}$ to be a family of optimal functions minimizing $\mathcal{L}_{\sigma^2}$ for their corresponding value of σ^2. The asymptotic expansions are applicable point-wise in x, that is, with any fixed x we look at the behavior as $\sigma^2 \to 0$.

Proof

This proof is done in two parts. In the first part, the objective is to get to Equation 23 that has to be satisfied for the optimum solution.

We will leave out the σ^2 indices from the expressions involving $r(x)$ to make the notation lighter. We have a more important need for indices k in $r_k(x)$ that denote the d components of $r(x) \in \mathbb{R}^d$.

We treat σ^2 as given and constant for the first part of this proof.

In the second part we work out the asymptotic expansion in terms of σ^2. We again work with the implicit dependence of $r(x)$ on σ^2.

(part 1 of the proof)

We make use of the Euler-Lagrange equation from the Calculus of Variations. We would refer the reader to either (Dacorogna, 2004) or Wikipedia for more on the topic. Let

$$f(x_1, \ldots, x_n, r, r_{x_1}, \ldots, r_{x_n}) = p(x) \left[\|r(x) - x\|_2^2 + \sigma^2 \left\| \frac{\partial r(x)}{\partial x} \right\|_F^2 \right]$$

where $x = (x_1, \ldots, x_d)$, $r(x) = (r_1(x), \ldots, r_d(x))$ and $r_{x_i} = \frac{\partial f}{\partial x_i}$.

We can rewrite the loss $\mathcal{L}(r)$ more explicitly as

$$\begin{aligned} \mathcal{L}(r) &= \int_{\mathbb{R}^d} p(x) \left[\sum_{i=1}^d (r_i(x) - x_i)^2 + \sigma^2 \sum_{i=1}^d \sum_{j=1}^d \frac{\partial r_i(x)}{\partial x_j}^2 \right] dx \\ &= \sum_{i=1}^d \int_{\mathbb{R}^d} p(x) \left[(r_i(x) - x_i)^2 + \sigma^2 \sum_{j=1}^d \frac{\partial r_i(x)}{\partial x_j}^2 \right] dx \end{aligned} \tag{21}$$

to observe that the components $r_1(x), \ldots, r_d(x)$ can each be optimized separately.

The Euler-Lagrange equation to be satisfied at the optimal $r : \mathbb{R}^d \to \mathbb{R}^d$ is

$$\frac{\partial f}{\partial r} = \sum_{i=1}^{d} \frac{\partial}{\partial x_i} \frac{\partial f}{\partial r_{x_i}}.$$

In our situation, the expressions from that equation are given by

$$\frac{\partial f}{\partial r} = 2(r(x) - x)p(x)$$

$$\frac{\partial f}{\partial r_{x_i}} = 2\sigma^2 p(x) \left[\begin{array}{cccc} \frac{\partial r_1}{\partial x_i} & \frac{\partial r_2}{\partial x_i} & \cdots & \frac{\partial r_d}{\partial x_i} \end{array} \right]^T$$

$$\begin{aligned} \frac{\partial}{\partial x_i} \left(\frac{\partial f}{\partial r_{x_i}} \right) &= 2\sigma^2 \frac{\partial p(x)}{\partial x_i} \left[\begin{array}{cccc} \frac{\partial r_1}{\partial x_i} & \frac{\partial r_2}{\partial x_i} & \cdots & \frac{\partial r_d}{\partial x_i} \end{array} \right]^T \\ &\quad + 2\sigma^2 p(x) \left[\begin{array}{cccc} \frac{\partial^2 r_1}{\partial x_i^2} & \frac{\partial^2 r_2}{\partial x_i^2} & \cdots & \frac{\partial^2 r_d}{\partial x_i^2} \end{array} \right]^T \end{aligned}$$

and the equality to be satisfied at the optimum becomes

$$(r(x) - x)p(x) = \sigma^2 \sum_{i=1}^{d} \left[\begin{array}{c} \frac{\partial p(x)}{\partial x_i} \frac{\partial r_1}{\partial x_i} + p(x) \frac{\partial^2 r_1}{\partial x_i^2} \\ \vdots \\ \frac{\partial p(x)}{\partial x_i} \frac{\partial r_d}{\partial x_i} + p(x) \frac{\partial^2 r_d}{\partial x_i^2} \end{array} \right]. \tag{22}$$

As Equation 21 hinted, the expression (22) can be decomposed into the different components $r_k(x) : \mathbb{R}^d \to \mathbb{R}$ that make r. For $k = 1, \ldots, d$ we get

$$(r_k(x) - x_k)p(x) = \sigma^2 \sum_{i=1}^{d} \left(\frac{\partial p(x)}{\partial x_i} \frac{\partial r_k(x)}{\partial x_i} + p(x) \frac{\partial^2 r_k(x)}{\partial x_i^2} \right).$$

As $p(x) \neq 0$ by hypothesis, we can divide all the terms by $p(x)$ and note that $\frac{\partial p(x)}{\partial x_i} / p(x) = \frac{\partial \log p(x)}{\partial x_i}$.

We get

$$r_k(x) - x_k = \sigma^2 \sum_{i=1}^{d} \left(\frac{\partial \log p(x)}{\partial x_i} \frac{\partial r_k(x)}{\partial x_i} + \frac{\partial^2 r_k(x)}{\partial x_i^2} \right). \tag{23}$$

This first thing to observe is that when $\sigma^2 = 0$ the solution is just $r_k(x) = x_k$, which translates into $r(x) = x$. This is not a surprise because it represents the perfect reconstruction value that we get when we the penalty term vanishes in the loss function.

(part 2 of the proof)

This linear partial differential Equation 23 can be used as a recursive relation for $r_k(x)$ to obtain a Taylor series in σ^2. The goal is to obtain an expression of the form

$$r(x) = x + \sigma^2 h(x) + o(\sigma^2) \quad \text{as} \quad \sigma^2 \to 0 \tag{24}$$

where we can solve for $h(x)$ and for which we also have that

$$\frac{\partial r(x)}{\partial x} = I + \sigma^2 \frac{\partial h(x)}{\partial x} + o(\sigma^2) \quad \text{as} \quad \sigma^2 \to 0.$$

We can substitute in the right-hand side of Equation 24 the value for $r_k(x)$ that we get from Equation 24 itself. This substitution would be pointless in any other situation where we are not trying to get a power series in terms of σ^2 around 0.

$$\begin{aligned}
r_k(x) &= x_k \quad + \sigma^2 \sum_{i=1}^{d} \left(\frac{\partial \log p(x)}{\partial x_i} \frac{\partial r_k(x)}{\partial x_i} + \frac{\partial^2 r_k(x)}{\partial x_i^2} \right) \\
&= x_k \quad + \sigma^2 \sum_{i=1}^{d} \left(\frac{\partial \log p(x)}{\partial x_i} \frac{\partial}{\partial x_i} \left(x_k + \sigma^2 \sum_{j=1}^{d} \left(\frac{\partial \log p(x)}{\partial x_j} \frac{\partial r_k(x)}{\partial x_j} + \frac{\partial^2 r_k(x)}{\partial x_j^2} \right) \right) \right) \\
&\qquad + \sigma^2 \sum_{i=1}^{d} \frac{\partial^2 r_k(x)}{\partial x_i^2} \\
&= x_k \quad + \sigma^2 \sum_{i=1}^{d} \frac{\partial \log p(x)}{\partial x_i} \mathbb{I}(i = k) + \sigma^2 \sum_{i=1}^{d} \frac{\partial^2 r_k(x)}{\partial x_i^2} \\
&\qquad + {\sigma^2}^2 \sum_{i=1}^{d} \sum_{j=1}^{d} \left(\frac{\partial \log p(x)}{\partial x_i} \frac{\partial}{\partial x_i} \left(\frac{\partial \log p(x)}{\partial x_j} \frac{\partial r_k(x)}{\partial x_j} + \frac{\partial^2 r_k(x)}{\partial x_j^2} \right) \right) \\
r_k(x) &= x_k \quad + \sigma^2 \frac{\partial \log p(x)}{\partial x_k} + \sigma^2 \sum_{i=1}^{d} \frac{\partial^2 r_k(x)}{\partial x_i^2} + {\sigma^2}^2 \rho(\sigma^2, x)
\end{aligned}$$

Now we would like to get rid of that $\sigma^2 \sum_{i=1}^{d} \frac{\partial^2 r_k(x)}{\partial x_i^2}$ term by showing that it is a term that involves only powers of ${\sigma^2}^2$ or higher. We get this by showing what we get by differentiating the expression for $r_k(x)$ in line (25) twice with respect to some l.

$$\frac{\partial r_k(x)}{\partial x_l} = \mathbb{I}(i = l) + \sigma^2 \frac{\partial^2 \log p(x)}{\partial x_l \partial x_k} + \sigma^2 \frac{\partial}{\partial x_l} \left(\sum_{i=1}^{d} \frac{\partial^2 r_k(x)}{\partial x_i^2} + \sigma^2 \rho(\sigma^2, x) \right)$$

$$\frac{\partial^2 r_k(x)}{\partial x_l^2} = \sigma^2 \frac{\partial^3 \log p(x)}{\partial x_l^2 \partial x_k} + \sigma^2 \frac{\partial}{\partial x_l^2} \left(\sum_{i=1}^{d} \frac{\partial^2 r_k(x)}{\partial x_i^2} + \sigma^2 \rho(\sigma^2, x) \right)$$

Since σ^2 is a common factor in all the terms of the expression of $\frac{\partial^2 r_k(x)}{\partial x_l^2}$ we get what we needed. That is,

$$r_k(x) = x_k + \sigma^2 \frac{\partial \log p(x)}{\partial x_k} + {\sigma^2}^2 \eta(\sigma^2, x).$$

This shows that

$$r(x) = x + \sigma^2 \frac{\partial \log p(x)}{\partial x} + o(\sigma^2) \quad \text{as} \quad \sigma^2 \to 0$$

and

$$\frac{\partial r(x)}{\partial x} = I + \sigma^2 \frac{\partial^2 \log p(x)}{\partial x^2} + o(\sigma^2) \quad \text{as} \quad \sigma^2 \to 0$$

which completes the proof. ■

Appendix D. Local Mean

In preliminary work (Bengio et al., 2012a), we studied how the optimal reconstruction could possibly estimate so-called local moments. We revisit this question here, with more appealing and precise results.

What previous work on denoising and contractive auto-encoders suggest is that regularized auto-encoders can *capture the local structure of the density* through the value of the encoding (or reconstruction) function and its derivative. In particular, Rifai et al. (2012); Bengio et al. (2012a) argue that the first and second derivatives tell us in which directions it makes sense to randomly move while preserving or increasing the density, which may be used to justify sampling procedures. This motivates us here to study so-called local moments as captured by the auto-encoder, and in particular the local mean, following the definitions introduced in Bengio et al. (2012a).

D.1 Definitions for Local Distributions

Let p be a continuous probability density function with support $\mathbb{R}^d$. That is, $\forall x \in \mathbb{R}^d$ we have that $p(x) \neq 0$. We define below the notion of a *local ball* $B_\delta(x_0)$, along with an associated *local density*, which is the normalized product of p with the indicator for the ball:

$$\begin{aligned} B_\delta(x_0) &= \{x \quad \text{s.t.} \quad \|x - x_0\|_2 < \delta\} \\ Z_\delta(x_0) &= \int_{B_\delta(x_0)} p(x)dx \\ p_\delta(x|x_0) &= \frac{1}{Z_\delta(x_0)} p(x) \mathbb{I}\left(x \in B_\delta(x_0)\right) \end{aligned}$$

where $Z_\delta(x_0)$ is the normalizing constant required to make $p_\delta(x|x_0)$ a valid pdf for a distribution centered on x_0. The support of $p_\delta(x|x_0)$ is the ball of radius δ around x_0 denoted by $B_\delta(x_0)$. We stick to the 2-norm in terms of defining the balls $B_\delta(x_0)$ used, but everything could be rewritten in terms of another p-norm to have slightly different formulas.

We use the following notation for what will be referred to as the first two *local moments* (i.e., local mean and local covariance) of the random variable described by $p_\delta(x|x_0)$.

$$m_\delta(x_0) \stackrel{def}{=} \int_{\mathbb{R}^d} x p_\delta(x|x_0)dx$$

$$C_\delta(x_0) \stackrel{def}{=} \int_{\mathbb{R}^d} (x - m_\delta(x_0))(x - m_\delta(x_0))^T p_\delta(x|x_0)dx$$

Based on these definitions, one can prove the following theorem.

Theorem 3 *Let p be of class C^3 and represent a probability density function. Let $x_0 \in \mathbb{R}^d$ with $p(x_0) \neq 0$. Then we have that*

$$m_\delta(x_0) = x_0 + \delta^2 \frac{1}{d+2} \left. \frac{\partial \log p(x)}{\partial x} \right|_{x_0} + o\left(\delta^3\right).$$

This links the local mean of a density with the score associated with that density. Combining this theorem with Theorem 2, we obtain that the optimal reconstruction function $r^*(\cdot)$ also estimates the local mean:

$$m_\delta(x) - x = \frac{\delta^2}{\sigma^2(d+2)} \left(r^*(x) - x\right) \; + A(\delta) + \delta^2 B(\sigma^2) \tag{25}$$

for error terms $A(\delta), B(\sigma^2)$ such that

$$\begin{aligned} A(\delta) &\in o(\delta^3) \quad \text{as} \quad \delta \to 0, \\ B(\sigma^2) &\in o(1) \quad \text{as} \quad \sigma^2 \to 0. \end{aligned}$$

This means that we can loosely estimate the *direction* to the local mean by the direction of the reconstruction:

$$m_\delta(x) - x \;\; \propto \;\; r^*(x) - x. \tag{26}$$

Appendix E. Asymptotic formulas for localized moments

Proposition 4 *Let p be of class C^2 and let $x_0 \in \mathbb{R}^d$. Then we have that*

$$Z_\delta(x_0) = \delta^d \frac{\pi^{d/2}}{\Gamma\left(1 + d/2\right)} \left[p(x_0) + \delta^2 \frac{Tr(H(x_0))}{2(d+2)} + o(\delta^3) \right]$$

where $H(x_0) = \left. \frac{\partial^2 p(x)}{\partial x^2} \right|_{x=x_0}$. Moreover, we have that

$$\frac{1}{Z_\delta(x_0)} = \delta^{-d} \frac{\Gamma\left(1 + d/2\right)}{\pi^{d/2}} \left[\frac{1}{p(x_0)} - \delta^2 \frac{1}{p(x_0)^2} \frac{Tr(H(x_0))}{2(d+2)} + o(\delta^3) \right].$$

Proof

$$\begin{aligned} Z_\delta(x_0) &= \int_{B_\delta(x_0)} \left[p(x_0) + \left. \frac{\partial p(x)}{\partial x} \right|_{x_0} (x - x_0) + \frac{1}{2!}(x - x_0)^T H(x_0)(x - x_0) \right. \\ &\quad \left. + \frac{1}{3!} D^{(3)} p(x_0)(x - x_0) + o(\delta^3) \right] dx \\ &= p(x_0) \int_{B_\delta(x_0)} dx \; + \; 0 \; + \; \frac{1}{2} \int_{B_\delta(x_0)} (x - x_0)^T H(x_0)(x - x_0) dx \; + \; 0 \; + \; o(\delta^{d+3}) \\ &= p(x_0) \delta^d \frac{\pi^{d/2}}{\Gamma\left(1 + d/2\right)} + \delta^{d+2} \frac{\pi^{d/2}}{4\Gamma\left(2 + d/2\right)} \mathrm{Tr}\left(H(x_0)\right) + o(\delta^{d+3}) \\ &= \delta^d \frac{\pi^{d/2}}{\Gamma\left(1 + d/2\right)} \left[p(x_0) + \delta^2 \frac{\mathrm{Tr}(H(x_0))}{2(d+2)} + o(\delta^3) \right] \end{aligned}$$

We use Proposition 10 to get that trace come up from the integral involving $H(x_0)$. The expression for $1/Z_\delta(x_0)$ comes from the fact that, for any $a, b > 0$ we have that

$$\begin{aligned}\frac{1}{a + b\delta^2 + o(\delta^3)} &= \frac{a^{-1}}{1 + \frac{b}{a}\delta^2 + o(\delta^3)} = \frac{1}{a}\left(1 - (\frac{b}{a}\delta^2 + o(\delta^3)) + o(\delta^4)\right) \\ &= \frac{1}{a} - \frac{b}{a^2}\delta^2 + o(\delta^3) \quad \text{as } \delta \to 0.\end{aligned}$$

by using the classic result from geometric series where $\frac{1}{1+r} = 1 - r + r^2 - \ldots$ for $|r| < 1$.
Now we just apply this to

$$\frac{1}{Z_\delta(x_0)} = \delta^{-d}\frac{\Gamma\left(1 + d/2\right)}{\pi^{d/2}} \frac{1}{\left[p(x_0) + \delta^2 \frac{\mathrm{Tr}(H(x_0))}{2(d+2)} + o(\delta^3)\right]}$$

and get the expected result.

■

Theorem 5 *Let p be of class C^3 and represent a probability density function. Let $x_0 \in \mathbb{R}^d$ with $p(x_0) \neq 0$. Then we have that*

$$m_\delta(x_0) = x_0 + \delta^2 \frac{1}{d+2} \left.\frac{\partial \log p(x)}{\partial x}\right|_{x_0} + o\left(\delta^3\right).$$

Proof

The leading term in the expression for $m_\delta(x_0)$ is obtained by transforming the x in the integral into a $x - x_0$ to make the integral easier to integrate.

$$m_\delta(x_0) = \frac{1}{Z_\delta(x_0)} \int_{B_\delta(x_0)} xp(x)dx = x_0 + \frac{1}{z_\delta(x_0)} \int_{B_\delta(x_0)} (x - x_0)p(x)dx.$$

Now using the Taylor expansion around x_0

$$\begin{aligned}m_\delta(x_0) &= x_0 + \frac{1}{Z_\delta(x_0)} \int_{B_\delta(x_0)} (x - x_0)\left[p(x_0) + \left.\frac{\partial p(x)}{\partial x}\right|_{x_0} (x - x_0)\right. \\ &\quad \left. + \frac{1}{2}(x - x_0)^T \left.\frac{\partial^2 p(x)}{\partial x^2}\right|_{x_0} (x - x_0) + o(\|x - x_0\|^2)\right] dx.\end{aligned}$$

Remember that $\int_{B_\delta(x_0)} f(x)dx = 0$ whenever we have a function f is anti-symmetrical (or "odd") relative to the point x_0 (i.e., $f(x - x_0) = f(-x - x_0)$). This applies to the terms $(x - x_0)p(x_0)$ and $(x - x_0)(x - x_0) \left.\frac{\partial^2 p(x)}{\partial x^2}\right|_{x=x_0} (x - x_0)^T$. Hence we use Proposition 9 to get

$$\begin{aligned} m_\delta(x_0) &= x_0 + \frac{1}{Z_\delta(x_0)} \int_{B_\delta(x_0)} \left[(x - x_0)^T \left.\frac{\partial p(x)}{\partial x}\right|_{x_0} (x - x_0) + o(\|x - x_0\|^3) \right] dx \\ &= x_0 + \frac{1}{Z_\delta(x_0)} \left(\delta^{d+2} \frac{\pi^{\frac{d}{2}}}{2\Gamma\left(2 + \frac{d}{2}\right)} \right) \left.\frac{\partial p(x)}{\partial x}\right|_{x_0} + o(\delta^3). \end{aligned}$$

Now, looking at the coefficient in front of $\left.\frac{\partial p(x)}{\partial x}\right|_{x_0}$ in the first term, we can use Proposition 4 to rewrite it as

$$\begin{aligned} & \frac{1}{Z_\delta(x_0)} \left(\delta^{d+2} \frac{\pi^{\frac{d}{2}}}{2\Gamma\left(2 + \frac{d}{2}\right)} \right) \\ = \; & \delta^{-d} \frac{\Gamma\left(1 + d/2\right)}{\pi^{d/2}} \left[\frac{1}{p(x_0)} - \delta^2 \frac{1}{p(x_0)^2} \frac{\mathrm{Tr}(H(x_0))}{2(d+2)} + o(\delta^3) \right] \delta^{d+2} \frac{\pi^{\frac{d}{2}}}{2\Gamma\left(2 + \frac{d}{2}\right)} \\ = \; & \delta^2 \frac{\Gamma\left(1 + \frac{d}{2}\right)}{2\Gamma\left(2 + \frac{d}{2}\right)} \left[\frac{1}{p(x_0)} - \delta^2 \frac{1}{p(x_0)^2} \frac{\mathrm{Tr}(H(x_0))}{2(d+2)} + o(\delta^3) \right] = \delta^2 \frac{1}{p(x_0)} \frac{1}{d+2} + o(\delta^3). \end{aligned}$$

There is no reason the keep the $-\delta^4 \frac{\Gamma\left(1+\frac{d}{2}\right)}{2\Gamma\left(2+\frac{d}{2}\right)} \frac{1}{p(x_0)^2} \frac{\mathrm{Tr}(H(x_0))}{2(d+2)}$ in the above expression because the asymptotic error from the remainder term in the main expression is $o(\delta^3)$. That would swallow our exact expression for δ^4 and make it useless.

We end up with

$$m_\delta(x_0) = x_0 + \delta^2 \frac{1}{d+2} \left.\frac{\partial \log p(x)}{\partial x}\right|_{x_0} + o(\delta^3).$$

■

Appendix F. Integration on balls and spheres

This result comes from *Multi-dimensional Integration : Scary Calculus Problems* from Tim Reluga (who got the results from *How to integrate a polynomial over a sphere* by Gerald B. Folland).

Theorem 6 *Let* $B = \left\{ x \in \mathbb{R}^d \,\middle|\, \sum_{j=1}^d x_j^2 \leq 1 \right\}$ *be the ball of radius* 1 *around the origin. Then*

$$\int_B \prod_{j=1}^d |x_j|^{a_j} \, dx = \frac{\prod \Gamma\left(\frac{a_j+1}{2}\right)}{\Gamma\left(1 + \frac{d}{2} + \frac{1}{2}\sum a_j\right)}$$

for any real numbers $a_j \geq 0$.

Corollary 7 *Let B be the ball of radius 1 around the origin. Then*

$$\int_B \prod_{j=1}^{d} x_j^{a_j} dx = \begin{cases} \frac{\prod \Gamma\left(\frac{a_j+1}{2}\right)}{\Gamma\left(1+\frac{d}{2}+\frac{1}{2}\sum a_j\right)} & \text{if all the } a_j \text{ are even integers} \\ 0 & \text{otherwise} \end{cases}$$

for any non-negative integers $a_j \geq 0$. Note the absence of the absolute values put on the $x_j^{a_j}$ terms.

Corollary 8 *Let $B_\delta(0) \subset \mathbb{R}^d$ be the ball of radius δ around the origin. Then*

$$\int_{B_\delta(0)} \prod_{j=1}^{d} x_j^{a_j} dx = \begin{cases} \delta^{d+\sum a_j} \frac{\prod \Gamma\left(\frac{a_j+1}{2}\right)}{\Gamma\left(1+\frac{d}{2}+\frac{1}{2}\sum a_j\right)} & \text{if all the } a_j \text{are even integers} \\ 0 & \text{otherwise} \end{cases}$$

for any non-negative integers $a_j \geq 0$. Note the absence of the absolute values on the $x_j^{a_j}$ terms.

Proof

We take the theorem as given and concentrate here on justifying the two corollaries.

Note how in Corollary 7 we dropped the absolute values that were in the original Theorem 6. In situations where at least one a_j is odd, we have that the function $f(x) = \prod_{j=1}^{d} x_j^{a_j}$ becomes odd in the sense that $f(-x) = -f(x)$. Because of the symmetrical nature of the integration on the unit ball, we get that the integral is 0 as a result of cancellations.

For Corollary 8, we can rewrite the integral by changing the domain with $y_j = x_j/\delta$ so that

$$\delta^{-\sum a_j} \int_{B_\delta(0)} \prod_{j=1}^{d} x_j^{a_j} dx = \int_{B_\delta(0)} \prod_{j=1}^{d} (x_j/\delta)^{a_j} \, dx = \int_{B_1(0)} \prod_{j=1}^{d} y^{a_j} \delta^d dy.$$

We pull out the δ^d that we got from the determinant of the Jacobian when changing from dx to dy and Corollary 8 follows.

■

Proposition 9 *Let $v \in \mathbb{R}^d$ and let $B_\delta(0) \subset \mathbb{R}^d$ be the ball of radius δ around the origin. Then*

$$\int_{B_\delta(0)} y < v, y > dy = \left(\delta^{d+2} \frac{\pi^{\frac{d}{2}}}{2\Gamma\left(2+\frac{d}{2}\right)} \right) v$$

where $< v, y >$ is the usual dot product.

Proof

We have that

$$y < v, y > \quad = \quad \begin{bmatrix} v_1 y_1^2 \\ \vdots \\ v_d y_d^2 \end{bmatrix}$$

which is decomposable into d component-wise applications of Corollary 8. This yields the expected result with the constant obtained from $\Gamma\left(\frac{3}{2}\right) = \frac{1}{2}\Gamma\left(\frac{1}{2}\right) = \frac{1}{2}\sqrt{\pi}$.

■

Proposition 10 *Let $H \in \mathbb{R}^{d \times d}$ and let $B_\delta(x_0) \subset \mathbb{R}^d$ be the ball of radius δ around $x_0 \in \mathbb{R}^d$. Then*

$$\int_{B_\delta(x_0)} (x - x_0)^T H (x - x_0) dx = \delta^{d+2} \frac{\pi^{d/2}}{2\Gamma\left(2 + d/2\right)} \, trace\left(H\right).$$

Proof

First, by substituting $y = \left(x - x_0\right)/\delta$ we have that this is equivalent to showing that

$$\int_{B_1(0)} y^T H y dy = \frac{\pi^{d/2}}{2\Gamma\left(2 + d/2\right)} \operatorname{trace}\left(H\right).$$

This integral yields a real number which can be written as

$$\int_{B_1(0)} y^T H y dy = \int_{B_1(0)} \sum_{i,j} y_i H_{i,j} y_j dy = \sum_{i,j} \int_{B_1(0)} y_i y_j H_{i,j} dy.$$

Now we know from Corollary 8 that this integral is zero when $i \neq j$. This gives

$$\sum_{i,j} H_{i,j} \int_{B_1(0)} y_i y_j dy = \sum_i H_{i,i} \int_{B_1(0)} y_i^2 dy = \operatorname{trace}\left(H\right) \frac{\pi^{d/2}}{2\Gamma\left(2 + d/2\right)}.$$

■

References

Y. Bengio. Learning deep architectures for AI. *Foundations & Trends in Mach. Learn.*, 2 (1):1–127, 2009.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *NIPS'2006*, 2007.

Yoshua Bengio and Olivier Delalleau. On the expressive power of deep architectures. In *ALT'2011*, 2011.

Yoshua Bengio, Guillaume Alain, and Salah Rifai. Implicit density estimation by local moment matching to sample from auto-encoders. Technical report, arXiv:1207.0057, 2012a.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. Technical report, arXiv:1206.5538, 2012b.

Yoshua Bengio, Yao Li, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. Technical Report arXiv:1305.6663, Universite de Montreal, 2013a.

Yoshua Bengio, Grégoire Mesnil, Yann Dauphin, and Salah Rifai. Better mixing via deep representations. In *ICML'13*, 2013b.

Lawrence Cayton. Algorithms for manifold learning. Technical Report CS2008-0923, UCSD, 2005.

B. Dacorogna. *Introduction to the Calculus of Variations*. World Scientific Publishing Company, 2004.

Karol Gregor, Arthur Szlam, and Yann LeCun. Structured sparse coding via lateral inhibition. In *NIPS'2011*, 2011.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

Aapo Hyvärinen. Estimation of non-normalized statistical models using score matching. *J. Machine Learning Res.*, 6, 2005.

Aapo Hyvärinen. Some extensions of score matching. *Computational Statistics and Data Analysis*, 51:2499–2512, 2007.

Viren Jain and Sebastian H. Seung. Natural image denoising with convolutional networks. In *NIPS'2008*, 2008.

K. Kavukcuoglu, M-A. Ranzato, R. Fergus, and Y. LeCun. Learning invariant features through topographic filter maps. In *CVPR'2009*, 2009.

Diederik Kingma and Yann LeCun. Regularized estimation of image statistics by score matching. In *NIPS'2010*, 2010.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML'2009*. 2009.

Hariharan Narayanan and Sanjoy Mitter. Sample complexity of testing the manifold hypothesis. In *NIPS'2010*. 2010.

B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision Research*, 37:3311–3325, 1997.

M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In *NIPS'2006*, 2007.

M. Ranzato, Y. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. In *NIPS'2007*, 2008.

Salah Rifai, Yann Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller. The manifold tangent classifier. In *NIPS'2011*, 2011a.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML'2011*, 2011b.

Salah Rifai, Yoshua Bengio, Yann Dauphin, and Pascal Vincent. A generative process for sampling contractive auto-encoders. In *ICML'2012*, 2012.

R. Salakhutdinov and G.E. Hinton. Deep Boltzmann machines. In *AISTATS'2009*, 2009.

Kevin Swersky, Marc'Aurelio Ranzato, David Buchman, Benjamin Marlin, and Nando de Freitas. On autoencoders and score matching for energy based models. In *ICML'2011*. 2011.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7), 2011.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML 2008*, 2008.

Journal of Machine Learning Research 15 (2014) 3775-3814 Submitted 5/13; Revised 5/14; Published 11/14

Revisiting Bayesian Blind Deconvolution

David Wipf DAVIDWIPF@GMAIL.COM
Visual Computing Group
Microsoft Research
Building 2, No. 5 Danling Street
Beijing, P.R. China, 100080

Haichao Zhang HCZHANG1@GMAIL.COM
School of Computer Science
Northwestern Polytechnical University
127 West Youyi Road
Xi'an, P.R. China, 710072

Editor: Lawrence Carin

Abstract

Blind deconvolution involves the estimation of a sharp signal or image given only a blurry observation. Because this problem is fundamentally ill-posed, strong priors on both the sharp image and blur kernel are required to regularize the solution space. While this naturally leads to a standard MAP estimation framework, performance is compromised by unknown trade-off parameter settings, optimization heuristics, and convergence issues stemming from non-convexity and/or poor prior selections. To mitigate some of these problems, a number of authors have recently proposed substituting a variational Bayesian (VB) strategy that marginalizes over the high-dimensional image space leading to better estimates of the blur kernel. However, the underlying cost function now involves both integrals with no closed-form solution and complex, function-valued arguments, thus losing the transparency of MAP. Beyond standard Bayesian-inspired intuitions, it thus remains unclear by exactly what mechanism these methods are able to operate, rendering understanding, improvements and extensions more difficult. To elucidate these issues, we demonstrate that the VB methodology can be recast as an unconventional MAP problem with a very particular penalty/prior that conjoins the image, blur kernel, and noise level in a principled way. This unique penalty has a number of useful characteristics pertaining to relative concavity, local minima avoidance, normalization, and scale-invariance that allow us to rigorously explain the success of VB including its existing implementational heuristics and approximations. It also provides strict criteria for learning the noise level and choosing the optimal image prior that, perhaps counter-intuitively, need not reflect the statistics of natural scenes. In so doing we challenge the prevailing notion of why VB is successful for blind deconvolution while providing a transparent platform for introducing enhancements and extensions. Moreover, the underlying insights carry over to a wide variety of other bilinear models common in the machine learning literature such as independent component analysis, dictionary learning/sparse coding, and non-negative matrix factorization.

Keywords: blind deconvolution, blind image deblurring, variational Bayes, sparse priors, sparse estimation

1. Introduction

Blind deconvolution problems involve the estimation of some latent sharp signal of interest given only an observation that has been compromised by an unknown filtering process. Although relevant algorithms and analysis apply in a general setting, this paper will focus on the particular case of blind image deblurring, where an unknown convolution or blur operator, as well as additive noise, corrupt the image capture of an underlying natural scene. Such blurring is an undesirable consequence that often accompanies the image formation process and may arise, for example, because of camera-shake during acquisition. Blind image deconvolution or deblurring strategies aim to recover a sharp image from only a blurry, compromised observation, a long-standing problem (Richardson, 1972; Lucy, 1974; Kundur and Hatzinakos, 1996) that remains an active research topic (Fergus et al., 2006; Shan et al., 2008; Levin et al., 2009; Cho and Lee, 2009; Krishnan et al., 2011). Moreover, applications extend widely beyond standard photography, with astronomical, bio-imaging, and other signal processing data eliciting particular interest (Zhu and Milanfar, 2013; Kenig et al., 2010).

Assuming a convolutional blur model with additive noise (Fergus et al., 2006; Shan et al., 2008), the low quality image observation process is commonly modeled as

$$\mathbf{y} = \mathbf{k} * \mathbf{x} + \mathbf{n}, \tag{1}$$

where $\mathbf{k}$ is the point spread function (PSF) or blur kernel, $*$ denotes the 2D convolution operator, and $\mathbf{n}$ is a zero-mean Gaussian noise term (although as we shall see, these assumptions about the noise distribution can easily be relaxed via the framework described herein). The task of blind deconvolution is to estimate both the sharp image $\mathbf{x}$ and blur kernel $\mathbf{k}$ given only the blurry observation $\mathbf{y}$, where we will mostly be assuming that $\mathbf{x}$ and $\mathbf{y}$ represent filtered (e.g., gradient domain) versions of the original pixel-domain images. Because $\mathbf{k}$ is non-invertible, some (typically) high frequency information is lost during the observation process, and thus even if $\mathbf{k}$ were known, the non-blind estimation of $\mathbf{x}$ is ill-posed. However, in the blind case where $\mathbf{k}$ is also unknown, the difficulty is exacerbated considerably, with many possible image/kernel pairs explaining the observed data equally well. This is analogous to the estimation challenges associated with a wide variety of bilinear models, where the observation model (1) is generalized to

$$\mathbf{y} = \mathbf{H}(\mathbf{k})\mathbf{x} + \mathbf{n}. \tag{2}$$

Here $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{k}$ can be arbitrary matrices or vectors, and $\mathbf{k}$ represents unknown parameters embedded in the linear operator $\mathbf{H}(\mathbf{k})$. Note that (1) represents a special case of (2) when $\mathbf{y}$ and $\mathbf{x}$ are vectorized images and $\mathbf{H}(\mathbf{k})$ is the Toeplitz convolution matrix associated with $\mathbf{k}$. Other important instances prevalent in the machine learning and signal processing literature include independent component analysis (ICA) (Hyvarinen and Oja, 2000), dictionary learning for sparse coding (Mairal et al., 2010), and non-negative matrix factorization (Lee and Seung, 2001).

To alleviate the intrinsic indeterminacy, prior assumptions must be adopted to constrain the space of candidate solutions, which naturally suggests a Bayesian framework. In Section 2, we briefly review the two most common classes of Bayesian algorithms for blind deconvolution used in the literature, (i) Maximum a Posteriori (MAP) estimation and (ii)

Variational Bayes (VB), and then later detail their fundamental limitations, which include heuristic implementational requirements and complex cost functions that are difficult to disentangle. Section 3 uses ideas from convex analysis to reformulate these Bayesian methods promoting greater understanding and suggesting useful enhancements, such as rigorous criteria for choosing appropriate image priors. Section 4 then situates our theoretical analysis within the context of existing analytic studies of blind deconvolution, notably the seminal work from Levin et al. (2009, 2011b), and discusses the relevance of natural image statistics. Learning noise variances is later addressed in Section 5, while experiments are carried out in Section 6 to provide corroborating empirical evidence for some of our theoretical claims. Finally, concluding remarks are contained in Section 7. While nominally directed at the challenges of blind deconvolution, we envision that the underlying principles analyses developed herein will nonetheless contribute to better understanding of generalized bilinear models in broad application domains.

2. MAP versus VB

As mentioned above, to compensate for the ill-posedness of the blind deconvolution problem, a strong prior is required for both the sharp image and kernel to regularize the solution space. Recently, natural image statistics over image gradients have been invoked to design prior (regularization) models (Roth and Black, 2009; Levin et al., 2007; Krishnan and Fergus, 2009; Cho et al., 2012), and MAP estimation using these priors has been proposed for blind deconvolution (Shan et al., 2008; Krishnan et al., 2011). While some specifications may differ, the basic idea is to find the mode (maximum) of

$$p(\mathbf{x},\mathbf{k}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x},\mathbf{k})p(\mathbf{x})p(\mathbf{k})}{p(\mathbf{y})} \propto p(\mathbf{y}|\mathbf{x},\mathbf{k})p(\mathbf{x})p(\mathbf{k}),$$

where $\mathbf{x}$ and $\mathbf{y}$ are now assumed to be vectorized gradient domain sharp and blurry images respectively, and $\mathbf{k}$ is the corresponding vectorized kernel.[1] $p(\mathbf{y}|\mathbf{x},\mathbf{k})$ is a Gaussian likelihood function with mean $\mathbf{k}*\mathbf{x}$ and covariance $\lambda\mathbf{I}$, and $p(\mathbf{x})$ and $p(\mathbf{k})$ are priors, with the former often assumed to be sparsity-promoting consistent with estimates of natural image statistics (Buccigrossi and Simoncelli, 1999; Levin et al., 2011b). After a $-2\log$ transformation, and ignoring constant factors, this is equivalent to computing

$$\min_{\mathbf{x},\mathbf{k}} -2\log p(\mathbf{x},\mathbf{k}|\mathbf{y}) \equiv \min_{\mathbf{x},\mathbf{k}} \frac{1}{\lambda}\|\mathbf{k}*\mathbf{x}-\mathbf{y}\|_2^2 + g_{\mathbf{x}}(\mathbf{x}) + g_{\mathbf{k}}(\mathbf{k}), \tag{3}$$

where $g_{\mathbf{x}}(\mathbf{x})$ is a penalty term over the desired image, typically of the form $g_{\mathbf{x}}(\mathbf{x}) = \sum_i g_x(x_i)$, while $g_{\mathbf{k}}(\mathbf{k})$ regularizes the blur kernel. Both penalties generally have embedded parameters that must be balanced along with the unknown λ. It is also typical to assume that $\sum_i k_i = 1$, with $k_i \geq 0$ and we will adopt this assumption throughout; however, Sections 3.5 and 3.7 will discuss a type of scale invariance such that this assumption becomes irrelevant in important cases.

Although straightforward, there are many problems with existing MAP approaches including ineffective global minima, e.g., poor priors may lead to degenerate global solutions

1. Even in vectorized form, we will still use $\mathbf{k}*\mathbf{x}$ to denote the standard 2D convolution operator, where the result is then subsequently vectorized.

like the delta kernel (frequently called the no-blur solution), or many suboptimal local minima and subsequent convergence issues. Therefore, the generation of useful solutions requires a delicate balancing of various factors such as dynamic noise levels, trade-off parameter values, and other heuristic regularizers such as salient structure selection (Shan et al., 2008; Cho and Lee, 2009; Krishnan et al., 2011) (we will discuss these issues more in Section 3).

To mitigate some of these shortcomings of MAP, the influential work by Levin et al. (2009) and others proposes to instead solve

$$\max_{\mathbf{k}} p(\mathbf{k}|\mathbf{y}) \equiv \min_{\mathbf{k}} -2\log p(\mathbf{y}|\mathbf{k})p(\mathbf{k}), \tag{4}$$

where $p(\mathbf{y}|\mathbf{k}) = \int p(\mathbf{x},\mathbf{y}|\mathbf{k})d\mathbf{x}$. This technique is sometimes referred to as Type II estimation in the statistics literature.[2] Once $\mathbf{k}$ is estimated in this way, $\mathbf{x}$ can then be obtained via conventional non-blind deconvolution techniques. One motivation for the Type II strategy is based on the inherent asymmetry in the dimensionality of the image relative to the kernel (Levin et al., 2009). By integrating out (or averaging over) the high-dimensional image, the estimation process can then more accurately focus on the few remaining low-dimensional parameters in $\mathbf{k}$.

The challenge with (4) is that evaluation of $p(\mathbf{y}|\mathbf{k})$ requires a marginalization over $\mathbf{x}$, which is a computationally intractable integral given realistic image priors. Consequently a variational Bayesian (VB) strategy is used to approximate the troublesome marginalization (Levin et al., 2011a). A similar idea has also been explored by a number of other authors (Miskin and MacKay, 2000; Fergus et al., 2006; Babacan et al., 2012). In brief, VB provides a convenient way of computing a rigorous upper bound on $-\log p(\mathbf{y}|\mathbf{k})$, which can then be substituted into (4) for optimization purposes leading to an approximate Type II estimator.

The VB methodology can be easily applied whenever the image prior $p(\mathbf{x})$ is expressible as a Gaussian scale mixture (GSM) (Palmer et al., 2006), meaning

$$p(\mathbf{x}) = \exp\left[-\frac{1}{2}g_{\mathbf{x}}(\mathbf{x})\right] = \prod_i \exp\left[-\frac{1}{2}g_x(x_i)\right] = \prod_i \int \mathcal{N}(x_i;0,\gamma_i)p(\gamma_i)d\gamma_i, \tag{5}$$

where each $\mathcal{N}(x_i;0,\gamma_i)$ represents a zero mean Gaussian with variance γ_i and prior distribution $p(\gamma_i)$. The role of this decomposition will become apparent below. Also, with some abuse of notation, $p(\gamma_i)$ may characterize a discrete distribution, in which case the integral in (5) can be reduced to a summation. Note that all prior distributions expressible via (5) will be supergaussian (Palmer et al., 2006), and therefore will to varying degrees favor a sparse $\mathbf{x}$ (we will return to this issue in Sections 3 and 4).

Given this $p(\mathbf{x})$, the negative log of $p(\mathbf{y}|\mathbf{k})$ can be upper bounded via

$$-\log p(\mathbf{y}|\mathbf{k}) \leq \underbrace{-\iint q(\mathbf{x},\boldsymbol{\gamma})\log\frac{p(\mathbf{x},\boldsymbol{\gamma},\mathbf{y}|\mathbf{k})}{q(\mathbf{x},\boldsymbol{\gamma})}d\mathbf{x}d\boldsymbol{\gamma}}_{F[q(\mathbf{x},\boldsymbol{\gamma}),\mathbf{k}]},$$

2. To be more specific, Type II estimation refers to the case where we optimize over one set of unknown variables after marginalizing out another set, in our case the image $\mathbf{x}$. In this context, standard MAP over both $\mathbf{x}$ and $\mathbf{k}$ via (3) can be viewed as Type I.

where $F[q(\mathbf{x},\boldsymbol{\gamma}),\mathbf{k}]$ is called the *free energy*, $q(\mathbf{x},\boldsymbol{\gamma})$ is an arbitrary distribution over $\mathbf{x}$, and $\boldsymbol{\gamma} = [\gamma_1,\gamma_2,\ldots]^T$, the vector of all the variances from (5). Equality is obtained when $q(\mathbf{x},\boldsymbol{\gamma}) = p(\mathbf{x},\boldsymbol{\gamma}|\mathbf{y},\mathbf{k})$. In fact, if we were able to iteratively minimize this F over $q(\mathbf{x},\boldsymbol{\gamma})$ and $\mathbf{k}$ (i.e., a form of coordinate descent), this would be exactly equivalent to the standard expectation-maximization (EM) algorithm for minimizing $-\log p(\mathbf{y}|\mathbf{k})$ with respect to $\mathbf{k}$, treating $\boldsymbol{\gamma}$ and $\mathbf{x}$ as hidden data and assuming $p(\mathbf{k})$ is flat within the specified constraint set mentioned previously (see Bishop 2006, Ch.9.4 for a detailed examination of this fact). However, optimizing over $q(\mathbf{x},\boldsymbol{\gamma})$ is intractable since $p(\mathbf{x},\boldsymbol{\gamma}|\mathbf{y},\mathbf{k})$ is generally not available in closed-form. Likewise, there is no closed-form update for $\mathbf{k}$, and hence no exact EM solution is possible.

The contribution of VB theory is to show that if we restrict the form of $q(\mathbf{x},\boldsymbol{\gamma})$ via structural assumptions, the updates can now actually be computed, albeit approximately. For this purpose the most common constraint is that $q(\mathbf{x},\boldsymbol{\gamma})$ must be factorized, namely, $q(\mathbf{x},\boldsymbol{\gamma}) = q(\mathbf{x})q(\boldsymbol{\gamma})$, sometimes called a mean-field approximation (Bishop, 2006, Ch.10.1). With this approximation we are effectively utilizing the revised (and looser) upper bound

$$-\log p(\mathbf{y}|\mathbf{k}) \leq -\iint q(\mathbf{x})q(\boldsymbol{\gamma})\log\frac{p(\mathbf{x},\boldsymbol{\gamma},\mathbf{y}|\mathbf{k})}{q(\mathbf{x})q(\boldsymbol{\gamma})}d\mathbf{x}d\boldsymbol{\gamma}, \tag{6}$$

which may be iteratively minimized over $q(\mathbf{x})$, $q(\boldsymbol{\gamma})$, and $\mathbf{k}$ independently while holding the other two fixed. In each case, closed-form updates are now possible, although because of the factorial approximation, we are of course no longer guaranteed to minimize $-\log p(\mathbf{y}|\mathbf{k})$.

Compared to the original Type II problem from (4), minimizing the bound from (6) is considerably simplified because the problematic marginalization over $\mathbf{x}$ has been effectively decoupled from $\boldsymbol{\gamma}$. However, when solving for $q(\mathbf{x})$ at each iteration, it can be shown that a full covariance matrix of $\mathbf{x}$ conditioned on $\boldsymbol{\gamma}$, denoted as $\mathbf{C}$, must be computed. While this is possible in closed form, it requires $O(m^3)$ operations, where m is the number of pixels in the image. Because this is computationally impractical for reasonably-sized images, a diagonal approximation to $\mathbf{C}$ must be adopted (Levin et al., 2011a). This assumption is equivalent to incorporating an additional factorization into the VB process such that now we are enforcing the constraint $q(\mathbf{x},\boldsymbol{\gamma}) = \prod_i q(x_i)q(\gamma_i)$. This leads to the considerably looser upper bound

$$-\log p(\mathbf{y}|\mathbf{k}) \leq -\iint \prod_i q(x_i)q(\gamma_i)\log\frac{p(\mathbf{x},\boldsymbol{\gamma},\mathbf{y}|\mathbf{k})}{\prod_i q(x_i)q(\gamma_i)}d\mathbf{x}d\boldsymbol{\gamma}.$$

In summary then, the full Type II approach can be approximated by minimizing the VB upper bound via the optimization problem

$$\min_{q(\mathbf{x},\boldsymbol{\gamma}),\mathbf{k}} \; F\left[q(\mathbf{x},\boldsymbol{\gamma}),\mathbf{k}\right], \quad \text{s.t. } q(\mathbf{x},\boldsymbol{\gamma}) = \prod_i q(x_i)q(\gamma_i). \tag{7}$$

The requisite update rules are shown in Algorithm 1.[3] Numerous methods fall within this category with some implementational differences, and the estimation steps are equivalent

3. For simplicity we have ignored image boundary effects when presenting the computation for c_j in Algorithm 1. In fact, the complete expression for c_j is described in Appendix A in the proof of Theorem 1. Additionally, Algorithm 1 in its present form includes a modest differentiability assumption on g_x for

Algorithm 1 VB Blind Deblurring (Levin et al., 2011a; Palmer et al., 2006; Babacan et al., 2012)

1: **Input:** blurry gradient domain image $\mathbf{y}$, noise level reduction factor β ($\beta > 1$), minimum noise level λ_0, image prior $p(\mathbf{x}) = \exp[-\frac{1}{2}g_{\mathbf{x}}(\mathbf{x})] = \prod_i \exp[-\frac{1}{2}g_x(x_i)]$

2: **Initialize:** blur kernel $\mathbf{k}$, noise level λ

3: **While** stopping criteria is not satisfied, repeat

- **Update sufficient statistics for $q(\boldsymbol{\gamma}) = \prod_i q(\gamma_i)$:**

$$\omega_i \triangleq \mathrm{E}_{q(\gamma_i)}[\gamma_i^{-1}] \leftarrow \frac{{g_x}'(\sigma_i)}{2\sigma_i},$$

  with $\sigma_i^2 \triangleq \mathrm{E}_{q(x_i)}[x_i^2] = \mu_i^2 + C_{ii}$.

- **Update sufficient statistics for $q(\mathbf{x}) = \prod_i q(x_i)$:**

$$\boldsymbol{\mu} \triangleq \mathrm{E}_{q(\mathbf{x})}[\mathbf{x}] \leftarrow \mathbf{A}^{-1}\mathbf{b}, \quad C_{ii} \triangleq \mathrm{Var}_{q(x_i)}[x_i] \quad \leftarrow A_{ii}^{-1},$$

  where $\mathbf{A} = \frac{\mathbf{H}^T\mathbf{H}}{\lambda} + \mathrm{diag}[\boldsymbol{\omega}]$, $\mathbf{b} = \frac{\mathbf{H}^T\mathbf{y}}{\lambda}$, $\mathbf{H}$ is the convolution matrix of $\mathbf{k}$.

- **Update k:**

$$\mathbf{k} \leftarrow \arg\min_{\mathbf{k}\geq 0} \|\mathbf{y} - \mathbf{W}\mathbf{k}\|_2^2 + \sum_j c_j k_j^2$$

  where $c_j = \sum_i C_{i+j,i+j}$ and $\mathbf{W}$ is the convolution matrix of $\boldsymbol{\mu}$.

- **Noise level reduction:** If $\lambda > \lambda_0$, then $\lambda \leftarrow \lambda/\beta$.

4: **Final Non-Blind Step:** In original image domain, estimate sharp image using fixed $\mathbf{k}$ from above

to simply inserting the kernel update rule and noise reduction heuristic from Levin et al. (2011a) into the general VB sparse estimation framework from Palmer et al. (2006). Results using this strategy for blind deblurring with different priors can be found in Babacan et al. (2012). Note that the full distributions for each $q(x_i)$ and $q(\gamma_i)$ are generally not needed; only certain sufficient statistics are required (certain means and variances, see Algorithm 1), analogous to standard EM. These can be efficiently computed using techniques from Palmer et al. (2006) for any $p(\mathbf{x})$ produced by (5). In the VB algorithm from Levin et al. (2011a), the sufficient statistic for $\boldsymbol{\gamma}$ is computed using an alternative methodology which applies only to finite Gaussian scale mixtures. However, the resulting updates are nonetheless equivalent to Algorithm 1 as shown in the proof of Theorem 1 presented later.

updating the sufficient statistics of $q(\gamma_i)$). Finally, while it is trivial to include multiple image filters in this pipeline (Levin et al., 2011a; Babacan et al., 2012), we avoid including such additional notation to simplify the exposition. Here we are already assuming that $\mathbf{y}$ and $\mathbf{x}$ represent blurry and sharp gradient domain images, obtained by simple horizontal and vertical first-order difference filters (see Section 3.1).

While possibly well-motivated in principle, the Type II approach relies on rather severe factorial assumptions which may compromise the original high-level justifications. In fact, at any minimizing solution denoted $q^*(x_i), q^*(\gamma_i), \forall i, \mathbf{k}^*$, it is easily shown that the gap between F and $-\log p(\mathbf{y}|\mathbf{k}^*)$ is given explicitly by

$$KL\left(\prod_i q^*(x_i)q^*(\gamma_i)||p(\mathbf{x}, \boldsymbol{\gamma}|\mathbf{y}, \mathbf{k}^*)\right), \tag{8}$$

where $KL(p_1||p_2)$ denotes the standard KL divergence between the distributions p_1 and p_2. Because the posterior $p(\mathbf{x}, \boldsymbol{\gamma}, |\mathbf{y}, \mathbf{k})$ is generally highly coupled (non-factorial), this divergence will typically be quite high, indicating that the associated approximation can be poor. We therefore have no reason to believe that this $\mathbf{k}^*$ is anywhere near the maximizer of $p(\mathbf{y}|\mathbf{k})$, which was the ultimate goal and motivation of Type II to begin with.

Other outstanding issues persist as well. For example, the free energy cost function, which involves both integration and function-valued arguments, is not nearly as transparent as the standard MAP estimation from (3). Moreover for practical use, VB depends on an appropriate schedule for reducing the noise variance λ during each iteration (see Algorithm 1), which implements a form of coarse-to-fine, multiresolution estimation scheme (Levin et al., 2011b) while potentially improving the convergence rate (Levin et al., 2011a).

It therefore becomes difficult to rigorously explain exactly why VB has often been empirically more successful than MAP in practice (see Babacan et al. 2012; Levin et al. 2011b,a for such comparisons), nor how to decide which image priors operate best in the VB framework.[4] While Levin *et al.* have suggested that at a high level, marginalization over the latent sharp image using natural-image-statistic-based priors is a good idea to overcome some of the problems faced by MAP estimation (Levin et al., 2009, 2011b), this argument only directly motivates substituting (4) for (3) rather than providing explicit rationalization for (7). Thus, we intend to more meticulously investigate the exact mechanism by which VB operates, explicitly accounting for all of the approximations and assumptions involved by drawing on convex analysis and sparse estimation concepts from Palmer et al. (2006); Wipf et al. (2011) (Section 4 will discuss direct comparisons with Levin *et al.* in detail). This endeavor then naturally motivates extensions to the VB framework and a simple prescription for choosing an appropriate image prior $p(\mathbf{x})$. Overall, we hope that we can further demystify VB providing an entry point for broader improvements such as robust non-uniform blur and noise estimation.

Several surprising, possibly counterintuitive conclusions emerge from this investigation which challenge some of the prevailing wisdom regarding why and how Bayesian algorithms can be advantageous for blind deconvolution. These include:

- The optimal image prior for blind deconvolution purposes using VB or MAP is likely not the one which most closely reflects natural images statistics. Rather, we argue that it is the distribution that most significantly discriminates between blurry and sharp images, meaning a prior such that, for some good sharp image estimate $\hat{\mathbf{x}}$, we

4. Note that, as discussed later, certain MAP algorithms can perform reasonably well when carefully balanced with additional penalty factors and tuning parameters added to (3). However, in direct comparisons using the same basic probabilistic model, VB can perform substantially better, even achieving state-of-the-art performance without additional tuning.

have $p(\hat{\mathbf{x}}) \gg p(\mathbf{k} * \hat{\mathbf{x}})$. Natural image statistics typically fail in this regard for explicit reasons, which apply to both MAP and VB, as discussed in Sections 3 and 4.

- The advantage of VB over MAP is not directly related to the dimensionality differences between $\mathbf{k}$ and $\mathbf{x}$ and the conventional benefits of marginalization over the latter. In fact, we prove in Section 3.2 that the underlying cost functions are formally equivalent in ideal noiseless environments given the factorial assumptions required by practical VB algorithms, and the same basic line of reasoning holds equally well in the noisy case. Instead, there is an intrinsic mechanism built into VB that allows bad locally minimizing solutions to be largely avoided even when using the highly non-convex, discriminative priors needed to distinguish between blurry and sharp images. This represents a new perspective on the relative advantages of VB.
- The VB algorithm can be reformulated in such a way that non-Gaussian noise models, non-uniform blur operators, and other extensions are easily incorporated, circumventing one important perceived advantage of MAP. In fact, we have already obtained practical success in more complex non-uniform and multi-image models using similar principles (Zhang et al., 2013; Zhang and Wipf, 2013).

3. Analysis of Variational Bayes

Drawing on ideas from Palmer et al. (2006); Wipf et al. (2011), in this section we will reformulate the VB methodology to elucidate its behavior. Simply put, we will demonstrate that VB is actually equivalent to using an unconventional MAP estimation-like cost function but with a particular penalty that links the image, blur kernel, and noise in a well-motivated fashion. This procedure removes the ambiguity introduced by the VB approximation, the subsequent diagonal covariance approximation, and the λ reduction heuristic that all contribute still somewhat mysteriously to the effectiveness of VB. It will then allow us to pinpoint the exact reasons why VB represents an improvement over conventional MAP estimations in the form of (3), and it provides us with a specific criteria for choosing the image prior $p(\mathbf{x})$.

3.1 Notation and Definitions

As mentioned above, and following many previous works (Fergus et al., 2006; Levin et al., 2011a), we will henceforth work entirely in the derivative domain of images, with the exception of an implicit final non-blind deconvolution step once the kernel $\mathbf{k}$ has been estimated. From a practical standpoint, these derivatives are computed by convolving the raw image with standard first-order horizontal and vertical difference filters $[1, -1]$ and $[1, -1]^T$. Given that convolution is a commutative operator, the blur kernel is unaltered. For latent sharp image derivatives of size $M \times N$ and blur kernel of size $P \times Q$, we denote the lexicographically ordered vector of the sharp image derivatives, blurry image derivatives, and blur kernel as $\mathbf{x} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^n$ and $\mathbf{k} \in \mathbb{R}^l$ respectively, with $m \triangleq MN$, $n \triangleq (M - P + 1)(N - Q + 1)$, and $l \triangleq PQ$. This assumes a single derivative filter. The extension to multiple filters, for example one for each image dimension as described above, follows naturally. For simplicity of notation however, we omit explicit referencing of multiple filters throughout this paper, although all related analysis directly follow through.

The likelihood model (1) can be rewritten as

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} = \mathbf{W}\mathbf{k} + \mathbf{n}, \tag{9}$$

where $\mathbf{H} \in \mathbb{R}^{n\times m}$ and $\mathbf{W} \in \mathbb{R}^{n\times l}$ are the Toeplitz convolution matrices constructed from the blur kernel and sharp image respectively. We introduce a matrix $\bar{\mathbf{I}} \in \mathbb{R}^{l\times m}$, where the j-th row of $\bar{\mathbf{I}}$ is a binary vector with 1 indicating that the j-th element of $\mathbf{k}$ (i.e., k_j) appears in the corresponding column of $\mathbf{H}$ and 0 otherwise. We define $\|\bar{\mathbf{k}}\|_2 \triangleq \sqrt{\sum_j k_j^2 \bar{I}_{ji}}$, which is equivalent to the norm of the i-th column of $\mathbf{H}$. It can also be viewed as the effective norm of $\mathbf{k}$ accounting for the boundary effects.[5] The element-wise magnitude of $\mathbf{x}$ is given by $|\mathbf{x}| \triangleq [|x_1|, |x_2|, \ldots]^T$.

Finally we introduce the definition of *relative concavity* (Palmer, 2003) which will serve subsequent analyses:

Definition 1 *Let u be a strictly increasing function on $[a, b]$. The function ν is **concave relative** to u on the interval $[a, b]$ if and only if*

$$\nu(y) \leq \nu(x) + \frac{\nu'(x)}{u'(x)}\left[u(y) - u(x)\right] \tag{10}$$

holds $\forall x, y \in [a, b]$.

In the following, we will use $\nu \prec u$ to denote that ν is concave relative to u on $[0, \infty)$. This can be understood as a natural generalization of the traditional notion of a concavity, in that a concave function is equivalently *concave relative to a linear function* per Definition 1. In general, if $\nu \prec u$, then when ν and u are set to have the same functional value and the same slope at any given point (i.e., by an affine transformation of u), then ν lies completely under u.

It is well-known that functions concave in $|\mathbf{x}|$ favor sparsity (meaning a strong preference for zero and relatively little distinction between nonzero values) (Rao et al., 2003; Wipf et al., 2011). The notion of relative concavity induces an ordering for many of the common sparsity promoting functions. Intuitively, a non-decreasing function ν of $|x_i|$ is more aggressive in promoting sparsity than some u if it is concave relative to u. For example, consider the class of ℓ_p functionals $\sum_i |x_i|^p$ that are concave in $|x_i|$ whenever $0 < p \leq 1$. The smaller p, the more a sparse $\mathbf{x}$ will be favored, with the extreme case $p \to 0$ producing the ℓ_0 norm (a count of the number of nonzero elements in $\mathbf{x}$), which is the most aggressive penalty for promoting sparsity. Meanwhile, using Definition 1 it is easy to verify that, as a function of $|\mathbf{x}|$, $\ell_{p_1} \prec \ell_{p_2}$ whenever $p_1 < p_2$.

5. Technically $\|\bar{\mathbf{k}}\|_2$ depends on i, the index of image pixels, but it only makes a difference near the image boundaries. We prefer to avoid an explicit notational dependency on i to keep the presentation concise, although the proofs in Appendix A do consider i-dependency when it is relevant. The subsequent analysis will also omit this dependency keeping in mind that all of the results nonetheless carry through in the general case. The same is true for the other quantities that depend on $\|\bar{\mathbf{k}}\|_2$, e.g., the ρ parameter defined later in (12).

3.2 Connecting VB with MAP

As mentioned previously, the VB algorithm of Levin et al. (2011a) can be efficiently implemented using any image prior expressible in the form of (5). However, for our purposes we require an alternative representation with roots in convex analysis. Based on Palmer et al. (2006), it can be shown that any prior given by (5) can also be represented as a maximization over scaled Gaussians with different variances leading to the alternative representation

$$p(x_i) = \exp\left[-\frac{1}{2}g_x(x_i)\right] = \max_{\gamma_i \geq 0} \mathcal{N}(x_i; 0, \gamma_i) \exp\left[-\frac{1}{2}f(\gamma_i)\right], \tag{11}$$

where $f(\gamma_i)$ is some non-negative energy function; the associated exponentiated factor is sometimes treated as a hyperprior, although it will not generally integrate to one. This f, which determines the form of g_x in (5), will ultimately play a central role in how VB penalizes images $\mathbf{x}$ as will be explored via the results of this section.

Theorem 1 *Consider the objective function*

$$\mathcal{L}(\mathbf{x}, \mathbf{k}) \triangleq \frac{1}{\lambda}\|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2 + \sum_i g_{\text{VB}}(x_i, \rho) + m \log \|\bar{\mathbf{k}}\|_2^2, \tag{12}$$

where

$$g_{\text{VB}}(x_i, \rho) \triangleq \min_{\gamma_i \geq 0}\left[\frac{x_i^2}{\gamma_i} + \log(\rho + \gamma_i) + f(\gamma_i)\right], \text{ and } \rho \triangleq \frac{\lambda}{\|\bar{\mathbf{k}}\|_2^2}. \tag{13}$$

Algorithm 1 minimizing (7) is equivalent to coordinate descent minimization of (12) over $\mathbf{x}$*,* $\mathbf{k}$*, and the latent variables* $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_m]^T$*.*

Proofs will be deferred to the Appendix A. This reformulation of VB closely resembles (3), with a quadratic data fidelity term combined with additive image and kernel penalties. The penalty on $\mathbf{k}$ in (12) is not unlike those incorporated into standard MAP schemes, meaning $g_{\mathbf{k}}(\mathbf{k})$ from (3). However, quite unlike MAP, for $\lambda > 0$ the penalty g_{VB} on the image pixels x_i is dependent on both the noise level λ and the kernel $\mathbf{k}$ through the parameter ρ, the ratio of the noise level to the squared kernel norm. The remainder of Section 3 will explore the consequences of this crucial, yet previously unexamined distinction from typical MAP formulations.

In contrast, with $\lambda = 0$, both MAP and VB possess a formally equivalent penalty on each x_i via the following corollary:

Corollary 1 *If* $\lambda = 0$*, then* $g_{\text{VB}}(x_i, 0) = g_x(x_i) \equiv -2 \log p(x_i)$*.*

Therefore the underlying VB cost function *is effectively no different than regular MAP from (3) in the noiseless setting*, a surprising conclusion that seems to counter much of the prevailing understanding of VB deconvolution algorithms. So then what exactly is the true advantage, if any, of VB over MAP? And is the advantage limited to cases when the noise level is significant? The remainder of Section 3 will address these questions, demonstrating that VB maintains a significant advantage over MAP, and that this advantage persists, perhaps paradoxically, even when the noise level is small or zero. These conclusions ultimately hinge on detailed properties of the image penalty g_{VB} in the context of practical deblurring pipelines. We will also examine the related issue of choosing the optimal image prior $p(\mathbf{x})$, which is equivalent to choosing the optimal f in (11).

3.3 Evaluating the VB Image Penalty g_{VB}

While in a few special cases g_{VB} can be computed in closed-form for general $\rho \neq 0$ leading to greater transparency, as we shall see below the VB algorithm and certain attendant analyses can nevertheless be carried through even when closed-form solutions for g_{VB} are not possible. Importantly, we can assess properties that may potentially affect the sparsity and quality of resulting solutions as λ and $\|\bar{\mathbf{k}}\|_2^2$ are varied.

A highly sparse prior, and therefore penalty function, is generally more effective in differentiating sharp images with fine structures from blurry ones (details in Section 4). Recall that concavity with respect to coefficient magnitudes is a signature property of such sparse penalties (Rao et al., 2003; Wipf et al., 2011). A potential advantage of MAP is that it is very straightforward to characterize the associated image penalty; namely, if g_x from (3) is a highly concave, nondecreasing function of each $|x_i|$, then we may expect that sparse image gradients will be heavily favored. And for two candidate image penalties ${g_x}^{(1)}$ and ${g_x}^{(2)}$, if ${g_x}^{(1)} \prec {g_x}^{(2)}$, then we may expect the former to promote an even sparser solution than the latter (provided we are not trapped at a bad local solution). Section 4 will argue that ${g_x}^{(1)}$ will then lead to a better estimate of $\mathbf{x}$ and $\mathbf{k}$.

In contrast, with VB it is completely unclear to what degree g_{VB} favors sparse solutions, except in the special case from the previous section when g_{VB} is equal to the MAP penalty g_x. We now explicitly describe sufficient and necessary conditions for g_{VB} to be a concave, nondecreasing function of $|x_i|$, which turn out to be much stricter than the conditions required for MAP.

Theorem 2 *The VB penalty g_{VB} will be a concave, non-decreasing function of $|x_i|$ for any ρ if and only if f from (11) is a concave, non-decreasing function on $[0, \infty)$. Moreover, at least $m-n$ elements of $\mathbf{x}$ will equal zero at any locally minimizing solution to (12) (however typically many more will equal zero in practice).*

Theorem 2 explicitly quantifies what class of image priors leads to a strong, sparsity-promoting $\mathbf{x}$ penalty when fully propagated through the VB framework. Yet while this attribute may anchor VB as a legitimate sparse estimator in the image (filter) domain given an appropriate f, it does not explain precisely why VB often produces superior results to MAP. In fact, the associated MAP penalty g_x (when generated from the same f) will actually promote sparse solutions under much weaker conditions as follows:

Corollary 2 *The MAP penalty g_x will be a concave, non-decreasing function of $|x_i|$ if and only if $\vartheta(z) \triangleq \log(z) + f(z)$ is a concave, non-decreasing function on $[0, \infty)$.*

The extra log factor implies that f itself need not be concave to ensure that g_x is concave. For example, the selection $f(z) = z - \log(z)$ it not concave and yet the associated g_x still will be since now $\vartheta(z) = z$, which is concave and non-decreasing as required by Corollary 2. The stronger proclivity for producing sparsity of MAP over VB is further quantified by the following:

Corollary 3 *Let f be a differentiable, non-decreasing function which induces the penalty functions g_x and g_{VB} associated with MAP and VB respectively. As $z \to \infty$, then $g_{\text{VB}}(z) - g_x(z) \to 0$, and therefore g_{VB} and g_x penalize large magnitudes of $\mathbf{x}$ equally. In contrast,*

for any z,$z' \geq 0$, if $z < z'$ then $g_{\text{VB}}(z) - g_x(z) \geq g_{\text{VB}}(z') - g_x(z')$. Therefore, as $z \to 0$, $g_{\text{VB}}(z) - g_x(z)$ is maximized, implying that the MAP penalty g_x favors zero-valued coefficients (sparsity) more heavily than g_{VB}.

This result implies that for a broad class of image priors, VB actually leads to a *weaker* enforcement of sparsity than the corresponding MAP estimator. This occurs because large magnitudes of any x_i are penalized nearly equivalently with VB and MAP, while small magnitudes are penalized much more aggressively with MAP. Taken together then, Corollaries 2 and 3 superficially suggest that perhaps MAP should be preferred over VB to the extent that we believe sparsity is important for distinguishing sharp and blurry images. However, a closer investigation will reveal why this conclusion is premature.

For this purpose we will consider closely the simplest choice for f which satisfies the conditions of Theorem 2, and a choice that has been advocated in the sparse estimation literature in different contexts: namely, a constant value, $f(\gamma) = b$. This in turn implies that the resulting prior $p(x_i)$ is a Jeffreys non-informative distribution on the coefficient magnitudes $|x_i|$ after solving the maximization from (11), and is attractive in part because there are no embedded hyperparameters (the constant b is irrelevant).[6] This selection for f leads to a particularly interesting closed-form penalty g_{VB} as follows:

Theorem 3 *In the special case where $f(\gamma_i) = b$, then*

$$g_{\text{VB}}(x_i, \rho) \equiv \frac{2|x_i|}{|x_i| + \sqrt{x_i^2 + 4\rho}} + \log\left(2\rho + x_i^2 + |x_i|\sqrt{x_i^2 + 4\rho}\right). \tag{14}$$

Figures 1 (a) and (b) display 1D and 2D plots of this penalty function. It is worth spending some time here to examine this particular selection for f (and therefore g_{VB}) in detail since it elucidates many of the mechanisms whereby VB, with all of its attendant approximations and heuristics, can affect improvement over MAP regardless of the true noise level.

In the limit as $\rho \to 0$, the first term in (14) converges to the indicator function $I[x_i \neq 0]$, and thus when we sum over i we obtain the ℓ_0 norm of $\mathbf{x}$, which represents a canonical measure of sparsity, or a count of the nonzero elements in a vector.[7] The second term in (14), when we again sum over i, converges to $\sum_i \log|x_i|$, ignoring a constant factor. Sometimes referred to as Gaussian entropy, this term can also be connected to the ℓ_0 norm via the relations $\|\mathbf{x}\|_0 \equiv \lim_{p\to 0} \sum_i |x_i|^p$ and $\lim_{p\to 0} \frac{1}{p}\sum_i (|x_i|^p - 1) = \sum_i \log|x_i|$ (Wipf et al., 2011). Thus the cumulative effect when ρ becomes small is an image prior that closely mimics the highly non-convex (in $|x_i|$) ℓ_0 norm which favors maximally sparse solutions. In contrast, when ρ becomes large, it can be shown that both terms in (14), when combined for all i, approach scaled versions of the convex ℓ_1 norm. Additionally, this scaling turns out to be principled in the sense described in Wipf and Wu (2012); Zhang and Wipf (2013).

For intermediate values of ρ between these two extremes, we obtain a g_{VB} that becomes *less* concave with respect to each $|x_i|$ as ρ increases in the formal sense of relative concavity discussed in Section 3.1. In particular, we have the following:

6. The Jeffreys prior is of the form $p(x) \propto 1/|x|$, which represents an improper distribution that does not integrate to one.
7. Although with $\rho = 0$, this term reduces to a constant, and therefore has no impact.

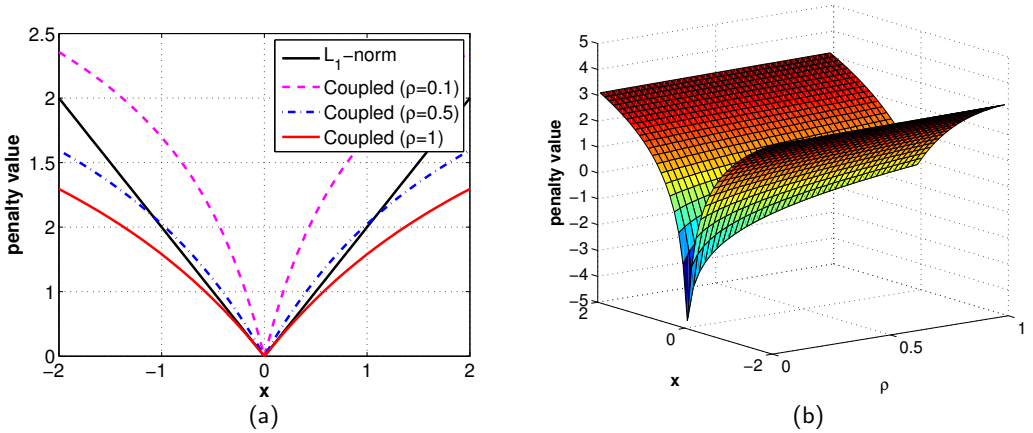


Figure 1: (a) A 1D example of the coupled penalty $g_{\text{VB}}(x, \rho)$ (normalized) with different ρ values assuming f is a constant. The ℓ_1 norm is included for comparison. (b) A 2D example surface plot of the coupled penalty function $g_{\text{VB}}(x, \rho)$; f is a constant.

Corollary 4 *If* $f(\gamma_i) = b$, *then* $g_{\text{VB}}^{\rho_1} \prec g_{\text{VB}}^{\rho_2}$ *for* $\rho_1 < \rho_2$.

Thus, as the noise level λ is increased, ρ increases and we have a penalty that behaves more like a convex (less sparse) function, and so becomes less prone to local minima. In contrast, as $\|\bar{\mathbf{k}}\|_2^2$ is increased, meaning that ρ is now reduced, the penalty actually becomes *more* concave with respect to $|x_i|$. This phenomena is in some ways similar to certain homotopy continuation sparse estimation schemes (e.g., Chartrand and Yin 2008), where heuristic hyperparameters are introduced to gradually introduce greater non-convexity into canonical compressive sensing problems, but without any dependence on the noise or other factors. The key difference here with VB however is that penalty shape modulation is explicitly dictated by both the noise level λ and the kernel $\mathbf{k}$ in an integrated fashion. [8]

To summarize then, the ratio ρ can be viewed as modulating a smooth transition of the penalty function shape from something akin to the non-convex ℓ_0 norm to a properly-scaled ℓ_1 norm. In contrast, conventional MAP-based penalties on $\mathbf{x}$ are independent from $\mathbf{k}$ or λ, and thus retain a fixed shape. The crucial ramifications of this coupling and ρ-controlled shape modification/augmentation exclusive to the VB framework will be addressed in the following two subsections. Other choices for f, which exhibit a partially muted form of this coupling, will be considered in Section 3.7, which will also address a desirable form of invariance that only exists when f is a constant.

3.4 Noise Dependency Analysis

The success of practical VB blind deconvolution algorithms is heavily dependent on some form of stagewise coarse-to-fine approach, whereby the kernel is repeatedly re-estimated at

8. After our original submission, a new MAP-related algorithm was published that applies a continuation strategy, not unlike Chartrand and Yin (2008)), to blind deblurring and achieves very promising results (Xu et al., 2013). However, unlike VB this algorithm requires two tuning parameters balancing kernel and image penalties and a fixed, pre-defined schedule for modulating the image penalty shape.

successively higher resolutions. At each stage, a lower resolution version is used to initialize the estimate at the next higher resolution. One way to implement this approach is to initially use large values of λ (regardless of the true noise level) such that only dominant, primarily low-frequency image structures dictate the optimization (Levin et al., 2009). During subsequent iterations as the blur kernel begins to reflect the correct coarse shape, λ can be gradually reduced to allow the recovery of more detailed, fine structures.

A highly sparse (concave) prior can ultimately be more effective in differentiating sharp images and fine structures than a convex one. Detailed supported evidence for this claim can be found in Fergus et al. (2006); Levin et al. (2007); Krishnan and Fergus (2009); Cho et al. (2012), as well as in Section 4 below. However, if such a prior is applied at the initial stages of estimation, the iterations are likely to become trapped at suboptimal local minima, of which there will always be a combinatorial number. Moreover, in the early stages, the effective noise level is actually high due to errors contained in the estimated blur kernel, and exceedingly sparse image penalties are likely to produce unstable solutions.

Given the reformulation outlined above, we can now argue that VB implicitly avoids these problems by beginning with a large λ (and therefore a large ρ), such that the penalty function is initially nearly convex in $|x_i|$ (see Figure 1). In this situation, the data fidelity term $\frac{1}{\lambda}\|\mathbf{y}-\mathbf{k}*\mathbf{x}\|_2^2$ from (12) is de-emphasized because of the inverse dependency on λ, and small edges and structures will be ignored. Hence an approximately convex penalty is generally sufficient for resolving the remaining strong edges. As the iterations proceed and fine structures need to be resolved, the penalty function is made less convex (more concave) as λ is reduced, but the risk of local minima and instability is ameliorated by the fact that we are likely to be already in the neighborhood of a desirable basin of attraction. Additionally, the implicit noise level (or modeling error) is now substantially lower.

This kind of automatic 'resolution' adaptive penalty shaping is arguably superior to conventional MAP approaches based on (3), where the concavity/shape of the induced separable penalty function is kept fixed regardless of the variation in the noise level or scale, i.e., at different resolutions across the coarse-to-fine hierarchy. In general, it would seem very unreasonable that the same penalty shape would be optimal across vastly different noise scales. This advantage over MAP can be easily illustrated by simple head-to-head comparisons where the underlying prior distributions are identical; Section 3.6 below contains one such example. Additionally, this phenomena can be significantly enhanced by automatically learning λ as discussed in Section 5.

3.5 Blur Dependency Analysis

There are many different viewpoints for understanding how the blur dependency of the VB penalty g_{VB} contributes to successful deblurring results. Here we consider potentially one of the most transparent.

Because the blur kernel appears judiciously in all three terms in (12), with a slight reparameterization and subsequent algebraic manipulation, we may consolidate terms leading to the equivalent revised formulation of (12) given by

$$\mathcal{L}(\tilde{\mathbf{x}},\tilde{\mathbf{k}}) \triangleq \frac{1}{\lambda}\left\|\mathbf{y}-\tilde{\mathbf{k}}*\tilde{\mathbf{x}}\right\|_2^2 + \sum_i g_{\text{VB}}(\tilde{x}_i,\lambda), \quad \text{s.t. } \|\tilde{\tilde{\mathbf{k}}}\|_2 = 1, \tag{15}$$

where $\tilde{x}_i \triangleq x_i \|\bar{\mathbf{k}}\|_2$ for all i and, with slight abuse of notation, $\tilde{\mathbf{k}}$ denotes a normalized kernel such that the resulting convolution matrix $\mathbf{H}$ has columns of unit norm. In other words, if $\mathbf{x}^*$ and $\mathbf{k}^*$ represent the optimal solution to (12), then $\tilde{\mathbf{x}}^* = \mathbf{x}^* \|\bar{\mathbf{k}}^*\|_2$ and $\tilde{\mathbf{k}}^* = \mathbf{k}^* / \|\bar{\mathbf{k}}^*\|_2$ are the optimal solution to (15), at least when g_{VB} is given by (14). Hence we see that, once the kernel operator is constrained to have unit ℓ_2 norm, *no additional kernel penalization is included whatsoever*. Consequently then, to the extent VB is successful, we observe that an additional kernel penalty, and any associated tuning parameter, is completely unnecessary. Additionally, with the kernel fixed, solving for $\tilde{\mathbf{x}}$ now represents a standardized sparse estimation problem, with a quadratic data-fit term now characterized by a design matrix $\mathbf{H}$ with consistent ℓ_2 normalized columns.

Note that essentially all previous blind deblurring algorithms assume what amounts to an ℓ_1 normalized kernel satisfying $\sum_i k_i = 1$ (since each element of the sum must be positive, the corresponding convolution matrix $\mathbf{H}$ will have ℓ_1 normalized columns except at the boundary). But in the context of a quadratic data fit term as used by deblurring algorithms, this is unlikely to be optimal as it will apply some implicit pressure on the estimated kernel towards the no-blur solution ($\mathbf{k} = \delta$), potentially counteracting, at least in part, the push for sparse image estimates. This occurs because kernels near the delta solution will increase the ℓ_2 column norms of $\mathbf{H}$ when the ℓ_1 norm is fixed, which then allows for relatively smaller image coefficients $\mathbf{x}$ by virtue of the quadratic data term. These smaller coefficients then reduce any non-decreasing penalty on the magnitudes of $\mathbf{x}$, reducing the overall cost function. Additionally, in much more complex non-uniform deblurring environments, this undesirable effect is considerably more pronounced (Zhang and Wipf, 2013).

In the context of VB however, this ℓ_1 normalization is implicitly switched to ℓ_2 normalization via the mechanism outlined above leading to (15), and hence it is entirely inconsequential to VB. In contrast, MAP has no such agency, and therefore it is not surprising that the majority of MAP algorithms explicitly include an additional ℓ_2 kernel penalty which helps to counteract movement towards no-blur solutions. The disadvantage of course is that an additional image-dependent tuning parameter is required as well to balance the resulting contribution. We could, however, alternatively consider replacing the ℓ_1 norm constraint in MAP with ℓ_2-norm constraints as in (15), although this complicates the optimization process considerably, whereas VB handles this automatically.[9]

3.6 Illustrative Example using 1D Signals

Here we will briefly illustrate some of the distinctions between MAP and VB discussed thus far where other confounding factors have been conveniently removed. For this purpose we consider a simplified noiseless situation where the optimal λ value is zero, and we consider the image prior produced when $f(\gamma) = b$ as introduced in Section 3.3 (later Section 3.7 will

9. One potential way around these complications for MAP would be to replace the non-convex constraint $\|\tilde{\bar{\mathbf{k}}}\|_2 = 1$ with the convex quadratic one $\|\tilde{\bar{\mathbf{k}}}\|_2 \leq 1$, ignoring boundary effects which would complicate things dramatically by requiring m additional constraints, one for each element of $\mathbf{x}$. While in uniform blur models this could be effective since the optimal solution should satisfy $\|\tilde{\bar{\mathbf{k}}}\|_2 = 1$ anyway, in non-uniform models this is unlikely the case, and such a substitution could still be difficult to implement and could undermine performance. But the central point remains that VB handles all of these situations naturally and seamlessly.

argue that this selection is in some sense optimal). For MAP we also include the kernel penalty $m \log \|\bar{\mathbf{k}}\|_2^2$ from (12) which will facilitate more direct comparisons with VB below. Given these assumptions, the associated MAP problem from (3) is easily shown to be

$$\min_{\mathbf{x},\mathbf{k}} \frac{1}{\lambda} \|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2 + \sum_i 2 \log |x_i| + 2m \log \|\bar{\mathbf{k}}\|_2, \tag{16}$$

where the image penalty is obtained by applying a $-2 \log$ transformation to (11) giving

$$-2 \log \left[\max_{\gamma_i \geq 0} \mathcal{N}(x_i; 0, \gamma_i) \right] \equiv 2 \log |x_i| \,. \tag{17}$$

Irrelevant additive constants have been excluded. Conveniently, since

$$\sum_i 2 \log |x_i| + 2m \log \|\bar{\mathbf{k}}\|_2 = \sum_i 2 \log \left(|x_i| \, \|\bar{\mathbf{k}}\|_2 \right), \tag{18}$$

we can reparameterize (16) using $\tilde{\mathbf{x}}$ such that the kernel penalty is removed and the constraint $\|\tilde{\bar{\mathbf{k}}}\|_2 = 1$ is enforced just as with VB. Consequently, in the limit as $\lambda \to 0$, based on the equivalency derived from Corollary 1, both VB and MAP are then effectively solving

$$\min_{\tilde{\mathbf{x}},\tilde{\mathbf{k}}} \sum_i \log |\tilde{x}_i| \,, \quad \text{s.t. } \mathbf{y} = \tilde{\mathbf{k}} * \tilde{\mathbf{x}}, \; \|\tilde{\bar{\mathbf{k}}}\|_2 = 1. \tag{19}$$

Moreover, given the arguments made in Section 3.3, the penalty on $\tilde{\mathbf{x}}$ is more or less equivalent to the ℓ_0 norm up to inconsequential scaling and translations. Thus, (19) effectively reduces to

$$\min_{\tilde{\mathbf{x}},\tilde{\mathbf{k}}} \|\tilde{\mathbf{x}}\|_0, \quad \text{s.t. } \mathbf{y} = \tilde{\mathbf{k}} * \tilde{\mathbf{x}}, \; \|\tilde{\bar{\mathbf{k}}}\|_2 = 1. \tag{20}$$

Therefore at this simplified, stripped-down level both VB and MAP are merely minimizing the ℓ_0 norm of $\mathbf{x}$ subject to the linear convolutional constraint. Of course we do not attempt to solve (20) directly, which is a difficult combinatorial problem in nature. Instead for both VB and MAP we begin with a large λ and gradually reduce it towards zero as part of a multi-resolution approach designed to avoid bad local minima as described in Section 3.4. For this reduction schedule of λ we use $\beta = 1.15$ in Algorithm 1 (this value is taken from Levin et al. 2011a).[10] While equivalent when $\lambda \to 0$, before λ becomes small the VB and MAP cost functions will behave very differently, leading to a radically different optimization trajectory terminating at different locally minimizing solutions to (20).

The superiority of the VB convergence path will now be demonstrated with a synthetic 1D signal composed of multiple spikes. This signal is convolved with two different blur kernels, one uniform and one random, creating two different blurry observations. Refer to Figure 2 (first row) for the ground-truth spike signal and associated blur kernels. We then apply the MAP and VB blind deconvolution algorithms, with the same prior (f equals a constant) and λ reduction schedule, to the blurry test signals and compare the quality of the

10. The corresponding MAP algorithm can be implemented by simply setting $\mathbf{C}$ to zero before the $q(\gamma_i)$ update in Algorithm 1, with guaranteed convergence to some local minima. For both MAP and VB, the γ sufficient statistic update is simply $\omega_i = \sigma_i^{-2}$ whenever f is a constant.

reconstructed blur kernels and signals. The recovery results are shown in Figure 2 (second and third rows), where it is readily apparent that VB produces superior estimation quality of both kernel and image. Additionally, the signal recovered by VB is considerably sparser than MAP, indicating that it has done a better job of optimizing (20), consistent with our previous analysis. This is not to say that MAP cannot potentially be effective with careful tuning and initialization (perhaps coupled with additional regularization factors or clever optimization schemes), only that VB is much more robust in its present form.

Note that this demonstrable advantage of VB is entirely based on an improved convergence path, since VB and MAP possess an identical constellation of local minima once $\lambda = 0$. Moreover, it is unrelated to any putative advantage of solving (4) over (3). We will revisit this latter point in Section 4.

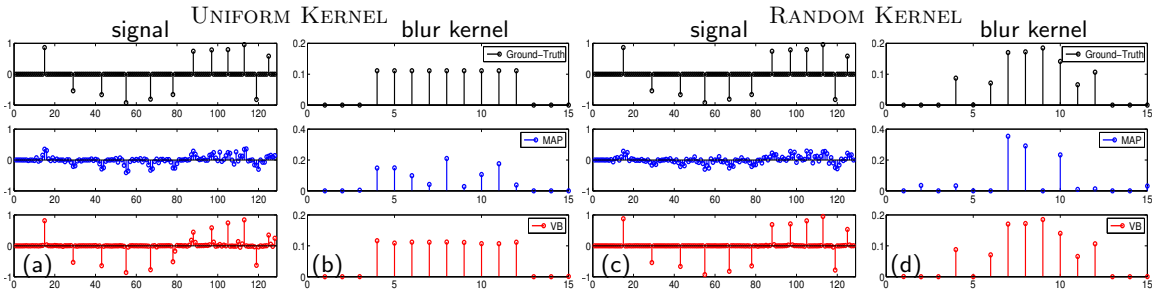


Figure 2: 1D deblurring example using MAP and VB approaches assuming the same underlying image prior $p(\mathbf{x})$. (a)-(b) results with a uniform blur kernel; (c)-(d) results with a random blur kernel.

3.7 Other Choices for f

Because essentially any sparse prior on $\mathbf{x}$ can be expressed using the alternative variational form from (11), choosing such a prior is tantamount to choosing f which then determines g_{VB}. Theorem 2 suggests that a concave, non-decreasing f is useful for favoring sparsity (assumed to be in the gradient domain). Moreover, Theorem 3 and subsequent analyses suggest that the simplifying choice where $f(\gamma) = b$ possesses several attractive properties regarding the relative concavity of the resulting g_{VB}. But what about other selections for f and therefore g_{VB}?

While directly working with g_{VB} can sometimes be limiting (except in certain special cases like $f(\gamma) = b$ from before), the variational form of (13) allows us to closely examine the relative concavity of a useful proxy. Let

$$\psi(\gamma_i, \rho) \triangleq \log(\rho + \gamma_i) + f(\gamma_i). \tag{21}$$

Then for fixed λ and $\mathbf{k}$ the VB estimation problem can equivalently be viewed as solving

$$\min_{\mathbf{x}, \boldsymbol{\gamma} \geq 0} \frac{1}{\lambda} \|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2 + \sum_i \left[\frac{x_i^2}{\gamma_i} + \psi(\gamma_i, \rho) \right]. \tag{22}$$

It now becomes clear that the sparsity of $\mathbf{x}$ and $\boldsymbol{\gamma}$ are intimated related. More concretely, assuming f is concave and non-decreasing (as motivated by Theorems 2 and 3), then there is actually a one-to-one correspondence in that whenever $x_i = 0$, the optimal γ_i equals zero as well, and vice versa.[11] Therefore we may instead examine the relative concavity of ψ for different ρ values, which will directly determine the sparsity of $\boldsymbol{\gamma}$ and in turn, the sparsity of $\mathbf{x}$. This then motivates the following result:

Theorem 4 *Let $\rho_1 < \rho_2$ and assume that f satisfies the conditions of Theorem 2. Then $\psi^{\rho_1} \prec \psi^{\rho_2}$ if and only if $f(\gamma) = a\gamma + b$, with $a \geq 0$.*

Thus, although we have not been able to formally establish a relative concavity result for all general g_{VB} directly, Theorem 4 provides a nearly identical analog allowing us to draw similar conclusions to those detailed in Sections 3.4 and 3.5 whenever a general affine f is adopted. Perhaps more importantly, it also suggests that as f deviates from an affine function, we may begin to lose some of the desirable effects regarding the described penalty shape modulation.

While previously we closely scrutinized the special affine case where $f(\gamma) = b$, it still remains to examine the more general affine form $f(\gamma) = a\gamma + b$, $a > 0$. In fact, it is not difficult to show that as a is increased, the resulting penalty on $\mathbf{x}$ increasingly resembles an ℓ_1 norm with lesser dependency on ρ, thus severely muting the effect of the shape modulation that appears to be so effective (see arguments above and empirical results section below). So there currently does not seem to be any advantage to choosing some $a > 0$ and we are left, out of the multitude of potential image priors, with the conveniently simple choice of $f(\gamma) = b$, where the value of b is inconsequential. Experimental results support this conclusion: namely, as a is increased from zero performance gradually degrades (results not shown for space considerations).

As a final justification for simply choosing $f(\gamma) = b$, there is a desirable form of invariance that uniquely accompanies this selection.

Theorem 5 *If $\mathbf{x}^*$ and $\mathbf{k}^*$ represent the optimal solution to (12) under the constraint $\sum_i k_i = 1$, then $\alpha^{-1}\mathbf{x}^*$ and $\alpha\mathbf{k}^*$ will always represent the optimal solution under the modified constraint $\sum_i k_i = \alpha$ if and only if $f(\gamma) = b$. Additionally, minimizing (12) is equivalent to minimizing (15) if and only if $f(\gamma) = b$.*

This is unlike the myriad of MAP estimation techniques or VB with other choices of f, where the exact calibration of the constraint can fundamentally alter the form of the optimal solution beyond a mere rescaling. Moreover, if such a constraint on $\mathbf{k}$ is omitted altogether, these other methods must then carefully tune associated trade-off parameters, so in one way or another this lack of invariance will require additional tuning.

Interestingly, Babacan et al. (2012) experiments with a variety of VB algorithms using different underlying image priors, and empirically find that f as a constant works best;

11. To see this first consider $x_i = 0$. The x_i^2/γ_i term can be ignored and so the optimal γ_i need only minimize $\log(\rho + \gamma_i) + f(\gamma_i)$, which is concave and non-decreasing whenever f is. Therefore the optimal γ_i is trivially zero. Conversely if $\gamma_i = 0$, then there is effectively an infinite penalty on x_i, and so the optimal x_i must also be zero.

however, no rigorous explanation is given for why this should be the case.[12] Thus, our results provide a powerful theoretical confirmation of this selection, along with a number of useful attendant intuitions.

3.8 Analysis Summary

To summarize this section, we have shown that the shape of the effective VB image penalty is explicitly controlled by the ratio of the noise variance to the squared kernel norm, and that in many circumstances this leads to a desired mechanism for controlling relative concavity and balancing sparsity, largely mitigating issues such as local minima that compromise the convergence of more traditional MAP estimators. We have then demonstrated a unique choice for the image prior (i.e., when f is constant) such that this mechanism is in some sense optimal and scale-invariant. Of course we readily concede that different choices for the image prior could still be useful when other factors are taken in to account. We also emphasize that none of this is meant to suggest that real imaging data follows a Jeffreys prior distribution (which is produced when f is constant). We will return to this topic in Section 4 below. Overall, this perspective provides a much clearer picture of how VB is able to operate effectively and how we might expect to optimize performance.

While space precludes a detailed treatment, many natural extensions to VB are suggested by these developments. For example, in the original formation of VB given by (7) it is not clear the best way to incorporate alternative noise models because the required integrations are no longer tractable. However, when viewed alternatively using (12) it becomes obvious that different data-fidelity terms can easily be substituted in place of the quadratic likelihood factor. Likewise, given additional prior knowledge about the blur kernel, there is no difficulty in substituting for the ℓ_2-norm on $\mathbf{k}$ or the uniform convolutional observation model to reflect additional domain knowledge. Thus, the proposed reformulation allows VB to inherit most of the transparent extensibility previously reserved for MAP.

We may also consider these ideas in the context of existing MAP algorithms, which adopt various structure selection heuristics, implicitly or explicitly, to achieve satisfactory performance (Shan et al., 2008; Cho and Lee, 2009; Xu and Jia, 2010). This can be viewed as adding additional image penalty terms and trade-off parameters to (3). For example, Shan et al. (2008) incorporates an extra local penalty on the latent image, such that the gradients of small-scale structures in the recovered image are close to those in the blurry image. Thus they will actually contribute less to the subsequent kernel estimation step, allowing larger structures to be captured first. Similarly, a bilateral filtering step is used for pruning out small scale structures in Cho and Lee (2009). Finally, Xu and Jia (2010) develop an empirical structure selection metric designed such that small scale structures can be pruned away by thresholding the corresponding response map, allowing subsequent kernel estimation to be dominated by only large-scale structures.

12. Based on a strong simplifying assumption that the covariance $\mathbf{C}$ from Algorithm 1 is a constant, Babacan et al. (2012) provides some preliminary discussion regarding possibly why VB may be advantageous over MAP. However, when $\mathbf{C}$ is constant, the analysis easily reduces to a standard penalized regression problem, and hence this material can already be found in the sparse estimation literature (e.g., see Palmer et al. 2006; Wipf et al. 2011 and related references). Our key contribution is to explicitly account for the actual dynamic nature of $\mathbf{C}$ and expose the true behavior of VB blind deblurring.

Generally speaking, existing MAP strategies face a trade-off: either they must adopt a highly sparse image prior needed for properly resolving fine structures (see Section 4) and then deal with the attendant constellation of problematic local minima,[13] or rely on a more smooth image prior augmented with compensatory structure-selection measures such as those described above to avoid bad global solutions. In contrast, we may interpret the coupled penalty function intrinsic to VB as a principled alternative with a transparent, integrated functionality for estimation at different resolutions without any additional penalty factors, trade-off parameters, or complexity.

4. Maximal Sparsity vs. Natural Image Statistics

Levin et al. (2009, 2011a,b), which represents the initial inspiration for our work, presents a compelling and highly influential case that joint MAP estimation over $\mathbf{x}$ and $\mathbf{k}$ generally favors a degenerate, no-blur solution, meaning that $\mathbf{k}$ will be a delta function, even when the assumed image prior $p(\mathbf{x})$ reflects the true underlying distribution of $\mathbf{x}$, meaning $p(\mathbf{x}) = p_{\text{true}}(\mathbf{x})$, and $p(\mathbf{k})$ is assumed flat in the feasible region.[14] In turn, this is presented as a primary argument for why MAP is inferior to VB. As this line of reasoning is considerably different from that given in Section 3, here we will take a closer look at these orthogonal perspectives in the hopes of providing a clarifying resolution.

To begin, it helps to revisit the formal analysis of MAP failure from Levin et al. (2011b), where the following specialized scenario is presented. Assume that a blurry image $\mathbf{y}$ is generated by $\mathbf{y} = \mathbf{k}^* * \mathbf{x}^*$, where $\|\mathbf{k}^*\|_2 \ll 1$ and each true sharp image gradient x_i^* is drawn iid from the *generalized Gaussian distribution* $p_{\text{true}}(x_i^*) \sim \exp(-|x_i^*|^p)$, $0 < p \leq 1$. Now consider the minimization problem

$$\min_{\mathbf{x},\mathbf{k}} \sum_i |x_i|^p \quad \text{s.t. } \mathbf{y} = \mathbf{k} * \mathbf{x}. \tag{23}$$

Solving (23) is equivalent to MAP estimation over $\mathbf{x}$ and $\mathbf{k}$ under the true image prior $p_{\text{true}}(\mathbf{x})$ and an implicitly assumed flat prior on $\mathbf{k}$ within the previously specified kernel constraint set. In the limit as the image grows arbitrarily large, (Levin et al., 2011b, Claim 2) proves that the no-blur delta solution $\{\mathbf{x} = \mathbf{y}, \mathbf{k} = \delta\}$ will be favored over the true solution $\{\mathbf{x} = \mathbf{x}^*, \mathbf{k} = \mathbf{k}^*\}$. Intuitively, this occurs because the blurring operator $\mathbf{k}$ contributes two opposing effects:

1. It reduces a measure of the image *sparsity*, which increases $\sum_i |y_i|^p$, and
2. It broadly reduces the overall image *variance*, which reduces $\sum_i |y_i|^p$.

Depending on the relative contributions, we may have the situation where the second effect dominates such that $\sum_i |y_i|^p$ may be less than $\sum_i |x_i^*|^p$, meaning the cost function value

13. Appropriate use of continuation methods such as the algorithm from Chartrand and Yin (2008) may help in this regard.

14. Note that Levin *et al.* frequently use $\text{MAP}_{x,k}$ to refer to joint MAP estimation over both $\mathbf{k}$ and $\mathbf{x}$ (Type I) while using MAP_k for MAP estimation of $\mathbf{k}$ alone after $\mathbf{x}$ has been marginalized out (Type II). In this terminology, MAP_k then represents the inference ideal that VB purports to approximate, equivalent to (4) herein.

at the blurred image is actually lower than at the true, sharp image. Consequently, MAP estimation may not be reliable.

Our conclusions then suggest a sort of paradox: in Section 3 we have argued that VB is actually equivalent to an unconventional form of MAP estimation over $\mathbf{x}$, but with an intrinsic mechanism for avoiding bad local minima, increasing the chances that a good global or near-global minima can be found. Moreover, at least in the noiseless case ($\lambda \to 0$), any such minima will be exactly equivalent to the standard MAP solution by virtue of Corollary 1. However, based on the noiseless analysis from Levin *et al.* above, any global MAP solution is unlikely to involve the true sharp image when the true image statistics are used for $p(\mathbf{x})$, meaning that VB performance should be poor as well even at a global solution. Thus how can we reconcile the positive performance of VB actually observed in practice, and avoidance of degenerate no-blur solutions, with Levin *et al.*'s characterization of the MAP cost function?

First, when analyzing MAP, Levin *et al.* consider only a flat prior on $\mathbf{k}$ within the constraint set $\sum_i k_i = 1$ and $k_i \geq 0$. However, MAP estimation may still avoid no-blur solutions when equipped with an appropriate non-flat kernel prior and associated trade-off parameter. Likewise under certain conditions described in Section 3.5, VB naturally substitutes in a quadratic normalization constraint for $\mathbf{k}$ that we have argued disfavors no-blur solutions automatically. Moreover, VB introduces this normalization in a convenient form devoid of additional tuning parameters.

Secondly, the argument in Levin *et al.* breaks down when the true sharp image $\mathbf{x}^*$ is actually sparse in the canonical sense, meaning the distribution of each element includes a delta function at zero, i.e.,

$$p_{\text{true}}(x_i^*) = \alpha\delta(x_i^*) + (1-\alpha)\rho(\mathbf{x}_i), \tag{24}$$

where ρ is an arbitrary distribution and $\alpha \in [0,1]$ is a constant. Clearly samples from (24) will include some elements exactly equal to zero with probability at least α.

Lemma 1 *Let $\mathbf{x}^*$ be distributed iid with elements drawn from (24) and let $\mathbf{y} = \mathbf{k}^* * \mathbf{x}^*$ for some non-negative kernel $\mathbf{k}^*$. Then with probability approaching one as the image size grows large*

$$\mathbf{k}^*, \mathbf{x}^* = \arg \min_{\mathbf{k},\mathbf{x}:\mathbf{y}=\mathbf{k}*\mathbf{x}} \log p_{\text{true}}(x_i^*) = \arg \min_{\mathbf{k},\mathbf{x}:\mathbf{y}=\mathbf{k}*\mathbf{x}} \|\mathbf{x}\|_0. \tag{25}$$

The proof is straightforward and we do not reproduce it here. Regardless, this result demonstrates that exactly sparse images can in fact be recovered using MAP or equivalently the ℓ_0 norm, the latter of which is actually blind to the distribution of non-zero coefficients $\rho(\mathbf{x}_i)$. Intuitively, this occurs because these measures are entirely immune to changes in variance and only sensitive to sparsity; hence any blurring operation will only increase either penalty function in the feasible region. So immediately we may conclude that, assuming we have some way of solving (25), we should not discount MAP or ℓ_0 minimization as a viable means for recovering sparse images. And importantly, the exact distribution of nonzero coefficients is irrelevant as long as some degree of sparsity exists.

Of course most practical images of interest are not exactly sparse. Rather, a commonly-reported estimate of true image gradient statistics is the generalized Gaussian distribution with $p \approx [0.5, 0.8]$, samples from which will have no exactly zero-valued elements (Buccigrossi and Simoncelli, 1999). In this regime then the original claim from Levin *et al.* will

hold and MAP estimation seems to have been discredited. However, we would argue that MAP can still be salvaged if we are willing to intentionally allow mismatch between the true image prior and the image prior which forms the basis of our MAP estimator. More specifically, we suggest replacing the true image statistics p_{true} with the ℓ_0 norm. However, solving (25) directly will obviously not be effective when there are no exactly zero-valued coefficients.

Fortunately there is a simple way around this. In the regime where $n \approx m$, meaning the sharp and blurry images $\mathbf{y}$ and $\mathbf{x}$ are large relative to the size of $\mathbf{k}$ and therefore of nearly equal dimension, the generalized Gaussian distribution with $p \approx [0.5, 0.8]$ is a *compressible distribution* in the sense described in Cevher (2009). In words, this means that the sorted magnitudes of samples from this distribution exhibit a power-law decay and hence can be well-approximated by sparse signals. Consequently, there will exist some sparse $\hat{\mathbf{x}}$ with $\|\hat{\mathbf{x}}\|_0 \ll m$ such that $\|\mathbf{y} - \mathbf{k}^* * \hat{\mathbf{x}}\|_2^2 < \epsilon$ for some small ϵ. In contrast, each element of the blurry image $\mathbf{y}$ is a summation of many elements of $\mathbf{x}^*$ via the blur operation, and therefore, by central limit theorem arguments each element, while not exactly iid, will approach samples from a Gaussian distribution (exactly so for large enough blur kernels), which is not a compressible distribution. Therefore, if we solve a relaxed version of (25) given by

$$\min_{\mathbf{x},\mathbf{k}} \|\mathbf{x}\|_0, \quad \text{s.t. } \|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2 < \epsilon, \tag{26}$$

with an appropriate choice for ϵ, then we are very likely to obtain the true blur kernel $\mathbf{k}^*$, and a close sparse approximation to $\mathbf{x}^*$. Conversely it is very *unlikely* that the solution will be $\mathbf{x} = \mathbf{y}$ and $\mathbf{k} = \delta$. Therefore, just because $\mathbf{x}^*$ may not be exactly sparse, we may nonetheless locate a sparse approximation $\hat{\mathbf{x}}$ that is sufficiently reasonable such that the unknown $\mathbf{k}^*$ can still be estimated accurately, facilitating a later non-blind, image domain estimation step.

Overall then, the success of the ℓ_0 norm penalty in the context of MAP estimation speaks to the following point: it is more important that *the assumed image prior* $p(\mathbf{x}) = \exp[-\frac{1}{2} g_{\mathbf{x}}(\mathbf{x})]$ *be maximally discriminative with respect to blurred and sharp images, as opposed to accurately reflecting the statistics of real images.* Mathematically, this implies that it is much more important that we have $p(\mathbf{k} * \mathbf{x}^*) \ll p(\hat{\mathbf{x}})$ for some $\hat{\mathbf{x}}$ such that $\mathbf{k} * \mathbf{x}^* \approx \mathbf{k} * \hat{\mathbf{x}}$, than we enforce $p(\mathbf{x}) = p_{\text{true}}(\mathbf{x})$, even if $p_{\text{true}}(\mathbf{x})$ were known exactly. This is because the sparsity/variance trade-off described above implies that it may often be the case that $p_{\text{true}}(\mathbf{k} * \mathbf{x}^*) > p_{\text{true}}(\hat{\mathbf{x}})$ leading to the no-blur solution.

Obviously from a practical standpoint solving (26) represents a difficult, combinatorial optimization problem with numerous local minima. However, to the extent that the VB image penalty g_{VB} approximates the ℓ_0 norm, the VB cost (12) can be viewed as an approximate Lagrangian form of (26), but augmented with an adaptive shape modulation that helps to circumvent these local minima. Thus we can briefly summarize largely why VB can be superior to MAP: *VB allows us to use a near-optimal image penalty, one that is maximally discriminative between blurry and sharp images, but with a reduced risk of getting stuck in bad local minima during the optimization process.* Overall, these conclusions provide a more complete picture of the essential differences between MAP and VB.

Before proceeding to the next section, we emphasize that none of the arguments presented herein discredit the use of natural image statistics when directly solving (4). In fact

Levin et al. (2011b) prove that when $p(\mathbf{x}) = p_{\text{true}}(\mathbf{x})$, then in the limit as the image grows large the MAP estimate for $\mathbf{k}$, after marginalizing over $\mathbf{x}$ (Type II), will equal the true $\mathbf{k}^*$. But there is no inherent contradiction with our results, since it should now be readily apparent that VB is fundamentally different than solving $\min_{\mathbf{k}} p(\mathbf{k}|\mathbf{y})$, and therefore justification for the latter cannot be directly transferred to justification for the former. This highlights the importance of properly differentiating various forms of Bayesian inference, both in the context of blind image deblurring and beyond to widespread application domains.

Natural image statistics are ideal in cases where $\mathbf{y}$ and $\mathbf{x}$ grow large and we are able to integrate out the unknown $\mathbf{x}$, benefiting from central limit arguments when estimating $\mathbf{k}$ alone. However, when we jointly compute MAP estimates of both $\mathbf{x}$ and $\mathbf{k}$ (Type I) as in (3), we enjoy no such asymptotic welfare since the number of unknowns increases proportionally with the sample size. One of the insights of our paper is to show that, at least in this regard, VB is on an exactly equal footing with Type I MAP, and thus we must look for theoretical VB justification elsewhere, leading to the analysis of relative concavity, local minima, invariance, maximal sparsity, etc. presented herein.

5. Learning λ

While existing VB blind deconvolution algorithms typically utilize some pre-assigned decreasing sequence for λ as described in Section 3.4 and noted in Algorithm 1, it is preferable to have λ learned automatically from the data itself as is common in other applications of VB. In the case of blind deblurring, we expect that such a learned λ, with an image-dependent trajectory, may better modulate the penalty curvature discussed in Section 3.4. In contrast, a fixed, pre-defined decreasing sequence is likely to be miscalibrated as it will not reflect the current quality of image and kernel estimates during each iteration. Additionally, the alternative strategy of learning λ has the conceptual appeal of an integrated cost function that is universally reduced even as λ is updated, unlike Algorithm 1 where the λ reduction step may in fact increase the overall cost.

However, current VB deblurring papers either do not mention such a seemingly obvious alternative (perhaps suggesting that the authors unsuccessfully tried such an approach) or explicitly mention that learning λ is problematic but without concrete details. For example, Levin et al. (2011b) observed that the noise level learning used in Fergus et al. (2006) represents a source of problems as the optimization diverges when the estimated noise level decreases too much. But there is no mention of why λ might decrease too much, and further details or analyses are absent.

Interestingly, the perspective presented herein provides some direct insights into how λ may be effectively learned. Consider minimization of the revised VB cost function (12) over $\mathbf{x}$, $\mathbf{k}$, and now λ as well. Because $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^n$ with $m > n$, for a fixed $\mathbf{k}$ there are an infinite number of candidate solutions such that $\mathbf{y} = \mathbf{k} * \mathbf{x}$ since the corresponding null-space of the convolution matrix is of dimension $m - n$. Therefore there exist an infinite set images $\mathbf{x}$ such that the term $\frac{1}{\lambda} \|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2$ in the VB cost function (12) can be minimized to exactly zero even in the limit as $\lambda \to 0$. Included in this set are basic feasible solutions (in the linear programming sense), each of which have at least $m - n$ elements of $\mathbf{x}$ equal to zero.

A problem then arises because it can be shown that $g_{\text{VB}}(0,\rho) \rightarrow -\infty$ as $\rho \rightarrow 0$ for all non-decreasing f.[15] Consequently, we can always trivially drive the VB cost function (12) to $-\infty$ using any basic feasible solution combined with $\lambda \rightarrow 0$, regardless of the quality of the solution. Now because of the disparity in dimensionality mentioned above, there will always be feasible solutions to $\mathbf{y} = \mathbf{k} * \mathbf{x}$ with at least $m - n$ or more elements of $\mathbf{x}$ equal to zero. Thus, at any one of these solutions the VB cost function (12) can then be driven to $-\infty$ with $\lambda \rightarrow 0$. Unless the true $\mathbf{x}$ actually has many exactly zero-valued elements, this will represent a globally degenerate minimizing solution for a broad class of f. And even for other choices for f, a slightly more subdued form of this same degeneracy will still exist since the VB-specific regularization fundamentally favors λ being small: essentially the $\log(\gamma_i + \rho)$ factor in (13) will always favor ρ, and therefore λ being small. The $1/\lambda$ weighting of $\|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2$ is not sufficient for counteracting this effect given the multitude of feasible solutions such that $\mathbf{y} = \mathbf{k} * \mathbf{x}$.

And even for other choices for f, a slightly more subdued form of this same degeneracy will still persist since the existing VB-specific regularization fundamentally favors λ being small: essentially the $\log(\gamma_i + \rho)$ factor in (13) will always favor ρ, and therefore λ being small. The $1/\lambda$ weighting of $\|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2$ is not sufficient for counteracting this effect given the multitude of feasible solutions such that $\mathbf{y} = \mathbf{k} * \mathbf{x}$.

However, these degeneracies can be circumvented with an additional penalty factor on λ that is naturally motivated by this framework. Specifically, we propose to append the penalty function

$$v(\lambda) = (n - m)\log\lambda + \frac{d}{\lambda} \tag{27}$$

to (12), where d is assumed to be a small positive constant. The first term in (27) directly counteracts the degeneracy of basic feasible solutions by providing an equal and opposite barrier to arbitrary solutions with $\lambda \rightarrow 0$ and $\|\mathbf{x}\|_0 = m - n$. Additionally, when f is a constant as we have argued previously represents a well-motivated selection, then it can be shown that this additional penalty represents a very principled approximation to what the true λ penalty should be if the original VB formulation from (6) were not factorized as in (7). Additionally, for other choices of f, (12) can be similarly modified to provide a consistent estimator for λ in the sense described in Wipf and Wu (2012).

As justification for the second term in (27), note that this added factor is proportional to $1/\lambda\,\|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2$, but acts as an interpretable barrier preventing λ from ever going below d/n, which remains a possibility even with the $(n - m)\log\lambda$ term in place. In fact it is easily shown (see Appendix B) that any λ minimizing the cost function (12) augmented with the penalty $v(\lambda)$ must satisfy $\lambda \geq d/n$, which can be viewed as a lower-bound on what $1/n\,\|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2$ should be.[16]

In practice, we have found the fixed value $d = n \times 10^{-4}$ to be highly effective across a wide range of images and testing scenarios, including all reported results in Section 6 and

15. Based on (13), it is clear that the optimizing γ_i value for computing $g_{\text{VB}}(0,\rho)$ will be $\gamma_i = 0$. When $\rho \rightarrow 0$, we then have $\log(\gamma_i + \rho) \rightarrow -\infty$, and therefore $g_{\text{VB}}(0,\rho) \rightarrow -\infty$. Graphically, Figure 1 (b) also reveals this effect, showing that if we were to jointly minimize over both x and ρ, the $\{0,0\}$ solution is heavily favored.

16. While it could be argued that setting d to a larger value could obviate the need for the $(n - m)\log\lambda$ penalty altogether, we would lose considerable interpretability, connection with the original VB problem, and from a practical standpoint, we would likely be saddled with a more sensitive tuning heuristic for d.

Algorithm 2 VB Blind Deblurring with Jeffreys Prior and Learned λ (VB+).

1: **Input:** blurry image $\mathbf{y}$, noise level estimation hyper-parameter $d = n \times 10^{-4}$
2: **Initialize:** blur kernel $\mathbf{k}$, noise level λ
3: **While** stopping criteria is not satisfied, modify ω_i and λ updates from Algorithm 1 with $\omega_i \leftarrow \sigma_i^{-2}, \forall i$ and $\lambda \leftarrow \frac{\|\mathbf{y}-\boldsymbol{\mu}*\mathbf{k}\|_2^2+\sum_i(\|\bar{\mathbf{k}}\|_2^2 \cdot C_{ii})+d}{n}$
4: **End**

all of the real-world, more complex non-uniform deblurring experiments from Zhang and Wipf (2013). Regardless, use of a single, fixed value is likely to be less burdensome than producing an entire λ reduction schedule, which also requires a user-specified minimal λ value anyway. Moreover, the VB update rules only require a slight modification to account for this additional term while retaining existing convergence properties (see Appendix B for the derivation). By estimating the noise level together with the image and kernel, we not only make the deblurring algorithm more noise-aware and mostly parameter-free. But more importantly, by initializing with a large value and allowing the iterations to learn the optimal reduction schedule, it offers a natural coarse-to-fine process for blind deblurring, which has been found as one of the crucial factors for blind deblurring algorithms as discussed above. The experimental results from Section 6 support this conclusion.

6. Experimental Results

We emphasize that the primary purpose of this paper is the formal analysis of existing VB blind deconvolution methodology, not the development and validation of an entirely practical system per se. Some empirical support for recent VB algorithms, complementary to our theoretical presentation, already exist (Babacan et al., 2012; Levin et al., 2011a; Zhang et al., 2013; Zhang and Wipf, 2013). Nonetheless, motivated by our results herein, we will briefly evaluate two simple refinements of Algorithm 1 that help corroborate some of our analytical findings while demonstrating that an extremely simplified version of VB, albeit with theoretically sound underpinnings, can outperform recent published state-of-the-art MAP and VB algorithms with considerably more complexity and/or manual parameters. In doing so, we hope to motivate the optimal usage of VB for more sophisticated and realistic blind deblurring problems.

To this end we will (i) use an image prior obtained when f is flat (Jeffreys prior) as motivated in Section 3 instead of a prior based on natural image statistics, and (ii) we will learn the λ parameter automatically per the discussion in Section 5. The revised estimation steps are summarized in Algorithm 2, which is obtained by adopting the basic procedure from Algorithm 1 under the special case $f(\gamma) = b$ and with the λ updates derived in Appendix B. We will refer to this algorithm as *VB+*. Note that estimation is performed in the gradient domain using the filters described in Section 3.1; however, the recovered kernel is applied to a non-blind deconvolution step to obtain the final latent image estimate. This filtering and final non-blind step, taken from Levin et al. (2011a), is standardized across all algorithms compared in this section, with the exception of approach from Babacan et al. (2012), which uses its own specially-tailored non-blind step.

Given this variant of VB, we reproduce the experiments from Levin et al. (2011a) using the useful benchmark test data from Levin et al. (2009).[17] This consists of 4 base images of size 255×255 and 8 different blurring effects, leading to a total of 32 blurry images. Ground truth blur kernels were estimated by recording the trace of focal reference points on the boundaries of the sharp images (see Levin et al. 2011b, Figure 7 and related text for details of the experimental setup and data collection). The kernel sizes range from 13×13 to 27×27. All evaluations are based on the SSD (Sum of Squared Difference) metric defined in Levin et al. (2009), which quantifies the error between estimated and the ground-truth images. To normalize for the fact that harder kernels give a larger image reconstruction error even when the true kernel is known (because the corresponding non-blind deconvolution problem is also harder), the SSD ratio between the image deconvolved with the estimated kernel and the image deconvolved with the ground-truth kernel is used as the final evaluation measure.

We first compare VB+ as described in Algorithm 2 with the related variational Bayesian methods from Fergus et al. (2006), Levin et al. (2011a), and Babacan et al. (2012), labeled VB-Fergus, VB-Levin, and VB-Babacan respectively. For VB-Fergus and VB-Levin, results directly accompany the Levin *et al.* data set; for VB-Babacan the results are produced using a script provided directly from the first author's website explicitly designed for producing competitive results with the Levin *et al.* data (note that this code contains an additional kernel penalty with added trade-off parameters set by the authors for working with this data set). While all three existing VB algorithms can be effective in practice, they have not been optimized with respect to the considerations provided herein. With VB-Fergus and VB-Levin, the adopted image priors are loosely based on the statistics of natural scenes and, as we have argued in Sections 3 and 4, may not be optimal. While VB-Babacan also involves a prior with f flat like VB+, it adopts the fixed λ reduction schedule originally from VB-Levin, and hence cannot exploit the full potential of VB.

The cumulative histogram of the SSD error ratios is shown in Figure 3(a) for all VB methods. The height of the bar indicates the percentage of images having error ratio below that level. High bars indicate better performance. As mentioned by Levin *et al.*, the results with error ratios above 2 may already have some visually implausible regions (Levin et al., 2009). VB+ can achieve close to 90% success with error ratio below 2, significantly higher than the others.

Regardless, all of the VB algorithms still exhibit reasonable performance, especially given that they do not benefit from any additional prior information or regularization heuristics that facilitate blur-adaptive structure selection (meaning the additional regularization based on domain knowledge added to Equation 3 that boost typical MAP algorithms as discussed previously). However, one curious phenomenon is that both VB-Fergus and VB-Levin experience a relatively large drop-off in performance when the error ratio reduces from 1.5 to 1.1. While it is difficult to be absolutely certain, one very plausible explanation for this decline relates to the prior selection employed by these algorithms. In both cases, the prior is based on a finite mixture of zero mean Gaussians with different variances roughly matched to natural image statistics. While such a prior does heavily favor approximately sparse signals, it will never produce any exactly sparse estimates at any resolution of the course-to-fine hierarchy, and hence, especially at high resolutions the penalty shape modu-

17. This data is available online at `http://www.wisdom.weizmann.ac.il/~levina/papers/LevinEtalCVPR09Data.rar`

lation effect of VB will be highly muted, as will be the beneficial sparsity/variance trade-off that accompanies more strongly sparse priors. Thus these algorithms may not be optimal for resolving extremely fine details, which is required for reliably producing image estimates with low error ratios. In contrast, to achieve high error ratios only lower resolution features need be resolved, and in this regime VB-Levin, which is also based directly on Algorithm 1, performs nearly as well as VB+. Also note that VB-Babacan performs relatively poorly at the higher error ratios, which is likely attributable to the fact that local minima and more catastrophic errors are a serious problem when using the highly non-convex Jeffreys distribution without a proper schedule for reducing λ that is tuned to the image prior.

We next compare VB+ with several state-of-the-art MAP algorithms from Shan et al. (2008), Xu and Jia (2010), and Cho and Lee (2009). Shan *et al.* (denoted MAP-Shan) adopts an additional local smoothness prior designed to reduce ringing artifacts. Xu *et al.* (MAP-Xu) includes two phases for kernel estimation and incorporates an explicit scheme for edge structure selection. Finally, Cho *et al.* (MAP-Cho) is also a carefully-engineered MAP approach coupled with structure selection and sharp edge prediction schemes, which help the algorithm to avoid the degenerate delta solution. Recall that previously we have argued that standard MAP algorithms may suffer from one of two problems: either the pixel-wise image prior is highly sparse and convergence to sub-optimal local solutions becomes a problem, or the prior is less sparse and global solutions do not sufficiently distinguish blurry from sharp images. All of the MAP algorithms tested here can be viewed as addressing this conundrum by including additional regularization schemes (priors) such that global or near global minima favor sharp images even when the basic pixel-wise image prior is convex (i.e., minimally sparse). This is a very different strategy than VB, which adopts a simpler underlying model with no additional regularizers beyond the canonical pixel-wise sparse prior. Figure 3(b) reveals that the simple VB strategy, when properly implemented, can still outperform specially tuned MAP estimates. Note that the results of MAP-Cho are from the data set accompanying Levin et al. (2011a) directly, while the results of MAP-Shan and MAP-Xu are produced using the software provided by the authors, for which we adjust the parameters carefully. For all algorithms we run every test image with the same parameters, similar to Levin et al. (2009, 2011b). Overall, VB+ obtains the highest reported result of any existing VB or MAP algorithm on this important benchmark.

7. Conclusion

This paper presents an insightful reformulation and subsequent analysis of MAP and VB blind deconvolution algorithms revealing why practical success is possible and suggesting valuable improvements for the latter. We summarize the contributions of this perspective as follows:

- Levin et al. (2009, 2011b) have provided an interesting analysis of VB and related MAP algorithms. We push the limits of understanding further, demonstrating that rigorous evaluation of VB and its associated priors cannot be separated from implementation heuristics, and we have meticulously examined the interplay of the relevant underlying algorithmic details employed by practical VB systems. Consequently, what may initially appear to be a plausible rationale for achieving high performance may

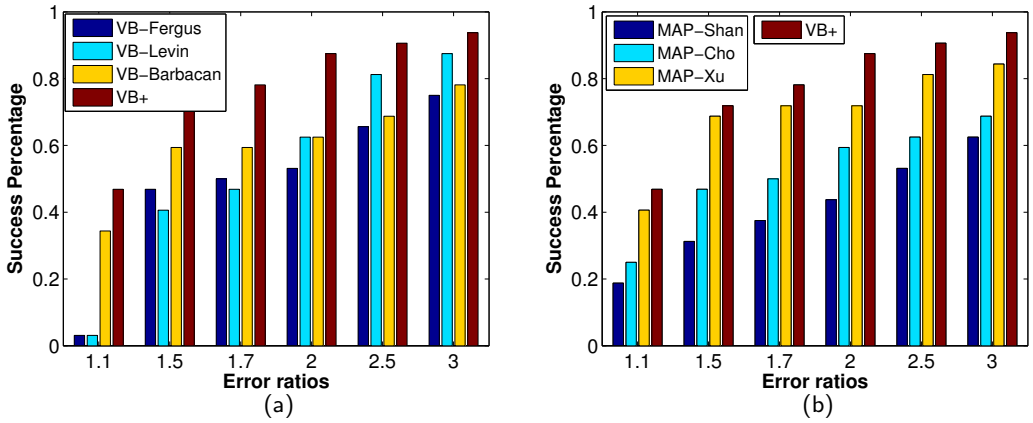


Figure 3: Evaluation of the restoration results: Cumulative histogram of the deconvolution error ratio across 32 test examples. The height of the bar indicates the percentage of images having error ratio below that level. High bars indicate better performance. (a) comparison with several other VB algorithms. (b) comparison with several state-of-the-art MAP algorithms.

have limited applicability given the assumptions required to implement scalable versions of VB.

- We have proven that in an ideal, noiseless setting, VB and MAP have an identical underlying cost function once the requisite approximations are accounted for (and they are intimately related even when noise is present). This is in direct contrast to conventional assumptions explaining the presumed performance advantages of VB.

- We carefully examine the underlying VB objective function in a transparent form, leading to principled criteria for choosing the optimal image prior. It is crucial to emphasize that this image prior need not, and generally should not, reflect the most accurate statistics of real imaging data. Instead, the preferred distribution is one that is most likely to guide VB iterations to high quality global solutions by strongly differentiating between blurry and sharp images. In this context, we have motivated a unique selection, out of the infinite set of possible sparse image priors, that simultaneously allows for maximal discrimination between $\mathbf{k}*\mathbf{x}$ and $\mathbf{x}$, displays a desirable form of scale invariance, and leads to an intrinsic coupling between the blur kernel, noise level, and image penalty such that bad local minima can largely be avoided. To the best of our knowledge, this represents a completely new viewpoint for understanding VB algorithms.

- The cause of failure when using standard MAP algorithms depends on the choice of image prior. If $-2\log p(\mathbf{x})$ is only marginally concave in $|\mathbf{x}|$, or is tuned to natural image statistics, then the problem is often that global or near-global solutions do not properly differentiate blurry from sharp images. In contrast, if $p(\mathbf{x})$ is highly sparse, while global solutions may be optimally selective for sharp image gradients,

convergence to bad local solutions is more-or-less inevitable. It is with the latter that VB offers a compelling advantage.

- We have derived and analyzed a simple yet powerful extension of VB deconvolution algorithms for learning the noise level.
- By reframing VB as a nearly parameter-free sparse regression problem in standard form, we demonstrate that it is no longer difficult to enhance performance and generality by inheriting additional penalty functions (such as those from Shan et al. 2008) or noise models (e.g., Laplacian, Poisson, etc.) commonly reserved for MAP. Moreover, we anticipate that these contributions will lead to a wider range of principled VB applications, such as non-uniform deconvolution (Whyte et al., 2012; Zhu and Milanfar, 2013) and multi-frame and video deblurring (Sroubek and Milanfar, 2012; Takeda and Milanfar, 2011). Preliminary results show tremendous promise (Zhang et al., 2013; Zhang and Wipf, 2013). Additionally, the analysis we conducted for blind deconvolution may well be relevant to other related bilinear models like robust dictionary learning in the presence of noise.

Overall, we hope that these observations will ensure that VB is not under-utilized in blind deconvolution and related tasks. We conclude by mentioning that, given the new perspective on VB provided herein, it may be possible to derive new blind deblurring algorithms and penalty functions that deviate from the VB script but nonetheless adopt some of its attractive properties. This is a direction of ongoing research.

8. Acknowledgements

We would especially like to thank Yi Ma and Aaron Hertzmann for providing some insightful comments on an earlier version of this manuscript, and to the reviewers for several useful suggestions.

Appendix A. Proofs

This section provides proofs of various technical results described in this paper.

A.1 Proof of Theorem 1

We begin with the cost function

$$\mathcal{L}(\mathbf{x}, \mathbf{k}, \boldsymbol{\gamma}) \triangleq \frac{1}{\lambda}\|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2 + \sum_i \left[\frac{x_i^2}{\gamma_i} + \log(\lambda + \|\bar{\mathbf{k}}\|_2^2\gamma_i) + f(\gamma_i)\right], \tag{28}$$

which is obtained starting with (12) and then simply removing the minimization over $\boldsymbol{\gamma}$ from the definition of g_{VB} in (13), plugging in the value of ρ, and simplifying. The basic strategy here will be to use a majorization-minimization approach (Hunter and Lange, 2004) akin to the concave-convex procedure (Yuille and Rangarajan, 2001) to derive coordinate-wise updates that are guaranteed to reduce or leave unchanged $\mathcal{L}(\mathbf{x}, \mathbf{k}, \boldsymbol{\gamma})$, and then show that these are in fact the same updates as Algorithm 1. In doing so we show that (12) is an equally valid explanatory cost function with which to interpret VB.

As an initial proposition, we may attempt to directly minimize $\mathcal{L}(\mathbf{x}, \mathbf{k}, \boldsymbol{\gamma})$ over $\mathbf{x}$, $\mathbf{k}$, and $\boldsymbol{\gamma}$ independently, in each case while holding the other two variables fixed. Beginning with $\mathbf{x}$, we collect relevant terms and find that we must solve

$$\min_{\mathbf{x}} \frac{1}{\lambda}\|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2 + \sum_i \frac{x_i^2}{\gamma_i}, \tag{29}$$

which has a convenient closed-form solution $\mathbf{x}^{\text{opt}}$ given by

$$\mathbf{x}^{\text{opt}} = \left[\frac{1}{\lambda}\mathbf{H}^T\mathbf{H} + \boldsymbol{\Gamma}^{-1}\right]^{-1} \frac{1}{\lambda}\mathbf{H}^T\mathbf{y}, \tag{30}$$

where $\boldsymbol{\Gamma} \triangleq \text{diag}[\boldsymbol{\gamma}]$ and $\mathbf{H}$ is the convolution matrix of the blur kernel defined in Section 3.1.

Next we consider updating $\boldsymbol{\gamma}$, where the associated cost function conveniently decouples so we may solve for each γ_i independently. For this purpose, we use the fact that

$$\lambda + \|\bar{\mathbf{k}}\|_2^2\gamma_i = \lambda\gamma_i\left(\frac{1}{\gamma_i} + \frac{\|\bar{\mathbf{k}}\|_2^2}{\lambda}\right) \tag{31}$$

to obtain the following minimization problem for each γ_i:

$$\min_{\gamma_i \geq 0} \frac{x_i^2}{\gamma_i} + \log\gamma_i + \log\left[\frac{\|\bar{\mathbf{k}}\|_2^2}{\lambda} + \gamma_i^{-1}\right] + f(\gamma_i), \tag{32}$$

where γ_i-independent terms are omitted. Because no closed-form solution is available, we instead use basic principles from convex analysis to form a strict upper bound that will facilitate subsequent optimization. In particular, we use

$$\frac{z_i}{\gamma_i} - \phi^*(z_i) \geq \log\left[\frac{\|\bar{\mathbf{k}}\|_2^2}{\lambda} + \gamma_i^{-1}\right], \tag{33}$$

which holds for all $z_i \geq 0$, where ϕ^* is the concave conjugate (Boyd and Vandenberghe, 2004) of the concave function $\phi(\alpha) \triangleq \log\left[\frac{\|\bar{\mathbf{k}}\|_2^2}{\lambda} + \alpha\right]$. It can be shown that equality in (33) is obtained using

$$z_i^{\text{opt}} = \frac{\partial\phi}{\partial\alpha}\bigg|_{\alpha=\gamma_i^{-1}} = \frac{1}{\frac{\sum_j k_j^2 \bar{I}_{ji}}{\lambda} + \gamma_i^{-1}}, \forall i, \tag{34}$$

where we have used the fact that $\|\bar{\mathbf{k}}\|_2^2 \triangleq \sum_j k_j^2 \bar{I}_{ji}$ is the squared norm of $\mathbf{k}$ reincorporating the i-dependent image boundary conditions (see Section 3.1), which will become somewhat relevant for a more comprehensive version of the proof. Plugging (33) into (32) we obtain the revised problem

$$\min_{\gamma_i \geq 0} \frac{x_i^2 + z_i}{\gamma_i} + \log\gamma_i + f(\gamma_i). \tag{35}$$

This sub-problem can be handled in multiple ways. First, if the underlying g_x associated with f (obtained from (11)) is differentiable, then (35) has a convenient closed-form solution obtained as follows. After a $\exp[-1/2(\cdot)]$ transformation (35) assumes the same variational

form as the sparse prior given by (11) evaluated at the point $\sqrt{x_i^2 + z_i}$, ignoring irrelevant constants. Consequently, based on Palmer et al. (2006) we know that the optimizing γ_i is given by

$$\gamma_i^{\text{opt}} = \left.\frac{2\sigma}{g_x{}'(\sigma)}\right|_{\sigma=\sqrt{x_i^2+z_i}}, \forall i. \tag{36}$$

This covers the vast majority of practical sparse priors (and all of those amenable to Algorithm 1). Secondly, if for some reason g_x is not differentiable at some point(s), then (35) may still be solved numerically as a 1D optimization problem, or perhaps analytically leveraging the structure of f. For example, if f is a non-decreasing function (as motivated in Section 3.3), then g_x will not be differentiable at zero. However, since $\gamma_i^{\text{opt}} = 0$ whenever $x_i^2 + z_i = 0$, so this does not pose a problem.

We now examine optimization over $\mathbf{k}$. Isolating terms, this requires that we solve

$$\min_{\mathbf{k}\geq 0} \frac{1}{\lambda}\|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2 + \sum_i \log\left[\frac{\|\bar{\mathbf{k}}\|_2^2}{\lambda} + \gamma_i^{-1}\right]. \tag{37}$$

There is no closed-form solution; however, as before we may use strict upper bounds derived from convex analysis for optimization purposes. Accounting again for the fact that $\|\bar{\mathbf{k}}\|_2^2 \triangleq \sum_j k_j^2 \bar{I}_{ji}$ actually depends on i, we choose

$$\Big(\sum_j k_j^2 \bar{I}_{ji}\Big) v_i - \varphi_i^*(v_i) \geq \log\Big[\frac{1}{\lambda}\Big(\sum_j k_j^2 \bar{I}_{ji}\Big) + \gamma_i^{-1}\Big], \tag{38}$$

which holds for all $v_i \geq 0$, where φ^* is the concave conjugate of the concave function $\varphi_i(\alpha) \triangleq \log\left[\frac{\alpha}{\lambda} + \gamma_i^{-1}\right]$. Similar to the $\boldsymbol{\gamma}$ updates from above, it can be shown that equality in (38) is obtained with the minimizing v_i given by

$$v_i^{\text{opt}} = \left.\frac{\partial \varphi_i}{\partial \alpha}\right|_{\alpha=\sum_j k_j^2 \bar{I}_{ji}} = \frac{z_i}{\lambda}, \forall i. \tag{39}$$

Plugging (38) and (39) into (37) leads to the quadratic optimization problem

$$\mathbf{k}^{\text{opt}} = \arg\min_{\mathbf{k}\geq 0} \frac{1}{\lambda}\|\mathbf{y}-\mathbf{W}\mathbf{k}\|_2^2 + \sum_i \frac{z_i}{\lambda}\left(\sum_j k_j^2 \bar{I}_{ji}\right) = \arg\min_{\mathbf{k}\geq 0} \|\mathbf{y}-\mathbf{W}\mathbf{k}\|_2^2 + \sum_j k_j^2 \left(\sum_i z_i \bar{I}_{ji}\right), \tag{40}$$

where $\mathbf{W}$ is the convolution matrix constructed from the image $\mathbf{x}$ (see Section 3.1). As a simple convex program, there exist many high-performance algorithms for solving (40).

To review, we would originally like to minimize $\mathcal{L}(\mathbf{x}, \mathbf{k}, \boldsymbol{\gamma})$ over $\mathbf{x}$, $\mathbf{k}$, and the latent variables $\boldsymbol{\gamma}$. To simplify the optimization we introduce additional latent variables $\mathbf{z} \triangleq [z_1, \ldots, z_m]^T$ and $\mathbf{v} \triangleq [v_1, \ldots, v_m]^T$, such that, after combining terms from above we are now equivalently minimizing

$$\begin{aligned}\mathcal{L}(\mathbf{x}, \mathbf{k}, \boldsymbol{\gamma}, \mathbf{z}, \mathbf{v}) \triangleq{}& \frac{1}{\lambda}\|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2 \\ &+ \sum_i \left[\frac{x_i^2 + z_i}{\gamma_i} + \log\gamma_i + f(\gamma_i) - \phi^*(z_i)\right] + \sum_i \left[\sum_j \left(k_j^2 \bar{I}_{ji}\right) v_i - \varphi_i^*(v_i)\right]\end{aligned} \tag{41}$$

over $\mathbf{x}$, $\mathbf{k}$, and the latent variables $\boldsymbol{\gamma}$, $\mathbf{z}$, and $\mathbf{v}$. The associated coordinate descent updates rules, meaning the cyclic iteration of (30), (34), (36), (39), and (40), are guaranteed to reduce or leave unchanged $\mathcal{L}(\mathbf{x}, \mathbf{k}, \boldsymbol{\gamma})$ by standard properties of majorization-minimization algorithms. And importantly, at least for our purposes, these updates are in one-to-one correspondence with those from Algorithm 1, albeit with some inconsequential differences in notation and statistical interpretation. Specifically, the $\boldsymbol{\gamma}$ update from (36) is equivalent to the $\boldsymbol{\omega}$ update in Algorithm 1, the $\mathbf{x}$ update from (30) is equivalent to the $\boldsymbol{\mu}$ update, the $\mathbf{z}$ update becomes equivalent to computing the diagonal of $\mathbf{C}$, and finally the $\mathbf{k}$ update from (40) is the same as that in Algorithm 1 but with the requisite boundary conditions explicitly incorporated via $\bar{\mathbf{I}}$.

Note that the $\boldsymbol{\omega}$ update from Algorithm 1 appears somewhat different from that originally presented in Levin et al. (2011a), which only considers the special case where the assumed image prior is a finite Gaussian scale mixture given by

$$p(x_i) = \sum_j \frac{\pi_j}{\sqrt{2\pi\bar{\gamma}_j}} \exp\left[-\frac{1}{2}\frac{x_i^2}{\bar{\gamma}_j}\right], \tag{42}$$

where $\pi_j \geq 0$ and $\sum_j \pi_j = 1$. However, using Palmer et al. (2006) it is easily shown that

$$\frac{2\sigma}{g_x{}'(\sigma)} = \left(\mathrm{E}_{p(\gamma|x_i=\sigma)}[\gamma^{-1}]\right)^{-1} = \frac{\sum_j \frac{\pi_j}{\sqrt{2\pi\bar{\gamma}_j}} \exp\left[-\frac{1}{2}\frac{\sigma^2}{\bar{\gamma}_j}\right]}{\sum_j \frac{\pi_j}{\sqrt{2\pi\bar{\gamma}_j}} \exp\left[-\frac{1}{2}\frac{\sigma^2}{\bar{\gamma}_j}\right]\frac{1}{\bar{\gamma}_j}} \tag{43}$$

such that formal equivalence with Levin et al. (2011a) is maintained.

In closing, we emphasize that the upper bounds utilized here were specifically chosen so as to establish a connection with Algorithm 1. However, once we have motivated that $\mathcal{L}(\mathbf{x}, \mathbf{k}, \boldsymbol{\gamma})$ is an equally valid cost function, other bounds can be used to potentially improve the convergence rate or other properties of the algorithm. This is a direction of future research. ■

A.2 Proof of Corollary 1

Here we omit the pixel-wise subscript i for simplicity. Likewise for later proofs where appropriate. From the definition of g_{VB} we know that $g_{\text{VB}}(x, 0) = \min_{\gamma \geq 0} \frac{x^2}{\gamma} + \log(\gamma) + f(\gamma)$. After a $-2\log$ transformation of (11), and ignoring constant terms, we have $g_x(x) = -2\log p(x) = \min_{\gamma \geq 0} \frac{x^2}{\gamma} + \log(\gamma) + f(\gamma)$, and so it follows that $g_{\text{VB}}(x, 0) = g_x(x)$. ■

A.3 Proof of Theorem 2

We first assume that f is a concave, non-decreasing function and express $g_{\text{VB}}(x, \rho)$ as

$$g_{\text{VB}}(x, \rho) \triangleq \min_{\gamma \geq 0} \frac{x^2}{\gamma} + \psi(\gamma), \tag{44}$$

where $\psi(\gamma) \triangleq \log(\rho+\gamma) + f(\gamma)$ is also a concave, non-decreasing function of γ (because $\log(\rho+\gamma)$ is). Thus we can always express $\psi(\gamma)$ as

$$\psi(\gamma) = \min_{z \geq 0} z\gamma - \psi^*(z), \tag{45}$$

where $\psi^*(z)$ is the concave conjugate (Boyd and Vandenberghe, 2004) of $\psi(\gamma)$. Therefore, it follows that

$$g_{\text{VB}}(x,\rho) = \min_{\gamma,z \geq 0} \frac{x^2}{\gamma} + z\gamma - \psi^*(z). \tag{46}$$

Optimizing over γ for fixed x and z, the optimal solution is

$$\gamma^{\text{opt}} = z^{-1/2}|x|. \tag{47}$$

Plugging this result into (46) gives

$$g_{\text{VB}}(x,\rho) = \min_{z \geq 0} \frac{x^2}{z^{-1/2}|x|} + zz^{-1/2}|x| - \psi^*(z) = \min_{z \geq 0} 2z^{1/2}|x| - \psi^*(z). \tag{48}$$

This implies that $g_{\text{VB}}(x,\rho)$ can be expressed as a minimum over upper-bounding hyperplanes in $|x|$, with different z implying different slopes. Any function expressible in this form is necessarily concave, and also non-decreasing since $z \geq 0$ (Boyd and Vandenberghe, 2004).

Now in the other direction, assume that $g_{\text{VB}}(x,\rho)$ is a concave, non-decreasing function of $|x|$. It then follows that

$$g_{\text{VB}}(x,\rho) = \min_{z \geq 0} 2z|x| + h(z) \tag{49}$$

for some function h. Using the fact that

$$2|x| = \min_{\alpha \geq 0} \frac{x^2}{\alpha} + \alpha \tag{50}$$

and defining $\gamma \triangleq \alpha z^{-1}$, we can re-express $g_{\text{VB}}(x,\rho)$ as

$$\begin{aligned} g_{\text{VB}}(x,\rho) &= \min_{\alpha,z \geq 0} z\left[\frac{x^2}{\alpha} + \alpha\right] + h(z) = \min_{\alpha,z \geq 0} \frac{x^2}{\alpha z^{-1}} + z\alpha + h(z) \\ &= \min_{\gamma,z \geq 0} \frac{x^2}{\gamma} + z^2\gamma + h(z) = \min_{\gamma \geq 0} \frac{x^2}{\gamma} + \varphi(\gamma), \end{aligned} \tag{51}$$

where $\varphi(\gamma) \triangleq \min_{z \geq 0} z^2\gamma + h(z)$ is necessarily a concave, non-decreasing function of γ by construction and arguments made previously. This implies that $\psi(\gamma)$ from (44) must be a concave, non-decreasing function of γ for all ρ. Of course as $\rho \to \infty$, $\log(z+\rho)$ becomes arbitrarily flat, with derivative approaching zero for all γ. Consequently, the only way to ensure that $\psi(\gamma)$ is concave and non-decreasing for any ρ is to require that f is a concave, non-decreasing function.

Finally, any locally minimizing solution $\mathbf{x}^{\text{opt}}$ to (12) must necessarily be a local minimum to

$$\min_{\mathbf{x}} \frac{1}{\lambda}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \sum_i g_{\text{VB}}(x_i,\rho). \tag{52}$$

If f is concave and non-decreasing, then so is $g_{\text{VB}}(x_i, \rho)$ based on the arguments presented above, and so (52) is a canonical sparse estimation problem with a separable concave in $|\mathbf{x}|$ regularizer. Based on (Rao et al., 2003, Theorem 1), we may then conclude that $m - n$ elements of $\mathbf{x}^{\text{opt}}$ will be zero at any local minimizer. ■

A.4 Proof of Corollaries 2 and 3

For Corollary 2, the proof in both directions follows from similar arguments to those used for proving Theorem 2. For Corollary 3, the proof follows from several modifications of the proof of Theorem 2 from Zhang et al. (2013). We omit details for the sake of brevity. ■

A.5 Proof of Theorem 3

For $f(\gamma) = b$, we have

$$g_{\text{VB}}(x, \rho) \equiv \min_{\gamma \geq 0} \underbrace{\frac{x^2}{\gamma} + \log(\rho + \gamma)}_{\varphi} \tag{53}$$

since constant terms are irrelevant. We first calculate the optimal γ by differentiating φ and equating terms to zero. Since

$$\frac{\partial \varphi}{\partial \gamma} = -\frac{x^2}{\gamma^2} + \frac{1}{\rho + \gamma}, \tag{54}$$

it follows after some algebra that

$$\gamma^{\text{opt}} = \frac{x^2 + |x|\sqrt{x^2 + 4\rho}}{2}. \tag{55}$$

Based on the unimodality of φ it follows that γ^{opt} represents the unique minimizer. Substituting (55) into (53) and omitting irrelevant constant factors, we have

$$g_{\text{VB}}(x, \rho) \equiv \frac{2|x|}{|x| + \sqrt{x^2 + 4\rho}} + \log\left(2\rho + x^2 + |x|\sqrt{x^2 + 4\rho}\right).$$

■

A.6 Proof of Corollary 4

Assuming $f(\gamma) = b$ and $\rho_1 < \rho_2$, we want to show that $g_{\text{VB}}^{\rho_1} \prec g_{\text{VB}}^{\rho_2}$. For this purpose it is sufficient to show that $\frac{\partial^2 g_{\text{VB}}^{\rho}(x)}{\partial x^2} / \frac{\partial g_{\text{VB}}^{\rho}(x)}{\partial x}$ is an increasing function of ρ, which represents an equivalent condition for relatively concavity to one given by Definition 1, assuming the requisite derivatives exist (Palmer, 2003).

Defining $\eta \triangleq \gamma^{-1}$, we have

$$g_{\text{VB}}^{\rho}(x) = \min_{\eta \geq 0} \eta x^2 + \log(\rho + \eta^{-1}) \tag{56}$$

where the optimal η^{opt} is given by the gradient of $g^{\rho}_{\text{VB}}(x)$ with respect to x^2, which follows from basic concave duality theory. Let $h^{\rho}(z) \triangleq g^{\rho}_{\text{VB}}(\sqrt{z})$. Then $\eta^{\text{opt}} = \frac{\partial h^{\rho}(z)}{\partial z}$. With $z \triangleq x^2$, we can readily compute the expression for $\frac{\partial g^{\rho}_{\text{VB}}}{\partial x}(x)$ via

$$\frac{\partial g^{\rho}_{\text{VB}}(x)}{\partial x} = \frac{\partial h^{\rho}(z)}{\partial z}\frac{dz}{dx} = 2x\frac{\partial h^{\rho}(z)}{\partial z} = \frac{x}{\rho}\left(\sqrt{1+\frac{4\rho}{x^2}}-1\right). \tag{57}$$

Using (57) it is also straightforward to derive $\frac{\partial^2 g^{\rho}_{\text{VB}}(x)}{\partial x^2}$ as

$$\frac{\partial^2 g^{\rho}_{\text{VB}}(x)}{\partial x^2} = 2\frac{\partial h^{\rho}(z)}{\partial z} - \frac{4}{x^2\sqrt{1+\frac{4\rho}{x^2}}}. \tag{58}$$

We must then show that

$$\frac{\partial^2 g^{\rho}_{\text{VB}}(x)/\partial x^2}{\partial g^{\rho}_{\text{VB}}(x)/\partial x} = \frac{1}{x} - \frac{\frac{4}{x^2\sqrt{1+\frac{4\rho}{x^2}}}}{\frac{x}{\rho}\left(\sqrt{1+\frac{4\rho}{x^2}}-1\right)} \tag{59}$$

is an increasing function of ρ. By neglecting irrelevant additive and multiplicative factors (and recall that $x \geq 0$ from the definition of g^{ρ}_{VB}), this is equivalent to showing that

$$\xi(\rho) = \frac{1}{\rho}\left(\sqrt{1+\frac{4\rho}{x^2}}-1\right) \tag{60}$$

is a decreasing function of ρ. It is easy to check that

$$\xi'(\rho) = \frac{\sqrt{1+\frac{4\rho}{x^2}}-1-\frac{2\rho}{x^2}}{\sqrt{1+\frac{4\rho}{x^2}}} < 0. \tag{61}$$

Therefore, $\xi(\rho)$ is a decreasing function of ρ, implying that $\frac{\partial^2 g^{\rho}_{\text{VB}}(x)}{\partial x^2}/\frac{\partial g^{\rho}_{\text{VB}}(x)}{\partial x}$ is an increasing function of ρ, completing the proof. ■

A.7 Proof of Theorem 4

For simplicity assume that f is twice differentiable. From the definition of relative concavity, $\psi^{\rho_1} \prec \psi^{\rho_2}$ if and only if $\frac{\partial^2 \psi^{\rho}(\gamma)}{\partial \gamma^2}/\frac{\partial \psi^{\rho}(\gamma)}{\partial \gamma}$ is an increasing function of ρ (Palmer, 2003). It is easy to show that

$$\xi(\rho) \triangleq \frac{\partial^2 \psi^{\rho}(\gamma)}{\partial \gamma^2}/\frac{\partial \psi^{\rho}(\gamma)}{\partial \gamma} = \frac{-\frac{1}{(\gamma+\rho)^2}+f''(\gamma)}{\frac{1}{\gamma+\rho}+f'(\gamma)}. \tag{62}$$

To avoid notation clutter, we let $\omega \triangleq \gamma+\rho$, so that the objective is then to prove that

$$\xi(\rho) = \frac{-\frac{1}{\omega^2}+f''(\gamma)}{\frac{1}{\omega}+f'(\gamma)} \tag{63}$$

is an increasing function of ρ, for all $\gamma, \rho \geq 0$ if and only if $f''(\gamma) = 0$ and $f'(\gamma) \geq 0$, or equivalently that f is affine with positive slope. For this purpose it suffices to examine conditions whereby

$$\xi'(\rho) = \frac{f''(\gamma)\omega^2 + 2f'(\gamma)\omega + 1}{\left(f'(\gamma)\omega^2 + \omega\right)^2} \geq 0, \forall \rho, \gamma \geq 0. \tag{64}$$

First, assume $f''(\gamma) = 0$. We also have that $f'(\gamma) \geq 0$ by virtue of the Theorem statement. Clearly (64) will always be true and so $\xi(\rho)$ must be an increasing function of ρ. In the other direction, assume that (64) is true for all ρ and γ. Because f is a concave function, $f''(\gamma) \leq 0$. Now consider the case where $f''(\gamma) < 0$. The denominator of (64) is always non-negative and can be ignored. For the numerator, allow ρ to become arbitrarily large while keeping γ fixed. The quadratic term will then dominate such that $\xi'(\gamma) < 0$, violating our assumption that $\xi'(\rho) \geq 0$. Therefore it must be that $f''(\gamma) = 0$.

To conclude, $\psi^{\rho_1} \prec \psi^{\rho_2}$ if and only if $f''(\gamma) = 0$ and $f'(\gamma) \geq 0$, which is equivalent to the requirement that $f(\gamma) = a\gamma + b$ with $a \geq 0$. ■

A.8 Proof of Theorem 5

Consider the VB cost function (12) with g_{VB} defined via (13). Given an optimal solution pair $\{\mathbf{x}^*, \mathbf{k}^*\}$, we equivalently want to prove that $\{\alpha^{-1}\mathbf{x}^*, \alpha\mathbf{k}^*\}$ is also always an optimal solution pair if and only if $f(\gamma_i) = b$.

First we assume that f is a constant. It is easy to see that the value of the data fidelity term in (12) is unchanged since

$$\frac{1}{\lambda}\left\|\mathbf{y} - \mathbf{k}^* * \mathbf{x}^*\right\|_2^2 \equiv \frac{1}{\lambda}\left\|\mathbf{y} - \alpha\mathbf{k}^* * \frac{\mathbf{x}^*}{\alpha}\right\|_2^2. \tag{65}$$

For the penalty terms, after defining $\bar{\gamma}_i \triangleq \alpha^2\gamma_i$ for each i and $\rho^* \triangleq \lambda/\|\bar{\mathbf{k}}^*\|_2^2$, we have

$$\begin{aligned}
g_{\text{VB}}\left(\frac{x_i^*}{\alpha}, \frac{\rho^*}{\alpha^2}\right) + \log\left(\alpha^2\left\|\bar{\mathbf{k}}^*\right\|_2^2\right) &= \min_{\gamma_i \geq 0} \frac{x_i^{*2}}{\alpha^2\gamma_i} + \log\left(\frac{\rho^*}{\alpha^2} + \gamma_i\right) + \log\left(\alpha^2\left\|\bar{\mathbf{k}}^*\right\|_2^2\right) \\
&= \min_{\bar{\gamma}_i \geq 0} \frac{x_i^{*2}}{\bar{\gamma}_i} + \log\left(\frac{\rho^*}{\alpha^2} + \frac{\bar{\gamma}_i}{\alpha^2}\right) + \log\alpha^2 + \log\left\|\bar{\mathbf{k}}^*\right\|_2^2 \\
&= \min_{\bar{\gamma}_i \geq 0} \frac{x_i^{*2}}{\bar{\gamma}_i} + \log(\rho^* + \bar{\gamma}_i) + \log\|\bar{\mathbf{k}}^*\|_2^2 \\
&\equiv g_{\text{VB}}(x_i^*, \rho^*) + \log\left(\|\bar{\mathbf{k}}^*\|_2^2\right),
\end{aligned} \tag{66}$$

Therefore, the rescaled solution pair $\{\alpha^{-1}\mathbf{x}^*, \alpha\mathbf{k}^*\}$ does not change the cost function value, and must therefore also represent an optimal solution.

On the other hand, assume that $\{\alpha^{-1}\mathbf{x}^*, \alpha\mathbf{k}^*\}$ is an optimal solution for any $\alpha > 0$, from which it must follow that

$$g_{\text{VB}}\left(\frac{x_i^*}{\alpha}, \frac{\rho}{\alpha^2}\right) + \log(\alpha^2\|\bar{\mathbf{k}}^*\|_2^2) = g_{\text{VB}}(x_i^*, \rho) + \log(\|\bar{\mathbf{k}}^*\|_2^2)$$

and therefore

$$\min_{\bar{\gamma}_i \geq 0} \frac{x_i^{*2}}{\bar{\gamma}_i} + \log(\rho + \bar{\gamma}_i) + \log \|\bar{\mathbf{k}}\|_2^2 + f\left(\frac{\bar{\gamma}_i}{\alpha^2}\right) = \min_{\gamma_i \geq 0} \frac{x_i^{*2}}{\gamma_i} + \log(\rho + \gamma_i) + \log \|\bar{\mathbf{k}}\|_2^2 + f(\gamma_i). \quad (67)$$

To satisfy the above equivalence for all possible $\mathbf{x}^*$, $\mathbf{k}^*$, and λ, f must be a constant (with the exception of an irrelevant, zero-measure discontinuity at zero). The second part of the theorem is likewise straightforward to show; however, we omit additional details. ■

Appendix B. Noise Level Estimation

As introduced in Section 5, we would like to minimize the VB cost function (12) after the inclusion of an additional λ-dependent penalty. This is tantamount to solving

$$\min_{\lambda \geq 0} \frac{1}{\lambda}\left[d + \|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2\right] + n \log \lambda + \sum_i \log\left(\frac{\|\bar{\mathbf{k}}\|_2^2}{\lambda} + \gamma_i^{-1}\right),$$

where VB factors irrelevant to λ estimation have been omitted. We set $d = n \times 10^{-4}$ for all simulations which leads to good performance. While there is no closed-form, minimizing solution for λ, similar to the $\boldsymbol{\gamma}$ updates described in the proof of Theorem 1, we may utilize a convenient upper bound for optimization purposes. Here we use

$$\frac{\theta}{\lambda} - \phi^*(\theta) \geq \sum_i \log\left(\frac{\|\bar{\mathbf{k}}\|_2^2}{\lambda} + \gamma_i^{-1}\right) \quad (68)$$

where ϕ^* is the concave conjugate of $\phi(\theta) \triangleq \sum_i \log\left(\theta \|\bar{\mathbf{k}}\|_2^2 + \gamma_i^{-1}\right)$. Equality is obtained with

$$\theta^{\text{opt}} = \left.\frac{\partial \phi}{\partial \theta}\right|_{\theta = \lambda^{-1}} = \sum_i \frac{\|\bar{\mathbf{k}}\|_2^2}{\frac{\|\bar{\mathbf{k}}\|_2^2}{\lambda} + \gamma_i^{-1}}. \quad (69)$$

To optimize over λ, we may iteratively solve

$$\min_{\lambda, \theta \geq 0} \frac{1}{\lambda}\left(\|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2 + d\right) + n \log \lambda + \frac{1}{\lambda}\theta - \phi^*(\theta). \quad (70)$$

For fixed θ, the minimizing λ is easily computed as

$$\lambda^{\text{opt}} = \frac{\|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2 + \theta + d}{n}, \quad (71)$$

where λ^{opt} has a lower bound of d/n. Thus we may set d so as to reflect some expectation regarding the minimal about of noise or modeling error. In practice, these updates can be merged into Algorithm 1 without disrupting the convergence properties (see Algorithm 2).

References

S.D. Babacan, R. Molina, M.N. Do, and A.K. Katsaggelos. Bayesian blind deconvolution with general sparse image priors. In *ECCV*, 2012.

C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

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

R.W. Buccigrossi and E.P. Simoncelli. Image compression via joint statistical characterization in the wavelet domain. *IEEE Trans. Image Processing*, pages 1688–1701, 1999.

V. Cevher. Learning with compressible priors. In *NIPS*, 2009.

R. Chartrand and W. Yin. Iteratively reweighted algorithms for compressive sensing. In *ICASSP*, 2008.

S. Cho and S. Lee. Fast motion deblurring. In *ACM SIGGRAPH ASIA*, 2009.

T.S. Cho, C.L. Zitnick, N. Joshi, S.B. Kang, R. Szeliski, and W.T. Freeman. Image restoration by matching gradient distributions. *IEEE Trans. PAMI*, 34(4):683–694, 2012.

R. Fergus, B. Singh, A. Hertzmann, S.T. Roweis, and W.T. Freeman. Removing camera shake from a single photograph. In *ACM SIGGRAPH*, 2006.

D.R. Hunter and K. Lange. A tutorial on MM algorithms. *Amer. Statist*, 58(1):30–37, 2004.

A. Hyvarinen and E. Oja. Independent component analysis: Algorithms and application. *Neural Networks*, (4-5):411–430, 2000.

T. Kenig, Z. Kam, and A. Feuer. Blind image deconvolution using machine learning for three-dimensional microscopy. *IEEE Trans. PAMI*, 32(12):2191–2204, 2010.

D. Krishnan and R. Fergus. Fast image deconvolution using hyper-Laplacian priors. In *NIPS*, 2009.

D. Krishnan, T. Tay, and R. Fergus. Blind deconvolution using a normalized sparsity measure. In *CVPR*, 2011.

D. Kundur and D. Hatzinakos. Blind image deconvolution. *IEEE Signal Process. Mag.*, 13 (3):43–64, 1996.

D.D. Lee and H.S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, 2001.

A. Levin, R. Fergus, F. Durand, and W. T. Freeman. Deconvolution using natural image priors. Technical report, MIT, 2007.

A. Levin, Y. Weiss, F. Durand, and W.T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *CVPR*, 2009.

A. Levin, Y. Weiss, F. Durand, and W.T. Freeman. Efficient marginal likelihood optimization in blind deconvolution. In *CVPR*, 2011a.

A. Levin, Y. Weiss, F. Durand, and W.T. Freeman. Understanding blind deconvolution algorithms. *IEEE Trans. PAMI*, 33(12):2354–2367, 2011b.

L.B. Lucy. An iterative technique for the rectification of observed distributions. *The Astronomical Journal*, 79:745–754, 1974.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *J. Machine Learning Research*, pages 19–60, 2010.

J.W. Miskin and D.J.C. MacKay. Ensemble learning for blind image separation and deconvolution. In *Advances in Independent Component Analysis*, 2000.

J.A. Palmer. Relative convexity. Technical report, UCSD, 2003.

J.A. Palmer, D.P. Wipf, K. Kreutz-Delgado, and B.D. Rao. Variational EM algorithms for non-Gaussian latent variable models. In *NIPS*, 2006.

B.D. Rao, K. Engan, S.F. Cotter, J.A. Palmer, and K. Kreutz-Delgado. Subset selection in noise based on diversity measure minimization. *IEEE Trans. Signal Processing*, 51(3): 760–770, 2003.

H.W. Richardson. Bayesian-Based Iterative Method of Image Restoration. *J. Optical Society of America*, 62(1):55–59, 1972.

S. Roth and M.J. Black. Fields of experts. *Int. J. Computer Vision*, 82(2):205–229, 2009.

Q. Shan, J. Jia, and A. Agarwala. High-quality motion deblurring from a single image. In *ACM SIGGRAPH*, 2008.

F. Sroubek and P. Milanfar. Robust multichannel blind deconvolution via fast alternating minimization. *IEEE Trans. Image Processing*, 21(4):1687–1700, 2012.

H. Takeda and P. Milanfar. Removing motion blur with space-time processing. *IEEE Trans. Image Processing*, 20(10):2990–3000, 2011.

O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. Non-uniform deblurring for shaken images. *Int. J. Computer Vision*, 98(2):168–186, 2012.

D.P. Wipf and Y. Wu. Dual-space analysis of the sparse linear model. In *NIPS*, 2012.

D.P. Wipf, B.D. Rao, and S.S. Nagarajan. Latent variable Bayesian models for promoting sparsity. *IEEE Trans. Info Theory*, 57(9):6236–6255, 2011.

L. Xu and J. Jia. Two-phase kernel estimation for robust motion deblurring. In *ECCV*, 2010.

L. Xu, S. Zheng, and J. Jia. Unnatural L0 sparse representation for natural image deblurring. In *CVPR*, 2013.

A.L. Yuille and A. Rangarajan. The concave-convex procedure (CCCP). In *NIPS*, 2001.

H. Zhang and D.P. Wipf. Non-uniform camera shake removal using a spatially-adaptive sparse penalty. In *NIPS*, 2013.

H. Zhang, D.P. Wipf, and Y. Zhang. Multi-image blind deblurring using a coupled adaptive sparse prior. In *CVPR*, 2013.

X. Zhu and P. Milanfar. Removing atmospheric turbulence via space-invariant deconvolution. *IEEE Trans. PAMI*, 35(1):157–170, 2013.

Journal of Machine Learning Research 15 (2014) 3815-3846 Submitted 1/13; Revised 5/14; Published 11/14

New Results for Random Walk Learning

Jeffrey C. Jackson JACKSONJ@DUQ.EDU
Karl Wimmer WIMMERK@DUQ.EDU
Duquesne University
600 Forbes Avenue
Pittsburgh, PA 15282-1754

Editor: John Shawe-Taylor

Abstract

In a very strong positive result for passive learning algorithms, Bshouty et al. showed that DNF expressions are efficiently learnable in the uniform random walk model. It is natural to ask whether the more expressive class of thresholds of parities (TOP) can also be learned efficiently in this model, since both DNF and TOP are efficiently uniform-learnable from queries. However, the time bounds of the algorithms of Bshouty et al. are exponential for TOP. We present a new approach to weak parity learning that leads to quasi-efficient uniform random walk learnability of TOP. We also introduce a more general random walk model and give two positive results in this new model: DNF is efficiently learnable and juntas are efficiently agnostically learnable.

Keywords: computational learning theory, Fourier analysis of Boolean functions, random walks, DNF learning, TOP learning

1. Introduction

Positive results in learning theory have often assumed that the learner has access to a membership oracle. Such a learner is *active*, actively choosing examples for which it would like information. Here we consider *passive* models where the examples are chosen randomly. A commonly studied passive model is the model where the learner has access to independently random labeled samples. In this document, we consider a model with intermediate power. We study the *random walk* model, where the learner has access to labeled samples drawn from a random walk on the Boolean cube.

Our work is another step in the line of successful Fourier-based algorithms for learning with respect to the uniform distribution. Although many of these algorithms have been active, the Low Degree Algorithm of Linial et al. (1993), the first major Fourier-based learner, was a passive learning algorithm. This algorithm uses only random examples to estimate all of the "low degree" Fourier coefficients of a target function. By showing that AC^0 circuits (and hence DNF expressions) are well-approximated by their Fourier coefficients of degree logarithmic in the learning parameters, Linial et al. proved that the Low Degree Algorithm can be used to learn AC^0 in quasi-polynomial time. Another recent passive algorithm is in a new model motivated by the smoothed analysis of algorithms initiated by Spielman and Teng (2004). In particular, Kalai et al. (2009) show how to learn DNF—and, agnosti-

cally, decision trees—with respect to so-called smoothed product distributions, essentially random perturbations of arbitrary product distributions.

When the learner is active and has access to membership queries, the learner has substantially more power. Kushilevitz and Mansour (1993) gave an algorithm for learning decision trees in polynomial time. The algorithm, based on earlier work of Goldreich and Levin (1989), uses membership queries to find the "heavy" (large magnitude) Fourier coefficients, and uses this information to construct a hypothesis. Gopalan et al. (2009) showed that this algorithm can be made to be efficient and return a hypothesis equivalent to the decision tree. Jackson (1997) gave the first polynomial-time algorithm for learning DNF formulas with membership queries. The algorithm combines the Fourier search of Kushilevitz and Mansour (1993) with the boost-by-majority algorithm of Freund (1995). Various improvements have been made to Jackson's algorithm. Recently, Kalai et al. (2009) showed how to improve this algorithm to not use boosting, but to find all heavy coefficients of one function and run an intricate procedure to construct a good hypothesis. Also, Feldman (2012) gave a simplified algorithm for this construction. In terms of agnostic learning, Gopalan et al. (2008) gave an algorithm for agnostically learning decision trees using membership queries. The possibility of agnostically learning DNF formulas with respect to the uniform distribution is left open.

A significant step forward in learning in the random walk model (and, in our opinion, a significant step in learning in any nontrivial passive model) was given by Bshouty et al. (2005). They showed that, in the random walk model, it is possible to efficiently learn the class of DNF expressions with respect to the uniform distribution. We define the oracle for the random walk model here.

Definition 1 *The random walk oracle proceeds as follows: the first example generated by the oracle is $\langle x, f(x)\rangle$ for a uniformly random x in $\{0,1\}^n$, and the initial internal state of the oracle is set to x. Every subsequent example is generated as follows:*

1. *Select $i \in [n]$ uniformly at random and select $b \in \{0,1\}$ uniformly at random.*
2. *Letting x be the internal state, set the new internal state x' to be $x'_j = x_j$ for $j \neq i$ and $x'_i = b$.*
3. *Return $\langle i, x', f(x')\rangle$.*

This transition of internal state is known as *updating* x. A more natural definition involves flipping rather than updating bits, where "$x'_i = b$" in the second item above is replaced with "$x'_i = 1 - x_i$". As Bshouty et al. (2005) point out, the definitions are essentially interchangeable for uniform random walks, and the updating oracle turns out to be more convenient to analyze. The accuracy of hypothesis h is assessed with respect to the uniform distribution over $\{0,1\}^n$, which is the stationary distribution of this random walk. We call learning in this setting the *uniform random walk model*.

However, Bshouty et al. left the efficient learnability of polynomial-weight threshold-of-parity (TOP) functions as an open problem for the uniform random walk model; we will refer to this problem as *learning TOP*; we define the class here.

Definition 2 *A function* $f : \{0,1\}^n \to \{-1,1\}$ *can be expressed as a* TOP of weight W *if it can be represented as the sign of a weighted sum of parity functions, where the weights are integers, and the sum of the magnitudes of the weights is bounded by* W*. The* TOP weight *of* f *is the minimum weight over all TOP representations of* f*.*

We will be mostly interested in the case that the weight is bounded by a polynomial in n. This class is equivalent to polynomial-weight polynomial threshold functions of *arbitrary* degree over $\{-1,1\}^n$. Additionally, Krause and Pudlák (1998) showed that a polynomial threshold function of weight w over $\{0,1\}^n$ can be represented as a polynomial threshold function of weight n^2w^4 over $\{-1,1\}^n$, so this class includes polynomial-weight polynomial threshold functions over $\{0,1\}^n$ (and this latter class includes polynomial-size DNF formulas). We remark that the parity function on n variables has TOP weight 1, which is minimal, while any DNF representation of this function requires 2^{n-1} terms, which is maximal (Lupanov, 1961). The efficient learnability of TOP is an intriguing question both because TOP is a much more expressive class than is DNF and because the membership query algorithm for efficiently learning DNF, the Harmonic Sieve (Jackson, 1997), can also efficiently learn TOP. We add that Roch (2007) showed that TOP is learnable from a modified random walk oracle that updates the bits repeatedly in some fixed order; this type of oracle is well-suited to the same analysis as Jackson's Harmonic Sieve.

Unfortunately, as Bshouty et al. point out, their approach does not seem to be capable of producing a positive TOP learning result. Actually, they give two algorithms, one using a random walk oracle and a second using a weaker oracle that can be viewed as making statistical queries to estimate noise sensitivity (which is, roughly, the probability that corrupting an input to a function changes the output). A proof due to Roch (2007) shows that time $2^{\Omega(n)}$ is required for the latter approach, and Bshouty et al.'s time bound for the former approach also becomes exponential when applied to learning even a single parity function on $\Omega(n)$ bits.

In somewhat more detail, the Bshouty et al. weak learning algorithm is loosely based on the algorithm of Goldreich and Levin (1989)—adopted for learning purposes by Kushilevitz and Mansour (1993) and therefore often referred to in learning papers as the **KM** algorithm—for locating all of the Fourier coefficients of a function whose magnitude exceeds some threshold. Bshouty et al. replace the influence-like estimates made by **KM** with noise sensitivity estimates, and they also employ a breadth-first search rather than **KM**'s depth-first approach. Each of these changes makes the modified **KM** unsuitable for finding a weak-approximating parity on $\Omega(n)$ bits: such an algorithm is essentially searching for a large-magnitude high-order Fourier coefficient, but high-order Fourier coefficients contribute only negligibly to low-order noise sensitivity estimates, which are the only estimates made by Bshouty et al.; and employing breadth-first search for a high-degree parity using the original **KM** influence-like estimates would lead to computing an exponential in n number of estimates before locating the parity.

We are aware of only one approach to weakly learning parity that differs fundamentally from **KM**. This approach is based on a clever algorithm due to Levin (1993) that randomly partitions the set of 2^n Fourier coefficients into polynomially many bins and succeeds in finding a weak approximator if one of the bins is dominated by a single coefficient, which happens with non-negligible probability. Variants of Levin's algorithm have been proposed

and analyzed by others (Bshouty et al., 2004; Feldman, 2007). But it is not at all clear how Levin's original algorithm or any of the variants could be adapted to use a random walk oracle rather than a membership oracle.

1.1 Our Results

Thus, it seems that a fundamentally new approach to weakly learning parity functions is needed in order to efficiently learn TOP in the random walk model. Our main result is the following:

Theorem 3 *In the uniform random walk model, there is an algorithm that learns TOP in time polynomial in n but exponential in $\log(s/\epsilon)$, where s is the minimal TOP-weight of the target function and ϵ is the desired accuracy of the approximation.*

Virtually every Fourier-based learning algorithm uses the following two procedures (possibly interleaving their operation):

1. *Fourier detection:* A procedure for finding appropriate (typically heavy) Fourier coefficients of some function (not necessarily the target function).

2. *Hypothesis construction:* A procedure for constructing a hypothesis given the identified Fourier coefficients and their estimated values.

In particular, the random walk algorithm of Bshouty et al. (2005) relies on two such procedures. Specifically, it employs the hypothesis construction method of the Harmonic Sieve (Jackson, 1997) as a black box but replaces the Sieve's Fourier detection algorithm with an algorithm that implements the Bounded Sieve (Bshouty and Feldman, 2002) using a random walk oracle. Although this bounded Fourier detection approach is sufficient for learning DNF, it is inadequate for learning TOP. Specifically, the parity function on all variables (which has TOP weight 1) has no nonzero Fourier coefficients among those that the Bounded Sieve considers, and the Fourier detection phase fails.

Our algorithm also borrows its hypothesis construction step from the Harmonic Sieve, but it introduces a new Fourier detection algorithm that uses a random walk oracle to locate heavy coefficients of arbitrary degree in quasipolynomial time. Our Fourier detection algorithm borrows some underlying Fourier ideas from **KM**, but differs markedly in how it employs these ideas. A key feature is that our algorithm can be viewed as an elimination algorithm: it locates a good approximating parity function by eliminating all other possibilities. The core of our algorithm can also be viewed as an agnostic learner for parity in the random walk model; it is efficient if the optimal parity is an $O(1)$-approximator to the target. With uniform random examples alone, the best algorithms for this problem run in time $2^{O(n/\log n)}$ (Feldman et al., 2009).

Finally, we introduce a more general random walk model based on p-biased distributions. We briefly show why Fourier-based methods cannot efficiently learn TOP in this model. On a positive note, we generalize existing efficient learning results for DNF and juntas, showing that these classes are also efficiently learnable (juntas agnostically and properly, quite similarly to the analysis in Gopalan et al. (2008)) in the product random walk model. Roch (2007) has shown a similar result for learning DNF from random walks over the

domain $[b]^n$ equipped with the uniform distribution. We also mention the work of Kalai et al. (2009) regarding the smoothed analysis model introduced by Spielman and Teng (2004). In their model, the product distribution itself is a randomly chosen perturbation of a product distribution. In the smoothed analysis model, DNF is efficiently learnable, although our random walk model and the smoothed analysis model are of incomparable strength.

We will assume that the hypothesis construction procedure of Gopalan et al. (2008) is used in our product distribution algorithm, although the subsequent improved version of their algorithm due to Feldman (2012) or even the boosting-based approach used in Jackson's Harmonic Sieve could also be used. All of these methods have comparable running times in this model. For the Fourier detection procedure, we generalize the method of Bshouty et al. (2005) to the product distribution setting, a setting they did not consider. We use an analysis similar to that of Bshouty et al. to show that the heavy low-degree Fourier coefficients of real-valued functions will, with high probability, be found by this procedure.

2. Preliminaries

All of our results are for versions of the well-known PAC model. In this model, the learner's goal is to recover an unknown target function $f : \{0,1\}^n \to \{-1,1\}$, where f is a member of some known class of functions $\mathcal{F}$. The class $\mathcal{F}$ is typically equipped with a size measure, such as minimum TOP weight or minimum number of terms in DNF representation or minimum decision tree size. The learner is given some sort of oracle access to f in the form of labeled examples $\langle x, f(x)\rangle$, and the accuracy of the hypothesis is measured with respect to a distribution D (that is related to the oracle).

Definition 4 *We say that $\mathcal{F}$ is* learnable *if there is an algorithm that, for every $\epsilon, \delta > 0$, and every target function $f \in \mathcal{F}$, given some sort of oracle access to f, produces a hypothesis $h : \{0,1\}^n \to \{-1,1\}$ such that*

$$\Pr_{\boldsymbol{x} \sim D}[h(\boldsymbol{x}) \neq f(\boldsymbol{x})] \leq \epsilon.$$

with probability $1 - \delta$. The randomness is over the oracle and the algorithm. We say that such an algorithm learns *$\mathcal{F}$.*

In this paper, we consider the case where the oracle access is a random walk oracle as described in Definition 1. In this case, the accuracy of the hypothesis is measured with respect to the stationary distribution of the random walk.

Definition 5 *If $\mathcal{F}$ is learnable for all $\epsilon \geq 1/2 - \gamma$ for some $\gamma > 0$, we say that $\mathcal{F}$ is* γ-weakly learnable. *Further, if $\mathbf{Pr}_{\boldsymbol{x} \sim D}[h(\boldsymbol{x}) \neq f(\boldsymbol{x})] \leq 1/2 - \gamma$, we say that h is a* γ-weak approximation to f.

The run time dependence on δ for our algorithms, like virtually all PAC algorithms, is logarithmic. As is standard to simplify the exposition, we will ignore the analysis of δ parameter, and simply assume that all estimates are correct. The δ parameter is then subsumed in $O()$ notation and will henceforth be ignored.

The more interesting and challenging case is the case where there are no restrictions on the target function. In this model, our goal is to develop an algorithm that returns a function h that is "competitive" with the best function from some fixed class $\mathcal{F}$; this is called agnostic learning.

Definition 6 *We say that $\mathcal{F}$ is* agnostically learnable *if there is an algorithm that, for every $\epsilon > 0$ and* every *target function $f : \{0,1\}^n \to \{-1,1\}$, given some sort of oracle access to f, produces a hypothesis $h : \{0,1\}^n \to \{-1,1\}$ such that*

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim D}[h(\boldsymbol{x}) \neq f(\boldsymbol{x})] \leq \min_{g\in\mathcal{F}} \mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim D}[g(\boldsymbol{x}) \neq f(\boldsymbol{x})] + \epsilon$$

We say that such an algorithm agnostically learns *$\mathcal{F}$.*

We will make extensive use of discrete Fourier analysis. For every vector $a \in \{0,1\}^n$, we define the function $\chi_a : \{0,1\}^n \to \{-1,1\}$ such that $\chi_a(x) = (-1)^{\sum_{i=1}^n a_i x_i} = (-1)^{a\cdot x}$. On any fixed input x, $\chi_a(x)$ returns the parity (1 for even parity, -1 for odd) of the subset of components of x indexed by 1's in a. The set of functions $\{\chi_a\}_{a\in\{0,1\}^n}$ forms an orthonormal basis for real-valued functions over $\{0,1\}^n$, where the inner product of two functions g and h is taken to be $\mathbf{E}_{\boldsymbol{x}\sim\mathcal{U}}[g(\boldsymbol{x})h(\boldsymbol{x})]$. Here, $\mathcal{U}$ represents the uniform distribution on $\{0,1\}^n$. For a function $g : \{0,1\}^n \to \mathbb{R}$, we define the Fourier coefficient $\hat{g}(a) = \mathbf{E}_{\boldsymbol{x}\sim\mathcal{U}}[g(\boldsymbol{x})\chi_a(\boldsymbol{x})]$. If g is Boolean (meaning that the codomain of g is $\{-1,1\}$) then it is not hard to see that $\hat{g}(a) = 1 - 2\mathbf{Pr}_{\boldsymbol{x}}[\chi_a(\boldsymbol{x}) \neq g(\boldsymbol{x})]$, where again the probability is taken with respect to the uniform distribution over $\{0,1\}^n$. Thus, if $|\hat{g}(a)| \geq \gamma$, then either $\mathbf{Pr}_{\boldsymbol{x}}[\chi_a(\boldsymbol{x}) \neq g(\boldsymbol{x})] \leq 1/2 - \gamma/2$ or $\mathbf{Pr}_{\boldsymbol{x}}[-\chi_a(\boldsymbol{x}) \neq g(\boldsymbol{x})] \leq 1/2 - \gamma/2$, which implies that either χ_a or $-\chi_a$ is a $(\gamma/2)$-weak approximation to g.

Our primary algorithms focus on finding *heavy* Fourier coefficients—finding a such that $|\hat{g}(a)| \geq \gamma$ for some threshold value $\gamma > 0$ that will be clear from context. As has just been shown, such algorithms can be viewed as $\gamma/2$-weak learning algorithms with respect to the uniform distribution. We use $\hat{\|}g\hat{\|}_2^2$ to denote $\sum_a \hat{g}^2(a)$ and $\|g\|_\infty$ to represent $\max_x |g(x)|$. Parseval's identity tells us that $\hat{\|}g\hat{\|}_2^2 = \mathbf{E}[g^2(x)]$, where the expectation is over uniform random choice of x; this implies that if g is Boolean, $\hat{\|}g\hat{\|}_2^2 = 1$. The notation $x \oplus y$ represents the bitwise exclusive-OR of the binary vectors x and y (assumed to be of the same length). In later sections, instead of $\hat{f}(a)$, it will be more convenient to write $\hat{f}(A)$, where $A \subset \{1,\ldots,n\}$ is the set of coordinates at which a is 1.

We will call a string in $\{0,1,*\}^n$ a *restriction* (we use such strings to represent certain subsets, which can be viewed as restrictions of larger sets). For example, $0**1*$ represents the restriction and could be viewed as representing the set of all 5-bit strings where the first bit is 0 and the fourth is 1. The *bits of a restriction* are those symbols that are not $*$'s. Note that an n-bit string is considered to be a restriction. For $1 \leq i \leq n$ and $b \in \{0,1,*\}$, we use the notation $\alpha + (i,b)$ to represent the restriction α' that is identical to α except that its ith symbol α'_i is b. We say that a restriction a is *consistent* with a restriction α if and only if for all i such that $\alpha_i \neq *$ it is the case that $a_i = \alpha_i$. A Fourier coefficient $\hat{f}(c)$ is consistent with restriction a if c is consistent with a. A sum over $a \in \alpha$ represents a sum over the set of all bit-strings a consistent with α. We use $|\alpha|$ to denote the number of non-$*$ characters in α.

As mentioned in the introduction, the Fourier detection portion of our algorithm will be a variation of the Kushilevitz-Mansour **KM** algorithm (Kushilevitz and Mansour, 1993). The **KM** algorithm proceeds as follows. The goal is, for a given $\theta > 0$ and $g : \{0,1\}^n \to \mathbb{R}$, to find all $a \in \{0,1\}^n$ such that $|\hat{g}(a)| \geq \theta$. The number of such coefficients is clearly at most $\hat{\|}g\hat{\|}_2^2/\theta^2$. The **KM** algorithm builds a binary tree with variables at each non-leaf node and restrictions at each leaf. The restriction at the leaf v is consistent with the variable assignments on the path from the root to v. The key idea is to estimate $\sum_{a\in\alpha} \hat{g}(a)^2$. When $\alpha = *^n$, we have $\sum_{a\in\alpha} \hat{g}(a)^2 = \hat{\|}g\hat{\|}_2^2$. If we find $\sum_{a\in\alpha} \hat{g}(a)^2 < \theta^2$, then we know that every desired Fourier coefficient is inconsistent with α. If $\sum_{a\in\alpha} \hat{g}(a)^2 \geq \theta^2$ and $\alpha_i = *$ for some i, we can refine our partition by replacing α with $\alpha + (i,0)$ and $\alpha + (i,1)$. Continuing in this fashion, we find all the heavy Fourier coefficients. This algorithm is efficient because the number of "active" restrictions at any time is $\hat{\|}g\hat{\|}_2^2/\theta^2$, so the number of leaves is at most $2\hat{\|}g\hat{\|}_2^2/\theta^2$. This algorithm can efficiently learn polynomial size decision trees, but cannot efficiently learn polynomial size DNF formulas.

3. Finding a Heavy Parity

In this section we present and analyze our core algorithm, which given a threshold value θ and a uniform random walk oracle for a function $g : \{0,1\}^n \to \mathbb{R}$ finds the index a of a Fourier coefficient such that $|\hat{g}(a)|$ (nearly) exceeds θ, if such a coefficient exists. The algorithm's time bound is exponential in $\log(\|g\|_\infty/\theta)$ but is otherwise polynomial. In later sections we use this algorithm to obtain other random walk results, agnostically learning parity (in polynomial time if the accuracy ϵ is constant) and learning TOP (in polynomial time for TOPs of constant size and given constant ϵ and with significantly better run-time bound than the previous algorithms of Bshouty et al. in the general case).

3.1 Utility Fourier Algorithms

Our algorithm will depend on two utility algorithms that are described now.

We first require an easy lemma, which will allow us to get uniform random examples given a uniform random walk oracle:

Lemma 7 *Let x be an arbitrary initial internal state of the random walk oracle. Then with probability at least $1 - 1/t$, after $n\ln(nt)$ examples have been drawn from the oracle, every bit of x will have been updated at least once.*

Proof The probability that the ith bit is not updated after $r = n\ln(nt)$ examples is $(1-1/n)^{n\ln(nt)} \leq \exp(-\ln(nt)) = 1/(nt)$. By the union bound, the probability that there is a bit not updated after $n\ln(nt)$ examples is at most $1/t$. ■

By making t sufficiently small, we can fold the failure probability of this "reset to random" procedure into the δ parameter with only polynomial impact on the algorithm's run time. We will therefore assume in the sequel that we can, whenever needed, reset the random walk to a uniform random internal state. Among other things, this implies that

an algorithm with access to a random walk oracle can efficiently draw independent random examples.

Our first utility algorithm is $\mathbf{FC}(g, a, \tau)$ (an abbreviation of **F**ourier **C**oefficient), an algorithm that takes a uniform random walk oracle for $g : \{0,1\}^n \to \mathbb{R}$, a vector $a \in \{0,1\}^n$, and $\tau > 0$ as input, and uses a uniform random set of examples to estimate the Fourier coefficient $\hat{g}(a) = \mathbf{E}_{\boldsymbol{x}}[g(\boldsymbol{x})\chi_a(\boldsymbol{x})]$ within an additive error τ of the true value. By a standard Hoeffding argument (Hoeffding, 1963), given a set of polynomial in n, $\|g\|_\infty$, and $1/\tau$ such examples, the mean of $g \cdot \chi_a$ over the sample provides a sufficiently accurate estimate. Therefore, **FC** can be implemented (with high probability) in time polynomial in n, $\|g\|_\infty$, and $1/\tau$.

Second, we will use $\mathbf{SSF}(g, \alpha, \tau)$ (an abbreviation of **S**um of **S**quared **F**ourier coefficients) to represent an algorithm that takes a random walk oracle for $g : \{0,1\}^n \to \mathbb{R}$, a restriction α, and $\tau > 0$ as input, and returns a value σ such that $|\sum_{a \in \alpha} \hat{g}^2(a) - \sigma| \leq \tau$. If α contains exactly k bits (or equivalently, k non-$*$ entries), then it follows from the analysis of the **KM** algorithm (Kushilevitz and Mansour, 1993) that $\sum_{a \in \alpha} \hat{g}^2(a) = \mathbf{E}_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}}[g(\boldsymbol{x} + \boldsymbol{y})g(\boldsymbol{x} + \boldsymbol{z})\chi_d(\boldsymbol{y} \oplus \boldsymbol{z})]$, where the expectation is over uniform choice of $\boldsymbol{x}$ from $\{0,1\}^{n-k}$ and $\boldsymbol{y}$ and $\boldsymbol{z}$ from $\{0,1\}^k$, where we use $\boldsymbol{x} + \boldsymbol{y}$ to mean the n-bit string formed by interleaving in the obvious way bits from $\boldsymbol{x}$ in positions where there are $*$'s in α with bits from $\boldsymbol{y}$ in non-$*$ positions, and where d is the k-bit string obtained by removing all of the $*$'s from α. Thus, **SSF** could be implemented given a membership oracle for g by randomly sampling the random variable $g(\boldsymbol{x} + \boldsymbol{y})g(\boldsymbol{x} + \boldsymbol{z})\chi_d(\boldsymbol{y} \oplus \boldsymbol{z})$ and computing the sample mean. Using Hoeffding's bound, a sample of size polynomial in $1/\tau$ and $\|g\|_\infty$ suffices to ensure a sample mean within τ of the true mean, and therefore the time bound for **SSF** is polynomial in n, $1/\tau$, and $\|g\|_\infty$. However, when using a random walk rather than membership oracle, we have a somewhat weaker result:

Lemma 8 *The random walk algorithm* $\boldsymbol{SSF}(g, \alpha, \tau)$ *can be implemented using* $\mathrm{poly}(n^{|\alpha|}, \|g\|_\infty, 1/\tau)$ *time and samples.*

Proof The algorithm uses examples to estimate $g(\boldsymbol{x} + \boldsymbol{y})g(\boldsymbol{x} + \boldsymbol{z})\chi_d(\boldsymbol{y} \oplus \boldsymbol{z})$. We begin by resetting the oracle to a random internal state. We then draw from the oracle until we observe a sequence of consecutive steps that collectively update all of and only the bits corresponding to non-$*$ characters in α (some of these bits might be updated more than once). The inputs before and after this sequence of steps give us the $\boldsymbol{x} + \boldsymbol{y}$ and $\boldsymbol{x} + \boldsymbol{z}$ values we need in order to obtain a sample from the random variable. For simplicity of analysis, we can imagine waiting for a sequence of consecutive steps that update the bits corresponding to non-$*$ characters in order, each being updated exactly once (this gives an upper bound on the time required for the actual algorithm, which will allow repeats and any ordering of updates). Then the probability that we see a sequence of $|\alpha|$ updates in order on the appropriate bits is exactly $1/n^{|\alpha|}$, and thus the running time is polynomial in $1/\tau$, $n^{|\alpha|}$, and $\|g\|_\infty$.

■

3.2 Intuitive Description

Given access to **SSF** and **FC**, we next describe the intuition behind our algorithm **PT** that finds a parity function weakly approximating the target, if such a function exists. If we could call **SSF** with arbitrary restrictions α, then in order to find a weak-approximating parity function we could simply employ **SSF** as in the **KM** algorithm, which operates in levels as follows (we also assume for simplicity of exposition in this intuition section that **SSF** computes the sum of squares of coefficients exactly). At the first level, **KM** calls (a membership-oracle based version of) **SSF** with the restrictions $0*^{n-1}$ and $1*^{n-1}$. Taking θ to be the magnitude of the desired coefficient, if either of the values returned by **SSF** is less than the threshold value θ^2 then we know that all coefficients with indices consistent with the corresponding restriction have magnitude less than θ. For instance, if **SSF** on $0*^{n-1}$ returns a value below the threshold, we know that, if any Fourier coefficient $\hat{f}(a)$ has magnitude exceeding θ, it must be the case that $a_1 = 1$. In this case, we can then continue by calling **SSF** on $10*^{n-2}$ and $11*^{n-2}$. If both returned values are above threshold, we could continue with computing the sums of squares of Fourier coefficients on the four restrictions $100*^{n-3}$, $101*^{n-3}$, $110*^{n-3}$ and $111*^{n-3}$. **KM** continues in this way, computing (estimates of) sums of squares of coefficients at level k for a set of restrictions each of which contains k bits. It is not hard to see that, at any level of this recursion, at most $\hat{\|}g\hat{\|}_2^2/\theta^2$ restrictions will survive the threshold test. Finally, after running its **SSF** on the set of restrictions at the $n-1$ level and thresholding, **KM** can run **FC** on any surviving restrictions (now full n-bit vectors) to locate the desired Fourier coefficient, if it exists.

The problem with this approach in our random walk setting is that our implementation of **SSF** has a running time exponential in the level, so we cannot afford to run **SSF** at all of the levels required by **KM**. This suggests that rather than adopting the depth-oriented approach of **KM**, we might be better served by a breadth-oriented approach.

For instance, imagine that there is one especially heavy coefficient $\hat{f}(c)$ and that the sum of squares of all other coefficients is very small. Then running **SSF** on the pair of restrictions $0*^{n-1}$ and $1*^{n-1}$ would reveal the first bit of c (0 if **SSF** run on the first restriction returns a large value and 1 otherwise), running the algorithm on the pair $*0*^{n-2}$ and $*1*^{n-2}$ reveals the second bit, and so on. Each of these calls to **SSF** is, in **KM** terms, at the first level, and therefore each can be run in time polynomial in n.

Of course, in general, the Fourier spectrum will not be so accommodating. For instance, we might be able to fix the first bit of any heavy coefficient as above, but perhaps when we call **SSF** on the pair of restrictions $*0*^{n-2}$ and $*1*^{n-2}$, both calls return values that exceed the threshold. In this case, we will further explore the coefficients consistent with one of these restrictions—let us say those consistent with $*0*^{n-2}$—by making second-level calls to **SSF**, beginning with calls on the pair of restrictions $*00*^{n-3}$ and $*01*^{n-3}$. If both of these calls return values below threshold, we know that any heavy coefficient $\hat{f}(c)$ must have $c_2 = 1$, so we can proceed with a breadthwise search at the first level. On the other hand, if exactly one of these restrictions—say $*01*^{n-3}$—returns a value above threshold, then we can continue a breadthwise search at the second level, next computing the sums of squares of coefficients consistent with $*0*0*^{n-4}$ and $*0*1*^{n-4}$.

If this breadthwise search succeeds at fixing a single bit for each of the remaining bit positions, then we will have a candidate vector c on which we can run **FC**. If c is rejected,

or if at any point in the second-level search both of a pair of returned values are below threshold, then we will as before be able to fix $c_2 = 1$ and will continue a breadthwise search at the first level. Finally, if at any point in the second-level search we encounter a pair of above-threshold return values from **SSF**, we will begin a third-level breadthwise search over the coefficients consistent with one of these restrictions. And so on.

What we have not specified thus far is which restriction we choose to further explore when both restrictions in a pair produce above-threshold values. The answer is that we choose the restriction corresponding to the *smaller* of the two sums of squares. This is perhaps somewhat counter-intuitive; after all, we are seeking a heavy coefficient, so it might seem that our search efforts should be focused on those sets of coefficients that are most likely to contain such a coefficient. However, to a large extent our algorithm attempts to locate a heavy coefficient by *eliminating* certain sets of coefficients from further consideration. Viewed this way, choosing to focus on sets of coefficients with small sums of squares makes good sense, as these are the sets most likely to be eliminated by refined searches. What is more, we are guaranteed that every time we increase the level of our search, we are searching on a set of coefficients that has sum of squares at most one half of the sum at the previous search level. Thus, the sums decrease exponentially quickly with increases in level, which allows us to limit our calls on **SSF** to relatively shallow levels.

With this background, we are ready to formally present our algorithm.

3.3 PT Algorithm

PT and its recursive helper function **PTH**, which performs the bulk of the work, are presented as Algorithms 1 and 2. The α parameter of **PTH** represents the bits that have been fixed (for purposes of further exploration) by earlier search levels. The a parameter is a restriction that is consistent with α and that further incorporates information learned to this point in the breadthwise search at the current level. In particular, a is a string of bits followed by a string of $*$'s, and it essentially tells us that the index of any heavy coefficient consistent with α must also be consistent with (begin with the bits of) a.

In this paper, we use the control flow statement **throw**, which takes a value as input. As used in this algorithm, "**throw** x" has the effect of returning the value x to the user, and the *entire* algorithm is terminated, regardless of the level of recursion at which the **throw** statement occurs. (**throw** will be used in a more general way in a later section.)

We prove the correctness of our algorithm using several lemmas.

Input: $\theta, \epsilon > 0$

Output: thrown value, if any, is $a \in \{0,1\}^n$ such that $|\hat{g}(a)| \geq \theta - \epsilon/2$; normal return guarantees that there is no a such that $|\hat{g}(a)| \geq \theta + \epsilon/2$.

1: **if** $\theta \leq \epsilon/2$ **then** {any coefficient will do}
2: **throw** 0^n
3: **else**
4: **PTH**$(*^n, *^n, 1, \theta, \epsilon)$
5: **return**
6: **end if**

Algorithm 1: PT

Input: $\alpha \in \{0,1,*\}^n$; $a \in \{0,1,*\}^n$, where a is consistent with α; $1 \leq i \leq n$; $\theta > 0$; $0 < \epsilon < 2\theta$

Output: thrown value, if any, is $c \in \{0,1\}^n$ such that $|\hat{g}(c)| \geq \theta - \epsilon/2$; normal return guarantees that there is no c consistent with a such that $|\hat{g}(c)| \geq \theta + \epsilon/2$.

while $i \leq n$ **do**
  $s_0 \leftarrow \mathbf{SSF}(g, \alpha + (i,0), \epsilon^2/16)$
  $s_1 \leftarrow \mathbf{SSF}(g, \alpha + (i,1), \epsilon^2/16)$
  if $s_0 < \theta^2$ and $s_1 < \theta^2$ **then**
    return
  else if $s_0 < \theta^2$ **then**
    $a \leftarrow a + (i,1)$
  else if $s_1 < \theta^2$ **then**
    $a \leftarrow a + (i,0)$
  else
    $b \leftarrow \text{argmin}_b(s_b)$
    $\mathbf{PTH}(\alpha + (i,b), a + (i,b), i+1, \theta, \epsilon)$
    $a \leftarrow a + (i, 1-b)$
  end if
  $i \leftarrow i + 1$
end while
if $|\mathbf{FC}(g, a, \epsilon/2)| \geq \theta$ **then**
  throw a
else
  return
end if

Algorithm 2: PTH

Lemma 9 *Algorithm **PT** always either throws a vector or returns normally (it does not loop infinitely).*

Proof Clearly, **PT** eventually terminates if **PTH** does. It is also easy to see that **PTH** terminates if called with parameter value i such that $i > n$. Assume, for induction, that for some fixed $k \geq -1$, **PTH** terminates when called with i such that $n - i \leq k$, and consider a call to **PTH** with i such that $n - i = k + 1$. Then every recursive call to **PTH** will be made with an argument value i such that $n - i \leq k$ and will therefore either return normally or throw a value, terminating **PH**. It is thus clear that, even if all recursive calls to **PTH** return normally, the while loop of **PTH** will eventually terminate, as will the call to **PTH** having $n - i = k + 1$. ■

Lemma 10 *For any execution of **PT**, in every call to **PTH** the α and i argument values will be such that the characters in locations i through n of α will all be $*$.*

Proof This is simple to verify. ■

Lemma 11 *The restriction on **PTH** that the a parameter value be consistent with the α value will be satisfied throughout any execution of **PT**. Furthermore, all values that a takes on during any given execution of **PTH** will be consistent with the value of the α parameter passed to this instantiation of **PTH**.*

Proof The call to **PTH** from **PT** clearly satisfies the restriction. Let us then assume, for induction, that the restriction on a is satisfied as long as the level of recursion never exceeds some fixed integer k, and consider an instance of **PTH** executing at recursion level k. Then since the value of i in the while loop of **PTH** is never less than i's initial parameter value, by Lemma 10 any bit restrictions applied to a at lines 7, 9, and 13 will be consistent with α. It follows that if **PTH** is called at line 12, initiating execution at recursion level $k + 1$, $a + (i, b)$ will be consistent with $\alpha + (i, b)$. ■

Lemma 12 *For any execution of **PT**, in every call to **PTH** with argument values α, a, i, θ, and ϵ, the depth of recursion at which the call is made will be exactly $|\alpha|$.*

Proof The only line for a recursive call to **PTH** is in line 12, and this is also the only line where α is modified. As noted in the proof of the previous lemma, the i and α parameter values within **PTH** will always be such that position i in α will be a $*$. Therefore, whenever this line is executed, the value of the α argument in the call—that is, the value of $\alpha + (i, b)$; call it α_1—will be such that $|\alpha_1| = |\alpha| + 1$. Thus, each recursive call to **PTH** increases $|\alpha|$ by exactly 1. ■

Lemma 13 *For any given execution of **PT**, no two distinct calls to **PTH** will have the same value for the α argument.*

Proof Suppose the claim is false and let α' be a restriction that is passed to **PTH** multiple times such that $|\alpha'|$ is as small as possible. By Lemma 10 every recursive call to **PTH** adds a non-$*$ bit to the α argument, so α' cannot be $*^n$. Thus, there is a maximum value—call it j—such that position j of α' is not a $*$. Also by Lemma 10, the multiple calls to **PTH** with argument value α' must have been such that the value of their α parameter was $\alpha'' = \alpha' + (j, *)$. This is a contradiction to the minimality of α' since $|\alpha''| = |\alpha'| - 1$. ■

Lemma 14 *For every fixed $g : \{0,1\}^n \to \mathbb{R}$ and $\theta, \epsilon > 0$, if **PT** run with these parameters throws a vector c, then $|\hat{g}(c)| \geq \theta - \epsilon/2$.*

Proof There are only two places in the algorithm that a value can be thrown. The first is at line 2 in **PT**, which is called only if $\theta \leq \epsilon/2$. In this case, $\theta - \epsilon/2 \leq 0$, so any vector c will have a corresponding Fourier coefficient satisfying the requirement of the lemma. The other throw is at line 18 of **PTH**. In order for this statement to throw a vector c, it must be the case that $|\mathbf{FC}(g, c, \epsilon/2)| \geq \theta$, which implies $|\hat{g}(c)| \geq \theta - \epsilon/2$. ■

We now need to show that a normal return implies that all the Fourier coefficients have low magnitude. We show an equivalent statement:

Lemma 15 *For any valid fixed g, θ, and ϵ, if there exists $c' \in \{0,1\}^n$ such that $|\hat{g}(c')| \geq \theta + \epsilon/2$, then **PT** executed on θ and ϵ and with access to a uniform random walk oracle for g throws some vector c such that $|\hat{g}(c)| \geq \theta - \epsilon/2$.*

Proof

Let us for the moment consider a function g for which there is one c' such that $|\hat{g}(c')| \geq \theta + \epsilon/2$ and all other coefficients $c \neq c'$ are such that $|\hat{g}(c)| < \theta - \epsilon/2$. By Lemmas 9 and 14, **PT** run on an oracle for such g either throws c' or returns. By Lemma 11, if **PTH** is called with a parameter α with which c' is not consistent, **PTH** will return normally. On the other hand, if c' is consistent with the α parameter of some call to **PTH**, then each time s_0 and s_1 are computed at lines 2 and 3 with some value assigned to i, c' will be consistent with $\alpha + (i, c'_i)$. Thus, when we compute $s_{c'_i}$, we get a value that is at least $(\theta + \epsilon/2)^2 - \epsilon^2/16 \geq \theta^2$.

Therefore, a call to **PTH** for which c' is consistent with the α parameter of the call will never execute the normal return at line 5. Furthermore, for such an instantiation of **PTH**, if any and all recursive calls to **PTH** at line 12 are made with $\alpha + (i, b)$ with which c' is not consistent, then all will return normally. In this case, the instance of **PTH** under consideration will eventually throw c' at line 18. On the other hand, if this instance makes a recursive call at line 12 using a $\alpha + (i, b)$ with which c' is consistent, then it can be seen (inductively) that that call must result in c' being thrown, either directly by the recursively called **PTH** itself or indirectly as a result of further recursion.

Next, consider the case in which multiple coefficients have magnitude of at least $\theta - \epsilon/2$ and at least one coefficient has magnitude at least $\theta + \epsilon/2$. Fix the index c' of one such coefficient and consider a call to **PTH** with a parameter α with which c' is consistent. The only point at which the above argument used the assumption of a single coefficient

heavier than $\theta - \epsilon/2$ was in inferring that a recursive call to **PTH** at line 12 would return normally given that c' was not consistent with the argument value $\alpha + (i, b)$. However, if this assumption of a normal return fails, by Lemmas 9 and 14 it must fail because the recursive call causes a vector c to be thrown such that $|\hat{g}(c)| \geq \theta - \epsilon/2$.

Summarizing, we have argued that whenever **PTH** is called with a restriction α such that $(\theta + \epsilon/2)$-heavy c' is consistent with α, the algorithm will throw some $(\theta - \epsilon/2)$-heavy vector c (which is not necessarily c'). Putting this together with the fact that c'—and every other vector—is consistent with the value $*^n$ assigned to α in the initial call to **PTH** gives the lemma. ■

Theorem 16 *For any $g : \{0,1\}^n \to \mathbb{R}$, given a uniform random walk oracle for g, $\boldsymbol{PT}(g, \theta, \epsilon)$ runs in time polynomial in $n^{\log(\|g\|_\infty/\theta)}$ and $1/\epsilon$, throws a value c' such that $|\hat{g}(c')| \geq \theta - \epsilon/2$ if there exists c such that $|\hat{g}(c)| \geq \theta + \epsilon/2$, and returns normally if $|\hat{g}(c)| < \theta - \epsilon/2$ for all c.*

Proof By Lemma 15, the theorem follows by establishing the run time bound. This, in turn, depends primarily on bounding the depth of recursion of the algorithm, which we do now.

Lemma 17 *The maximum depth of the recursion when $\boldsymbol{PT}$ is called with fixed parameter value θ and access to random walk oracle for fixed real-valued function g is $O(\log(\|g\|_\infty/\theta))$.*

Proof Consider an arbitrary instance of **PTH** that is executing at line 11. For $b \in \{0,1\}$, define $t_b = \sum_{a \in \alpha + (i,b)} \hat{g}(a)^2$. In line 11, ideally we would like to assign the value for b that minimizes $t_b/(t_b + t_{1-b})$. If we could replace the $\epsilon^2/16$ in **SSF** with 0, then the recursion at line 12 would always be called with this minimizing b and we would have $t_b/(t_b + t_{1-b}) \leq 1/2$. Thus, for any call to **PTH**, it would always be the case that

$$\sum_{c \in \alpha} \hat{g}(c)^2 \leq \hat{\|}g\hat{\|}_2^2 2^{-|\alpha|},$$

where we are using the fact (Lemma 12) that the recursion depth is $|\alpha|$. If the right hand side of this inequality were less than θ^2, then no more recursive calls would be made. Therefore, we have that $|\alpha|$ is at most $\lceil \log_2(\hat{\|}g\hat{\|}_2^2/\theta^2) \rceil$. By Parseval's, $\hat{\|}g\hat{\|}_2^2 \leq L_\infty^2(g)$, which implies that the maximum depth of recursion is $O(\log(\|g\|_\infty/\theta))$ as required.

Of course, when using the actual **SSF** we can have estimation error. Since the error tolerance is $\epsilon^2/16$, if we reach line 11 then we can only assume for each $b \in \{0,1\}$ that $t_b \geq \theta^2 - \epsilon^2/16$. Recalling that $\theta \geq \epsilon/2$, we have $t_b \geq (3/4)\theta^2$. Furthermore, it could be that the value assigned to b at line 11 would actually correspond to the larger of the two sums of squares; the most we can say is that $t_b \leq t_{1-b} + \epsilon^2/8$. Combining these observations yields that for the value of b chosen at line 11

$$\frac{t_b}{t_b + t_{1-b}} \leq \frac{t_b}{2t_b - \epsilon^2/8} \leq \frac{t_b}{2t_b - \theta^2/2} \leq \frac{(3/4)\theta^2}{2(3/4)\theta^2 - \theta^2/2} = \frac{3}{4}$$

and it follows that

$$\sum_{a \in \alpha} \hat{g}(a)^2 \leq \hat{\|}g\hat{\|}_2^2 (4/3)^{-|\alpha|}.$$

The same reasoning as in the error-free case gives the lemma. ■

From Lemma 13 we know that each time **PTH** is called it is passed a distinct value for the α argument. From Lemma 17 we also know that the maximum depth of the recursion is $O(\log(\|g\|_\infty/\theta))$, so it follows that the number of recursive calls made is at most $n^{O(\log(\|g\|_\infty/\theta))}$. Each recursive call uses at most one call to **FC**, which runs in polynomial time, and two calls to **SSF**, which run in time polynomial in $n^{\log(\|g\|_\infty/\theta)}$ and $1/\epsilon$. The theorem follows. ■

4. TOP Learning

The Harmonic Sieve (Jackson, 1997) learns—from a membership oracle and with accuracy measured with respect to the uniform distribution—Threshold of Parity (TOP) functions in time polynomial in their weight w as well as in n and $1/\epsilon$ (recall from Definition 2 that the TOP weight of a function f is the minimum weight representation of that function as a threshold of an integer-weighted sum of parity functions). The algorithm's proof of correctness is based in part on the following fact (Jackson, 1997):

Fact 18 *For every f of TOP weight w and every distribution D over $\{0,1\}^n$, there exists a parity χ_a such that*

$$|E_D[f\chi_a]| \geq \frac{1}{2w+1}.$$

Defining $g \equiv 2^n fD$, it follows that for any TOP of weight w, there exists a such that $|\hat{g}(a)| \geq 1/(2w+1)$. The original Sieve uses its membership queries in two ways: 1) to obtain uniform random examples for purposes of estimating hypothesis accuracy; 2) to implement **KM** in order to locate heavy a's for D's (and hence g's) defined by a certain boosting algorithm. The original Sieve boosting algorithm (and some other boosting algorithms which could be used instead and give asymptotically better bounds) has the property that, when learning with respect to the uniform distribution, the D's it defines all have the property that $2^n \max_x D(x)$ is polynomial in $1/\epsilon$. It follows that any g defined using such a D has $\|g\|_\infty$ that is also polynomial in $1/\epsilon$.

It is a simple matter, then, to replace the membership-query **KM** algorithm in the Sieve with the random-walk **PT** algorithm. Since $1/\theta$ is $O(w)$ (we can assume w is known, since a simple binary search technique can be used otherwise), in the context of TOP learning **PT** will run in time polynomial in $n^{\log(w/\epsilon)}$. And as noted earlier, the uniform random examples required by the Sieve can be obtained using a uniform random walk oracle with $\tilde{O}(n)$ run-time cost per example. We therefore obtain the following:

Theorem 19 *TOP is learnable in the uniform random walk model in time $n^{O(\log(w/\epsilon))}$.*

When employing certain boosting algorithms, the Harmonic Sieve produces a TOP as its hypothesis. Thus, TOP is actually *properly* learnable in the uniform random walk model in the stated time.

Input: $\epsilon > 0$
Output: throws a such that $|\hat{f}(o)| - |\hat{f}(a)| \leq 2\epsilon$
1: $\theta_l \leftarrow 1$
2: **try**
3: **loop**
4: **PT**(θ_l, ϵ)
5: $\theta_l \leftarrow \theta_l/2$
6: **end loop**
7: **catch** a
8: **end try-catch**
9: **if** $\theta_l = 1$ or $\theta_l \leq \epsilon$ **then**
10: **throw** a
11: **end if**
12: $\theta_u \leftarrow 2\theta_l$
13: **while** $\theta_u - \theta_l > \epsilon$ **do**
14: **try**
15: **PT**$((\theta_l + \theta_u)/2, \epsilon)$
16: $\theta_u \leftarrow (\theta_l + \theta_u)/2$
17: **catch** a
18: $\theta_l \leftarrow (\theta_l + \theta_u)/2$
19: **end try-catch**
20: **end while**
21: **throw** a

Algorithm 3: AGPARITY

5. Agnostic Parity Learning

It is straightforward to employ **PT** to agnostically learn parity in the uniform random walk model. First, some analysis. Let $o \equiv \text{argmax}_a |\hat{f}(a)|$. That is, either χ_o or $-\chi_o$ is the optimal approximating parity; let $\pm\chi_o$ represent the optimal. We will use **PT** to give a agnostic learning that is *proper*; that is, we will actually output a parity function as our hypothesis. Thus, we want an $a \in \{0,1\}^n$ such that $\mathbf{Pr}[\pm\chi_a \neq f] \leq \text{Opt} + \epsilon = \mathbf{Pr}[\pm\chi_o \neq f] + \epsilon$, where $\pm\chi_a$ represents either χ_a or $-\chi_a$, depending on which better approximates f. Since for any a, $\mathbf{Pr}[\chi_a \neq f] = (1 - \hat{f}(a))/2$, we can achieve our goal if we find an a such that $|\hat{f}(o)| - |\hat{f}(a)| \leq 2\epsilon$.

The **AGPARITY** algorithm (Algorithm 3) achieves this goal, as proved in the following lemma. Note that this algorithm uses **try-catch** blocks. Any **throw** occurring within such a block is "caught" by the **catch** statement ending the block, and execution proceeds normally from that point with the thrown value assigned to the variable specified in the **catch** statement. On the other hand, if normal sequential execution encounters a **catch**

statement, the **catch** is ignored and execution continues immediately after the next **end try-catch** statement. **throw** statements occurring outside a **try-catch** block behave as before, returning the specified value to the user and terminating the entire procedure.

Lemma 20 *For any $\epsilon > 0$ and any $f : \{0,1\}^n \to \{-1,1\}$, Algorithm 3 throws a vector a such that $|\hat{f}(o)| - |\hat{f}(a)| \leq 2\epsilon$. Thus, **AGPARITY** agnostically learns parity.*

Proof Recall (Lemma 15) that if $\mathbf{PT}(\theta_u, \epsilon)$ returns normally then there is no a such that $|\hat{f}(a)| \geq \theta_u + \epsilon/2$. Thus, in the case of a normal return, $|\hat{f}(o)| < \theta_u + \epsilon/2$. On the other hand, if a call to $\mathbf{PT}(\theta_l, \epsilon)$ throws a vector a, then $|\hat{f}(a)| \geq \theta_l - \epsilon/2$ by Lemma 14. Therefore, if we are able to find two threshold values $0 < \theta_l < \theta_u$ such that $\theta_u - \theta_l \leq \epsilon$, $\mathbf{PT}(\theta_l, \epsilon)$ throws a vector a, and $\mathbf{PT}(\theta_u, \epsilon)$ returns normally, then we will have that $|\hat{f}(o)| - |\hat{f}(a)| \leq 2\epsilon$, as desired. Algorithm 3 consists for the most part of a search for such threshold values.

The algorithm begins (lines 2 through 8) by searching for a value θ_l at which $\mathbf{PT}(\theta_l, \epsilon)$ throws some vector, beginning with $\theta_l = 1$ and halving until a suitable threshold value is found. This search must terminate after $O(\log(1/\epsilon))$ iterations, since **PT** will certainly throw a vector once $\theta_l \leq \epsilon/2$. If the first call (with $\theta_l = 1$) throws a vector a, then **AGPARITY** can in turn also throw a without any further search, since in this case $|\hat{f}(a)| \geq 1 - \epsilon/2$ and, for any Boolean f, $|\hat{f}(o)|$ can be at most 1. Similarly, if $\theta_l \leq \epsilon$ when a vector a is first thrown, then the previous call to **PT** used a threshold value (call it θ_u) that was at most ϵ greater than θ_l (since $\theta_u = 2\theta_l$), and this call returned normally. Therefore, by the analysis of the previous paragraph, a can safely be thrown by **AGPARITY**.

On the other hand, if $\epsilon < \theta_l < 1$ when a is thrown, then we will set θ_u to $2\theta_l$ (line 12). Throughout the remainder of the algorithm, θ_l will be the largest threshold value at which **PT** has thrown a vector, and θ_u will be the smallest value at which **PT** has returned normally. **AGPARITY** uses binary search to refine these values until they are within ϵ of one another. That is, it will call **PT** on the midpoint between the threshold values, updating θ_l if **PT** throws a vector and updating θ_u otherwise. Once the threshold values are sufficiently near one another, **AGPARITY** will throw the final vector a that it has caught from **PT**. It follows immediately from the earlier analysis that this a satisfies the requirements of the lemma.

■

We next analyze the run time. Clearly, the second loop of Algorithm 3, like the first, makes $O(\log(1/\epsilon))$ calls to **PT**, since the initial difference $\theta_u - \theta_l$ is less than 1 and the difference is halved each iteration. Therefore, the total number of calls to **PT** is $O(\log(1/\epsilon))$, and these calls obviously dominate the run time. Since the threshold value in each call is at least $\epsilon/4$ and f is Boolean, the runtime of each call to **PT** is $O(n^{\log(1/\epsilon)})$ by Theorem 16. Therefore, we have shown:

Theorem 21 *The class of all parity functions can be agnostically learned in the uniform random walk model in time $O(n^{\log(1/\epsilon)})$.*

6. Product Random Walk Model

We next turn to results for a generalization of the uniform random walk model to certain non-uniform walks. In this section, we give definitions and some preliminary observations. Subsequent sections generalize existing learning results to this model. We study product distributions, whose study in the context of learning began with Furst et al. (1991).

From this point forward, we will slightly change notation: we will index Fourier coefficients by subsets of $[n]$ rather than vectors in $\{0,1\}^n$; identifying the string $a \in \{0,1\}^n$ with the set $S = \{i | a_i = 1\}$. Further, we will take the domain of our functions over the hypercube to be $\{-1,1\}^n$ instead of $\{0,1\}^n$.

6.1 Properties of Product Distributions

A product distribution over $\{-1,1\}^n$ with parameter $\mu = [-1,1]^n$ is a distribution $\mathcal{D}$ where $\mathbf{E}_{\mathcal{D}}[x_i] = \mathbf{Pr}_{\mathcal{D}}[x_i = 1] - \mathbf{Pr}_{\mathcal{D}}[x_i = -1] = \mu_i$ for all i, and the bits $\{x_i\}_{i=1}^n$ are independent. We will denote such a distribution as $\mathcal{D}_\mu$. With respect to $\mathcal{D}_\mu$, we define the inner product of two functions f and g to be $\mathbf{E}_{\boldsymbol{x} \sim \mathcal{D}_\mu}[f(\boldsymbol{x})g(\boldsymbol{x})]$.

Given a vector $\mu \in [-1,1]^n$, a string $x \in \{-1,1\}^n$, and an index $i \in [n]$, define the function

$$z_i(x) = \frac{x_i - \mu_i}{\sqrt{1-\mu_i^2}}$$

and for a set $S \subseteq [n]$, define $\phi_S = \prod_{i \in S} z_i$. The work of Bahadur (1961) shows that the 2^n functions ϕ_S form an orthonormal basis on real valued functions on $\{-1,1\}^n$ with respect to the inner product defined in this section. We define the Fourier coefficient of $f : \{-1,1\}^n \to \mathbb{R}$ as

$$\widetilde{f}(S) = \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \mathcal{D}_\mu}[f(S)\phi_S].$$

Notice that when $\mu = 0^n$, we recover the uniform distribution. Indeed, many theorems from Fourier analysis with respect to the uniform distribution are true in arbitrary product distributions, such as Parseval's identity:

$$\mathop{\mathbf{E}}_{\boldsymbol{x} \sim \mathcal{D}_\mu}[f(\boldsymbol{x})^2] = \sum_S \widetilde{f}(S)^2.$$

We will frequently need to bound the maximum absolute value of these ϕ_S functions, so we will assume that the product distribution $\mathcal{D}_\mu$ is such that μ is in $[-1+c, 1-c]^n$; we refer to such a product distribution as c-bounded. The uniform distribution is the unique 1-bounded product distribution.

With the standard Fourier coefficients, if $\hat{f}(S)$ is heavy then the corresponding parity χ_S (or its negation) is a weak approximator to f. But ϕ_S is not a Boolean function, so it cannot necessarily be directly used as a hypothesis if $\widetilde{f}(S)$ is heavy. However, Jackson (1997) shows how to efficiently produce a weak approximating Boolean function from a heavy $\widetilde{f}(S)$ as long as $|S|$ is logarithmic and we are working in a c-bounded product distribution with c bounded away from 0, which will be the case for our results.

We can still estimate Fourier coefficients using a modified version of **FC**. Specifically, when given $g : \{-1,1\}^n \to \mathbb{R}$ as input with tolerance parameter τ, we simply estimate $\mathbf{E}[g(\boldsymbol{x})\phi_S(\boldsymbol{x})]$ to within $\pm\tau$. A simple application of a Hoeffding bound tells us that we can do this from a random walk oracle, but the number of samples required to ensure confidence in our estimate is $\text{poly}(c^{-k}, 1/\tau)$ for a c-bounded product distribution. Thus, the factor c^{-k} will often show up in our time complexity bounds. We will refer to this procedure as $\mathbf{FC}_\mu$.

6.2 Product Random Walk Oracle

It is not immediately clear how one would even define a random walk oracle with respect to product distributions. Fortunately for us, a very straightforward random walk has desirable properties. We use the same updating steps as before. Regardless of μ, we pick a coordinate i uniformly at random and replace it with a bit chosen from the one-dimensional product distribution with mean μ_i. For $\mu_i = 1/2$, we replace the ith coordinate with a uniform random bit as in the previous sections.

It is straightforward to show that the stationary distribution of this random walk is the $\mathcal{D}_\mu$:

Lemma 22 *The stationary distribution of the aforementioned random walk is $\mathcal{D}_\mu$, and the mixing time of this random walk is $O(n \log n)$, independent of μ.*

Proof

It is easy to compute that with respect to $\mathcal{D}_\mu$, we have $\mathbf{Pr}_{\boldsymbol{x}\sim\mathcal{D}_\mu}[\boldsymbol{x}_i = b] = \frac{1}{2} + \frac{1}{2}b\mu_i$ for $b \in \{-1,1\}$. For a string $x \in \{-1,1\}^n$, we let $\mathcal{D}_\mu(x)$ denote the probability mass assigned to x. Further, let $x^{(i)}$ be x with the ith bit flipped. Let $\boldsymbol{\alpha}$ be the current state of the random walk, and $\boldsymbol{\beta}$ the next state. It is easy to see that

$$\frac{\mathcal{D}_\mu(x^{(i)})}{\mathcal{D}_\mu(x)} = \frac{\frac{1}{2} - \frac{1}{2}x_i\mu_i}{\frac{1}{2} + \frac{1}{2}x_i\mu_i}$$

using the fact that $\mathcal{D}_\mu$ is a product distribution. Also, we have $\mathbf{Pr}[\boldsymbol{\beta} = x^{(i)}|\boldsymbol{\alpha} = x] = \frac{1}{n}(\frac{1}{2} - \frac{1}{2}x_i\mu_i)$, since the random walk step must update bit i and update that bit to $-x_i$. We have a similar expression when $x^{(i)}$ and x are reversed, and ultimately we have

$$\begin{aligned}
\mathcal{D}_\mu(x^{(i)})\mathbf{Pr}[\boldsymbol{\beta} = x|\boldsymbol{\alpha} = x^{(i)}] &= \mathcal{D}_\mu(x^{(i)})\frac{1}{n}\left(\frac{1}{2} + \frac{1}{2}x_i\mu_i\right) \\
&= \mathcal{D}_\mu(x)\frac{1}{n}\left(\frac{1}{2} - \frac{1}{2}x_i\mu_i\right) \\
&= \mathcal{D}_\mu(x)\mathbf{Pr}[\boldsymbol{\beta} = x^{(i)}|\boldsymbol{\alpha} = x)].
\end{aligned}$$

It follows that the chain is reversible with respect to $\mathcal{D}_\mu$, and this $\mathcal{D}_\mu$ is the stationary distribution of the Markov chain.

The mixing follows since we are updating coordinates uniformly at random, so just as in the previous sections, all n coordinates are (with high probability) updated after $O(n \log n)$ steps.

■

6.3 The Noise Sensitivity Model

We will actually prove our product random walk results in a weaker learning model. The *product ρ-noise sensitivity model* is defined similarly to the ρ-noise sensitivity model introduced by Bshouty et al. (2005). Specifically, each call to the oracle for this model will return a 5-tuple $\langle \boldsymbol{x}, f(\boldsymbol{x}), \boldsymbol{y}, f(\boldsymbol{y}), U \rangle$, where $\boldsymbol{x}$ is generated at random from $\mathcal{D}_\mu$-biased distribution and $\boldsymbol{y}$ is constructed from $\boldsymbol{x}$ as follows: for each $i \in n$, independently and with probability $1-\rho$ update each $\boldsymbol{x}_i$ by choosing a new value from the distribution with parameter μ_i for it (if a bit of $\boldsymbol{x}$ is not updated, it is unchanged in $\boldsymbol{y}$). U is the set of coordinates of the bits updated. The accuracy of the hypothesis produced will be measured with respect to $\mathcal{D}_\mu$. We refer to this model as pρNS for short. The following lemma is a generalization of Proposition 10 of Bshouty et al. (2005).

Lemma 23 *For any fixed $0 < \rho < 1$, any algorithm in the pρNS model can be simulated in the pRW model at the cost of a $O(n \log n)$ factor in run time.*

Proof Our random walk can be described in the following way: given the current string $\boldsymbol{\alpha}$, the next string $\boldsymbol{\beta}$ is $\boldsymbol{\alpha}$ with probability $\mathcal{D}_\mu(\boldsymbol{\alpha}) := (1/n)\sum(\frac{1}{2} + \frac{1}{2}\boldsymbol{\alpha}_i\mu_i)$, and $\boldsymbol{\beta} = \boldsymbol{\alpha}^{(i)}$ with probability $(1/n)(\frac{1}{2} - \frac{1}{2}\boldsymbol{\alpha}_i\mu_i)$ for all i. Under this distribution, we can assign a bit that got updated according to μ_i. If $\boldsymbol{\beta} \neq \boldsymbol{\alpha}$, then the updated bit must be the unique bit where $\boldsymbol{\beta}_i \neq \boldsymbol{\alpha}_i$. Otherwise, we randomly choose a bit that was updated (but was updated to its original value). Here, we choose i with probability $(\frac{1}{2} + \frac{1}{2}\boldsymbol{\alpha}_i\mu_i)/\mathcal{D}_\mu(\boldsymbol{\alpha})$. The total probability that i is selected is $(1/n)(\frac{1}{2} - \frac{1}{2}\mu_i) + (1/n)(\frac{1}{2} + \frac{1}{2}\mu_i) = 1/n$, regardless of the value of $\boldsymbol{\alpha}$. Thus, we can treat our distribution as uniformly choosing a bit, and updating.

Fix $\rho \in (0,1)$. To simulate an oracle for $p\rho NS$ from pRW, we first draw a $\mathcal{D}_\mu$ random labeled example $\langle x, f(x) \rangle$ by updating every bit, which happens after $O(n \log n)$ samples. To get $\langle y, f(y) \rangle$, we first select a random number u from $\textbf{Binomial}(n, 1-\rho)$, and update precisely u of the bits of x. We then repeatedly draw examples from the pRW oracle until exactly u distinct bits have been updated. The resulting point is as if a random subset of u bits had been updated. We take this point to be y, its label to be $f(y)$, and U to be the set of bits updated. The resulting 5-tuple $\langle x, f(x), y, f(y), U \rangle$ is consistent with the distribution given by the $p\rho NS$ oracle. Note that we achieve a slowdown of at most $O(n \log n)$; the worst case is when $u = n$, and we need a new example from $\mathcal{D}_\mu$. ■

7. Positive and Negative Results in pRW

Having introduced the pRW model, we would like to transfer our uniform random walk result for agnostically learning parity to the pRW model. Unfortunately, this is not possible using Fourier methods, due to the "smearing" of the Fourier spectrum of parity under product distributions where the bits are not uniformly distributed. Here, we think of parity as a function from $\{-1,1\}^n$ into $\{-1,1\}$, where $\chi_{[n]}(x) = \prod x_i$. We also define $\chi_i(x) = x_i$ for any $1 \leq i \leq n$. We will restrict ourselves to distributions where $\mu_i = \mu_j$ for all $1 \leq i \leq j \leq n$, so the only parameter is μ_1.

Claim 24 *Let $\chi_{[n]}$ be the parity function on n bits. With respect to $\mathcal{D}_\mu$ where $\mu_1 = \mu_j$ for $1 \leq j \leq n$, the Fourier coefficient $\widetilde{f}(S)$ of $f = \chi_{[n]}$ is*

$$\mu_1^{n-|S|}(\sqrt{1-\mu_1^2})^{|S|}.$$

Proof Assume we are working under $\mathcal{D}_\mu$ where $\mu_1 = \mu_j$ for all $1 \leq j \leq n$; all expectations here are with respect to this distribution. Then

$$\begin{aligned}
\widetilde{f}(S) &= \mathbf{E}[\chi_{[n]}\phi_S] \\
&= \mathbf{E}[\prod_{i\in[n]} \chi_i \prod_{i\in S} \phi_i] \\
&= \mathbf{E}[\prod_{i\in S} \chi_i\phi_i \prod_{i\notin S} \chi_i] \\
&= \prod_{i\in S} \mathbf{E}[\chi_i\phi_i] \prod_{i\notin S} \mathbf{E}[\chi_i].
\end{aligned}$$

It is straightforward to check that $\mathbf{E}[\chi_i] = \mu_1$ and $\mathbf{E}[\chi_i\phi_i] = \frac{1}{\sqrt{1-\mu_1^2}}((1+\mu_1)(\frac{1}{2}-\frac{1}{2}\mu_1)+(1-\mu_1)(\frac{1}{2}+\frac{1}{2}\mu_1)) = \sqrt{1-\mu_1^2}$, proving the claim. ■

When μ_1 is bounded away from -1, 0 and 1 by a constant, the expression $\sqrt{1-\mu_1^2}$ is bounded away from 0 and 1 by a constant, so every $\widetilde{f}(S)$ is exponentially small. Thus, our agnostic parity algorithm, which relies on finding heavy Fourier coefficients, cannot succeed in the product random walk model (for most values of μ). Although there is a simple algorithm in the case of learning parities, this "smearing" will still happen for functions with all high-degree terms when written as a multilinear polynomial; equivalently, when all the Fourier coefficients with respect to the uniform distribution occur on vectors of high Hamming weight.

Despite this negative beginning, we are able to obtain two positive results for pRW. We begin by observing that many of the algorithms used in learning under the uniform distribution using Fourier analysis techniques work by estimating certain (possibly weighted) sums of squares of Fourier coefficients. Often, the algorithm efficiently estimates these sums by estimating expectations, and the correctness of these expectations depends only on the orthogonality of the χ_S functions. When only orthogonality is used, it is straightforward to extend the algorithm to product distributions. However, the structural theorems used to prove correctness may not follow, as we have just seen in the case of parity. In addition, the complexity of the algorithm may increase when extending to product distributions.

We will show two positive learning results in pRW (via results in $p\rho NS$): DNF formulas can be efficiently learned in the pRW model, and juntas can be efficiently agnostically properly learned in the pRW model. The efficiency of these algorithms degrades as c decreases. The proofs are given in the next two sections. Both of these results rely on using Fourier detection to find heavy Fourier coefficients of low degree; such an algorithm is known as the *Bounded Sieve*.

8. Bounded Sieve

We begin this section with the definition of the Bounded Sieve, first appearing in Bshouty and Feldman (2002):

Definition 25 *Let $f : \{-1,1\}^n \to \mathbb{R}$. An algorithm with some form of oracle access to f is said to perform the Bounded Sieve (with respect to $\mathcal{D}$) if, given input parameters $\theta > 0$ and $\ell \in [n]$, it outputs a list of subsets of $[n]$ such that every set $S \subseteq [n]$ satisfying $|S| \leq \ell$ and $\widetilde{f}(S)^2 \geq \theta$ appears in the list. Further, every set in the list satisfies $|S| \leq \ell$ and $\widetilde{f}(S)^2 \geq \theta/2$.*

In short, an algorithm that performs the Bounded Sieve (which we will simply refer to as the Bounded Sieve) is a Fourier detection algorithm that is only required to detect heavy low-degree Fourier coefficients. While access to the Bounded Sieve isn't enough to even learn constant-size TOP, Bshouty and Feldman (2002) showed that it is sufficient to learn polynomial-size DNF formulas with respect to the uniform distribution.

The approach taken by Bshouty and Feldman (2002) is boosting-based as is the Harmonic Sieve of Jackson (1997), so the actual implementation involves a interleaving of the Fourier detection phase (which is the weak learner) with the hypothesis construction phase (which is the boosting algorithm). Feldman (2012) gives an algorithm that runs each stage only once: it is sufficient to find all heavy low-degree Fourier coefficients of the target function; this corresponds to running the Bounded Sieve once. The contribution of Feldman (2012) is a hypothesis construction procedure that uses no extra training examples. We also note that Feldman (2012) gives a membership query algorithm for learning DNF under product distributions and claims that the algorithm also succeeds in the random walk model, although no random walk model for product distributions is clearly defined. We proceed to verify that the Bounded Sieve indeed can be performed in the product random walk model that we have defined.

Theorem 26 (essentially Bshouty and Feldman (2002), see also Feldman (2012)) *Let $\mathcal{A}$ be an algorithm performing the Bounded Sieve (with respect to a c-bounded product distribution $\mathcal{D}_\mu$) which runs in time $\mathrm{poly}(n, \|f\|_\infty, \theta, \ell)$. Then there is a $\mathrm{poly}(n, s^{\log(2/(2-c))}, \epsilon^{-\log(2/(2-c))}, c^{-\log(2/(2$ time algorithm which learns n-variable, s-term DNF formulas to error ϵ using $\mathcal{A}$ as a black box and independent uniform random examples.*

We note that Bshouty and Feldman (2002) do not mention product distributions, but combining the work of Jackson (1997) on product distributions with their work almost immediately yields the above theorem.

Our proofs use the even weaker $p\rho NS$ oracle, As in Bshouty et al. (2005), our first goal is to estimate quantities of the form

$$\mathcal{T}(I) := \sum_{S : S \supseteq I} \rho^{|S|} \widetilde{f}(S)^2.$$

Since we only care about finding heavy Fourier coefficients $\widetilde{f}(S)$ with $|S| \leq \ell$, each such Fourier coefficient contributes at least $\rho^\ell \theta$ to $\mathcal{T}(I)$ when $S \supseteq I$. For applications to learning DNF we take ℓ to be $O(\log n)$ and θ to be inverse polynomial, so $\rho^{|S|}\widetilde{f}(S)^2$ is at least inverse polynomial in n for the indices of Fourier coefficients that we are interested

in. As an aside, the algorithm is an approximate, low-degree version of the Kushilevitz-Mansour (Kushilevitz and Mansour, 1993) algorithm applied to the "noisy" version of f; specifically, the function $\sum_{S \subseteq [n]} \rho^{|S|} \widetilde{f}(S) \phi_S$.

We now work towards efficiently estimating $\mathcal{T}(I)$. Given a $p\rho NS$ oracle and a set $I \subseteq [n]$, let $\mathcal{D}_\rho^{(I)}$ be the distribution of pairs $(\boldsymbol{x}, \boldsymbol{y})$ as follows: $\boldsymbol{x}$ is a random string from $\mathcal{D}_\mu$ (we will suppress the dependence on μ for clarity), and $\boldsymbol{y}$ is formed from $\boldsymbol{x}$ by updating each bit in I with probability 1 and updating each bit not in I with probability $1 - \rho$. Using a $p\rho NS$ oracle, we can simulate this distribution. We simply keep drawing $\langle \boldsymbol{x}, f(\boldsymbol{x}), \boldsymbol{y}, f(\boldsymbol{y}), U \rangle$ until we get a 5-tuple with $I \subseteq U$. With high probability, we need at most $\mathrm{poly}((1-\rho)^{|I|})$ examples until this happens. Then $(\boldsymbol{x}, \boldsymbol{y})$ is our desired draw from $\mathcal{D}_\rho^{(I)}$.

Define $\mathcal{T}'(I) = \mathbf{E}_{(\boldsymbol{x},\boldsymbol{y}) \sim \mathcal{D}_\rho^{(I)}}[f(\boldsymbol{x}) f(\boldsymbol{y})]$. It is easy to see that with a $p\rho NS$ oracle we can estimate $\mathcal{T}'(I)$ (with a $\mathrm{poly}((1-\rho)^{|I|})$ slowdown). Now we prove the analog of Claim 12 from Bshouty et al. (2005) in the product setting. Our proof will demonstrate that we only use orthonormality of the ϕ_S functions to achieve this result.

Input: $\theta > 0, \ell > 0$
Output: The returned list contains all indices of heavy or almost-heavy Fourier coefficients S with $|S| \leq \ell$.
1: **return** **EST**$(\theta, \ell, \emptyset, \emptyset)$

Algorithm 4: BOUNDED-SIEVE

Input: $\theta > 0, \ell > 0, I \subseteq [n], \mathcal{L} \subseteq 2^{[n]}$
Output: The list $\mathcal{L}$ is augmented with indices of heavy or almost-heavy Fourier coefficients S with $S \supseteq I$ and $|S| \leq \ell$.
1: Estimate $\widetilde{f}(I)^2$ to within $\pm\theta/4$.
2: **if** the estimate is at least $3\theta/4$ **then**
3: $\quad \mathcal{L} \leftarrow \mathcal{L} \cup \{I\}$
4: **end if**
5: **if** $|I| \geq \ell$ **then**
6: $\quad$ **return** $\mathcal{L}$
7: **end if**
8: Estimate $\mathcal{T}(I) = \sum_{S : S \supseteq I} \rho^{|S|} \widetilde{f}(S)^2$ to within $\pm\rho^\ell \theta/4$.
9: **if** the estimate is at least $3\rho^\ell\theta/4$ **then**
10: $\quad$ **for all** $i \in ([n] \setminus I)$ **do**
11: $\quad\quad \mathcal{L} \leftarrow \mathcal{L} \cup$ **EST**$(\theta, \ell, I \cup \{i\}, \mathcal{L})$
12: $\quad$ **end for**
13: **end if**
14: **return** $\mathcal{L}$

Algorithm 5: EST

Claim 27 $\mathcal{T}'(I) = \sum_{S \bigcap I = \emptyset} \rho^{|S|} \widetilde{f}(S)^2$.

Proof All expectations in this proof are over $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}_\rho^{(I)}$.

$$\begin{aligned}\mathbf{E}[f(\boldsymbol{x})f(\boldsymbol{y})] &= \mathbf{E}[(\sum_S \widetilde{f}(S)\phi_S(\boldsymbol{x}))(\sum_T \widetilde{f}(T)\phi_T(\boldsymbol{y}))] \\ &= \sum_S \sum_T \widetilde{f}(S)\widetilde{f}(T)\mathbf{E}[\phi_S(\boldsymbol{x})\phi_T(\boldsymbol{y})] \\ &= \sum_S \sum_T \widetilde{f}(S)\widetilde{f}(T)\mathbf{E}[\prod_{i\in S} z_i(\boldsymbol{x}) \prod_{j\in T} z_j(\boldsymbol{y})],\end{aligned}$$

where the second equality holds because the ϕ_S functions are orthonormal. Because we are working over product distributions, $z_i(\boldsymbol{x})$ and $z_j(\boldsymbol{x})$ are independent when $i \neq j$, and $\boldsymbol{x}$ can be replaced with $\boldsymbol{y}$ in either or both cases. Notice that if $S \setminus T$ or $T \setminus S$ is nonempty, the expectation is 0. We will assume without loss of generality that $T \setminus S$ is nonempty and $j \in T \setminus S$. Then $z_j(\boldsymbol{y})$ is independent of every other term in the expectation, and $\mathbf{E}[z_j(\boldsymbol{y})]$ is 0. So the only nonzero terms in the sum occur when $S = T$, and the sum becomes

$$\sum_S \widetilde{f}(S)^2 \mathbf{E}[\prod_{i\in S} z_i(\boldsymbol{x})z_i(\boldsymbol{y})] = \sum_S \widetilde{f}(S)^2 \prod_{i\in S} \mathbf{E}[z_i(\boldsymbol{x})z_i(\boldsymbol{y})] \tag{1}$$

using independence again.

Recall that we are trying to show $\mathcal{T}'(I) = \sum_{S \bigcap I=\emptyset} \rho^{|S|} \widetilde{f}(S)^2$, and the distribution with which we are taking expectations respect to is $\mathcal{D}_\rho^{(I)}$. To evaluate the expectation, we will consider two cases. If $i \in I$, then the ith bit is updated with certainty. In this case, $z_i(\boldsymbol{x})$ and $z_i(\boldsymbol{y})$ are independent and thus $\mathbf{E}[z_i(\boldsymbol{x})z_i(\boldsymbol{y})] = \mathbf{E}[z_i(\boldsymbol{x})]\mathbf{E}[z_i(\boldsymbol{y})] = 0$. For $i \notin I$, we see that $\boldsymbol{y}_i$ is updated with probability $1 - \rho$. The probability distribution on $z_i(\boldsymbol{x})z_i(\boldsymbol{y})$ is as follows:

$$z_i(\boldsymbol{x})z_i(\boldsymbol{y}) = \begin{cases} \dfrac{1-\mu_i}{1+\mu_i} & \text{with probability } (\frac{1}{2}+\frac{1}{2}\mu_i)(\rho + (1-\rho)(\frac{1}{2}+\frac{1}{2}\mu_i)) \\ \dfrac{1+\mu_i}{1-\mu_i} & \text{with probability } (\frac{1}{2}-\frac{1}{2}\mu_i)(\rho + (1-\rho)(\frac{1}{2}-\frac{1}{2}\mu_i)) \\ -1 & \text{with probability } 2(\frac{1}{2}+\frac{1}{2}\mu_i)(\frac{1}{2}-\frac{1}{2}\mu_i)(1-\rho) \end{cases}$$

The first case corresponds to both bits being 1, the second for both bits being -1, and the third for when the bits are different. It is straightforward to verify that $\mathbf{E}[z_i(\boldsymbol{x})z_i(\boldsymbol{y})] = \rho$. So the sum in Equation 1 reduces to

$$\sum_S \widetilde{f}(S)^2 \prod_{i\in S} \mathbf{E}[z_i(\boldsymbol{x})z_i(\boldsymbol{y})] = \sum_{S:S\bigcap I=\emptyset} \rho^{|S|} \widetilde{f}(S)^2$$

as claimed. ■

Equipped with the Fourier interpretation of $\mathcal{T}'(I)$, we can now prove that $\mathcal{T}(I)$ can be efficiently estimated.

Lemma 28 *In Step 8, $\mathcal{T}(I)$ can be efficiently estimated.*

Proof Define $\mathcal{T}''(I) = \sum_{S:S\bigcap I\neq\emptyset} \rho^{|S|}\widetilde{f}(S)^2$. This quantity is easy to estimate, as $\mathcal{T}''(I) + \mathcal{T}'(I) = \sum_S \rho^{|S|}\widetilde{f}(S)^2 = \mathcal{T}'(\emptyset)$. To estimate $\mathcal{T}(I) = \sum_{S:S\supseteq I} \rho^{|S|}\widetilde{f}(S)^2$ within γ, we can estimate $\mathcal{T}''(J)$ for every $J \subseteq I$ to within $\gamma/2^{|I|}$. We apply inclusion-exclusion, resulting in

$$\mathcal{T}(I) = \sum_{J\subseteq I; J\neq\emptyset} (-1)^{|J|-1}\mathcal{T}''(J).$$

There are $2^{|I|}$ many subsets, and the coefficients of $\mathcal{T}''(J)$ above are at most 1 in absolute value, so the error is at most γ. ■

Given that we can estimate $\mathcal{T}(I)$, our algorithm will perform a breadth-first search, similar to Kushilevitz and Mansour (1993). We think of the set $2^{[n]}$ as a graph in the natural way; the nodes are identified with subsets of $[n]$, and two nodes T and U are adjacent if the symmetric difference of T and U has cardinality 1. We will refer to a node I as *active* when the recursive procedure is called with I as the third parameter. We remark that in a breadth-first search of this graph starting at the empty set, the previously undiscovered nodes are supersets of the current node.

Starting at $I = \emptyset$, at each active node, the algorithm estimates $\widetilde{f}(I)^2$ to within $\theta/4$ and $\mathcal{T}(I)$ to within $\rho^\ell\theta/4$. The second estimate uses our procedure outlined in Lemma 28 and takes time $\mathrm{poly}(n, (1-\rho)^\ell, 2^{|I|}, \rho^\ell\theta/2, c^{-\ell})$. The first estimate is performed via $\mathbf{FC}_\mu$ as described earlier; the required running time can be bounded by $\mathrm{poly}(n, \|f\|_\infty, 1/\theta, c^{-|I|})$ by applying the Hoeffding bound (Hoeffding, 1963). The $c^{-|I|}$ term comes from the fact that under a c-bounded product distribution $\mathcal{D}_\mu$, $|\phi_I(x)| \leq \left(\sqrt{\frac{2-c}{c}}\right)^{|I|} = \mathrm{poly}(c^{-|I|})$ for every x. If $\mathbf{FC}_\mu$ returns that $\widetilde{f}(I)^2$ has magnitude at least $\theta/2$ then the algorithm adds I to the list of f's heavy Fourier coefficients. Thus if $\widetilde{f}(I)^2 \geq \theta$ then I will certainly be added to the list.

The breadth-first search proceeds to the neighbors of I only if $|I| < \ell$ and the estimate of $\mathcal{T}(I)$ is at least $3\rho^\ell\theta/4$. The proof is complete given two claims: first, we claim the algorithm finds all Fourier coefficients $\widetilde{f}(S)^2 \geq \theta$ and $|S| \leq \ell$; and second, we claim the algorithm ends its search after visiting at most $\mathrm{poly}(\|f\|_\infty, 1/\theta, (1-\rho)^{-\ell})$ sets I.

For the first claim, note that if $|S| \leq \ell$ and $\widetilde{f}(S)^2 \geq \theta$, then for $I \subseteq S$, we have

$$\mathcal{T}(I) = \sum_{T:T\supseteq I} \rho^{|T|}\widetilde{f}(T)^2 \geq \rho^{|S|}\widetilde{f}(S)^2 \geq \rho^\ell\theta.$$

It follows that the breadth-first search will reach S. The algorithm will estimate $\widetilde{f}(S)^2$, and the set S is added to the list $\mathcal{L}$.

For the second claim, we give an upper bound on the number of active nodes that the algorithm considers. First, a lemma:

Lemma 29 *For any $f : \{-1,1\}^n \to \mathbb{R}$, $0 \leq j \leq n$, and $\rho \in (0,1)$, we have $\sum_{|I|=j} \mathcal{T}(I) \leq \|f\|_\infty^2 \rho^j (1-\rho)^{-(j+1)}$.*

Proof We straightforwardly calculate:

$$\begin{aligned}
\sum_{|I|=j} \mathcal{T}(I) &= \sum_{|I|=j} \sum_{S \supseteq I} \rho^{|S|} \widetilde{f}(S)^2 \\
&= \sum_{|S| \geq j} \binom{|S|}{j} \rho^{|S|} \widetilde{f}(S)^2 \\
&\leq \left(\sum_{|S| \geq j} \widetilde{f}(S)^2 \right) \sum_{t=j}^{\infty} \binom{t}{j} \rho^t \\
&= \left(\sum_{|S| \geq j} \widetilde{f}(S)^2 \right) (1/\rho)(\frac{\rho}{1-\rho})^{j+1} \\
&\leq \|f\|_2^2 (1/\rho)(\frac{\rho}{1-\rho})^{j+1} \\
&\leq \|f\|_\infty^2 \rho^j (1-\rho)^{-(j+1)},
\end{aligned}$$

where the third equality follows from generating function identities and the fact that $\rho \in (0,1)$.

■

This implies that the number of active nodes at layer j in the breadth-first search can be at most:

$$\frac{\sum_{|I|=j} \mathcal{T}(I)}{\rho^j \theta/2} = \frac{\|f\|_\infty^2 \rho^j (1-\rho)^{-(j+1)}}{\rho^j \theta/2} = 2\|f\|_\infty^2 (1/\theta)(1-\rho)^{-(j+1)}.$$

Since $j \leq \ell$, the total number of nodes the breadth-first search ever encounters is at most $(\ell+1) \cdot 2\|f\|_\infty^2 (1/\theta)(1-\rho)^{-(\ell+1)} = \text{poly}(n, \|f\|_\infty, 1/\theta, (1-\rho)^{-\ell})$, as claimed.

Now that the Bounded Sieve is established, we get the following result, which is a restatement of Theorem 26:

Theorem 30 *The class of s-term DNF formulas over n variables can be learned with error in the $p\rho NS$ and pRW models in time* $\text{poly}(n, s^{\log(2/(2-c))}, \epsilon^{-\log(2/(2-c))}, c^{-\log(2/2-c)})$.

For the Fourier detection phase, we also remark that it seems difficult to transfer the running time guarantee of the **EKM** algorithm of Kalai et al. (2009). Although their paper does not specifically address membership query algorithms, their **EKM** procedure can be used as a membership query algorithm for finding heavy Fourier coefficients in product distributions with a running time independent of c. However, in our product random walk model, methods as rejection sampling would incur a slowdown on the order of $\text{poly}(2/(2-c)^\ell)$, where ℓ is the size of the sets of Fourier coefficients we consider. In the case that c is bounded away from 0, these extra factors are virtually constant compared to the other factors in the running time.

9. Agnostically Learning Juntas

In the case of agnostically learning juntas, we can use the Bounded Sieve, but we have to do slightly more work to show that this is useful. The overall complexity with our implementation of the Bounded Sieve yields a running time of $\text{poly}(n, k^k, \epsilon^{-k}, c^{-k})$ for properly agnostically learning juntas with respect to a c-bounded product distribution $\mathcal{D}_\mu$.

Input: $\epsilon, k > 0$
Output: returned function is a k-junta with error ϵ in computing f
1: Use Algorithm 4 to find $\mathcal{L}$, all Fourier coefficients $\hat{f}(S)$ such that $|S| \leq \ell = k$ and $\hat{f}(S)^2 \geq \theta = \epsilon^2 2^{-k}$.
2: Use $\mathbf{FC}_\mu$ to estimate $\widetilde{f}(S)$ to within $\pm\theta/4$ for each $S \in \mathcal{L}$, and call the estimate $\widehat{g}(S)$.
3: Let g be the function such that $g(x) = \sum_{S \in \mathcal{S}} \widehat{g}(S)\phi_S(x)$.
4: Let $R = \{i | \sum_{S:S \ni i} |S| \widetilde{g}(S)^2 \geq \epsilon^2/k\}$, and let $g'(x) = \sum_{S \subseteq R} \widehat{g}(S)\phi_S(x)$.
5: For every $K \subseteq R$ of size k, let $h'_K = \text{sgn}(g'_K)$ and estimate $\text{err}_f(h'_K)$.
6: Return h'_K which minimizes $\text{err}_f(h'_K)$.

Algorithm 6: JUNTAS

Our proof is very similar to the proof of Gopalan et al. (2008). In fact, our algorithm is virtually the same. However, rather than working in the model of uniform distribution plus membership queries, we extend this algorithm to product distributions as well as restricting our oracle access to a $p\rho NS$ oracle.

The algorithm of Gopalan et al. (2008) makes use of its membership queries by using KM to identify all Fourier coefficients $\hat{f}(S)$ of heavy magnitude with $|S| \leq k$. Since the size of S is bounded for their purposes, it suffices to use the Bounded Sieve, just as we have already done. In showing above that DNF is efficiently learnable in the $p\rho NS$ model, we have effectively also shown that the Bounded Sieve works even in the $p\rho NS$ model. The Fourier methods used in their algorithm again only use orthogonality and are not specific to the uniform distribution. Therefore, after translating expectations to the correct product distribution, the algorithm is almost immediate.

Suppose we wish to agnostically learn a k-junta. We will start by running the Bounded Sieve in $p\rho NS$, stopping at level k and setting the threshold $\theta = \epsilon^2 2^{-k}$. In this fashion, we find all heavy Fourier coefficients S of f with $|S| \leq k$. Let $\mathcal{S}$ be the set of heavy Fourier coefficients found, and set $g(x) = \sum_{S \in \mathcal{S}} \widetilde{g}(S)\phi_S(x)$. Note that we can only estimate g, so we use $\widehat{g}(S)$ to denote our estimate of $\widetilde{g}(S)$. Following the argument of Gopalan et al. (2008), let R be the set of coordinates with high "low-degree influences"; we have $i \in R$ iff $\sum_{S \in \mathcal{S}, S \ni i} \widetilde{g}(S)^2 \leq \epsilon^2/k$. Finally, for every set $K \subseteq R$ of size k, estimate the error of $\text{sgn}(g'_K)$ on f, where $g'_K = \sum_{S \subseteq K} \widetilde{g}(S)\phi_S(x)$. The function $\text{sgn}(g'_K)$ of least error over choices of $K \subseteq R$ is our hypothesis.

We will now prove the correctness of this algorithm. We remind the reader that for a function $g : \{-1, 1\} \to \mathbb{R}$ we define $\|g\|_1$ to be $\mathbf{E}_{\boldsymbol{x}}[|g(\boldsymbol{x})|]$; in this section, the expectation is over the relevant product distribution $\mathcal{D}_\mu$. We will prove a sequence of lemmas very similar to those of Gopalan et al. (2008). First, an analog of their Lemma 13:

Lemma 31 *Given $K \subset [n]$, and $f : \{-1,1\}^n \to \{-1,1\}$, let $f_K(x) = \sum_{S \subseteq K} \widetilde{f}(S)\phi_S(x)$. The K-junta that minimizes $\mathrm{err}_f(\cdot)$ is given by $h_K(x) := \mathrm{sgn}(f_K(x))$. Also, $\mathrm{err}_f(h_K) = \frac{1}{2}(1 - \|f_K\|_1)$.*

Proof The proof indeed follows similarly to Lemma 13 of Gopalan et al. (2008). The major difference is that in the $\{\phi_S\}$ basis, x_i is not a unbiased bit. However, the ϕ_S functions are orthonormal, which is the property used to derive (in the ϕ basis) that $\mathbf{E}[\phi_S(x)|x_K = u] = \mathbf{1}[S \subseteq K]\phi_S(u)$. The rest of the argument follows nearly directly by changing $\hat{f}$ to $\widetilde{f}$. We provide the details here.

Let us fix a value $u \in \{-1,1\}^k$. As in the work of Gopalan et al. (2008), by $\boldsymbol{x}|\boldsymbol{x}_K = u$ we denote the random variable x where the indices in K are set according to u and the rest are uniformly random. This identifies a sub-cube $C_K(u)$ of $\{-1,1\}^n$. Note that any function depending only on the coordinates in K will be a constant function when restricted to $C_K(u)$. Hence, the agreement with f is maximized by the function $g_K : \{-1,1\}^k \to \{-1,1\}$, where

$$g_K(u) = \mathrm{sgn}(\mathbf{E}[f(\boldsymbol{x})|\boldsymbol{x}_K = u]) = \mathrm{sgn}(\sum_{S \subseteq [n]} \widetilde{f}(S)\mathbf{E}[\phi_S(\boldsymbol{x})|\boldsymbol{x}_K = u].$$

Using the fact that we are working in a product distribution, the expected value of $\phi_{\{i\}}(\boldsymbol{x})$ is 0 if $i \notin K$. Thus,

$$\mathbf{E}[\phi_S(\boldsymbol{x})|\boldsymbol{x}_K = u] = \begin{cases} \phi_S(u) & \text{if } S \subseteq K \\ 0 & \text{otherwise} \end{cases}$$

Hence $\mathbf{E}[f(\boldsymbol{x})|\boldsymbol{x}_K = u] = \sum_{S \subseteq K} \widetilde{f}(S)\phi_S(u)$, which implies that $g_K(u) = \mathrm{sgn}(f_K(u)) = h_K(u)$. Since $\mathbf{E}[f(\boldsymbol{x})|\boldsymbol{x}_K = u] = f_K(u)$ and $f(x) \in \{-1,1\}$, we have

$$\mathbf{Pr}[f(\boldsymbol{x}) = \mathrm{sgn}(f_K(\boldsymbol{x}))|\boldsymbol{x}_K = u] = \tfrac{1}{2} + \tfrac{1}{2}(|f_K(u)|) \qquad \text{and}$$
$$\mathbf{Pr}[f(\boldsymbol{x}) \neq \mathrm{sgn}(f_K(\boldsymbol{x}))|\boldsymbol{x}_K = u] = \tfrac{1}{2} - \tfrac{1}{2}(|f_K(u)|).$$

Averaging over all $u \in \{-1,1\}^k$ with respect to the underlying product distribution and observing that the product distribution induces the appropriate product distribution on $\boldsymbol{x}_K$, we obtain $\mathrm{err}_f(h_K) = \mathbf{Pr}[h_K(x) \neq f(x)] = \frac{1}{2} - \frac{1}{2}\mathbf{E}[|f_K(\boldsymbol{x})] = \frac{1}{2}(1 - \|f_K\|_1)$. ■

Their Lemma 14 is also easily generalized:

Lemma 32 *Let $g_K : \{-1,1\}^n \to \mathbb{R}$ be such that $\|f_K(x) - g_K(x)\|_1 < \epsilon$, and let $h'_K = \mathrm{sgn}(g_K)$. Writing $h_K = \mathrm{sgn}(f_K)$, we have $\mathrm{err}_f(h'_K) \leq \mathrm{err}_f(h_K) + 2\epsilon$.*

Proof

Fix $x_k = u$. Then from the previous lemma, we have

$$\mathrm{err}_f(h'_K|x_K = u) = \begin{cases} \mathrm{err}_f(h_K|x_K = u) & \text{if } \mathrm{sgn}(g_K(u)) = h_K(u) \\ \mathrm{err}_f(h_K|x_K = u) + 2|f_K(u)| & \text{if } \mathrm{sgn}(g_K(u)) \neq h_K(u) \end{cases}$$

We claim that in both cases,

$$\mathrm{err}_f(h'_K|x_K = u) \leq \mathrm{err}_f(h_K|x_K = u) + 2|f_K(u) - g_K(u)|.$$

The first case is easy to see; in the second case, we use the fact that $\mathrm{sgn}(g_K) \neq h_K = \mathrm{sgn}(f_K)$, so $|f_K(u)| \leq |f_K(u) - g_K(u)|$.

Averaging over all choices of u with respect to the underlying product distribution, we get

$$\mathrm{err}_f(h'_K|x_K = u) \leq \mathrm{err}_f(h_K) + 2\mathbf{E}[|f_K(\boldsymbol{x}) - g_K(\boldsymbol{x})|] \leq \mathrm{err}_f(h_K) + 2\epsilon.$$

■

And finally, we can prove correctness of the algorithm by proving an analogue of Theorem 15 in the work of Gopalan et al. (2008):

Theorem 33 *The described algorithm agnostically learns k-juntas to error* $\mathrm{Opt} + 6\epsilon$ *in the $p\rho NS$ model in time* $\mathrm{poly}(n, k^k, \epsilon^{-k}, ((2-c)/c)^k)$ *with respect to a c-bounded product distribution $\mathcal{D}_\mu$.*

Proof Let K be the set such that $h_K = \mathrm{sgn}(f_K)$ has the least error Opt. After we search for heavy Fourier coefficients using the Bounded Sieve, we are guaranteed that $|\widehat{g}(S) - \widetilde{f}(S)| \leq \theta$ for all $S \subseteq K$. Hence $\mathbf{E}[g_K(x) - f_K(x))^2] = \sum_{S \subseteq K} (\widetilde{f}(S)^2 - \widehat{g}(S))^2 \leq 2^k\theta \leq \epsilon^2$. The running time of the Bounded Sieve is certainly bounded by $\mathrm{poly}(n, k^k, \epsilon^{-k}, ((2-c)/c)^k)$.

In the next step of the algorithm, we restrict ourselves to considering variables in the set R, where $i \in R$ iff $\sum_{i \in S, |S| \leq k} \widehat{g}(S)^2 \geq \epsilon^2/k$. Note that even if all k variables from the set K are dropped, the total Fourier mass involving these variables is at most $k(\epsilon^2/k) = \epsilon^2$. Hence, $\mathbf{E}[g_K(\boldsymbol{x}) - g'_K(\boldsymbol{x}))^2] \leq \epsilon^2$.

Finally, we can bound $\mathbf{E}[|f_K(x) - g'_K(x)|]$ in the following manner:

$$\begin{aligned}
\mathbf{E}[|f_K(\boldsymbol{x}) - g'_K(\boldsymbol{x})|] &\leq \mathbf{E}[|f_K(\boldsymbol{x}) - g_K(\boldsymbol{x})|] + \mathbf{E}[|g_K(\boldsymbol{x}) - g'_K(\boldsymbol{x})|] \\
&\leq (\mathbf{E}[(f_K(\boldsymbol{x}) - g_K(\boldsymbol{x}))^2])^{1/2} + (\mathbf{E}[g_K(\boldsymbol{x}) - g'_K(\boldsymbol{x}))^2])^{1/2} \\
&\leq \epsilon + \epsilon \\
&= 2\epsilon.
\end{aligned}$$

Thus by the previous lemma, $\mathrm{err}_f(g'_K) \leq \mathrm{Opt} + 4\epsilon$. We estimate $\mathrm{err}_f(h'_K)$ for every $K \subseteq R$ to within $\pm\epsilon$. There must be at least one choice such that our error estimate is at most $\mathrm{Opt} + 5\epsilon$, it follows that the true error is at most $\mathrm{Opt} + 6\epsilon$ as required.

To bound the running time, we show that $|R| = O(k^2/\epsilon^2)$. Recall that we estimate each Fourier coefficient so that our estimate of $\widetilde{f}(S)^2$ (which we called $\widehat{g}(S)^2$) for $S \in \mathcal{S}$ is correct to within $\pm\theta/4$. The Bounded Sieve will only return Fourier coefficients such that $\widetilde{f}(S)^2 \geq \theta/2$, so it follows that $\widehat{g}(S)^2 \leq (3\theta/4)/(\theta/2)\widetilde{f}(S)^2 = (3/2)\widetilde{f}(S)^2 \leq 2\widetilde{f}(S)^2$ for each $S \in \mathcal{S}$.

Now we observe that

$$\sum_{i\in[n]}\Big(\sum_{i\in S,|S|\le k}\widehat{\widetilde{g}}(S)^2\Big)\le\sum_{i\in[n]}\Big(\sum_{i\in S,|S|\le k}2\widetilde{g}(S)^2\Big)\le\sum_{|S|\le k}2|S|\widetilde{g}(S)^2\le 2k\sum_{S}\widetilde{g}(S)^2\le 2k.$$

Thus, at most $2k/(\epsilon^2/k)=2k^2/\epsilon^2$ variables can satisfy $\sum_{i\in S,|S|\le k}\widehat{\widetilde{g}}(S)^2\ge\epsilon^2/k$. Hence $|R|\le 2k^2/\epsilon^2$. It follows that there are at most $\binom{|R|}{k}\le(2ek^2/\epsilon^2)^k$ many choices for K. Each estimation in Step 2 of the algorithm can be done in $\mathrm{poly}(n,1/\epsilon,c^{-k})$ time, so the overall running time beyond the Bounded Sieve is $\mathrm{poly}(n,\epsilon^{-k},k^k,c^{-k})$.

■

10. Further Work

Although we have made progress on learning TOP in the uniform random walk model, it would of course be preferable to have a polynomial-time algorithm. In the product model, parity of a logarithmic number of bits can be agnostically learned, and it is obvious that n-bit parity is learnable by a "statistical query-like" algorithm that simply observes which bits are relevant during a random walk. Can TOP be learned (quasi)-efficiently in this model? Can we remove the condition that the product distribution is c-bounded in the product random walk model?

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. CCF-0728939. The second author thanks Ryan O'Donnell for helpful discussions. The authors thank the anonymous referees for valuable comments and suggestions which helped improve this manuscript.

References

Raghu Raj Bahadur. A representation of the joint distribution of responses to n dichotomous items. In H. Solomon, editor, *Studies in Item Analysis and Prediction*, pages 158–168. Stanford University Press, 1961.

Nader Bshouty and Vitaly Feldman. On using extended statistical queries to avoid membership queries. *Journal of Machine Learning Research*, 2:359–395, 2002.

Nader Bshouty, Jeffrey C. Jackson, and Christino Tamon. More efficient PAC learning of DNF with membership queries under the uniform distribution. *Journal of Computer and System Sciences*, 68(1):205–234, 2004.

Nader Bshouty, Elchanan Mossel, Ryan O'Donnell, and Rocco Servedio. Learning DNF from Random Walks. *Journal of Computer and System Sciences*, 71(3):250–265, 2005.

Vitaly Feldman. Attribute-efficient and non-adaptive learning of parities and DNF expressions. *Journal of Machine Learning Research*, 8:1431–1460, 2007.

Vitaly Feldman. Learning DNF expressions from Fourier spectrum. *Journal of Machine Learning Research - Proceedings Track*, 23:17.1–17.19, 2012.

Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. On agnostic learning of parities, monomials, and halfspaces. *SIAM Journal of Computing*, 39(2):606–645, 2009.

Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

Merrick Furst, Jeffrey Jackson, and Sean Smith. Improved learning of AC^0 functions. In *Proc. 4th Annual Conference on Learning Theory (COLT)*, pages 317–325, 1991.

Oded Goldreich and Leonid Levin. A hard-core predicate for all one-way functions. In *Proc. 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 25–32, 1989.

Parikshit Gopalan, Adam Kalai, and Adam Klivans. Agnostically learning decision trees. In *Proc. 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 527–536, 2008.

Parikshit Gopalan, Ryan O'Donnell, Rocco Servedio, Amir Shpilka, and Karl Wimmer. Testing Fourier dimensionality and sparsity. In *Proc. 36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 500–512, 2009.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

Jeffrey C. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computing and Sys. Sci.*, 55(3):414–440, 1997.

Adam Tauman Kalai, Alex Samorodnitsky, and Shang-Hua Teng. Learning and smoothed analysis. In *Proc. 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 395–404. IEEE Computer Society, 2009. ISBN 978-0-7695-3850-1.

Matthias Krause and Pavel Pudlák. Computing boolean functions by polynomials and threshold circuits. *Computational Complexity*, 7(4):346–370, 1998.

Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, December 1993.

Leonid A. Levin. Randomness and nondeterminism. *Journal of Symbolic Logic*, 58(3): 1102–1103, 1993.

Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform and learnability. *Journal of the ACM*, 40(3):607–620, 1993.

Oleg Lupanov. Implementing the algebra of logic functions in terms of constant depth formulas in the basis &, $\vee$, $\neg$. *Dokl. Ak. Nauk. SSSR*, 136:1041–1042, 1961.

Sébastien Roch. On learning thresholds of parities and unions of rectangles in random walk models. *Random Structures and Algorithms*, 31(4):406–417, 2007.

Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

Journal of Machine Learning Research 15 (2014) 3847-3870 Submitted 12/12; Revised 7/14; Published 11/14

Transfer Learning Decision Forests for Gesture Recognition

Norberto A. Goussies NGOUSSIE@DC.UBA.AR
Sebastián Ubalde SEUBALDE@DC.UBA.AR
Marta Mejail MARTA@DC.UBA.AR
Departamento de Computación, Pabellón I
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Autónoma de Buenos Aires, C1428EGA
Argentina

Editor: Isabelle Guyon, Vassilis Athitsos, and Sergio Escalera

Abstract

Decision forests are an increasingly popular tool in computer vision problems. Their advantages include high computational efficiency, state-of-the-art accuracy and multi-class support. In this paper, we present a novel method for transfer learning which uses decision forests, and we apply it to recognize gestures and characters. We introduce two mechanisms into the decision forest framework in order to transfer knowledge from the source tasks to a given target task. The first one is mixed information gain, which is a data-based regularizer. The second one is label propagation, which infers the manifold structure of the feature space. We show that both of them are important to achieve higher accuracy. Our experiments demonstrate improvements over traditional decision forests in the ChaLearn Gesture Challenge and MNIST data set. They also compare favorably against other state-of-the-art classifiers.

Keywords: decision forests, transfer learning, gesture recognition

1. Introduction

Machine learning tools have achieved significant success in many computer vision tasks, including face detection (Viola and Jones, 2004), object recognition (Felzenszwalb et al., 2010), character recognition (LeCun et al., 1998) and gesture recognition (Guyon et al., 2013). Those tasks are often posed as a classification problem, namely identifying to which of a set of categories a new observation belongs. Such classifiers are usually learned from scratch using a training data set collected for the task. A major advantage of using machine learning tools is that they tend to deal robustly with the complexities found in real data.

However, in many cases it is difficult to create new training data sets for each new computer vision task. Although the problem remains unsolved, some progress has already been made in certain computer vision tasks, such as object recognition (Fei-Fei et al., 2006) and action recognition (Seo and Milanfar, 2011). The key insight is to try to replicate the ability of the human brain, which is capable of learning new concepts applying previously acquired knowledge.

Transfer learning aims at extracting the knowledge from one or more source tasks, and applying that knowledge to a target task. As opposed to multi-task learning, rather than

simultaneously learning the source and target tasks, transfer learning focus more on learning the target task. The roles of the source and target tasks are not symmetric (Pan and Yang, 2010). The goal is to exploit the knowledge extracted from the source tasks so as to improve the generalization of the classifier in the target task.

Many examples can be found in computer vision where transfer learning can be truly beneficial. One example is optical character recognition, which seeks to classify a given image into one of the characters of a given alphabet. Most methods have focused on recognizing characters from the English alphabet (LeCun et al., 1998). The recognition of characters from other alphabets, such as French, implies collecting a new training data set (Grosicki and Abed, 2011). In that case, it would be helpful to transfer the classification knowledge into the new domain.

The need for transfer learning also arises in gesture recognition (Guyon et al., 2013), which aims at recognizing a gesture instance drawn from a gesture vocabulary. For example, a gesture vocabulary may consist of Italian gestures or referee signals. In this case, the classifier needs to predict the gesture of the vocabulary that corresponds to a given video. Again, it would be interesting to improve the performance of a system by exploiting the knowledge acquired from similar vocabularies.

In this paper, we present a novel method for transfer learning which extends the decision forests framework (Breiman, 2001; Criminisi et al., 2012), and we apply it to transfer knowledge from multiple source tasks to a given target task. We introduce two mechanisms in order to transfer knowledge from the source tasks to the target task. The first one is mixed information gain, which is a data-based regularizer. The second one is label propagation, which infers the manifold structure of the feature space.

Decision forests have certain properties that make them particularly interesting for computer vision problems. First, decision forests are multi-class classifiers; therefore it is not necessary to train several binary classifiers for a multi-class problem. Second, they are fast both to train and test. Finally, they can be parallelized, which makes them ideal for GPU (Sharp, 2008) and multi-core implementations.

The first key contribution is to revise the criterion for finding the parameters of each internal node of the decision forests in the transfer learning setting. The novel criterion exploits the knowledge from the source tasks and the target task to find the parameters for each internal node of the decision forests. The additional information penalizes split functions with a high information gain in the target task and a low information gain in the source tasks. We prove that the novel criterion is beneficial.

The second key contribution is to propagate labels through leaves in order to infer the manifold structure of the feature space. The aim of this step is to assign a predictive model to the leaves without training samples of the target task after the trees of the decision forest are grown. We create a fully connected graph, for each tree in the forest, where the nodes are the leaves of the tree and the weight of each edge takes into account the training data reaching the leaves. An implicit assumption of this step is that nearby leaves should have similar predictive models.

We extensively validate our approach in two challenging data sets. First, our experiments in the ChaLearn gesture challenge data set (Guyon et al., 2012) show that our method does not have a uniform margin of improvement over all the tasks. However, we demonstrate that when there are source tasks related to the target task, we obtain greater

improvements. Second, our experiments in the MNIST data set (LeCun et al., 1998) show that greater improvements are obtained, compared to classification decision forests, when there are only a few training samples.

This paper is organized as follows. We summarize previous work on transfer learning in Section 2. Section 3 describes the novel transfer learning decision forest in, illustrates its performance on some artificial data sets, and proves some properties of the mixed information gain. In Section 4 we show how the transfer learning decision forests can be used to recognize gestures when there is only one training sample. We present our experiments on the ChaLearn data set and the MNIST data set in Section 5. Finally, Section 6 details our conclusions.

2. Related Work

In the following we will review transfer learning techniques which have been applied to computer vision problems. A recent survey (Pan and Yang, 2010) provides a comprehensive overview of the developments for classification, regression and clustering. In recent years, the computer vision community has become increasingly interested in using transfer learning techniques, especially for object recognition (Levi et al., 2004; Sudderth et al., 2005; Fei-Fei et al., 2006; Bart and Ullman, 2005; Torralba et al., 2007; Quattoni et al., 2008; Bergamo and Torresani, 2010; Gopalan et al., 2011; Saenko et al., 2010; Tommasi et al., 2014).

A variety of methods have been proposed in the generative probabilistic setting (Fei-Fei et al., 2006; Sudderth et al., 2005). These models consider the relationships between different object parts during the training process. The key idea is to share some parameters or prior distributions between object categories, using the knowledge from known classes as a generic reference for newly learned models. The association of objects with distributions over parts can scale linearly (Sudderth et al., 2005), or exponentially (Fei-Fei et al., 2006).

Moreover, discriminative models have been extended to the transfer learning setting (Dai et al., 2007; Yao and Doretto, 2010; Aytar and Zisserman, 2011; Tommasi et al., 2014; Lim et al., 2011; Torralba et al., 2007). Transfer learning has been applied to the SVM framework, during the training process of the target detector the previously learned template is introduced as a regularizer into the cost function (Tommasi et al., 2014; Aytar and Zisserman, 2011). Based on boosting (Freund and Schapire, 1997) a framework that allows users to utilize a small amount of newly labeled data has been developed (Dai et al., 2007). Later, the framework has been extended for handling multiple sources (Yao and Doretto, 2010) .

More similar to our method, instance transfer approaches (Pan and Yang, 2010) consider source and target data together during the training process. A loss function for borrowing examples from other classes in order to augment the training data of each class has been proposed by Lim et al. (2011). A method for learning new visual categories is described by Quattoni et al. (2008), using only a small subset of reference prototypes for a given set of tasks. As mentioned earlier, a boosting-based algorithm that allows knowledge to be effectively transferred from old to new data has been proposed by Dai et al. (2007) and extended later by Yao and Doretto (2010). The effectiveness of the novel algorithm is analyzed both theoretically and empirically. In this paper, we develop an instance transfer

approach that exploits source and target data to find the parameters of each internal node of the decision forest.

Few researchers have addressed the problem of transfer learning using decision forests or trees. Leistner et al. (2009) extends random forests to semi-supervised learning. In order to incorporate unlabeled data a maximum margin approach is proposed, which is optimized using a deterministic annealing-style technique. Wang et al. (2008) proposed to treat each input attribute as extra task to bias each component decision tree in the ensemble. Pei et al. (2013) proposed a novel criterion for node splitting to avoid the rank deficiency in learning density forests for lipreading. The method proposed by won Lee and Giraud-Carrier (2007) learns a new task by traversing and transforming a decision tree previously learned for a related task. The transfer learning decision tree learns the target task from a partial decision tree model induced by ID3 (Quinlan, 1986). In this paper, we follow a different approach, first we consider the source and target data when we build each tree of the decision forest. Second, decision forests reduce the variance of the classifier aggregating the results of multiple random decision trees.

Our approach shares some features with the work by Faddoul et al. (2012), who propose to transfer learning with boosted C4.5 decision trees. The main difference is that their method reduces the variance of the decision trees by means of boosting, which has been shown to be less robust against label noise when compared with decision forests (Breiman, 2001; Leistner et al., 2009). In addition, we use label propagation to learn the manifold structure of the feature space, and assign predictive models only to the leaves of the trees.

There has been a growing interest in applying transfer learning techniques to gesture recognition. A method for transfer learning in the context of sign language is described by Farhadi et al. (2007). A set of labeled words in the source and target data is shared so as to build a word classifier for a new signer on a set of unlabeled target words. A transfer learning method for conditional random fields is implemented to exploit information in both labeled and unlabeled data to learn high-level features for gesture recognition by Liu et al. (2010). More recently, the ChaLearn Gesture Competition (Guyon et al., 2013) provided a benchmark of methods that apply transfer learning to gesture recognition. Several approaches submitted to the competition have been published (Malgireddy et al., 2013; Lui, 2012; Wan et al., 2013).

3. Transfer Learning Decision Forests

We consider $N+1$ classification tasks $T_0, \ldots, T_N$ over the instance space $\mathbb{R}^d$ and label sets $\mathcal{Y}_0, \ldots, \mathcal{Y}_N$. We are interested in solving the classification task T_0 using the knowledge of the other tasks in order to improve classification accuracy. Our transfer learning algorithm will take as input the training set $S = \{(\mathbf{x}_i, \mathbf{y}_i, j) | \mathbf{x}_i \in \mathbb{R}^d,\ \mathbf{y}_i \in \mathcal{Y}_j,\ j \in \{0, \ldots, N\},\ 1 \leq i \leq M\}$. The projected sets $T_j S = \{(\mathbf{x}_i, \mathbf{y}_i) | \mathbf{x}_i \in \mathbb{R}^d,\ \mathbf{y}_i \in \mathcal{Y}_j, (\mathbf{x}_i, \mathbf{y}_i, j) \in S\}$ are the training sets for each task T_j. The empirical histogram for a training set S of a task T is defined as $\hat{p}_{TS}(\mathbf{y}) = \frac{1}{|TS|} \sum_{(\mathbf{x}', \mathbf{y}') \in TS} \delta_{\mathbf{y}'}(\mathbf{y})$ where $\delta_{\mathbf{y}'}(\mathbf{y})$ is the Kronecker delta and the empirical entropy is defined as $\mathcal{H}(TS) = -\sum_{\mathbf{y} \in \mathcal{Y}} \hat{p}_{TS}(\mathbf{y}) \log(\hat{p}_{TS}(\mathbf{y}))$, we will note $\hat{p}_S(\mathbf{y})$ or $\mathcal{H}(S)$ to make the notation simpler when it is convenient and unambiguous.

The goal is to find a decision forest $\mathcal{F} = \{F_1, \ldots, F_T\}$, defined as an ensemble of T decision trees F, which minimizes the classification error. A decision tree F is a strictly

binary tree in which each node k represents a subset R_k in the instance space $\mathbb{R}^d$ and all the leaves ∂F form a partition $\mathcal{P}$ of $\mathbb{R}^d$. In addition, each leaf $k \in \partial F$ of a decision tree F has a predictive model associated with it: $p_F(\mathbf{y}|\mathbf{x} \in R_k)$. The internal nodes $k \in F^\circ$ of a decision tree have a linear split function: $h(\mathbf{x}, \boldsymbol{\theta}_k) = \mathbf{x} \cdot \boldsymbol{\theta}_k$, where $\boldsymbol{\theta}_k$ are the parameters of node k. The subset represented by the left child k_L of node k is defined as $R_{k_L} = R_k^L = \{\mathbf{x} \in \mathbb{R}^d | \mathbf{x} \in R_k \wedge h(\mathbf{x}, \boldsymbol{\theta}_k) < 0\}$ and, similarly, we define $R_{k_R} = R_k^R = \{\mathbf{x} \in \mathbb{R}^d | \mathbf{x} \in R_k \wedge h(\mathbf{x}, \boldsymbol{\theta}_k) \geq 0\}$ as the subset represented by the right child k_R. The training set reaching node k is defined as $S_k = \{(\mathbf{x}, \mathbf{y}, j) \in S | \mathbf{x} \in R_k\}$.

3.1 Training

The training algorithm of a decision forest $\mathcal{F}$ consists in training each of the trees $F \in \mathcal{F}$ independently, introducing a certain level of randomness in the training process in order to de-correlate individual tree predictions and improve generalization.

We grow each tree using an extension of the classical training algorithm (Criminisi et al., 2012). The algorithm follows a top-down approach, optimizing the parameters $\boldsymbol{\theta}$ of the root node in the beginning and recursively processing the child nodes. The recursion is stopped when all the items in the training set have the same labels, or the maximum depth D is reached, or the number of points reaching the node is below the minimum number of points allowed κ.

In this paper, we adapt the procedure for optimizing the parameters $\boldsymbol{\theta}_k$ for each node $k \in F^\circ$ to the transfer learning setting (Pan and Yang, 2010). The difference between the classification decision forest (Criminisi et al., 2012) and the transfer learning decision forest is the objective function. In the former, the information gain is used to find the best parameters, taking into account only one task. By contrast, in this paper we use the mixed information gain function as described in Section 3.1.1.

The partition $\mathcal{P}$ defined by the leaves ∂F after making a tree F grow might contain regions R with no training samples of the target task T_0. Therefore, we cannot define a predictive model for those regions. In order to overcome this issue we infer the labels from the regions that have training samples of task T_0, as described in Section 3.1.2.

3.1.1 Mixed Information Gain

We believe that valuable knowledge can be transferred from the source tasks $T_1, \ldots, T_N$ to the target task T_0, as it happens with humans. For example, it is simpler to learn a new sign language if another sign language has already been learned. In other words, there is latent information that can be understood as common sense.

In our formulation, this common sense information is included in the process of making each tree $F \in \mathcal{F}$ in the forest grow. The main idea is, therefore, to find parameters $\boldsymbol{\theta}_k$ for each $k \in F^\circ$ in order to obtain a partition $\mathcal{P}$ of the feature space $\mathbb{R}^d$ such that, in each region $R \in \mathcal{P}$, the training samples of each task T have the same label. This aims at improving the generalization capabilities of each tree independently, since each region $R \in \mathcal{P}$ is found using more training samples, and is more general because it is encouraged to split the training samples of several tasks simultaneously.

Unfortunately, this is a very difficult problem. For this reason, we use a greedy heuristic which consist in recursively choosing for each internal node $k \in F^\circ$ the parameters $\boldsymbol{\theta}_k$ of the

split function $h(\mathbf{x}, \boldsymbol{\theta}_k)$, which makes the training samples reaching the child nodes as "pure" as possible. The information gain achieved by splitting the training set TS_k reaching the internal node $k \in F^\circ$ of a task T using parameter $\boldsymbol{\theta}_k$ is computed using the information gain function

$$\mathcal{I}(TS_k, \boldsymbol{\theta}_k) = \mathcal{H}(TS_k) - \sum_{i \in \{L,R\}} \frac{|TS_k^i|}{|TS_k|} \mathcal{H}(TS_k^i)$$

where $TS_k^L = \{(\mathbf{x}, \mathbf{y}) | (\mathbf{x}, \mathbf{y}) \in TS_k \wedge h(\mathbf{x}, \boldsymbol{\theta}_k) < 0\}$ and $TS_k^R = \{(\mathbf{x}, \mathbf{y}) | (\mathbf{x}, \mathbf{y}) \in TS_k \wedge h(\mathbf{x}, \boldsymbol{\theta}_k) \geq 0\}$. In this paper, the parameters $\boldsymbol{\theta}_k$ of each internal node $k \in F^\circ$ are found maximizing the information gain of all the tasks $T_0, \ldots, T_N$ simultaneously

$$\boldsymbol{\theta}_k^* = \arg \max_{\boldsymbol{\theta}_k \in \mathcal{T}_k} (1 - \gamma) \mathcal{I}(T_0 S_k, \boldsymbol{\theta}_k) + \gamma \sum_{n=1}^{N} p_{n,k} \mathcal{I}(T_n S_k, \boldsymbol{\theta}_k) \tag{1}$$

where γ is a scalar parameter that weights the two terms, $\mathcal{T}_k \subset \mathbb{R}^d$ is a small subset of the instance space available when training the internal node $k \in F^\circ$, and $p_{n,k}$ is the fraction of samples of the source task T_n in the samples reaching the node k, $p_{n,k} = \frac{|T_n S_k|}{\sum_{j=1}^{N} |T_j S_k|}$.

The maximization of (1) is achieved using randomized node optimization (Criminisi et al., 2012). We perform an exhaustive search over subset $\mathcal{T}_k$ of the feature space parameters $\mathbb{R}^d$. The size of the subset is a training parameter noted as $\rho = |\mathcal{T}_k|$. The randomized node optimization is a key aspect of the decision forest model, since it helps to de-correlate individual tree predictions and to improve generalization.

The first term of the objective function in (1) is the information gain associated with the training samples reaching node k for the target task T_0. This term encourages the parameters $\boldsymbol{\theta}_k$ to find a split function $h(\mathbf{x}, \boldsymbol{\theta}_k)$ that decreases the entropy of the training set of the target task T_0 reaching the children nodes of k.

Additional information is introduced into the second term of the objective function in (1) for the purposes of increasing the generalization performance. This information encourages the parameters $\boldsymbol{\theta}_k$ to make the training samples of source tasks reaching the descendant nodes of k as pure as possible. The key idea is that this term penalizes split functions $h(\mathbf{x}, \boldsymbol{\theta}_k)$ with a high information gain in the target task T_0 and a low information gain in the source tasks $T_1, \ldots, T_N$. Those splits might have a high information gain in the target task T_0 only because the training set for task T_0 is limited, and if we choose them the generalization performance will decrease.

A key insight of our work is an alternative representation of the second term in (1). It is possible to consider all the source tasks $T_1, \ldots, T_N$ together concatenating the label sets $\mathcal{Y}_1, \ldots, \mathcal{Y}_N$, denoted by $\mathcal{Y}_{1\ldots N} = \oplus_{n=1}^{N} \mathcal{Y}_n$. The new task is noted as $T_{1\ldots N}$ and the training sample is noted as $T_{1\ldots N} S = \{(\mathbf{x}, \mathbf{y}) | (\mathbf{x}, \mathbf{y}, j) \in S, j \in \{1, \ldots, N\}, \mathbf{y} \in \oplus_{n=1}^{N} \mathcal{Y}_n\}$. Using the generalized grouping rule of the entropy (Cover and Thomas, 2006) an alternative expression for the second term in (1) is found

$$\mathcal{I}(T_{1\ldots N} S_k, \boldsymbol{\theta}_k) = \sum_{n=1}^{N} p_{n,k} \mathcal{I}(T_n S_k, \boldsymbol{\theta}_k).$$

This equation relates the information gain of several source tasks $T_1, \ldots, T_N$ to the information gain of another source task $T_{1\ldots N}$. An important consequence of this equation

is that we can combine the training set of the simpler tasks $T_1, \ldots, T_N$ to obtain a larger training set for another source task $T_{1\ldots N}$. Therefore, increasing the number of training samples per source task or the number of source tasks has a similar effect.

This observation has previously been made in the multi-task learning literature (Faddoul et al., 2012). However, Faddoul et al. (2012) avoids the high variance of the decision trees by using the boosting framework, whereas we use a different approach, based on decision forest, for the same purpose.

We explain in more detail how the combination of the information gain of tasks $T_0, \ldots, T_N$ for finding the optimal parameters $\boldsymbol{\theta}_k$ improves the generalization properties of the decision forests. The parameters $\boldsymbol{\theta}_k$ are found using an empirical estimation of the entropy $\mathcal{H}(S_k)$ of the training samples S_k reaching node k and its children. Consequently, errors in estimating entropy can result in very different trees. Tighter bounds for the expected entropy are found by increasing the number of training samples, as explained in Theorem 1.

Theorem 1 *Let P be a probability distribution on $\mathbb{R}^d \times \mathcal{Y}$ such that the marginal distribution over $\mathcal{Y}$ is a categorical distribution with parameters $p_1, \ldots, p_{|\mathcal{Y}|}$, and suppose $S_K = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_K, \mathbf{y}_K)\}$ is the set generated by sampling K times from $\mathbb{R}^d \times \mathcal{Y}$ according to P. Let $\mathcal{H}(P) = -\sum_{y=1}^{|\mathcal{Y}|} p_y \log(p_y)$ be the entropy of distribution P. Then $\mathbb{E}(\mathcal{H}(S_K)) + \sum_{\mathbf{y}\in\mathcal{Y}} p_{\mathbf{y}} \log\left(1 + \frac{1-p_{\mathbf{y}}}{Kp_{\mathbf{y}}}\right) \leq \mathcal{H}(P) \leq \mathbb{E}(\mathcal{H}(S_K))$.*

This theorem is proved in the Appendix A.

Theorem 1 shows that the empirical entropy $\mathcal{H}(S_K)$ is closer to the entropy of the distribution P when the training set is larger, since when $K \to \infty$, $\log\left(1 + \frac{1-p_{\mathbf{y}}}{Kp_{\mathbf{y}}}\right) \to 0$. Therefore, if we assume that the source tasks are related to the target task i.e., both have a similar distribution P, using Theorem 1 we can conclude that the mixed information gain (1) finds parameters $\boldsymbol{\theta}_k$ that achieve lower generalization errors than the traditional information gain $\mathcal{I}(T_0 S_k, \boldsymbol{\theta}_k)$.

To gain some insight into how the mixed information gain works, Figure 1 considers a toy problem with two tasks, each with two labels. It is intuitively clear that the problem of estimating the information gain of a split with only a few training samples of the target task is that there are a lot of possible splits with the same empirical information gain but different generalization capabilities. Our goal is to discover which split to use, and we intend to choose the one with the best generalization capability. In Figure 1 all the splits have the same information gain but different mixed information gain. When, in our formulation, we use the additional training samples from the source tasks to compute the information gain of a split, some of the splits are penalized for having a low information gain in the source task and, thus, this allows us to find a split with increased generalization.

One of the major problems with decision trees is their high variance. A small change in the training data can often result in a very different series of splits. The major reason for this instability is the hierarchical nature of the process: the effect of an error in the top split is propagated down to all the splits below it (Hastie et al., 2003). Decision forests (Breiman, 2001) build a large collection of de-correlated decision trees, and hence reduce the variance averaging the prediction of each of them. The mixed information gain is a complementary approach for reducing their variance which increases the generalization of each tree independently. It is important to note that the mixed information preserves the

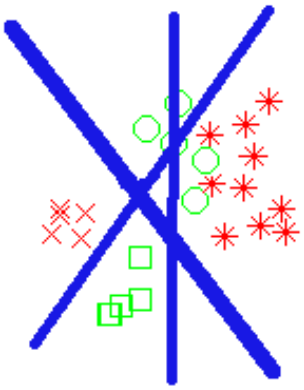


Figure 1: Illustration of mixed information gain on a toy problem in which there are two tasks, each with two labels. The thickness of the blue lines indicates the mixed information gain of the split (all the splits have the same information gain). Task T_0 has two green labels ($\mathcal{Y}_0 = \{\times, *\}$) and task T_1 has two red labels ($\mathcal{Y}_1 = \{\bigcirc, \square\}$).

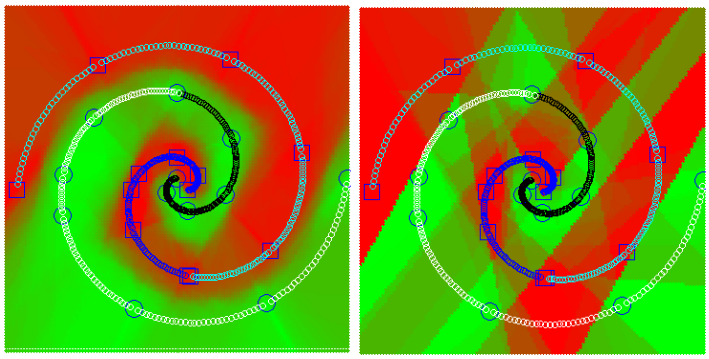
diversity of the forests, which is essential to improve the generalization error. The random nature of the random node optimization (Criminisi et al., 2012) used to optimize (1) allows us to keep a high diversity among the trees.

Figures 2a and 2b compare the output classification on all the points in a rectangular section of the feature space for a decision forest classifier and for our transfer learning decision forest classifier. Both decision forests were trained with the same maximum depth $D = 8$, and have the same number of trees $|\mathcal{F}| = 100$. The data set for the target and source task is organized in the shape of a two-arm spiral. We can see that the classification decision forests have serious generalization problems since, even when all the training data of the target task is correctly classified, the spiral structure is not predicted accurately. In contrast, the spiral structure is predicted by the transfer learning decision forests as shown in Figure 2a.

3.1.2 Label Propagation

For each leaf $k \in \partial F$ of each tree $F \in \mathcal{F}$, we must have a predictive model $p_F(\mathbf{y}|\mathbf{x} \in R_k)$ that estimates the probability of label $\mathbf{y} \in \mathcal{Y}_0$ given a previously unseen test input $\mathbf{x} \in R_k \subseteq \mathbb{R}^d$. This poses a problem when we make each tree grow using the mixed information gain because we may end up with leaves $k \in \partial F$ that have no training samples of the target task T_0 to estimate the predictive model $p_F(\mathbf{y}|\mathbf{x} \in R_k)$. In this paper we use label propagation to assign a predictive model $p_F(\mathbf{y}|\mathbf{x} \in R_k)$ to those leaves.

We are given a set of leaves $\mathcal{U} \subseteq \partial F$ without training samples of the target task T_0 and a set of leaves $\mathcal{L} \subseteq \partial F$ with training samples of the target task T_0. The goal is to obtain a predictive model $p_F(\mathbf{y}|\mathbf{x} \in R_k)$ for the leaves $k \in \mathcal{U}$ avoiding the propagation of labels through low density regions but, at the same time, propagating labels between nearby



(a) (b)

Figure 2: Left: Output classification of a transfer learning decision forest, tested on all points in a rectangular section of the feature space. The color associated with each test point is a linear combination of the colors (red and green) corresponding to the two labels ($\square, \bigcirc$) in the target task. The training data for the target task is indicated with big markers and the training data for the source task is indicated with small markers. Right: Output classification of a decision forest tested in the same feature space section as before but trained using only data for the target task.

leaves. We construct a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \partial F$ is the vertex set and $\mathcal{E}$ is the edge set with each edge $\mathbf{e}_{ij} \in \mathcal{E}$ representing the relationship between nodes $i, j \in \partial F$.

Edge $\mathbf{e}_{ij} \in \mathcal{E}$ is weighted taking into account the training samples of tasks $T_0, \ldots, T_N$. For each leaf $k \in \partial F$ we define the estimated mean $\boldsymbol{\mu}_k$ and estimated covariance Σ_k using the training samples reaching the node

$$\begin{aligned} \boldsymbol{\mu}_k &= \frac{1}{|S_k|} \sum_{(\mathbf{x},\mathbf{y},j) \in S_k} \mathbf{x} \\ \Sigma_k &= \sum_{(\mathbf{x},\mathbf{y},j) \in S_k} \sum_{(\mathbf{x}',\mathbf{y}',j') \in S_k} (\mathbf{x} - \boldsymbol{\mu}_k)(\mathbf{x}' - \boldsymbol{\mu}_k)^T. \end{aligned}$$

We use the estimated mean $\boldsymbol{\mu}_k$ and estimated covariance Σ_k to define the weight between two nodes $\mathbf{e}_{ij} \in \mathcal{E}$

$$\mathbf{e}_{ij} = \frac{1}{2} \left(d_{ij}^T \Sigma_i d_{ij} + d_{ij}^T \Sigma_j d_{ij} \right)$$

where $d_{ij} = \boldsymbol{\mu}_i - \boldsymbol{\mu}_j$ is the difference between the estimated mean of the leaves $i, j \in \partial F$. Weight $\mathbf{e}_{ij} \in \mathcal{E}$ is the symmetric Mahalanobis distance. We use it to discourage the propagation of labels through low density regions. For each node $k \in \mathcal{U}$ we find the shortest path in graph $\mathcal{G}$ to all the nodes in $\mathcal{L}$. Let $s_k^* \in \mathcal{L}$ be the node with the shortest path to node k. We assign the predictive model $p_F(\mathbf{y}|\mathbf{x} \in R_{s_k^*})$ to $p_F(\mathbf{y}|\mathbf{x} \in R_k)$.

Label propagation methods are usually at least quadratic $\mathcal{O}(n^2)$ in terms of the number of training samples, making them slow when a large number of training samples is avail-

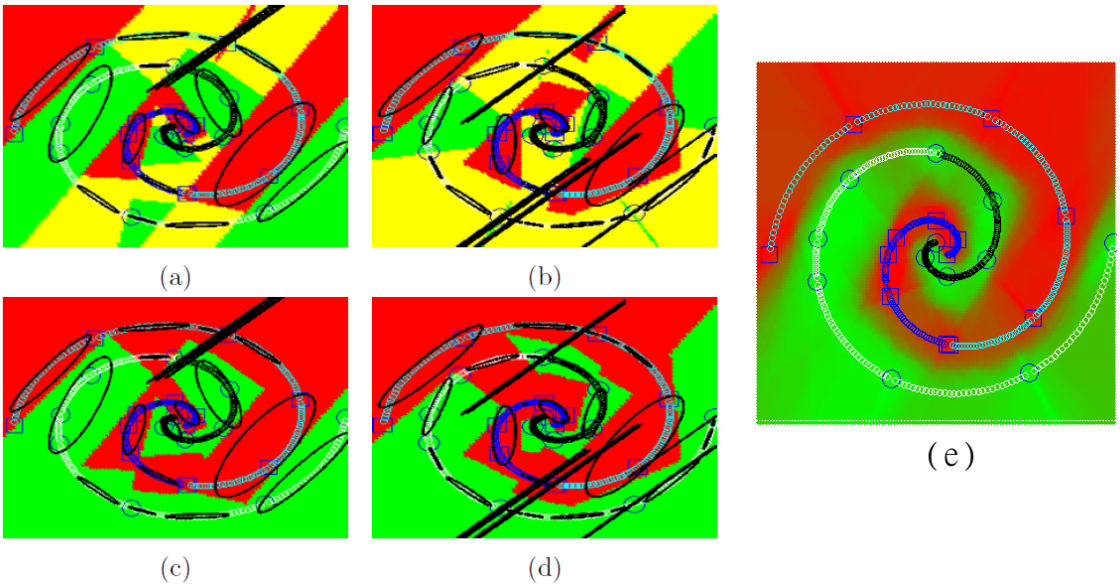


Figure 3: Illustration of the label propagation procedure between regions, as before the training data for the target task is indicated with big markers and the training data for the source task is indicated with small markers. The ellipses in black are the isocontours of a Gaussian distribution learned by maximum likelihood for each region using the training samples in the region. (a, b) show the predictive model for two different trees $F \in \mathcal{F}$ before propagating labels. The color associated with each region is a linear combination of the colors (red and green) corresponding to the two labels ($\square, \bigcirc$) in the target task. The regions in yellow are the ones without training data of the target task. (c, d) show the predictive model after the label propagation. (e) Output classification of the final transfer learning decision forest.

able. We avoid this problem by propagating the predictive model of the leaves, instead of propagating the labels of the training samples.

We illustrate the behavior of label propagation in Figure 3 using a 2D toy example. We consider the same two-arm spiral problem of Figure 2 which has data that follow a complex structure. We show the predictive models for the regions of two randomly grown trees before and after propagating labels. We observe that the predictive models are propagated following the intrinsic structure of the data, as a consequence of taking into account the training data of each region.

3.2 Testing

The predictive model of all the trees $F \in \mathcal{F}$ is combined to produce the final prediction of the forest

$$P_{\mathcal{F}}(\mathbf{y} = y|\mathbf{x}) = \frac{1}{|\mathcal{F}|} \sum_{F \in \mathcal{F}} P_F(\mathbf{y} = y|\mathbf{x}).$$

Let $l_F : \mathbb{R}^d \to \partial F$ be the function that, given a sample $\mathbf{x} \in \mathbb{R}^d$, returns the leaf such that $\mathbf{x} \in R_{l_F(\mathbf{x})}$. The prediction for a tree F is:

$$P_F(\mathbf{y} = y|\mathbf{x}) = P_F\left(\mathbf{y} = y|\mathbf{x} \in R_{l_F(\mathbf{x})}\right).$$

Finally, let $k \in \partial F$ be the leaf that is reached by sample $\mathbf{x} \in \mathbb{R}^d$. The class distribution for that leaf is:

$$P_F(\mathbf{y} = y|\mathbf{x} \in R_k) = \begin{cases} \hat{p}_{T_0 S_k}(y) & \textbf{if } T_0 S_k \neq \emptyset \\ \hat{p}_{T_0 S_{s_k^*}}(y) & \textbf{otherwise} \end{cases}.$$

Thus, $P_F(\mathbf{y} = y|\mathbf{x})$ is the empirical histogram of the training samples of the target task T_0 reaching node $l_F(\mathbf{x})$ if any. Otherwise, $P_F(\mathbf{y} = y|\mathbf{x})$ is the empirical histogram associated with the node that has the shortest path to $l_F(\mathbf{x})$.

4. Gesture Recognition

Gesture recognition is one of the open challenges in computer vision. There is a big number of potential applications for this problem, including surveillance, smart-homes, rehabilitation, entertainment, animation and human–robot interaction and sign language recognition just to mention a few. The task of gesture recognition is to determine the gesture label that best describes a gesture instance, even when performed by different people, from various viewpoints and in spite of large differences in manner and speed.

To reach that goal, many approaches combine vision and machine learning tools. Computer vision tools are employed to extract features that provide robustness to distracting cues and that, at the same time, are discriminative. Machine learning is used to learn a statistical model from those features, and to classify new examples using the models learned. This poses a problem in gesture recognition since it is difficult to collect big data sets to learn statistical models. Therefore, in this paper we perform experiments aimed at showing that our transfer learning decision forests are useful to mitigate this problem.

Recently, the ChaLearn competition (Guyon et al., 2012) provided a challenging data set to evaluate whether transfer learning algorithms can improve their classification performance using similar gesture vocabularies. The data set is organized into batches, with only one training example of each gesture in each batch. The goal is to automatically predict the gesture labels for the remaining gesture sequences (test examples). The gestures of each batch are drawn from a small vocabulary of 8 to 12 unique gestures, when we train a classifier to predict the labels of a target batch (or task) T_0 we use the training samples of T_0 and of the other batches $T_1, \ldots, T_N$.

Each batch of the ChaLearn competition includes 100 recorded gestures grouped in sequences of 1 to 5 gestures performed by the same user (Guyon et al., 2012). There is only one gesture in the training sequences, but there might be more than one gesture in the testing sequences. Therefore in order to use the method described in this section we need to temporally segment the testing sequences. To this end, we use the dynamic time warping (DTW) implementation given by the organizers.

In this section, we describe the features and the classifiers used to validate our approach, as well as their application to the ChaLearn competition (Guyon et al., 2012). First, Section 4.1 describes the features, and then, Section 4.2 describes the classifier.

$\tau \setminus \xi$	16	24	32	40
1	32.61 ± 0.14 %	32.61 ± 0.24 %	30.35 ± 0.22 %	29.26 ± 0.26 %
4	**30.43 ± 0.17 %**	31.52 ± 0.15 %	29.26 ± 0.15 %	28.17 ± 0.19 %
8	**30.43 ± 0.13 %**	**27.35 ± 0.16 %**	**28.09 ± 0.14 %**	**27.06 ± 0.18 %**
12	32.12 ± 0.23 %	32.61 ± 0.29 %	34.78 ± 0.31 %	29.35 ± 0.33 %
16	33.72 ± 0.28 %	32.61 ± 0.29 %	34.78 ± 0.25 %	30.43 ± 0.31 %

Table 1: Classification error in the test set of the *devel*11 batch for different combination of MHI parameters. In all the experiments we leave the the spatial resolution of each frame fixed to $\omega_1 \times \omega_2 = 16 \times 12$.

4.1 Motion History Images

Given a depth video V where $V(x, y, t)$ is the depth of the pixel with coordinates (x, y) at the tth frame. We compute the motion history image (MHI) (Bobick and Davis, 1996, 2001; Ahad et al., 2012) for each frame using the following function:

$$H_\tau(x, y, t) = \begin{cases} \tau & \text{if } |V(x, y, t) - V(x, y, t-1)| \geq \xi \\ \max(0, H_\tau(x, y, t-1) - 1) & \text{otherwise} \end{cases}$$

where τ defines the temporal extent of the MHI, and ξ is a threshold employed to perform the foreground/background segmentation at frame t. The result is a scalar-valued image for each frame of the original video V where pixels that have moved more recently are brighter. MHI H_τ represents the motion in an image sequence in a compact manner, the pixel intensity is a function of the temporal history of motion at that point. A common problem when computing MHI H_τ using the color channel is the presence of textured objects in the image sequence; here we use the depth video V to overcome this issue. This is a relevant problem in gesture recognition, because, as a result of the clothes texture, the MHI is noisy (Ahad et al., 2012).

An interesting property of the MHI is that it is sensitive to the direction of motion; hence it is well suited for discriminating between gestures with an opposite direction. An advantage of the MHI representation is that a range of times may be encoded in a single frame, and thus, the MHI spans the time scale of human gestures. After computing MHI H_τ we reduce the spatial resolution of each frame to $\omega_1 \times \omega_2$ pixels. Then, we flatten the MHI for each frame and obtain a feature $\mathbf{x}_m \in \mathbb{R}^{\omega_1 \omega_2}$.

Figure 4 contrasts the result of computing the MHI using the RGB channel with the one obtained using the depth channel. In the first row, we see that the clothes texture generates noise in the MHI computed using the RGB channel. In the second row, the MHI of the RGB channel is noisy because of the shadow from the moving arm. Both problems are avoided using the depth channel for computing the MHI. The parameters to compute the MHI in all the cases were $\tau = 15$, and $\xi = 30$. Table 1 shows the classification error in the test set of the *devel*11 batch of the ChaLearn competition, after training a decision forest with the following parameters $D = 8, T = 50$.

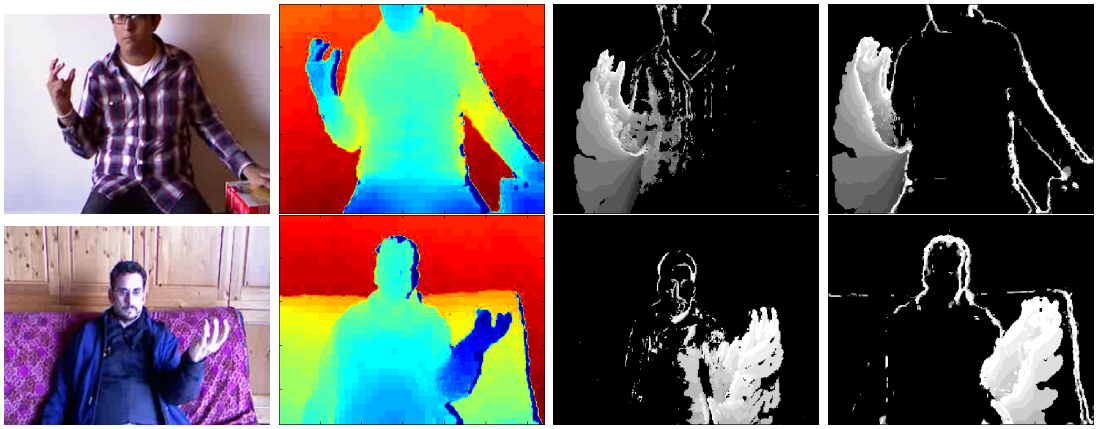


Figure 4: Comparison of the MHI computed using the depth channel or the RGB channel for two different training videos of the ChaLearn competition. The first two columns show the RGB channel and the depth channel, whereas the third and fourth columns show the MHI computed using the RGB channel and the MHI computed using the depth channel, respectively.

4.2 Naive Bayes

A main research trend in gesture recognition is to train hidden Markov models (HMMs) and theirs variants (Bowden et al., 2004; Kurakin et al., 2012), in order to exploit the temporal relation of a gesture. A drawback of this approach is that many training samples are required to train the large number of parameters of an HMM. Additionally, recognition rates might not improve significantly (Li et al., 2008). This limitation has been recognized by Bowden et al. (2004) and a two-stage classifier was proposed to obtain one-shot learning.

Since in the ChaLearn competition (Guyon et al., 2012) there is only one labeled training sample of each gesture, we use the naive Bayes model which has a smaller number of parameters than HMM. We use transfer learning decision forests to predict the probability that each frame will be part of a given gesture. We combine the predictions of the transfer learning decision forests for each frame using the naive Bayes model. An advantage of the naive Bayes assumption is that it is not sensitive to irrelevant frames (the probabilities for all the labels will be quite similar).

Given a video V of an isolated gesture, we want to find its label $\mathbf{y} \in \mathcal{Y}_0$. Assuming that the class prior $p(\mathbf{y})$ is uniform we have:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}_0} p(\mathbf{y}|V) = \arg\max_{\mathbf{y}\in\mathcal{Y}_0} p(V|\mathbf{y})$$

Let $\mathbf{x}_1, \ldots, \mathbf{x}_M$ denote the MHI for each frame of a video V with M frames. We assume the naive Bayes model i.e., that the features $\mathbf{x}_1, \ldots, \mathbf{x}_M$ are i.i.d. given the label $\mathbf{y}$, namely:

$$p(V|\mathbf{y}) = p(\mathbf{x}_1, \ldots, \mathbf{x}_M|\mathbf{y}) = \prod_{m=1}^{M} p(\mathbf{x}_m|\mathbf{y}) = \prod_{m=1}^{M} p(\mathbf{y}|\mathbf{x}_m)\frac{p(\mathbf{x}_m)}{p(\mathbf{y})} \tag{2}$$

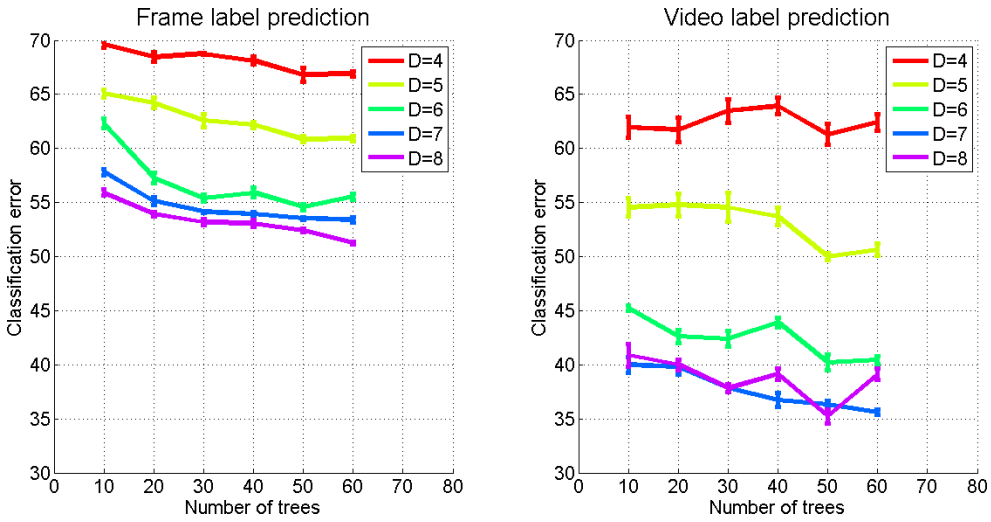


Figure 5: Effect of the training parameters for the frame label classification error $p(\mathbf{y}|\mathbf{x})$ (left) and video label classification error $p(\mathbf{y}|V)$ (right) in the devel11 batch using the transfer learning decision forests.

We compute the probability $p(\mathbf{y}|\mathbf{x}_m)$ using our proposed transfer learning decision forest $\mathcal{F}$. The data set for training the forest $\mathcal{F}$ consists of all the frames in each training video in the target task T_0 and source tasks $T_1, \ldots, T_N$. We propose to use the frames of the training videos in the source tasks to obtain a better classifier for each frame.

Taking the logarithm in (2) and ignoring the constant terms we obtain the following decision rule:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}_0} p(\mathbf{y}|V) = \arg\max_{\mathbf{y}\in\mathcal{Y}_0} \sum_{m=1}^{M} \log\left(p_{\mathcal{F}}(\mathbf{y}|\mathbf{x}_m)\right)$$

Note that we use the same forest $\mathcal{F}$ for computing the label distribution of all the frames in video V. For this reason, given a frame $\mathbf{x}$, we expect distribution $p_{\mathcal{F}}(\mathbf{y}|\mathbf{x})$ to be multimodal, which is an issue for several statistical methods. However, since the transfer learning decision forest has a predictive model for each leaf of its tree, it can deal with this type of distribution without major problems

Figure 5 compares the classification error when predicting the label of a frame $p(\mathbf{y}|\mathbf{x})$ with the classification error when predicting the label of a video $p(\mathbf{y}|V)$, for different combinations of training parameters in the devel11 batch. We observe that the maximum depth D has a larger impact to predict the label of a video than the number of trees $|\mathcal{F}|$. Moreover, the classification error when predicting the label of a frame is greater than the classification error when predicting the label of a video. This means, as expected, that some frames are more discriminative than others, and that the misclassification of some frames is not a decisive factor for classifying a video correctly.

5. Experiments

In this section we present a series of experiments on the ChaLearn Gesture Challenge (Guyon et al., 2012) and MNIST data set (LeCun et al., 1998) to assess the performance of our proposed algorithm.

5.1 ChaLearn Gesture Challenge

Here, we evaluate the transfer learning decision forests on the ChaLearn Gesture Challenge. First, we compare the results obtained for different parameters of the transfer learning decision forests, and then we compare these results with the ones reported in related works. For the MHI computation in this section, we set the temporal extent $\tau = 8$, the threshold $\xi = 25$, and reduce the spatial resolution of each frame to $\omega_1 \times \omega_2 = 16 \times 12$ pixels.

5.1.1 Transfer Decision Learning Parameters

To obtain a general idea of the effect of the training parameters, Figure 6 evaluates the classification error for different combinations of training parameters. We report the average classification error obtained in the *devel* batches. We use the temporal segmentation of the videos provided by the ChaLearn competition organizers. The experiments show that when the mixing coefficient γ is between 25% and 50%, the classification error is the smallest. This means that we obtain improvements when transferring knowledge from related tasks but, nevertheless, we still need to make the decision trees grow using information of the target task.

It is important to remark that when $\gamma = 0$ we are not using the training data of the source tasks and our mixed information gain simplifies to the usual information gain, thus, only the label propagation extension is being used. The classification error for the case $\gamma = 0$ indicates that we achieve an improvement using the label propagation alone. We obtain an additional improvement when γ is between 25% and 75%, therefore we can conclude that both extensions are important to reduce the classification error.

The maximum depth of the trees is a highly relevant parameter for the transfer learning decision forests, and has some influence for the classification decision forests. As expected, the greater the maximum depth, the smaller the classification error. It is interesting to observe that the difference in the classification error between different values of the mixing coefficients γ is reduced when the maximum depth is increased.

Figure 7 shows the confusion matrices for the classifiers of the transfer learning decision forests (TLDFs) and the decision forests (DFs) in the batches *devel*06, *devel*11 and *devel*14. To train the TLDFs, we set the number of trees $T = 50$, the maximum depth $D = 8$, the mixing coefficient $\gamma = 25\%$, and the size of the subset $|\mathcal{T}| = 50$. In these batches the TLDFs classifier shows improvements over the DFs classifier. The improvement is not uniform for all the gestures of the batches, but only for some of them. This is because not all the gestures can benefit from the training data of the source tasks. Only the gestures that have, at least, one similar gesture in a source task show improvements.

The confusion matrix for the *devel*06 batch in Figure 7 shows significant improvements in the classification of the last gesture. Figure 8 shows a representative image of that gesture and similar gestures in the *devel*13 and *devel*15 batches. The person in front of the camera

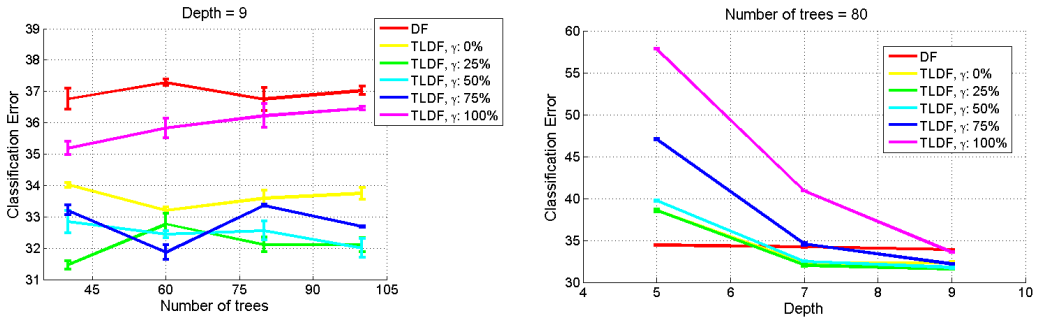


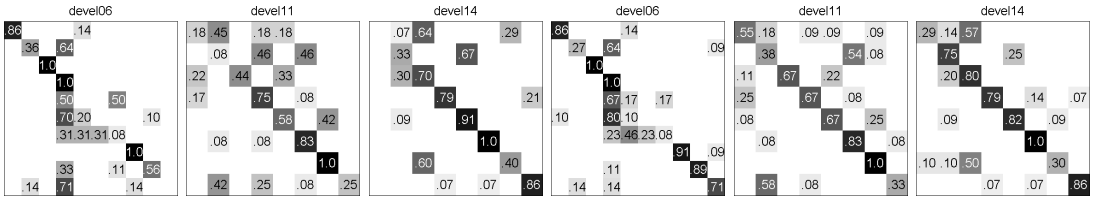
Figure 6: Comparison of the classification error using different combination of training parameters.

moves the left hand to a fixed position and then shows a similar pattern of the fingers, for all these gestures. The frames of these gestures are usually found in the same leaf after training the decision forest.

5.1.2 Devel and Final Data

Table 2 compares our results for the development batches of the ChaLearn Challenge with the ones previously reported by Lui (2012) and Malgireddy et al. (2013), using the evaluation procedure of the ChaLearn competition (Guyon et al., 2012). To train the TLDFs, we set the number of trees $T = 50$, the maximum depth $D = 8$, the mixing coefficient $\gamma = 25\%$, and the size of the search space $|\mathcal{T}| = 50$. As shown in Table 2, for most batches, our transfer learning decision forests obtain improvements over the DFs, and for some batches, they obtain the smallest errors.

Table 3 compares our results for the final evaluation data with the final results of the ChaLearn competition (Guyon et al., 2013). The Joewan team proposed a novel feature which fuses RGB-D data and is invariant to scale and rotation (Wan et al., 2013). Most of the other teams have not described their approach in a publication.



(a) (b) (c) (d) (e) (f)

Figure 7: Comparison of the confusion matrices obtained using the DF (a),(b),(c) and TLDF (d),(e),(f) classifiers on the *devel*06, *devel*11 and *devel*14 batches.

Figure 8: Similar gestures in different batches. The first, second and third rows show a gesture in the *devel*06, *devel*13 and *devel*15 batches respectively. The first column shows the RGB image for a representative frame of the video, the second column shows the corresponding depth image and the last column shows the MHI.

	devel01	devel02	devel03	devel04	devel05	devel06	devel07	devel08	devel09	devel10
Principal motion	6.67%	33.33%	71.74%	24.44%	**2.17%**	43.33%	23.08%	10.11%	19.78%	56.04%
Lui (2012)	–	–	–	–	–	–	–	–	–	–
Malgireddy et al. (2013)	13.33%	35.56%	71.74%	**10.00%**	9.78%	**37.78%**	18.68%	**8.99%**	**13.19%**	**50.55%**
DF	4.44%	28.89%	65.22%	25.56%	3.26%	48.89%	19.78%	17.98%	19.78%	59.34%
TLDF	**3.89%**	**25.00%**	**62.50%**	13.89%	4.89%	45.00%	**14.29%**	10.11%	15.38%	60.99%

	devel11	devel12	devel13	devel14	devel15	devel16	devel17	devel18	devel19	devel20	Avg.
Principal motion	**29.35%**	21.35%	12.50%	39.13%	40.22%	34.48%	48.91%	44.44%	60.44%	39.56%	33.15%
Lui (2012)	–	–	–	–	–	–	–	–	–	–	**24.09%**
Malgireddy et al. (2013)	35.87%	22.47%	**9.09%**	28.26%	**21.74%**	31.03%	**30.43%**	40.00%	**49.45%**	**35.16%**	28.73%
DF	42.39%	23.60%	19.32%	45.65%	26.09%	31.03%	53.26%	40.00%	60.44%	46.15%	34.14%
TLDF	39.13%	**19.10%**	25.00%	**27.71%**	31.52%	**27.01%**	45.11%	**38.33%**	54.95%	67.22%	31.55%

Table 2: Comparison of reported results using the Levenshtein distance.

Team	Private score set on final set #1	For comparison score on final set #2
alfnie	0.0734	0.0710
Joewan	0.1680	0.1448
Turtle Tamers	0.1702	0.1098
Wayne Zhang	0.2303	0.1846
Manavender	0.2163	0.1608
HIT CS	0.2825	0.2008
Vigilant	0.2809	0.2235
Our Method	0.2834	0.2475
Baseline method 2	0.2997	0.3172

Table 3: ChaLearn results of round 2.

	1/-1	2/-2	3/-3	4/-4	5/-5	6/-6
Adaboost (Faddoul et al., 2012)	91.77±1.89%	83.14±2.35%	82.96±1.24%	83.98±1.41%	78.42±0.69%	88.95±1.60%
MTL (Quadrianto et al., 2010)	96.80±1.91%	69.95±2.68%	74.18±5.54%	71.76±5.47%	57.26±2.72%	80.54±4.53%
MT-Adaboost (Faddoul et al., 2012)	96.80±0.56%	86.87±0.68%	87.68±1.04%	**90.38±0.71%**	84.25±0.73%	92.88±0.90%
Our approach	**97.23±0.44%**	**96.74±0.31%**	**93.29±0.96%**	90.10±1.23%	**92.79±1.62%**	**97.35±0.45%**

	7/-7	8/-8	9/-9	0/-0	Avg.
Adaboost (Faddoul et al., 2012)	87.11±0.90%	77.51±1.90%	81.84±1.85%	93.66±1.29%	84.93%
MTL (Quadrianto et al., 2010)	77.18±9.43%	65.85±2.50%	65.38±6.09%	97.81±1.01%	75.67%
MT-Adaboost (Faddoul et al., 2012)	92.81±0.57%	85.28±1.73%	**86.90±1.26%**	97.14±0.42%	90.10%
Our approach	**95.55±1.39%**	**91.99±1.30%**	84.76±1.67%	**98.05±0.28%**	**93.78%**

Table 4: Comparison of the accuracies on the MNIST data set.

5.2 MNIST

The MNIST (LeCun et al., 1998) data set has been used to compare transfer learning results (Quadrianto et al., 2010; Faddoul et al., 2012). A small sample of the training set is used to simulate the situation when only a limited number of labeled examples is available. For each digit $0 \dots 9$, we consider a binary task where label $+1$ means that the example belongs to the digit associated with the respective task, and label -1 means the opposite. We randomly choose 100 training samples for each task and test them on the $10,000$ testing samples. The experiments are repeated ten times and the results are summarized in Table 4. We train the TLDFs with $D = 6, T = 40, \gamma = 50\%$, and we do not apply any preprocessing to the sample images. The experiments show that our approach achieves better results than state-of-the-art methods in terms of transfer learning.

To analyze the influence of the number of training samples, we compare the classification error of the TLDFs with the classification error of the DFs. Figure 9 plots the classification error as a function of the number of training samples for each classifier. As we did previously, we compute the classification error using the 10000 test samples of the MNIST data set. We see that the classification error of the TLDF is smaller than that of the DF. In addition, it is interesting to note that the gap between both classifiers is larger when the number of training samples is smaller, thus suggesting that the TLDF is more suitable than DF for small training samples.

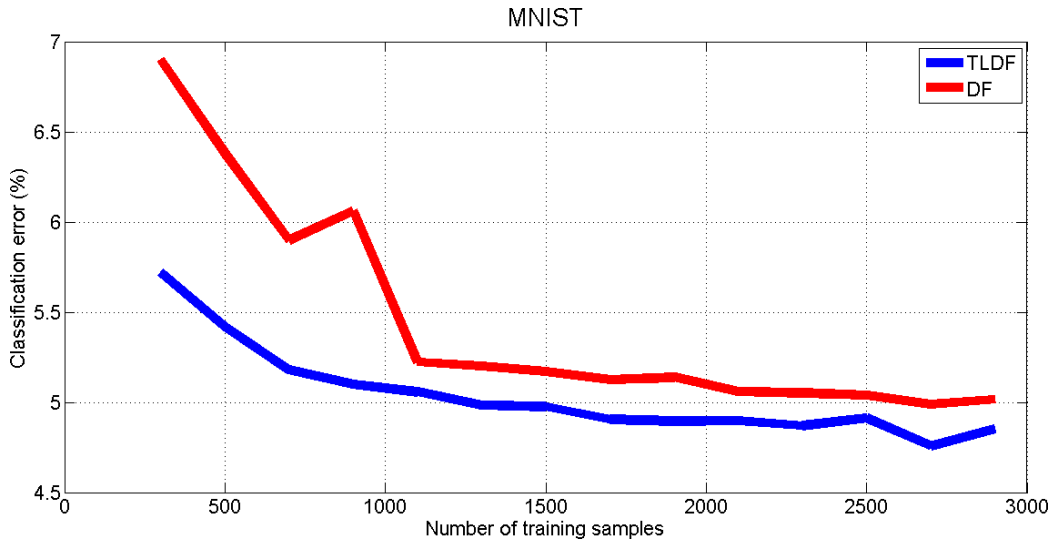


Figure 9: This figure evaluates the classification error as a function of the number of training samples.

6. Conclusions

In this paper we have introduced a novel algorithm to transfer knowledge from multiple source tasks to a given target task. The result is a classifier that can exploit the knowledge from similar tasks to improve the predictive performance on the target task. Two extensions were made to the decision forest framework in order to extract knowledge from the source tasks. We showed that both extensions are important in order to obtain smaller classification errors. The major improvements are obtained when there are only a few training samples.

We have applied the algorithm to two important computer vision problems and the results show that the proposed algorithm outperforms decision forests (which are a state-of-the-art method). We believe that transfer learning algorithms will be an essential component of many computer vision problems.

Acknowledgements

We would like to thank Zicheng Liu and Julio Jacobo-Berlles for their feedback and assistance.

Appendix A

We prove Theorem 1. First, we prove $\mathbb{E}(\mathcal{H}(S_K)) + \sum_{\mathbf{y}\in\mathcal{Y}} p_{\mathbf{y}} \log\left(1 + \frac{1-p_{\mathbf{y}}}{K p_{\mathbf{y}}}\right) \leq \mathcal{H}(P)$

By definition of the empirical entropy and linearity of the expectation, we have:

$$\mathbb{E}(\mathcal{H}(S_K)) = -\mathbb{E}\left[\sum_{\mathbf{y}\in\mathcal{Y}} \hat{p}_{S_K}(\mathbf{y}) \log(\hat{p}_{S_K}(\mathbf{y}))\right] = -\sum_{\mathbf{y}\in\mathcal{Y}} \mathbb{E}\left[\hat{p}_{S_K}(\mathbf{y}) \log(\hat{p}_{S_K}(\mathbf{y}))\right]$$

Using the definitions of the empirical histogram $\hat{p}_{S_K}(\mathbf{y})$ and the expectation:

$$-\sum_{\mathbf{y}\in\mathcal{Y}} \mathbb{E}\left[\hat{p}_{S_K}(\mathbf{y}) \log(\hat{p}_{S_K}(\mathbf{y}))\right] = -\sum_{\mathbf{y}\in\mathcal{Y}} \sum_{j=0}^{K} P\left(\hat{p}_{S_K}(\mathbf{y}) = \frac{j}{K}\right) \frac{j}{K} \log \frac{j}{K}$$

Assuming that the samples are iid, then:

$$= -\sum_{\mathbf{y}\in\mathcal{Y}} \sum_{j=0}^{K} \binom{K}{j} p_{\mathbf{y}}^{j} (1-p_{\mathbf{y}})^{K-j} \frac{j}{K} \log \frac{j}{K}$$

Note that, in this equation, $p_{\mathbf{y}}$ is the true probability of distribution P. After some algebraic manipulations, we obtain the following:

$$= -\sum_{\mathbf{y}\in\mathcal{Y}} p_{\mathbf{y}} \sum_{j=0}^{K-1} \binom{K-1}{j} p_{\mathbf{y}}^{j} (1-p_{\mathbf{y}})^{K-1-j} \log \frac{j+1}{K}$$

$$= -\sum_{\mathbf{y}\in\mathcal{Y}} p_{\mathbf{y}} \sum_{j=0}^{K-1} P\left(\hat{p}_{S_K}(\mathbf{y}) = \frac{j}{K}\right) \log \frac{j+1}{K}$$

Applying Jensen's inequality for the convex function $-\log(x)$, we obtain:

$$\geq -\sum_{\mathbf{y}\in\mathcal{Y}} p_{\mathbf{y}} \log \left(\sum_{j=0}^{K-1} P\left(\hat{p}_{S_K}(\mathbf{y}) = \frac{j}{K}\right) \frac{j+1}{K} \right)$$

$$= -\sum_{\mathbf{y}\in\mathcal{Y}} p_{\mathbf{y}} \log \frac{(K-1)p_{\mathbf{y}} + 1}{K}$$

$$= -\sum_{\mathbf{y}\in\mathcal{Y}} p_{\mathbf{y}} \log \left(p_{\mathbf{y}} + \frac{1-p_{\mathbf{y}}}{K} \right)$$

$$= -\sum_{\mathbf{y}\in\mathcal{Y}} p_{\mathbf{y}} \log \left(p_{\mathbf{y}} \left(1 + \frac{1-p_{\mathbf{y}}}{K p_{\mathbf{y}}} \right) \right)$$

$$= -\sum_{\mathbf{y}\in\mathcal{Y}} p_{\mathbf{y}} \log p_{\mathbf{y}} - \sum_{\mathbf{y}\in\mathcal{Y}} p_{\mathbf{y}} \log \left(1 + \frac{1-p_{\mathbf{y}}}{K p_{\mathbf{y}}} \right)$$

$$= \mathcal{H}(P) - \sum_{\mathbf{y}\in\mathcal{Y}} p_{\mathbf{y}} \log \left(1 + \frac{1-p_{\mathbf{y}}}{K p_{\mathbf{y}}} \right)$$

Now we prove $\mathcal{H}(P) \leq \mathbb{E}(\mathcal{H}(S_K))$.

By definition of the empirical entropy and linearity of the expectation, we have:

$$\mathbb{E}(\mathcal{H}(S_K)) = -\mathbb{E}\left[\sum_{\mathbf{y}\in\mathcal{Y}} \hat{p}_{S_K}(\mathbf{y}) \log(\hat{p}_{S_K}(\mathbf{y})) \right] = -\sum_{\mathbf{y}\in\mathcal{Y}} \mathbb{E}\left[\hat{p}_{S_K}(\mathbf{y}) \log(\hat{p}_{S_K}(\mathbf{y}))\right]$$

Applying Jensen's inequality for the convex function $x \log x$, we obtain the following:

$$\leq -\sum_{\mathbf{y} \in \mathcal{Y}} \mathbb{E}\left[\hat{p}_{S_K}(\mathbf{y})\right] \log(\mathbb{E}\left[\hat{p}_{S_K}(\mathbf{y})\right])$$

Since $\mathbb{E}\left[\hat{p}_{S_K}(\mathbf{y})\right] = p_{\mathbf{y}}$, we have:

$$= -\sum_{\mathbf{y} \in \mathcal{Y}} p_{\mathbf{y}} \log(p_{\mathbf{y}}) = \mathcal{H}(P)$$

References

M. A. R. Ahad, J. K. Tan, H. Kim, and S. Ishikawa. Motion history image: its variants and applications. *Machine Vision and Applications*, 23(2):255–281, 2012.

Y. Aytar and A. Zisserman. Tabula rasa: model transfer for object category detection. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, 2011.

E. Bart and S. Ullman. Cross-generalization: learning novel classes from a single example by feature replacement. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, 2005.

A. Bergamo and L. Torresani. Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 181–189, 2010.

A. Bobick and J. Davis. An appearance-based representation of action. In *Proceedings of the International Conference on Pattern Recognition*, pages 307–312, 1996.

A. Bobick and J. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine*, 2001.

R. Bowden, D. Windridge, T. Kadir, A. Zisserman, and J. M. Brady. A linguistic feature vector for the visual interpretation of sign language. In *Proceedings of the European Conference on Computer Vision*, 2004.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2006.

A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: a unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2-3):81–227, 2012.

W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *Proceedings of the International Conference on Machine Learning*, pages 193–200, New York, NY, USA, 2007.

J. B. Faddoul, B. Chidlovskii, R. Gilleron, and F. Torre. Learning multiple tasks with boosted decision trees. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, 2012.

A. Farhadi, D. Forsyth, and R. White. Transfer learning in sign language. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 1–8, 2007.

L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine*, 28(4):594–611, 2006.

P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine*, 32(9):1627–1645, 2010.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: an unsupervised approach. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 999–1006, 2011.

E. Grosicki and H. E. Abed. ICDAR 2011 - French handwriting recognition competition. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 1459–1463, 2011.

I. Guyon, V. Athitsos, P. Jangyodsuk, B. Hamner, and H. J. Escalante. ChaLearn gesture challenge: design and first results. In *Workshop on Gesture Recognition and Kinect Demonstration Competition*, 2012.

I. Guyon, V. Athitsos, P. Jangyodsuk, H. J. Escalante, and B. Hamner. Results and analysis of the chalearn gesture challenge 2012. In *Advances in Depth Image Analysis and Applications*, volume 7854 of *Lecture Notes in Computer Science*, pages 186–204. Springer, 2013.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2003.

A. Kurakin, Z. Zhang, and Z. Liu. A real-time system for dynamic hand gesture recognition with a depth sensor. In *Proceedings of the European Signal Processing Conference*, pages 1980–1984, 2012.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

C. Leistner, A. Saffari, J. Santner, and H. Bischof. Semi-supervised random forests. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 506–513, 2009.

K. Levi, M. Fink, and Y. Weiss. Learning from a small number of training examples by exploiting object categories. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Workshop*, pages 96–102, 2004.

W. Li, Z. Zhang, and Z. Liu. Graphical modeling and decoding of human actions. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*, pages 175–180, 2008.

J. J. Lim, R. Salakhutdinov, and A. Torralba. Transfer learning by borrowing examples for multiclass object detection. In *Proceedings of the Advances in Neural Information Processing Systems*, 2011.

J. Liu, K. Yu, Y. Zhang, and Y. Huang. Training conditional random fields using transfer learning for gesture recognition. In *Proceedings of the IEEE International Conference on Data Mining*, pages 314 – 323, 2010.

Y. M. Lui. Human gesture recognition on product manifolds. *Journal of Machine Learning Research*, 13:3297–3321, Nov 2012.

M. R. Malgireddy, I. Nwogu, and V. Govindaraju. Language-motivated approaches to action recognition. *Journal of Machine Learning Research*, 14:2189–2212, 2013.

S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010. ISSN 1041-4347.

Y. Pei, T.-K. Kim, and H. Zha. Unsupervised random forest manifold alignment for lipreading. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013.

N. Quadrianto, A. J. Smola, T. Caetano, S. Vishwanathanand, and J. Petterson. Multitask learning without label correspondences. In *Proceedings of the Advances in Neural Information Processing Systems*, 2010.

A. Quattoni, M. Collins, and T. Darrell. Transfer learning for image classification with sparse prototype representations. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 1 – 8, 2008.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *Proceedings of the European Conference on Computer Vision*, pages 213–226, 2010.

H. J. Seo and P. Milanfar. Action recognition from one example. *IEEE Transactions on Pattern Analysis and Machine*, 33(5):867–882, 2011.

T. Sharp. Implementing decision trees and forests on a GPU. In *Proceedings of the European Conference on Computer Vision*, pages 595–608, 2008.

E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky. Learning hierarchical models of scenes, objects, and parts. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1331–1338, 2005.

T. Tommasi, F. Orabona, and B. Caputo. Learning categories from few examples with multi model knowledge transfer. *IEEE Transactions on Pattern Analysis and Machine*, 36(5):928–941, 2014.

A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine*, 29(5): 854–869, 2007.

P. A. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

J. Wan, Q. Ruan, W. Li, and S. Deng. One-shot learning gesture recognition from RGB-D data using bag of features. *Journal of Machine Learning Research*, 14:2549–2582, 2013.

Q. Wang, L. Zhang, M. Chi, and J. Guo. MTForest: ensemble decision trees based on multi-task learning. In *Proceedings of the European Conference on Artificial Intelligence*, pages 122–126, 2008.

J. won Lee and C. Giraud-Carrier. Transfer learning in decision trees. In *Proceedings of the International Joint Conference on Neural Networks*, 2007.

Y. Yao and G. Doretto. Boosting for transfer learning with multiple sources. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 1855 – 1862, 2010.

Journal of Machine Learning Research 15 (2014) 3871-3914 Submitted 4/13; Revised 4/14; Published 11/14

Semi-Supervised Eigenvectors for Large-Scale Locally-Biased Learning

Toke J. Hansen TJHA@DTU.DK
Department of Applied Mathematics and Computer Science
Technical University of Denmark
Richard Petersens Plads, 2800 Lyngby, Denmark

Michael W. Mahoney MMAHONEY@STAT.BERKELEY.EDU
International Computer Science Institute and Dept. of Statistics
University of California
Berkeley, CA 94720-1776, USA

Editor: Mikhail Belkin

Abstract

In many applications, one has side information, e.g., labels that are provided in a semi-supervised manner, about a specific target region of a large data set, and one wants to perform machine learning and data analysis tasks "nearby" that prespecified target region. For example, one might be interested in the clustering structure of a data graph near a prespecified "seed set" of nodes, or one might be interested in finding partitions in an image that are near a prespecified "ground truth" set of pixels. Locally-biased problems of this sort are particularly challenging for popular eigenvector-based machine learning and data analysis tools. At root, the reason is that eigenvectors are inherently global quantities, thus limiting the applicability of eigenvector-based methods in situations where one is interested in very local properties of the data.

In this paper, we address this issue by providing a methodology to construct *semi-supervised eigenvectors* of a graph Laplacian, and we illustrate how these locally-biased eigenvectors can be used to perform *locally-biased machine learning*. These semi-supervised eigenvectors capture successively-orthogonalized directions of maximum variance, conditioned on being well-correlated with an input seed set of nodes that is assumed to be provided in a semi-supervised manner. We show that these semi-supervised eigenvectors can be computed quickly as the solution to a system of linear equations; and we also describe several variants of our basic method that have improved scaling properties. We provide several empirical examples demonstrating how these semi-supervised eigenvectors can be used to perform locally-biased learning; and we discuss the relationship between our results and recent machine learning algorithms that use global eigenvectors of the graph Laplacian.

Keywords: semi-supervised learning, spectral clustering, kernel methods, large-scale machine learning, local spectral methods, locally-biased learning

1. Introduction

In many applications, one has information about a specific target region of a large data set, and one wants to perform common machine learning and data analysis tasks "nearby" the pre-specified target region. In such situations, eigenvector-based methods such as those

that have been popular in machine learning in recent years tend to have serious difficulties. At root, the reason is that eigenvectors, e.g., of Laplacian matrices of data graphs, are inherently *global* quantities, and thus they might not be sensitive to very *local* information. Motivated by this, we consider the problem of finding a set of locally-biased vectors—we will call them *semi-supervised eigenvectors*—that inherit many of the "nice" properties that the leading nontrivial global eigenvectors of a graph Laplacian have—for example, that capture "slowly varying" modes in the data, that are fairly-efficiently computable, that can be used for common machine learning and data analysis tasks such as kernel-based and semi-supervised learning, etc.—so that we can perform what we will call *locally-biased machine learning* in a principled manner.

1.1 Locally-Biased Learning

By *locally-biased machine learning*, we mean that we have a data set, e.g., represented as a graph, and that we have information, e.g., given in a semi-supervised manner, that certain "regions" of the data graph are of particular interest. In this case, we may want to focus predominantly on those regions and perform data analysis and machine learning, e.g., classification, clustering, ranking, etc., that is "biased toward" those pre-specified regions. Examples of this include the following.

- *Locally-biased community identification.* In social and information network analysis, one might have a small "seed set" of nodes that belong to a cluster or community of interest (Andersen and Lang, 2006; Leskovec et al., 2008); in this case, one might want to perform link or edge prediction, or one might want to "refine" the seed set in order to find other nearby members.

- *Locally-biased image segmentation.* In computer vision, one might have a large corpus of images along with a "ground truth" set of pixels as provided by a face detection algorithm (Eriksson et al., 2007; Mahoney et al., 2012; Maji et al., 2011); in this case, one might want to segment entire heads from the background for all the images in the corpus in an automated manner.

- *Locally-biased neural connectivity analysis.* In functional magnetic resonance imaging applications, one might have small sets of neurons that "fire" in response to some external experimental stimulus (Norman et al., 2006); in this case, one might want to analyze the subsequent temporal dynamics of stimulation of neurons that are "nearby," either in terms of connectivity topology or functional response, members of the original set.

In each of these examples, the data are modeled by a graph—which is either "given" from the application domain or is "constructed" from feature vectors obtained from the application domain—and one has information that can be viewed as semi-supervised in the sense that it consists of exogeneously-specified "labels" for the nodes of the graph. In addition, there are typically a relatively-small number of labels and one is interested in obtaining insight about the data graph nearby those labels.

These examples present considerable challenges for standard global spectral techniques and other traditional eigenvector-based methods. (Such eigenvector-based methods have received attention in a wide range of machine learning and data analysis applications in recent

years. They have been useful, for example, in non-linear dimensionality reduction Belkin and Niyogi 2003; Coifman et al. 2005; in kernel-based machine learning Schölkopf and Smola 2001; in Nyström-based learning methods Williams and Seeger 2001; Talwalkar and Rostamizadeh 2010; spectral partitioning Pothen et al. 1990; Shi and Malik 2000; Ng et al. 2001, and so on.) At root, the reason is that eigenvectors are inherently global quantities, thus limiting their applicability in situations where one is interested in very local properties of the data. That is, very local information can be "washed out" and essentially invisible to these globally-optimal vectors. For example, a sparse cut in a graph may be poorly correlated with the second eigenvector and thus invisible to a method based only on eigenvector analysis. Similarly, if one has semi-supervised information about a specific target region in the graph, as in the above examples, one might be interested in finding clusters near this prespecified local region in a semi-supervised manner; but this local region might be essentially invisible to a method that uses only global eigenvectors. Finally, one might be interested in using kernel-based methods to find "local correlations" or to regularize with respect to a "local dimensionality" in the data, but this local information might be destroyed in the process of constructing kernels with traditional kernel-based methods.

1.2 Semi-Supervised Eigenvectors

In this paper, we provide a methodology to construct what we will call *semi-supervised eigenvectors* of a graph Laplacian; and we illustrate how these locally-biased eigenvectors (locally-biased in the sense that they will be well-correlated with the input seed set of nodes or that most of their "mass" will be on nodes that are "near" that seed set) inherit many of the properties that make the leading nontrivial global eigenvectors of the graph Laplacian so useful in applications. In order to make this method useful, there should ideally be a "knob" that allows us to interpolate between very local and the usual global eigenvectors, depending on the application at hand; we should be able to use these vectors in common machine learning pipelines to perform common machine learning tasks; and the intuitions that make the leading k nontrivial global eigenvectors of the graph Laplacian useful should, to the extent possible, extend to the locally-biased setting. To achieve this, we will formulate an optimization ansatz that is a variant of the usual global spectral graph partitioning optimization problem that includes a natural locality constraint as well as an orthogonality constraint, and we will iteratively solve this problem.

In more detail, assume that we are given as input a (possibly weighted) data graph $G = (V, E)$, an indicator vector s of a small "seed set" of nodes, a *correlation parameter* $\kappa \in [0, 1]$, and a positive integer k. Then, informally, we would like to construct k vectors that satisfy the following bicriteria: first, each of these k vectors is well-correlated with the input seed set; and second, those k vectors describe successively-orthogonalized directions of maximum variance, in a manner analogous to the leading k nontrivial global eigenvectors of the graph Laplacian. (We emphasize that the seed set s of nodes, the integer k, and the correlation parameter κ are part of the input; and thus they should be thought of as being available in a semi-supervised manner.) Somewhat more formally, our main algorithm, Algorithm 1 in Section 3, returns as output k semi-supervised eigenvectors; each of these is the solution to an optimization problem of the form of Generalized LocalSpectral in Figure 1, and thus each "captures" (say) κ/k of the correlation with the seed set. Our

main theoretical result, described in Section 3, states that these vectors define successively-orthogonalized directions of maximum variance, conditioned on being κ/k-well-correlated with an input seed set s; and that each of these k semi-supervised eigenvectors can be computed quickly as the solution to a system of linear equations. To extend the practical applicability of this basic result, we will in Section 4 describe several heuristic extensions of our basic framework that will make it easier to apply the method of semi-supervised eigenvectors at larger size scales. These extensions involve using the so-called Nyström method, computing one locally-biased eigenvector and iteratively "peeling off" successive components of interest, as well as performing random walks that are "local" in a stronger sense than our basic method considers.

Finally, in order to illustrate how the method of semi-supervised eigenvectors performs in practice, we also provide a detailed empirical evaluation using a wide range of both small-scale as well as larger-scale data.

1.3 Related Work

From a technical perspective, the work most closely related to ours is the recently-developed "local spectral method" of Mahoney et al. (2012). The original algorithm of Mahoney et al. (2012) introduced a methodology to construct a locally-biased version of the *leading* non-trivial eigenvector of a graph Laplacian and also showed (theoretically and empirically in a social network analysis application) that that the resulting vector could be used to partition a graph in a locally-biased manner. From this perspective, our extension incorporates a natural orthogonality constraint that successive vectors need to be orthogonal to previous vectors. Subsequent to the work of Mahoney et al. (2012), Maji et al. (2011) applied the algorithm of Mahoney et al. (2012) to the problem of finding locally-biased cuts in a computer vision application. Similar ideas have also been applied somewhat differently. For example, Andersen and Lang (2006) use locally-biased random walks, e.g., short random walks starting from a small seed set of nodes, to find clusters and communities in graphs arising in Internet advertising applications; Leskovec et al. (2008) used locally-biased random walks to characterize the local and global clustering structure of a wide range of social and information networks; and Joachims (2003) developed the Spectral Graph Transducer, which performs transductive learning via spectral graph partitioning.

The objectives in both (Joachims, 2003) and (Mahoney et al., 2012) are constrained eigenvalue problems that can be solved by finding the smallest eigenvalue of an asymmetric generalized eigenvalue problem; but in practice this procedure can be highly unstable (Gander et al., 1989). The algorithm of Joachims (2003) reduces the instabilities by performing all calculations in a subspace spanned by the d smallest eigenvectors of the graph Laplacian; whereas the algorithm of Mahoney et al. (2012) performs a binary search, exploiting the monotonic relationship between a control parameter and the corresponding Lagrange multiplier. The form of our optimization problem also has similarities to other work in computer vision applications: e.g., (Yu and Shi, 2002) and (Eriksson et al., 2007) find good conductance clusters subject to a set of linear constraints.

In parallel, Belkin and Niyogi (2003) and a large body of subsequent work including (Coifman et al., 2005) used (the usual global) eigenvectors of the graph Laplacian to perform dimensionality reduction and data representation, in unsupervised and semi-supervised

settings (Tenenbaum et al., 2000; Roweis and Saul, 2000; Zhou et al., 2004). Typically, these methods construct some sort of local neighborhood structure around each data point, and they optimize some sort of global objective function to go "from local to global" (Saul et al., 2006). In some cases, these methods can be understood in terms of data drawn from an hypothesized manifold (Belkin and Niyogi, 2008), and more generally they have proven useful for denoising and learning in semi-supervised settings (Belkin and Niyogi, 2004; Belkin et al., 2006). These methods are based on spectral graph theory (Chung, 1997); and thus many of these methods have a natural interpretation in terms of diffusions and kernel-based learning (Schölkopf and Smola, 2001; Kondor and Lafferty, 2002; Szummer and Jaakkola, 2002; Chapelle et al., 2003; Ham et al., 2004). These interpretations are important for the usefulness of these global eigenvector methods in a wide range of applications. As we will see, many (but not all) of these interpretations can be ported to the "local" setting, an observation that was made previously in a different context (Cucuringu and Mahoney, 2011).

Many of these diffusion-based spectral methods also have a natural interpretation in terms of spectral ranking (Vigna, 2009). "Topic sensitive" and "personalized" versions of these spectral ranking methods have also been studied (Haveliwala, 2003; Jeh and Widom, 2003); and these were the motivation for diffusion-based methods to find locally-biased clusters in large graphs (Spielman and Teng, 2004; Andersen et al., 2006; Mahoney et al., 2012). Our optimization ansatz is a generalization of the linear equation formulation of the PageRank procedure (Page et al., 1999; Mahoney et al., 2012; Vigna, 2009); and its solution involves Laplacian-based linear equation solving, which has been suggested as a primitive is of more general interest in large-scale data analysis (Teng, 2010).

1.4 Outline of the Paper

In the next section, Section 2, we will provide notation and some background and discuss related work. Then, in Section 3 we will present our main algorithm and our main theoretical result justifying the algorithm; and in Section 4 we will present several extensions of our basic method that will help for certain larger-scale applications of the method of semi-supervised eigenvectors. In Section 5, we present an empirical analysis, including both toy data to illustrate how the "knobs" of our method work, as well as applications to realistic machine learning and data analysis problems.

2. Background and Notation

Let $G = (V, E, w)$ be a connected undirected graph with $n = |V|$ vertices and $m = |E|$ edges, in which edge $\{i, j\}$ has weight w_{ij}. For a set of vertices $S \subseteq V$ in a graph, the *volume of* S is $\mathrm{vol}(S) \stackrel{\mathrm{def}}{=} \sum_{i \in S} d_i$, in which case the *volume of the graph* G is $\mathrm{vol}(G) \stackrel{\mathrm{def}}{=} \mathrm{vol}(V) = 2m$. In the following, $A_G \in \mathbb{R}^{V \times V}$ will denote the adjacency matrix of G, while $D_G \in \mathbb{R}^{V \times V}$ will denote the diagonal degree matrix of G, i.e., $D_G(i,i) = d_i = \sum_{\{i,j\} \in E} w_{ij}$, the weighted degree of vertex i. The Laplacian of G is defined as $L_G \stackrel{\mathrm{def}}{=} D_G - A_G$. (This is also called the combinatorial Laplacian, in which case the normalized Laplacian of G is $\mathcal{L}_G \stackrel{\mathrm{def}}{=} D_G^{-1/2} L_G D_G^{-1/2}$.)

The Laplacian is the symmetric matrix having quadratic form $x^T L_G x = \sum_{ij \in E} w_{ij}(x_i - x_j)^2$, for $x \in \mathbb{R}^V$. This implies that L_G is positive semidefinite and that the all-one vector $1 \in \mathbb{R}^V$ is the eigenvector corresponding to the smallest eigenvalue 0. The generalized eigenvalues of $L_G x = \lambda_i D_G x$ are $0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_N$. We will use v_2 to denote smallest non-trivial eigenvector, i.e., the eigenvector corresponding to λ_2; v_3 to denote the next eigenvector; and so on. We will overload notation to use $\lambda_2(A)$ to denote the smallest non-zero generalized eigenvalue of A with respect to D_G. Finally, for a matrix A, let A^+ denote its (uniquely defined) Moore-Penrose pseudoinverse. For two vectors $x, y \in \mathbb{R}^n$, and the degree matrix D_G for a graph G, we define the *degree-weighted inner product* as $x^T D_G y \stackrel{\text{def}}{=} \sum_{i=1}^n x_i y_i d_i$. In particular, if a vector x has unit norm, then $x^T D_G x = 1$. Given a subset of vertices $S \subseteq V$, we denote by 1_S the indicator vector of S in $\mathbb{R}^V$ and by 1 the vector in $\mathbb{R}^V$ having all entries set equal to 1.

3. Optimization Approach to Semi-Supervised Eigenvectors

In this section, we provide our main technical results: a motivation and statement of our optimization ansatz; our main algorithm for computing semi-supervised eigenvectors; and an analysis that our algorithm computes solutions of our optimization ansatz.

3.1 Motivation for the Program

Recall the optimization perspective on how one computes the leading nontrivial global eigenvectors of the normalized Laplacian $\mathcal{L}_G$ or, equivalently, of the leading nontrivial generalized eigenvectors of L_G. The first nontrivial eigenvector v_2 is the solution to the problem GlobalSpectral that is presented on the left of Figure 1. Equivalently, although GlobalSpectral is a non-convex optimization problem, strong duality holds for it and it's solution may be computed as v_2, the leading nontrivial generalized eigenvector of L_G. (In this case, the value of the objective is λ_2, and global spectral partitioning involves then doing a "sweep cut" over this vector and appealing to Cheeger's inequality.) The next eigenvector v_3 is the solution to GlobalSpectral, augmented with the constraint that $x^T D_G v_2 = 0$; and in general the t^{th} generalized eigenvector of L_G is the solution to GlobalSpectral, augmented with the constraints that $x^T D_G v_i = 0$, for $i \in \{2, \ldots, t-1\}$. Clearly, this set of constraints and the constraint $x^T D_G 1 = 0$ can be written as $x^T D_G X = 0$, where 0 is a $(t-1)$-dimensional all-zeros vector, and where X is an $n \times (t-1)$ orthogonal matrix whose i^{th} column equals v_i (where $v_1 = 1$, the all-ones vector, is the first column of X).

Also presented in Figure 1 is LocalSpectral, which includes a constraint that the solution be well-correlated with an input seed set. This LocalSpectral optimization problem was introduced in Mahoney et al. (2012), where it was shown that the solution to LocalSpectral may be interpreted as a locally-biased version of the second eigenvector of the Laplacian.[1] In particular, although LocalSpectral is not convex, it's solution can be computed efficiently as the solution to a set of linear equations that generalize the popular

1. In Mahoney et al. (2012), the locality constraint was actually a quadratic constraint, and thus a somewhat involved analysis was required. In retrospect, given the form of the solution, and in light of the discussion below, it is clear that the quadratic part was not "real," and thus we present this simpler form of LocalSpectral here. This should make the connections with our Generalized LocalSpectral objective more immediate.

GlobalSpectral	LocalSpectral	Generalized LocalSpectral
$\begin{aligned} \text{minimize} \quad & x^T L_G x \\ \text{s.t} \quad & x^T D_G x = 1 \\ & x^T D_G 1 = 0 \end{aligned}$	$\begin{aligned} \text{minimize} \quad & x^T L_G x \\ \text{s.t} \quad & x^T D_G x = 1 \\ & x^T D_G 1 = 0 \\ & x^T D_G s \geq \sqrt{\kappa} \end{aligned}$	$\begin{aligned} \text{minimize} \quad & x^T L_G x \\ \text{s.t} \quad & x^T D_G x = 1 \\ & x^T D_G X = 0 \\ & x^T D_G s \geq \sqrt{\kappa} \end{aligned}$

Figure 1: Left: The usual GlobalSpectral partitioning optimization problem; the vector achieving the optimal solution is v_2, the leading nontrivial generalized eigenvector of L_G with respect to D_G. Middle: The LocalSpectral optimization problem, which was originally introduced in Mahoney et al. (2012); for $\kappa = 0$, this coincides with the usual global spectral objective, while for $\kappa > 0$, this produces solutions that are biased toward the seed vector s. Right: The Generalized LocalSpectral optimization problem we introduce that includes both the locality constraint and a more general orthogonality constraint. Our main algorithm for computing semi-supervised eigenvectors will iteratively compute the solution to Generalized LocalSpectral for a sequence of X matrices. In all three cases, the optimization variable is $x \in \mathbb{R}^n$.

Personalized PageRank procedure; in addition, by performing a sweep cut and appealing to a variant of Cheeger's inequality, this locally-biased eigenvector can be used to perform locally-biased spectral graph partitioning (Mahoney et al., 2012).

3.2 Our Main Algorithm

We will formulate the problem of computing semi-supervised vectors in terms of a primitive optimization problem of independent interest. Consider the Generalized LocalSpectral optimization problem, as shown in Figure 1. For this problem, we are given a graph $G = (V, E)$, with associated Laplacian matrix L_G and diagonal degree matrix D_G; an indicator vector s of a small "seed set" of nodes; a *correlation parameter* $\kappa \in [0, 1]$; and an $n \times \nu$ constraint matrix X that may be assumed to be an orthogonal matrix. We will assume (without loss of generality) that s is properly normalized and orthogonalized so that $s^T D_G s = 1$ and $s^T D_G 1 = 0$. While s can be a general unit vector orthogonal to 1, it may be helpful to think of s as the indicator vector of one or more vertices in V, corresponding to the target region of the graph.

In words, the problem Generalized LocalSpectral asks us to find a vector $x \in \mathbb{R}^n$ that minimizes the variance $x^T L_G x$ subject to several constraints: that x is unit length; that x is orthogonal to the span of X; and that x is $\sqrt{\kappa}$-well-correlated with the input seed set vector s. In our application of Generalized LocalSpectral to the computation of semi-supervised eigenvectors, we will iteratively compute the solution to Generalized LocalSpectral, updating X to contain the already-computed semi-supervised eigenvectors. That is, to compute the first semi-supervised eigenvector, we let $X = 1$, i.e., the

n-dimensional all-ones vector, which is the trivial eigenvector L_G, in which case X is an $n \times 1$ matrix; and to compute each subsequent semi-supervised eigenvector, we let the columns of X consist of 1 and the other semi-supervised eigenvectors found in each of the previous iterations.

To show that Generalized LocalSpectral is efficiently-solvable, note that it is a quadratic program with only one quadratic constraint and one linear equality constraint.[2] In order to remove the equality constraint, which will simplify the problem, let's change variables by defining the $n \times (n-\nu)$ matrix F as

$$\{x : X^T D_G x = 0\} = \{x : x = F\hat{x}\}.$$

That is, F is a span for the null space of X^T; and we will take F to be an orthogonal matrix. In particular, this implies that $F^T F$ is an $(n-\nu) \times (n-\nu)$ Identity and FF^T is an $n \times n$ Projection. Then, with respect to the $\hat{x}$ variable, Generalized LocalSpectral becomes

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & \hat{x}^T F^T L_G F y \\ \text{subject to} \quad & \hat{x}^T F^T D_G F \hat{x} = 1, \\ & \hat{x}^T F^T D_G s \geq \sqrt{\kappa}. \end{aligned} \tag{1}$$

In terms of the variable x, the solution to this optimization problem is of the form

$$\begin{aligned} x^* &= cF \left(F^T \left(L_G - \gamma D_G\right) F\right)^+ F^T D_G s \\ &= c \left(FF^T \left(L_G - \gamma D_G\right) FF^T\right)^+ D_G s, \end{aligned} \tag{2}$$

for a normalization constant $c \in (0, \infty)$ and for some γ that depends on $\sqrt{\kappa}$. The second line follows from the first since F is an $n \times (n-\nu)$ orthogonal matrix. This so-called "S-procedure" is described in greater detail in Chapter 5 and Appendix B of (Boyd and Vandenberghe, 2004). The significance of this is that, although it is a non-convex optimization problem, the Generalized LocalSpectral problem can be solved by solving a linear equation, in the form given in Eqn. (2).

Returning to our problem of computing semi-supervised eigenvectors, recall that, in addition to the input for the Generalized LocalSpectral problem, we need to specify a positive integer k that indicates the number of vectors to be computed. In the simplest case, we would assume that we would like the correlation to be "evenly distributed" across all k vectors, in which case we will require that each vector is $\sqrt{\kappa/k}$-well-correlated with the input seed set vector s; but this assumption can easily be relaxed, and thus Algorithm 1 is formulated more generally as taking a k-dimensional vector $\kappa = [\kappa_1, \ldots, \kappa_k]^T$ of correlation coefficients as input.

To compute the first semi-supervised eigenvector, we will let $X = 1$, the all-ones vector, in which case the first nontrivial semi-supervised eigenvector is

$$x_1^* = c \left(L_G - \gamma_1 D_G\right)^+ D_G s, \tag{3}$$

2. Alternatively, note that it is an example of an constrained eigenvalue problem (Gander et al., 1989). We thank the numerous individuals who pointed this out to us subsequent to our dissemination of the original version of this paper.

where γ_1 is chosen to saturate the part of the correlation constraint along the first direction. (Note that the projections FF^T from Eqn. 2 are not present in Eqn. 3 since by design $s^T D_G 1 = 0$.) That is, to find the correct setting of γ_1, it suffices to perform a binary search over the possible values of γ_1 in the interval $(-\mathrm{vol}(G), \lambda_2(G))$ until the correlation constraint is satisfied, that is, until $(s^T D_G x_1)^2$ is sufficiently close to κ_1.

To compute subsequent semi-supervised eigenvectors, i.e., at steps $t = 2, \ldots, k$ if one ultimately wants a total of k semi-supervised eigenvectors, then one lets X be the $n \times t$ matrix of the form

$$X = [1, x_1^*, \ldots, x_{t-1}^*], \tag{4}$$

where $x_1^*, \ldots, x_{t-1}^*$ are successive semi-supervised eigenvectors; and the projection matrix FF^T is of the form

$$FF^T = I - D_G X (X^T D_G D_G X)^{-1} X^T D_G,$$

due to the the degree-weighted inner norm.

Then, by Eqn. (2), the t^{th} semi-supervised eigenvector takes the form

$$x_t^* \quad = \quad c \left(FF^T (L_G - \gamma_t D_G) FF^T \right)^+ D_G s.$$

Algorithm 1 Main algorithm to compute semi-supervised eigenvectors

Require: $L_G, D_G, s, \kappa = [\kappa_1, \ldots, \kappa_k]^T, \epsilon$ such that $s^T D_G 1 = 0$, $s^T D_G s = 1$, $\kappa^T 1 \leq 1$
1: $X = [1]$
2: **for** $t = 1$ to k **do**
3: $\quad FF^T \leftarrow I - D_G X (X^T D_G D_G X)^{-1} X^T D_G$
4: $\quad \top \leftarrow \lambda_2$ where $FF^T L_G FF^T v_2 = \lambda_2 FF^T D_G FF^T v_2$
5: $\quad \perp \leftarrow -\mathrm{vol}(G)$
6: $\quad$ **repeat**
7: $\quad\quad \gamma_t \leftarrow (\perp + \top)/2$ (Binary search over γ_t)
8: $\quad\quad x_t \leftarrow (FF^T (L_G - \gamma_t D_G) FF^T)^+ FF^T D_G s$
9: $\quad\quad$ Normalize x_t such that $x_t^T D_G x_t = 1$
10: $\quad\quad$ **if** $(x_t^T D_G s)^2 > \kappa_t$ **then** $\perp \leftarrow \gamma_t$ **else** $\top \leftarrow \gamma_t$ **end if**
11: $\quad$ **until** $\|(x_t^T D_G s)^2 - \kappa_t\| \leq \epsilon$ **or** $\|(\perp + \top)/2 - \gamma_t\| \leq \epsilon$
12: $\quad$ Augment X with x_t^* by letting $X = [X, x_t^*]$.
13: **end for**

In more detail, Algorithm 1 presents pseudo-code for our main algorithm for computing semi-supervised eigenvectors. The algorithm takes as input a graph $G = (V, E)$, a seed set s (which could be a general vector $s \in \mathbb{R}^n$, subject for simplicity to the normalization constraints $s^T D_G 1 = 0$ and $s^T D_G s = 1$, but which is most easily thought of as an indicator vector for the local "seed set" of nodes), a number k of vectors we want to compute, and a vector of locality parameters $(\kappa_1, \ldots, \kappa_k)$, where $\kappa_i \in [0, 1]$ and $\sum_{i=1}^k \kappa_i = 1$ (where, in the simplest case, one could choose $\kappa_i = \kappa/k$, $\forall i$, for some $\kappa \in [0, 1]$). Several things should be noted about our implementation of our main algorithm. First, as we will discuss in more detail below, we compute the projection matrix FF^T only *implicitly*. Second, a naïve approach to Eqn. (2) does not immediately lead to an efficient solution, since $D_G s$ will not be in the span of $(FF^T (L_G - \gamma D_G) FF^T)$, thus leading to a large residual. By changing variables so that $x = FF^T y$, the solution becomes

$$x_t^* \propto FF^T (FF^T (L_G - \gamma_t D_G) FF^T)^+ FF^T D_G s.$$

Since FF^T is a projection matrix, this expression is equivalent to

$$x_t^* \propto \left(FF^T(L_G - \gamma_t D_G)FF^T\right)^+ FF^T D_G s. \tag{5}$$

Third, regarding the solution x_i, we exploit that $FF^T(L_G - \gamma_i D_G)FF^T$ is an SPSD matrix, and we apply the conjugate gradient method, rather than computing the explicit pseudoinverse. That is, in the implementation we never explicitly represent the dense matrix FF^T, but instead we treat it as an operator and we simply evaluate the result of applying a vector to it on either side. Fourth, we use that λ_2 can never decrease (here we refer to λ_2 as the smallest non-zero eigenvalue of the modified matrix), so we only recalculate the upper bound for the binary search when an iteration saturates without satisfying $\|(x_t^T D_G s)^2 - \kappa_t\| \leq \epsilon$. Estimating the bound is critical for the semi-supervised eigenvectors to be able to interpolate all the way to the global eigenvectors of the graph, so in Section 3.4 we return to a discussion on efficient strategies for computing the leading nontrivial eigenvalue of L_G projected down onto the space perpendicular to the previously computed solutions.

From this discussion, it should be clear that Algorithm 1 solves the semi-supervised eigenvector problem by solving in an iterative manner optimization problems of the form of Generalized LocalSpectral; and that the running time of Algorithm 1 boils down to solving a sequence of linear equations.

3.3 Discussion of Our Main Algorithm

There is a natural "regularization" interpretation underlying our construction of semi-supervised eigenvectors. To see this, recall that the first step of our algorithm can be computed as the solution of a set of linear equations

$$x^* = c\,(L_G - \gamma D_G)^+ D_G s, \tag{6}$$

for some normalization constant c and some γ that can be determined by a binary search over $(-\mathrm{vol}(G), \lambda_2(G))$; and that subsequent steps compute the analogous quantity, subject to additional constraints that the solution be orthogonal to the previously-computed vectors. The quantity $(L_G - \gamma D_G)^+$ can be interpreted as a "regularized" version of the pseudoinverse of L, where $\gamma \in (-\infty, \lambda_2(G))$ serves as the regularization parameter. This interpretation has recently been made precise: Mahoney and Orecchia (2011) show that running a PageRank computation—as well as running other diffusion-based procedures—*exactly* optimizes a regularized version of the GlobalSpectral (or LocalSpectral, depending on the input seed vector) problem; and (Perry and Mahoney, 2011) provide a precise statistical framework justifying this.

The usual interpretation of PageRank involves "random walkers" who uniformly (or non-uniformly, in the case of Personalized PageRank) "teleport" with a probability $\alpha \in (0, 1)$. As described in (Mahoney et al., 2012), choosing $\alpha \in (0, 1)$ corresponds to choosing $\gamma \in (-\infty, 0)$. By rearranging Eqn. (6) as

$$\begin{aligned} x^* &= c\,((D_G - A_G) - \gamma D_G)^+ D_G s \\ &= \frac{c}{1-\gamma}\left(D_G - \frac{1}{1-\gamma}A_G\right)^+ D_G s \\ &= \frac{c}{1-\gamma} D_G^{-1}\left(I - \frac{1}{1-\gamma}A_G D_G^{-1}\right)^+ D_G s, \end{aligned}$$

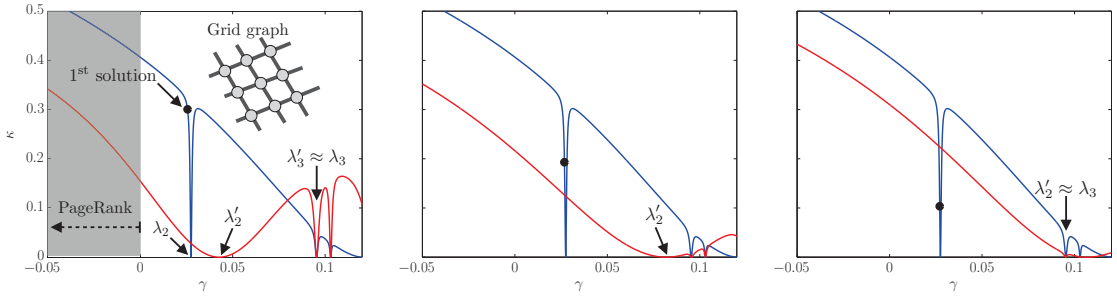
we recognize $A_G D_G^{-1}$ as the standard random walk matrix, and it becomes immediate that the solution based on random walkers,

$$x^* = \frac{c}{1-\gamma} D_G^{-1} \left(I + \sum_{i=1}^{\infty} \left(\frac{1}{1-\gamma} D_G^{-1} A_G \right)^i \right) D_G s,$$

is divergent for $\gamma > 0$. Since $\gamma = \lambda_2(G)$ corresponds to no regularization and $\gamma \to -\infty$ corresponds to heavy regularization, viewing this problem in terms of solving a linear equation is formally more powerful than viewing it in terms of random walkers. That is, while all possible values of the regularization parameter—and in particular the (positive) value $\lambda_2(\cdot)$—are achievable algorithmically by solving a linear equation, only values in $(-\infty, 0)$ are achievable by running a PageRank diffusion. In particular, if the optimal value of γ that saturates the κ-dependent locality constraint is negative, then running the PageRank diffusion could find it; otherwise, the "best" one could do will still not saturate the locality constraint, in which case some of the intended correlation is "unused."

An important technical and practical point has to do with the precise manner in which the i^{th} vector is well-correlated with the seed set s. In our formulation, this is captured by a *locality parameter* γ_i that is chosen (via a binary search) to "saturate" the correlation condition, i.e., so that the i^{th} vector is κ/k-well-correlated with the input seed set. As a general rule, successive γ_is need to be chosen that successive vectors are *less* well-localized around the input seed set. (Alternatively, depending on the application, one could choose this parameter so that successive γ_is are equal; but this will involve "sacrificing" some amount of the κ/k correlation, which will lead to the last or last few eigenvectors being very poorly-correlated with the input seed set. These tradeoffs will be described in more detail below.) Informally, if s is a seed set consisting of a small number of nodes that are "nearby" each other, then to maintain a given amount of correlation, we must "view" the graph over larger and larger size scales as we compute more and more semi-supervised eigenvectors. More formally, we need to let the value of the regularization parameter γ at the i^{th} round, we call it γ_i, vary for each $i \in \{1, \ldots, k\}$. That is, γ_i is not pre-specified, but it is chosen via a binary search over the region $(-\mathrm{vol}(G), \lambda_2(\cdot))$, where $\lambda_2(\cdot)$ is the leading nontrivial eigenvalue of L_G projected down onto the space perpendicular to the previously-computed vectors (which is in general larger than $\lambda_2(G)$). In this sense, our semi-supervised eigenvectors are both "locally-biased", in a manner that depends on the input seed vector and correlation parameter, and "regularized", in a manner that depends on the local graph structure.

To illustrate the previous discussion, Figure 2 considers the two-dimensional grid. In each subfigure, the blue curve shows the correlation with a single seed node as a function of γ for the leading semi-supervised eigenvector, and the black dot illustrates the value of γ for three different values of the locality parameter κ. This relationship between κ and γ is in general non-convex, but it is monotonic for $\gamma \in (-\mathrm{vol}(G), \lambda_2(G))$. The red curve in each subfigure shows the decay for the second semi-supervised eigenvector. Recall that it is perpendicular to the first semi-supervised eigenvector, that the decay is monotonic for $\gamma \in (-\mathrm{vol}(G), \lambda_2'(G))$, and that $\lambda_2 < \lambda_2' \leq \lambda_3$. In Figure 2(a), the first semi-supervised eigenvector is not "too" close to λ_2, and so λ_2' (i.e., the second eigenvalue of the next semi-supervised eigenvector) increases just slightly. In Figure 2(b), we consider a locality



(a) $(\boldsymbol{x}_1^T \boldsymbol{D}_G \boldsymbol{s})^2 = 0.3$ (b) $(\boldsymbol{x}_1^T \boldsymbol{D}_G \boldsymbol{s})^2 = 0.2$ (c) $(\boldsymbol{x}_1^T \boldsymbol{D}_G \boldsymbol{s})^2 = 0.1$

Figure 2: Interplay between the γ parameter and the correlation κ that a semi-supervised eigenvector has with a seed $\boldsymbol{s}$ on a two-dimensional grid. In Figure 2(a)-2(c), we vary the locality parameter for the leading semi-supervised eigenvector, which in each case leads to a value of γ which is marked by the black dot on the blue curve. This allows us to illustrate the influence on the relationship between γ and κ on the next semi-supervised eigenvector. Figure 2(a) also highlights the range ($\gamma < 0$) in which Personalized PageRank can be used for computing solutions to semi-supervised eigenvectors.

parameter that leads to a value of γ that is closer to λ_2, thereby increasing the value of λ_2'. Finally, in Figure 2(c), the locality parameter is such that the leading semi-supervised eigenvector almost coincides with $\boldsymbol{v}_2$; this results in $\lambda_2' \approx \lambda_3$, as required if we were to compute the global eigenvectors.

3.4 Bounding the Binary Search

For the following derivations it is more convenient to consider the normalized graph Laplacian, in which case we define the first solution as

$$y_1 = c\,(\mathcal{L}_G - \gamma_1 I)^+ D_G^{1/2} s, \tag{7}$$

where $x_1^* = D_G^{-1/2} y_1$. This approach is convenient since the projection operator with null space defined by previous solutions can be expressed as $FF^T = I - YY^T$, assuming that $Y^T Y = 1$. That is, Y is of the form

$$Y = [D_G^{1/2}, y_1^*, \ldots, y_{t-1}^*],$$

where y_i^* are successive solutions to Eqn. (7). In the following the type of projection operator will be implicit from the context, i.e., when working with the combinatorial graph Laplacian $FF^T = I - D_G X (X^T D_G D_G X)^{-1} X^T D_G$, whereas for the normalized graph Laplacian $FF^T = I - YY^T$.

For the normalized graph Laplacian $\mathcal{L}_G$, the eigenvalues of $\mathcal{L}_G v = \lambda v$ equal the eigenvalues of the generalized eigenvalue problem $L_G v = \lambda D_G v$. The binary search employed in Algorithm 1 uses a monotonic relationship between the $\gamma \in (-\mathrm{vol}(G), \lambda_2(\cdot))$ parameter and the

correlation with the seed $x^T D_G s$, that can be deduced from the KKT-conditions (Mahoney et al., 2012). Note, that if the upper bound for the binary search $\top = \lambda_2(FF^T\mathcal{L}_G FF^T)$ is not determined with sufficient precision, the search will (if we underestimate $\top$) fail to satisfy the constraint, or (if we overestimate $\top$) fail to converge because the monotonic relationship no longer hold.

By Lemma 1 in Appendix A it follows that $\lambda_2(FF^T\mathcal{L}_G FF^T) = \lambda_2(\mathcal{L}_G + \omega YY^T)$ when $\omega \to \infty$. Since the latter term is a sum of two PSD matrices, the value of the upper bound can only increase as stated by Lemma 2 in Appendix A. This is an important property, because if we do not recalculate $\top$, the previous value is guaranteed to be an underestimate, meaning that the objective will remain convex. Thus, it may be more efficient to first recompute $\top$ when the binary search fails to satisfy $(x^T D_G s)^2 = \kappa$, meaning that $\top$ must be recomputed to increase the search range.

We compute the value for the upper bound of the binary search by transforming the problem in such a way that we can determine the greatest eigenvalue of a new system (fast and robust), and from that, deduce the new value of $\top = \lambda_2(FF^T\mathcal{L}_G FF^T)$. We do so by expanding the expression as

$$\begin{aligned} FF^T\mathcal{L}_G FF^T &= FF^T\left(I - D_G^{-1/2}A_G D_G^{-1/2}\right)FF^T \\ &= FF^T - FF^T D_G^{-1/2}A_G D_G^{-1/2}FF^T \\ &= I - \left(FF^T D_G^{-1/2}A_G D_G^{-1/2}FF^T + YY^T\right). \end{aligned}$$

Since all columns of Y will be eigenvectors of $FF^T\mathcal{L}_G FF^T$ with zero eigenvalue, these will all be eigenvectors of $FF^T D_G^{-1/2}A_G D_G^{-1/2}FF^T + YY^T$ with eigenvalue 1. Hence, the largest algebraic eigenvalue $\lambda_{\mathrm{LA}}(FF^T D_G^{-1/2}A_G D_G^{-1/2}FF^T)$ can be used to compute the upper bound for the binary search as

$$\top = \lambda_2(FF^T\mathcal{L}_G FF^T) = 1 - \lambda_{\mathrm{LA}}(FF^T D_G^{-1/2}A_G D_G^{-1/2}FF^T). \tag{8}$$

The reason for not considering the largest magnitude eigenvalue, is that A_G may be indefinite. Finally, with respect to our implementation we emphasize that FF^T is used as a projection operator, and not represented explicitly.

4. Extension of Our Main Algorithm And Implementation Details

In this section, we present two variants of our main algorithm that are more well-suited for very large-scale applications; the first uses a column-based low-rank approximation, and the second uses random walk ideas. In Section 4.1, we describe how to use the Nyström method, which constructs a low-rank approximation to the kernel matrix by sampling columns, to construct a general solution for semi-supervised eigenvectors, where the low-rankness is exploited for very efficient computation. Then, in Section 4.2, we describe a "Push-peeling heuristic," based on the efficient Push algorithm by Andersen et al. (2006). The basic idea is that if, rather than iteratively computing locally-biased semi-supervised eigenvectors using the procedure described in Algorithm 1, we instead compute solutions to LocalSpectral and then construct the semi-supervised eigenvectors by "projecting away" pieces of these solutions, then we can take advantage of local random walks that have improved algorithmic properties.

4.1 A Nyström-Based Low-rank Approach

Here we describe the use of the recently-popular Nyström method to speed up the computation of semi-supervised eigenvectors. We do so by considering how a low-rank decomposition can be exploited to yield solutions to the Generalized LocalSpectral objective in Figure 1, where the running time largely depends on a matrix-vector product. These methods are most appropriate when the kernel matrix is reasonably well-approximated by a low-rank matrix (Drineas and Mahoney, 2005; Gittens and Mahoney, 2012; Williams and Seeger, 2000).

Given some low-rank approximation $\mathcal{L}_G \approx I - V\Lambda V^T$, we apply the Woodbury matrix identity, and we derive an explicit solution for the leading semi-supervised eigenvector

$$\begin{aligned} y_1 &\approx c\left((1-\gamma)I - V\Lambda V^T\right)^+ D_G^{1/2}s \\ &\approx c\left(\frac{1}{1-\gamma}I + \frac{1}{(1-\gamma)^2}V\left(\Lambda^{-1} - \frac{1}{1-\gamma}I\right)^{-1}V^T\right)D_G^{1/2}s \\ &\approx \frac{c}{1-\gamma}\left(I + V\Sigma V^T\right)D_G^{1/2}s, \end{aligned}$$

where $\Sigma_{ii} = \frac{1}{\frac{1-\gamma}{\lambda_i}-1}$. In order to compute efficiently the subsequent semi-supervised eigenvectors we must accommodate for the projection operator $FF^T = I - YY^T$, while yet exploiting the explicit closed-form inverse $(\mathcal{L}_G - \gamma I)^+ \approx \frac{1}{1-\gamma}\left(I + V\Sigma V^T\right)$. However, the projection operator complicates the expression, since the previous solution can be spanned by multiple global eigenvectors, so leveraging from the low-rank decomposition is more difficult for the inverse $(FF^T(\mathcal{L}_G - \gamma I)FF^T)^+$.

Conveniently, we can decouple the projection operator by treating the orthogonality constraint using a Lagrangian approach, such that the solution can be expressed as

$$y_t = c\left(\mathcal{L}_G - \gamma I + \omega YY^T\right)^+ D_G^{1/2}s,$$

where $\omega \geq 0$ denotes the associated Lagrange multiplier, and where the sign is deduced from the KKT conditions. Applying the Woodbury matrix identity is now straightforward

$$\left(P_\gamma + \omega YY^T\right)^+ = {P_\gamma}^+ - \omega {P_\gamma}^+ Y\left(I + \omega Y^T {P_\gamma}^+ Y\right)^+ Y^T {P_\gamma}^+,$$

where for notational convenience we have introduced $P_\gamma = \mathcal{L}_G - \gamma I$. By decomposing $Y^T {P_\gamma}^+ Y$ with an eigendecomposition USU^T the equation simplifies as follows

$$\begin{aligned} \left(P_\gamma + \omega YY^T\right)^+ &= {P_\gamma}^+ - \omega {P_\gamma}^+ Y\left(I + \omega USU^T\right)^+ Y^T {P_\gamma}^+ \\ &= {P_\gamma}^+ - {P_\gamma}^+ YU\Omega U^T Y^T {P_\gamma}^+, \end{aligned}$$

where $\Omega_{ii} = \frac{1}{\frac{1}{\omega}+S_{ii}}$. Note how this result gives a well defined way of controlling the amount of "orthogonality", and by Lemma 1 in Appendix A, we get exact orthogonality in the limit of $\omega \to \infty$, in which case the expression simplifies to

$$\left(P_\gamma + \omega YY^T\right)^+ = {P_\gamma}^+ - {P_\gamma}^+ Y(Y^T {P_\gamma}^+ Y)^+ Y^T {P_\gamma}^+.$$

Using the explicit expression for ${P_\gamma}^+$, the solution now only involves matrix-vector products and the inverse of a small matrix

$$y_t = c\left({P_\gamma}^+ - {P_\gamma}^+ Y (Y^T {P_\gamma}^+ Y)^+ Y^T {P_\gamma}^+\right) D_G^{1/2} s. \tag{9}$$

To conclude this section, let us also consider how we can optimize the efficiency of the calculation of $\lambda_2(FF^T\mathcal{L}_G FF^T)$ used for bounding the binary search in Algorithm 1. According to Eqn. (8) the bound can be calculated efficiently as $\top = 1-\lambda_{\mathrm{LA}}(FF^T D_G^{-1/2} A_G D_G^{-1/2} FF^T)$. However, by substituting with $D_G^{-1/2} A_G D_G^{-1/2} \approx V\Lambda V^T$, we can exploit low-rankness since

$$\top = 1 - \lambda_{\mathrm{LA}}(FF^T V\Lambda V^T FF^T) = 1 - \lambda_{\mathrm{LA}}(\Lambda^{1/2} V^T FF^T V \Lambda^{1/2}),$$

where the latter is a much smaller system.

4.2 A Push-Peeling Heuristic

Here we present a variant of our main algorithm that exploits the connections between diffusion-based procedures and eigenvectors, allowing semi-supervised eigenvectors to be efficiently computed for large networks. This is most well-known for the leading nontrivial eigenvectors of the graph Laplacian (Chung, 1997); but recent work has exploited these connections in the context of performing locally-biased spectral graph partitioning (Spielman and Teng, 2004; Andersen et al., 2006; Mahoney et al., 2012). In particular, we can compute the locally-biased vector using the first step of Algorithm 1, or alternatively we can compute it using a locally-biased random walk of the form used in (Spielman and Teng, 2004; Andersen et al., 2006). Here we present a heuristic that works by peeling off components from a solution to the PageRank problem, and by exploiting the regularization interpretation of γ, we can from these components obtain the subsequent semi-supervised eigenvectors.

Specifically, we focus on the Push algorithm by Andersen et al. (2006). This algorithm approximates the solution to PageRank very efficiently, by exploiting the local modifications that occur when the seed is highly concentrated. This makes our algorithm very scalable and applicable for large-scale data, since only the local neighborhood near the seed set will be touched by the algorithm. In comparison, by solving the linear system of equations we explicitly touch all nodes in the graph, even though most spectral rankings will be below the computational precision (Boldi and Vigna, 2011).

We adapt a similar notation as in Andersen et al. (2006) and start by defining the usual PageRank vector $\mathrm{pr}(\alpha, s_{\mathrm{pr}})$ as the unique solution of the linear system

$$\mathrm{pr}(\alpha, s_{\mathrm{pr}}) = \alpha s_{\mathrm{pr}} + (1-\alpha) A_G D_G^{-1} \mathrm{pr}(\alpha, s_{\mathrm{pr}}), \tag{10}$$

where α is the teleportation parameter, and s_{pr} is the sparse starting vector. For comparison, the push algorithm by Andersen et al. (2006) computes an approximate PageRank vector $\mathrm{pr}_\epsilon(\alpha', s_{\mathrm{pr}})$ for a slightly different system

$$\mathrm{pr}_\epsilon(\alpha', s_{\mathrm{pr}}) = \alpha' s_{\mathrm{pr}} + (1-\alpha') W \mathrm{pr}_\epsilon(\alpha', s_{\mathrm{pr}}),$$

where $W = \frac{1}{2}(I + A_G D_G^{-1})$ and not the usual random walk matrix AD_G^{-1} as used in Eqn. (10). However, these equations are only superficially different, and equivalent up to a change

of the respective teleportation parameter. Thus, it is straightforward to verify that these teleportation parameters and the γ parameter of Eqn. (6) are related as

$$\alpha = \frac{2\alpha'}{1+\alpha'} \Leftrightarrow \alpha' = \frac{\alpha}{2-\alpha} \Leftrightarrow \alpha' = \frac{\gamma}{\gamma-2},$$

and that the leading semi-supervised eigenvector for $\gamma \in (-\infty, 0)$ can be approximated as

$$x_1^* \approx \frac{c}{-\gamma} D_G^{-1} \mathrm{pr}_\epsilon \left(\frac{\gamma}{\gamma-2}, D_G s \right).$$

To generalize subsequent semi-supervised eigenvectors to this diffusion based framework, we need to accommodate for the projection operator such that subsequent solutions can be expressed in terms of graph diffusions. By requiring distinct values of γ for all semi-supervised eigenvectors, we may use the solution for the leading semi-supervised eigenvector and then systematically "peel off" components, thereby obtaining the solution of one of the consecutive semi-supervised eigenvectors. By Lemma 5, in Appendix A the general solution in Eqn. (5) can be approximated by

$$x_t^* \approx c \left(I - XX^T D_G\right) \left(L_G - \gamma_t D_G\right)^+ D_G s, \tag{11}$$

under the assumption that all γ_k for $1 < k \leq t$ are sufficiently apart. If we think about γ_k as being distinct eigenvalues of the generalized eigenvalue problem $L_G x_k = \gamma_k D_G x_k$, then it is clear that Eqn. (11), correctly computes the sequence of generalized eigenvectors. This is explained by the fact that $(L_G - \gamma_t D_G)^+ D_G s$ can be interpreted as the first step of the Rayleigh quotient iteration, where γ_t is the estimate of the eigenvalue, and $D_G s$ is the estimate of the eigenvector. Given that the estimate of the eigenvalue is right, this algorithm will in the initial step compute the corresponding eigenvector, and the operator $\left(I - XX^T D_G\right)$ will be superfluous, as the global eigenvectors are already orthogonal in the degree-weighted norm. To quantify the failure modes of the approximation, let us consider what happens when γ_2 starts to approach γ_1. What constitutes the second solution for a particular value of γ_2 is the perpendicular component with respect to the projection onto the solution given by γ_1. As γ_2 approaches γ_1, this perpendicular part diminishes and the solution becomes ill-posed. Fortunately, we can easily detect such issues during the binary search in Algorithm 1, and in general the approximation has turned out to work very well in practice as our experimental results in Section 5 show.

In terms of the approximate PageRank vector $\mathrm{pr}_\epsilon(\alpha', s_{\mathrm{pr}})$, the general approximate solution takes the following form

$$x_t^* \approx c \left(I - XX^T D_G\right) D_G^{-1} \mathrm{pr}_\epsilon \left(\frac{\gamma_t}{\gamma_t - 2}, D_G s \right). \tag{12}$$

As already stated in Section 3.3, the impact of using a diffusion based procedure is that we cannot interpolate all the way to the global eigenvectors, and that the main challenge is that the solutions do not become too localized. The ϵ parameter of the Push algorithm controls the threshold for propagating mass away from the seed set and into the adjacent nodes in the graph. If the threshold is too high, the solution will be very localized and make it difficult to find more than a few semi-supervised eigenvectors, as characterized by

Lemma 3 in Appendix A, because the leading ones will then span the entire space of the seed set. As the choice of ϵ is important for the applicability of our algorithm, we will in Section 5 investigate the influence of this parameter on large data graphs.

To conclude this section, we consider an important implementation detail that have been omitted so far. In the work of Mahoney et al. (2012) the seed vector was defined to be perpendicular to the all-ones vector, and for the sake of consistency we have chosen to define it in the same way. The impact of projecting the seed set to a space that is perpendicular to the all-ones vector is that the resulting seed vector is no longer sparse, making the use of the Push algorithm in Eqn. (12) inefficient. The seed vector can, however, without loss of generality, be defined as $s \propto D_G^{-1/2}\left(I - v_0 v_0^T\right) s_0$ where s_0 is the sparse seed, and $v_0 \propto \operatorname{diag}\left(D_G^{1/2}\right)$ is the leading eigenvector of the normalized graph Laplacian (corresponding to the all-ones vector of the combinatorial graph Laplacian). If we substitute with this expression for the seed in Eqn. (12), it follows by plain algebra (see Appendix B) that

$$x_t^* \approx c\left(I - XX^T D_G\right)\left(D_G^{-1}\mathrm{pr}_\epsilon\left(\frac{\gamma_t}{\gamma_t - 2}, D_G^{1/2} s_0\right) - D_G^{-1/2} v_0 v_0^T s_0\right). \tag{13}$$

Now the Push algorithm is only defined on the sparse seed set making the the expression very scalable. Finally, the Push algorithm maintains a queue of high residual nodes that are yet to be processed. The order in which nodes are processed influences the overall running time, and in Boldi and Vigna (2011) preliminary experiments showed that a FIFO queue resulted in the best performance for large values of γ, as compared to a priority queue that scales logarithmically. For this reason we have chosen to use a FIFO queue in our implementation.

5. Empirical Results

In this section, we provide a detailed empirical evaluation of the method of semi-supervised eigenvectors and how it can be used for locally-biased machine learning. Our goal is two-fold: first, to illustrate how the "knobs" of the method work; and second, to illustrate the usefulness of the method in real applications. To do this, we consider several classes of data.

- **Toy data.** In Section 5.1, we consider one-dimensional examples of the popular "small world" model (Watts and Strogatz, 1998). This is a parameterized family of models that interpolates between low-dimensional grids and random graphs; and, as such, it allows us to illustrate the behavior of the method and its various parameters in a controlled setting.

- **Congressional voting data.** In Section 5.2, we consider roll call voting data from the United States Congress that are based on (Poole and Rosenthal, 1991). This is an example of realistic data set that has relatively-simple global structure but nontrivial local structure that varies with time (Cucuringu and Mahoney, 2011); and thus it allows us to illustrate the method in a realistic but relatively-clean setting.

- **Handwritten image data.** In Section 5.3, we consider data from the MNIST digit data set (Lecun and Cortes). These data have been widely-studied in machine learning and related areas and they have substantial "local heterogeneity." Thus, these

data allow us to illustrate how the method may be used to perform locally-biased versions of common machine learning tasks such as smoothing, clustering, and kernel construction.

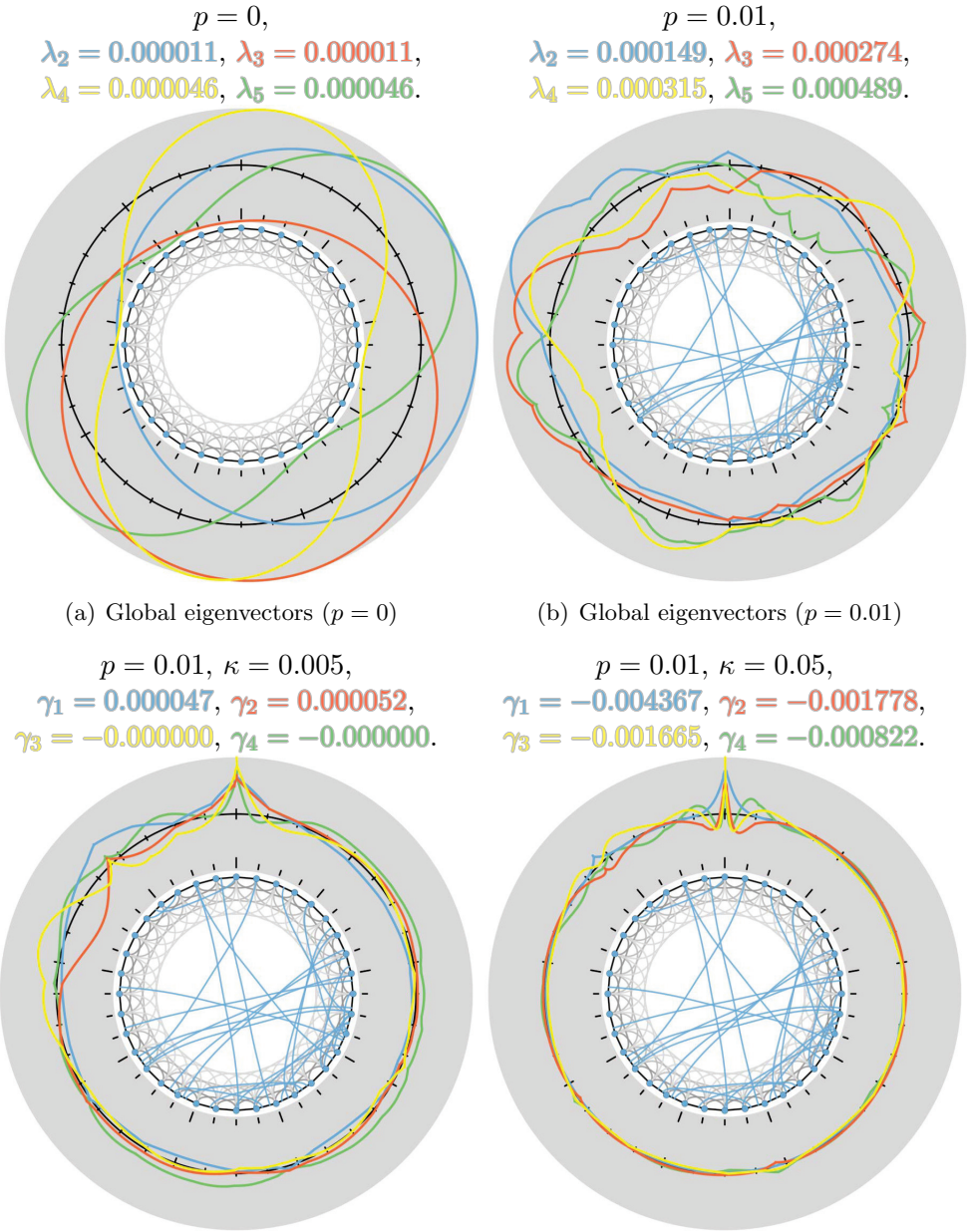
- **Large-scale network data.** In Section 5.4, we consider large-scale network data, and demonstrate significant performance improvements of the push-peeling heuristic compared to solving the same equations using a conjugate gradient solver. These improvements are demonstrated on data sets from the DIMACS implementation challenge, as well as on large web-crawls with more then 3 billion non-zeros in the adjacency matrix (Paolo et al., 2004, 2011; Paolo and Sebastiano, 2004).

5.1 Small-World Data

The first data sets we consider are networks constructed from the so-called small-world model. This model can be used to demonstrate how semi-supervised eigenvectors focus on specific target regions of a large data graph to capture slowest modes of local variation; and it can also be used to illustrate how the "knobs" of the method work, e.g., how κ and γ interplay, in a practical setting. In Figure 3, we plot the usual global eigenvectors, as well as locally-biased semi-supervised eigenvectors, around illustrations of non-rewired and rewired realizations of the small-world graph, i.e., for different values of the rewiring probability p and for different values of the locality parameter κ.

To start, in Figure 3(a) that we show a graph with no randomly-rewired edges ($p = 0$) and a parameter κ such that the global eigenvectors are obtained. This yields a symmetric graph with eigenvectors corresponding to orthogonal sinusoids, i.e., the first two capture the slowest mode of variation and correspond to a sine and cosine with equal random phase-shift (up to a rotational ambiguity). In Figure 3(b), random edges have been added with probability $p = 0.01$ and the parameter κ is still chosen such that the global eigenvectors—now of the rewired graph—are obtained. Note the many small kinks in the eigenvectors at the location of the randomly added edges. Note also the slow mode of variation in the interval on the top left; a normalized-cut based on the leading global eigenvector would extract this region, since the remainder of the ring is more well-connected due to the random rewiring.

In Figure 3(c), we see the same graph realization as in Figure 3(b), except that the semi-supervised eigenvectors have a seed node at the top of the circle, i.e., at "12 o-clock," and the locality parameter $\kappa_t = 0.005$, which corresponds to moderately well-localized eigenvectors. As with the global eigenvectors, the locally-biased semi-supervised eigenvectors are of successively-increasing (but still localized) variation. Note also that the neighborhood around "11 o-clock" contains more mass, e.g., when compared with the same parts of the circle in Figure 3(b) or with other parts of the circle in Figure 3(c), even though it is not very near the seed node in the original graph geometry. The reason for this is that this region is well-connected with the seed via a randomly added edge, and thus it is close in the modified graph topology. Above this visualization, we also show the value of γ_t that saturates κ_t, i.e., γ_t is the Lagrange multiplier that defines the effective locality κ_t. Not shown is that if we kept reducing κ_t, then γ_t would tend towards λ_{t+1}, and the respective semi-supervised eigenvectors would tend towards the global eigenvectors that are illustrated in Figure 3(b). Finally, in Figure 3(d), the desired locality is increased to $\kappa = 0.05$ (which



(a) Global eigenvectors ($p = 0$)

(b) Global eigenvectors ($p = 0.01$)

(c) Semi-supervised eigenvectors

(d) Semi-supervised eigenvectors

Figure 3: Illustration of small-world graphs with rewiring probability of $p = 0$ or $p = 0.01$ and with different values of the κ parameter. For each subfigure, the data consist of 3600 nodes, each connected to it's 8 nearest-neighbors. In the center of each subfigure, we show the nodes (blue) and edges (black and light gray are the local edges, and blue are the randomly-rewired edges). We wrap around the plots (black x-axis and gray background), visualizing the 4 smallest semi-supervised eigenvectors. Eigenvectors are color coded as **blue**, **red**, **yellow**, and **green**, starting with the one having the smallest eigenvalue.

has the effect of decreasing the value of γ_t), making the semi-supervised eigenvectors more localized in the neighborhood of the seed. It should be clear that, in addition to being determined by the locality parameter, we can think of γ as a regularizer biasing the global eigenvectors towards the region near the seed set. That is, variation in eigenvectors that are near the initial seed (in the modified graph topology) are most important, while variation that is far away from the initial seed matters much less.

5.2 Congressional Voting Data

The next data set we consider is a network constructed from a time series of roll call voting patterns from the United States Congress that are based on Poole and Rosenthal (1991). This is a particularly well-structured social network for which there is a great deal of meta-information, and it has been studied recently with graph-based methods (Mucha et al., 2010; Waugh et al., 2009; Cucuringu and Mahoney, 2011). Thus, it permits a good illustration of the method of semi-supervised eigenvectors in a real application (Poole, Fall 2005). This data set is known to have nontrivial time-varying structure at different time steps, and we will illustrate how the method of semi-supervised eigenvectors can perform locally-biased classification with a traditional kernel-based algorithm.

In more detail, we evaluate our method by considering the known Congress data-set containing the roll call voting patterns in the U.S Senate across time. We considered Senates in the 70^{th} Congress through the 110^{th} Congress, thus covering the years 1927 to 2008. During this time, the U.S went from 48 to 50 states, hence the number of senators in each of these 41 Congresses was roughly the same. We constructed an $N \times N$ adjacency matrix, with $N = 4196$ (41 Congresses each with ≈ 100 Senators) where $A_{ij} \in [0, 1]$ represents the extent of voting agreement between legislators i and j, and where identical senators in adjacent Congresses are connected with an inter-Congress connection strength. We then considered the Laplacian matrix of this graph, constructed in the usual way (Cucuringu and Mahoney, 2011).

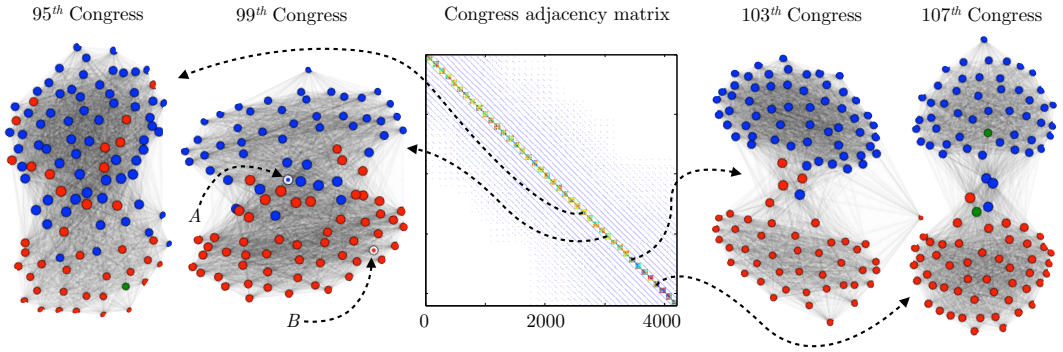


Figure 4: Shows the Congress adjacency matrix, along with four of the individual Congresses. Nodes are scaled according to their degree, blue nodes correspond to Democrats, red to Republicans, and green to Independents.

Figure 4 visualizes the adjacency matrix, along with four of the individual Congresses, color coded by party. This illustrates that these data should be viewed—informally—as a structure (depending on the specific voting patterns of each Congress) evolving along a one-dimensional temporal axis, confirming the results of Cucuringu and Mahoney (2011). Note that the latter two Congresses are significantly better described by a simple two-clustering than the former two Congresses, and an examination of the clustering properties of each of the 40 Congresses reveals significant variation in the local structure of individual Congresses, in a manner broadly consistent with Poole (Fall 2005) and Poole and Rosenthal (1991). In particular, the more recent Congresses are significantly more polarized.

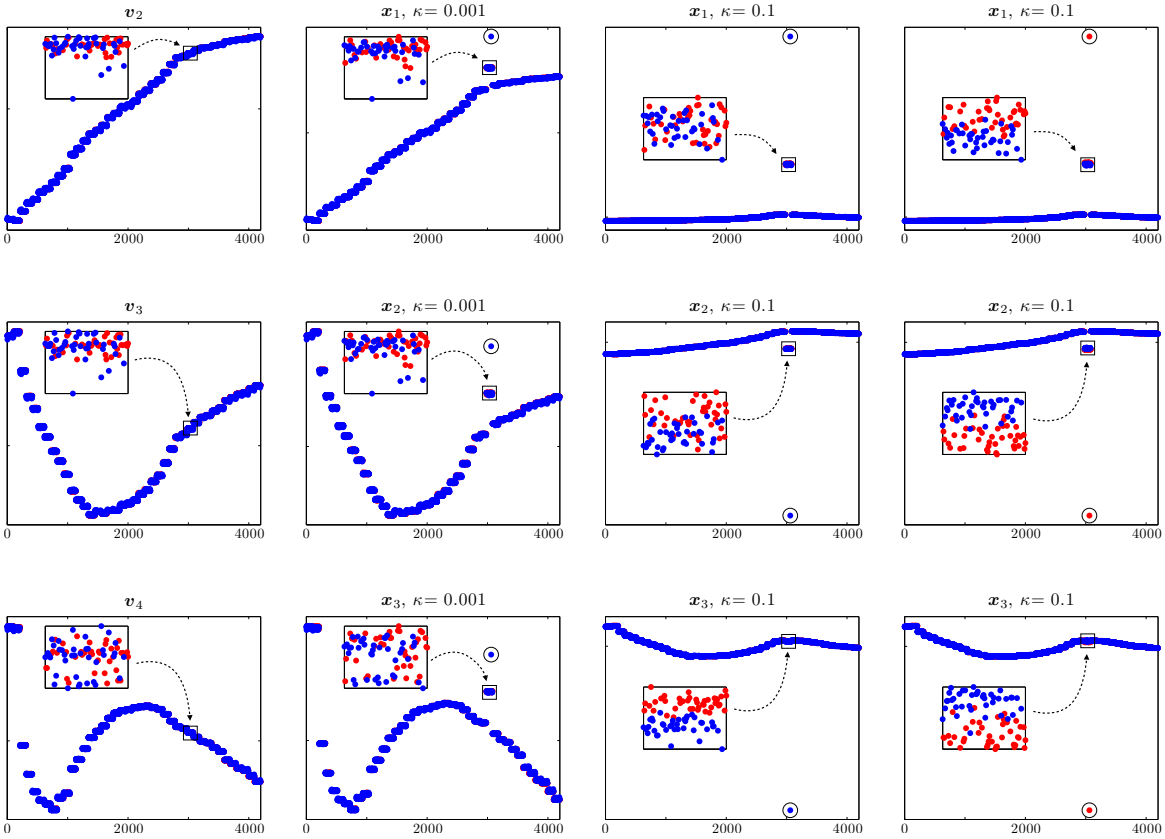


Figure 5: First column: The leading three nontrivial global eigenvectors. Second column: The leading three semi-supervised eigenvectors seeded (circled node) in an articulation point between the two parties in the 99^{th} Congress (see Figure 4), for correlation $\kappa = 0.001$. Third column: Same seed as previous column, but for a correlation of $\kappa = 0.1$. Notice the localization on the third semi-supervised eigenvector. Fourth column: Same correlation as the previous column, but for another seed node well within the cluster of Republicans. Notice the localization on all three semi-supervised eigenvectors.

The first vertical column of Figure 5 illustrates the first three global eigenvectors of the full data set, illustrating fluctuations that are sinusoidal and consistent with the one-

dimensional temporal scaffolding. Also shown in the first column are the values of that eigenfunction for the members of the 99^{th} Congress, illustrating that there is *not* a good separation based on party affiliations. The next three vertical columns of Figure 5 illustrate various localized eigenvectors computed by starting with a seed node in the 99^{th} Congress. For the second column, we visualize the semi-supervised eigenvectors for a very low correlation ($\kappa = 0.001$), which corresponds to only a weak localization—in this case one sees eigenvectors that look very similar to the global eigenvectors, and the elements of the eigenvector on that Congress do not reveal partitions based on the party cuts.

The third and fourth column of Figure 5 illustrate the semi-supervised eigenvectors for a much higher correlation ($\kappa = 0.1$), meaning a much stronger amount of locality. In particular, the third column starts with the seed node marked A in Figure 4, which is at the articulation point between the two parties, while the fourth column starts with the seed node marked B, which is located well within the cluster of Republicans. In both cases the eigenvectors are much more strongly localized on the 99^{th} Congress near the seed node, and in both cases one observes the partition into two parties based on the elements of the localized eigenvectors. Note, however, that when the initial seed is at the articulation point between two parties then the situation is much noisier: in this case, this "partitionability" is seen only on the third semi-supervised eigenvector, while when the initial seed is well within one party then this is seen on all three eigenvectors. Intuitively, when the seed set is strongly within a good cluster, then that cluster tends to be found with semi-supervised eigenvectors (and we will observe this again below). This is consistent with the diffusion interpretation of eigenvectors. This is also consistent with Cucuringu and Mahoney (2011), who observed that the properties of eigenvector localization depended on the local structure of the data around the seed node, as well as the larger scale structure around that local cluster.

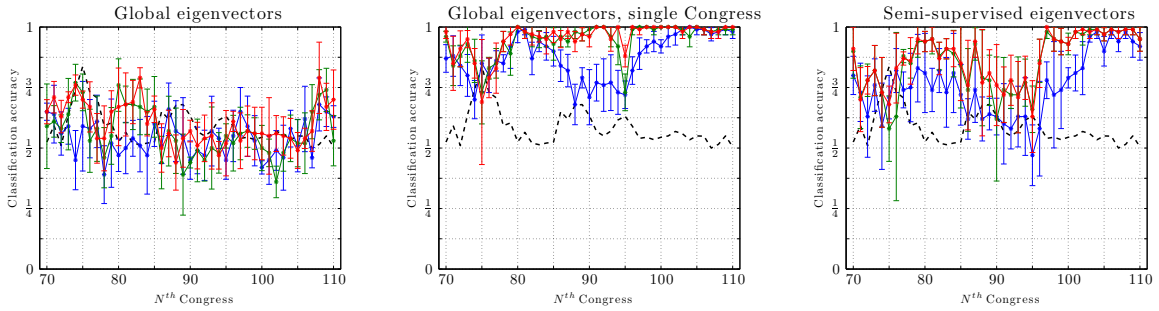


Figure 6: Classification accuracy measured in individual Congresses. For each Congress we perform 5-fold cross validation based on ≈ 80 samples and leave out the remaining 20 samples to estimate an unbiased test error. Error bars are obtained by resampling and they correspond to 1 standard deviation. For each approach we consider features based on the 1^{st} (blue), 2^{nd} (green), and 3^{rd} (red) smallest eigenvector(s), excluding the all-one vector. We also plot the probability of the most probable class as a baseline measure (black) as some Congresses are very imbalanced.

To illustrate how these structural properties manifest themselves in a more traditional machine learning task, we also consider the classification task of discriminating between Democrats and Republicans in single Congresses, i.e., we measure to what extent we can extract local discriminative features. To do so, we apply L_2-regularized L_2-loss support vector classification with a linear kernel, where features are extracted using the global eigenvectors of the entire data set, global eigenvectors from a single Congress (best case measure), and our semi-supervised eigenvectors. Figure 6 illustrates the classification accuracy for 1, 2, and 3 eigenvectors. As reported by Cucuringu and Mahoney (2011), locations that exhibit discriminative information are best found on low-order eigenvectors of this data, explaining why the classifier based global eigenvectors performs poorly. In the classifier based on global eigenvectors in the single Congress we exploit *a priori* knowledge to extract the relevant data, that in a usual situation would be impossible. Hence, this is simply to define a baseline point of reference for the best case classification accuracy. The classifier based on semi-supervised eigenvectors is seeded using a few training samples and performs in-between the two other approaches. Compared to our point of reference, Congresses in the range 88 to 96 do worse with the semi-supervised eigenvectors; whereas for Congresses after 100 the semi-supervised approach almost performs on par, even for a single single eigenvector. This is consistent with the visualization in Figure 4 illustrating that earlier Congresses are less cleanly separable, as well as with empirical evidence indicating heterogeneity due to Southern Democrats in earlier Congresses and the recent increase in party polarization in more recent Congresses, as described in Poole (Fall 2005) and Poole and Rosenthal (1991).

5.3 MNIST Digit Data

The next data set we consider is the well-studied MNIST data set containing $60,000$ training digits and $10,000$ test digits ranging from 0 to 9; and, with these data, we demonstrate the use of semi-supervised eigenvectors as a feature extraction preprocessing step in a traditional machine learning setting. We construct the full $70,000 \times 70,000$ k-NN graph, with $k = 10$ and with edge weights given by $w_{ij} = \exp(-\frac{4}{\sigma_i^2}\|x_i - x_j\|^2)$, where σ_i^2 is the Euclidean distance of the i^{th} node to it's nearest neighbor; and from this we define the graph Laplacian in the usual way. We then evaluate the semi-supervised eigenvectors in a transductive learning setting by disregarding the majority of labels in the entire training data. We use a few samples from each class to seed our semi-supervised eigenvectors as well as a few others to train a downstream classification algorithm. For this evaluation, we use the Spectral Graph Transducer (SGT) of Joachims (2003); and we choose to use it for two main reasons. First, the transductive classifier is inherently designed to work on a subset of global eigenvectors of the graph Laplacian, making it ideal for validating that the localized basis constructed by the semi-supervised eigenvectors can be more informative when we are solely interested in the "local heterogeneity" near a seed set. Second, using the SGT based on global eigenvectors is a good point of comparison, because we are only interested in the effect of our subspace representation. (If we used one type of classifier in the local setting, and another in the global, the classification accuracy that we measure would obviously be confounded.) As in Joachims (2003), we normalize the spectrum of both global and semi-supervised eigenvectors by replacing the eigenvalues with some monotonically increasing function. We use $\lambda_i = \frac{i^2}{k^2}$, i.e., focusing on ranking among smallest cuts; see (Chapelle

et al., 2003). Furthermore, we fix the regularization parameter of the SGT to $c = 3200$, and for simplicity we fix $\gamma = 0$ for all semi-supervised eigenvectors, implicitly defining the effective $\kappa = [\kappa_1, \ldots, \kappa_k]^T$. Clearly, other correlation distributions κ and other values of γ parameter may yield subspaces with even better discriminative properties (which is an issue to which we will return in Section 5.3.2 in greater detail).

	#Semi-supervised eigenvectors for SGT						#Global eigenvectors for SGT					
Labeled points	1	2	4	6	8	10	1	5	10	15	20	25
1 : 1	0.39	0.39	0.38	0.38	0.38	0.36	0.50	0.48	0.36	0.27	0.27	0.19
1 : 10	0.30	0.31	0.25	0.23	0.19	0.15	0.49	0.36	0.09	0.08	0.06	0.06
5 : 50	0.12	0.15	0.09	0.08	0.07	0.06	0.49	0.09	0.08	0.07	0.05	0.04
10 : 100	0.09	0.10	0.07	0.06	0.05	0.05	0.49	0.08	0.07	0.06	0.04	0.04
50 : 500	0.03	0.03	0.03	0.03	0.03	0.03	0.49	0.10	0.07	0.06	0.04	0.04

Table 1: Classification error for discriminating between 4s and 9s for the SGT based on, respectively, semi-supervised eigenvectors and global eigenvectors. The first column from the left encodes the configuration, e.g., 1:10 interprets as 1 seed and 10 training samples from each class (total of 22 samples—for the global approach these are all used for training). When the seed is well-determined and the number of training samples moderate (50:500), then a single semi-supervised eigenvector is sufficient; whereas for less data, we benefit from using multiple semi-supervised eigenvectors. All experiments have been repeated 10 times.

5.3.1 Discriminating Between Pairs of Digits

Here, we consider the task of discriminating between two digits; and, in order to address a particularly challenging task, we work with $4s$ and $9s$. (This is particularly challenging since these two classes tend to overlap more than other combinations since, e.g., a closed 4 can resemble a 9 more than an open 4.) Hence, we expect that the class separation axis will not be evident in the leading global eigenvector, but instead it will be "buried" further down the spectrum; and we hope to find a "locally-biased class separation axis" with locally-biased semi-supervised eigenvectors. Thus, this example will illustrate how semi-supervised eigenvectors can represent relevant heterogeneities in a local subspace of low dimensionality. See Table 1, which summarizes our classification results based on, respectively, semi-supervised eigenvectors and global eigenvectors, when we use the SGT. See also Figure 7 and Figure 8, which illustrate two realizations for the 1:10 configuration. In these two figures, the training samples are fixed; and, to demonstrate the influence of the seed, we have varied the seed nodes. In particular, in Figure 7 the seed nodes s_+ and s_- are located well-within the respective classes; while in Figure 8, they are located much closer to the boundary between the two classes. As intuitively expected, when the seed nodes fall well within the classes to be differentiated, the classification is much better than when the seed nodes are located closer to the boundary between the two classes. See the caption in these figures for further details.

5.3.2 Effect of Choosing The κ Correlation/Locality Parameter

Here, we discuss the effect of the choice of the correlation/locality parameter κ at different steps of Algorithm 1, e.g., how $\{\kappa_t\}_{t=1}^k$ should be distributed among the k components.

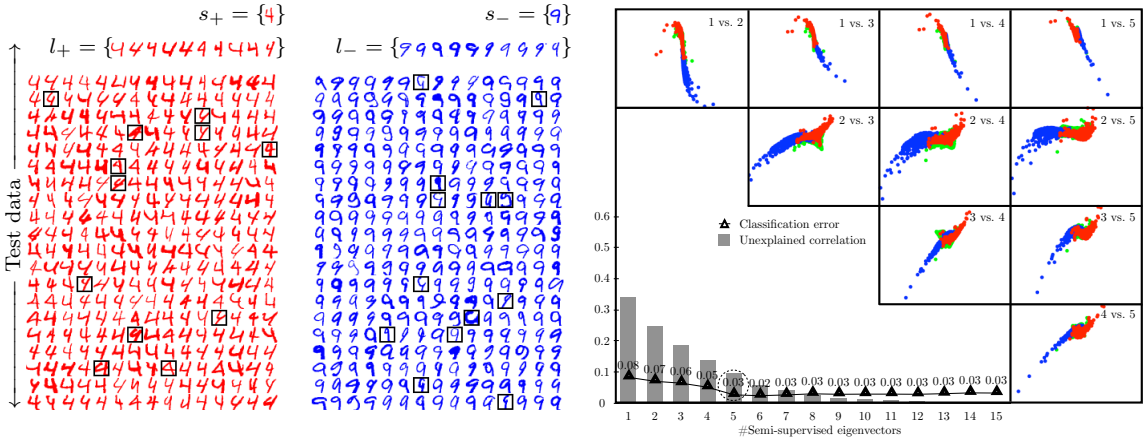


Figure 7: Discrimination between 4s and 9s. Left: Shows a subset of the classification results for the SGT based on 5 semi-supervised eigenvectors seeded in s_+ and s_-, and trained using samples l_+ and l_-. Misclassifications are marked with black frames. Right: Visualizes all test data spanned by the first 5 semi-supervised eigenvectors, by plotting each component as a function of the others. Red (blue) points correspond to 4 (9), whereas green points correspond to remaining digits. As the seed nodes are good representatives, we note that the eigenvectors provide a good class separation. We also plot the error as a function of local dimensionality, as well as the unexplained correlation, i.e., initial components explain the majority of the correlation with the seed (effect of $\gamma = 0$). The particular realization based on the leading 5 semi-supervised eigenvectors yields an error of ≈ 0.03 (dashed circle).

For example, will the downstream classifier benefit the most from a uniform distribution or will there exist some other nonuniform distribution that is better? Although this will be highly problem specific, one might hope that in realistic applications the classification performance is not too sensitive to the actual choice of distribution. To investigate the effect in our example of discriminating between 4s and 9s, we consider 3 semi-supervised eigenvectors for various κ distributions. Our results are summarized in Figure 9.

Figures 9(a), 9(b), and 9(c) show, for the global eigenvectors and for semi-supervised eigenvectors, where the κ vector has been chosen to be very nonuniform and very uniform, the top three (global or semi-supervised) eigenvectors plotted against each other as well as the ROC curve for the SGT classifier discriminating between 4s and 9s; and Figure 9(d) shows the test error as the κ vector is varied over the unit simplex. In more detail, red (respectively, blue) corresponds to 4s (respectively, 9s), and green points are the remaining digits; and, for Figures 9(b) and 9(c), the semi-supervised eigenvectors are seeded using 50 samples from each target class (4s vs. 9s) and having a non-uniform distribution of κ, as specified. As seen from the visualization of the semi-supervised eigenvectors in Figures 9(b) and 9(c), the classes are much better separated than by using the global eigenvectors, which are shown in Figure 9(a). For example, this is supported by the Area Under the Curve

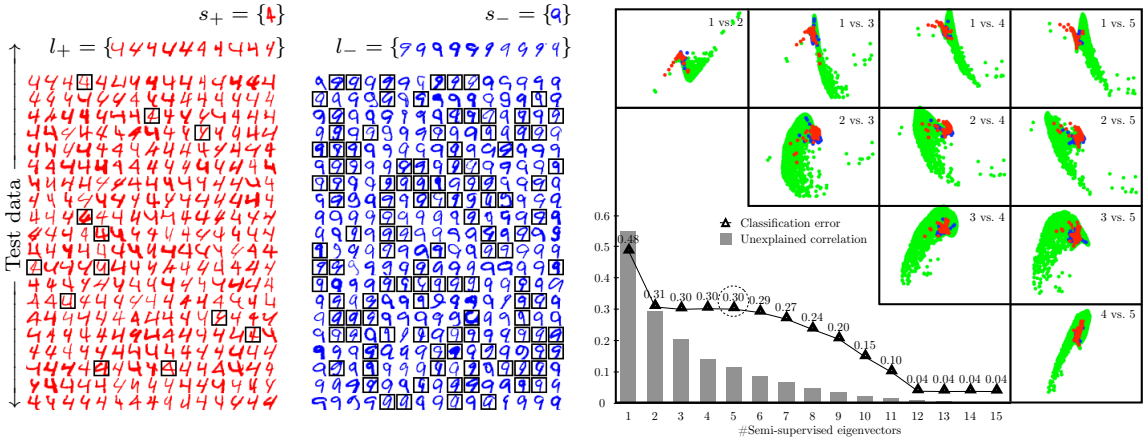
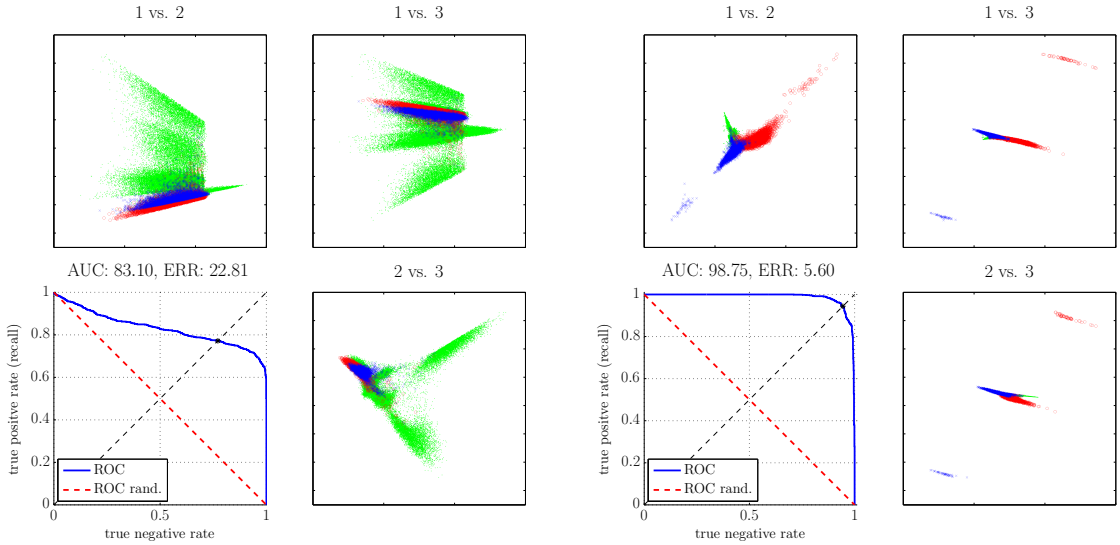


Figure 8: Discrimination between 4s and 9s. See the general description in Figure 7. Here we illustrate an instance where the s_+ shares many similarities with s_-, i.e., s_+ is on the boundary of the two classes. This particular realization achieves a classification error of ≈ 0.30 (dashed circle). In this constellation we first discover localization on low order semi-supervised eigenvectors (≈ 12 eigenvectors), which is comparable to the error based on global eigenvectors (see Table 1), i.e., further down the spectrum we recover from the bad seed and pickup the relevant mode of variation.

(AUC) and Error Rate (ERR), being the point on the Receiver Operating Characteristic (ROC) curve that corresponds to having an equal probability of miss-classifying a positive or negative sample, which is a fair estimate as the classes in the MNIST data set is fairly balanced. For Figure 9(c), where we use a uniform distribution of κ, the classifier performs slightly better than in Figure 9(b), which uses the non-uniform κ distribution (but both semi-supervised approaches are significantly better than the using the global eigenvectors). For Figure 9(d), we see the test error on the simplex defined by κ. To obtain this plot we sampled 500 different κ distributions according to a uniform Dirichlet distribution. With the exception of one extreme very nonuniform corner, the classification accuracy is not too sensitive to the choice of κ distribution. Thus, if we think of the semi-supervised eigenvectors as a locally-regularized version of the global eigenvectors, the desired discriminative properties are not too sensitive to the details of the locally-biased regularization.

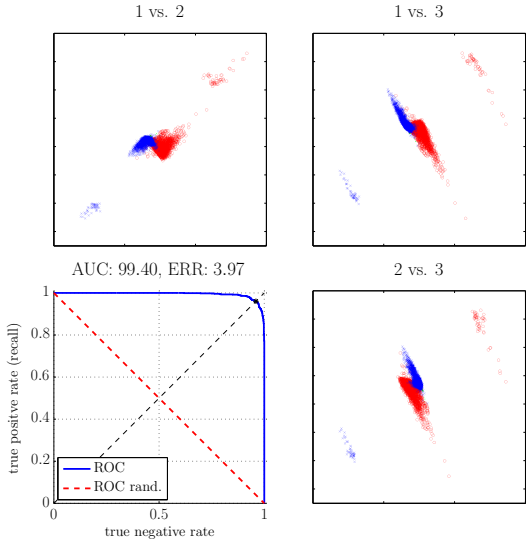
5.3.3 Effect of Approximately Computing Semi-Supervised Eigenvectors

Here, we discuss of the push-peeling procedure from Section 4 that is designed to compute efficient approximations to the semi-supervised eigenvectors by using local random walks to compute an approximation to personalized PageRank vectors. Consider Figure 10, which shows results for two values of the ϵ parameter (i.e., the parameter in the push algorithm that implicitly determines how many nodes will be touched). Using the same graph as defined previously, we compute 3 semi-supervised eigenvectors seeding using 50 samples from each class (4s vs. 9s). However, in this case, we fix the regularization parameter

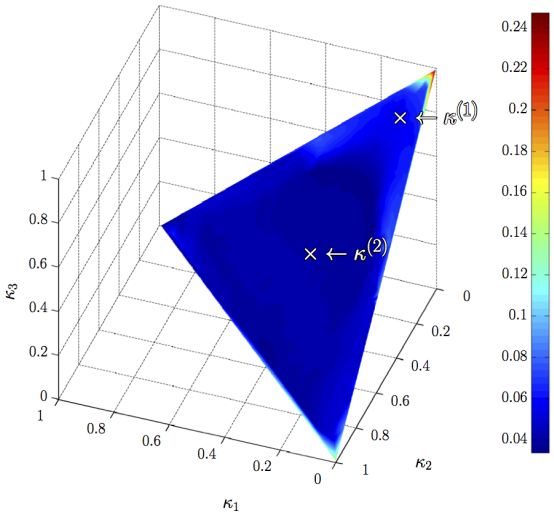


(a) Global eigenvectors

(b) Semi-supervised, with $\kappa^{(1)} = (\frac{1}{10}, \frac{1}{10}, \frac{8}{10})$



(c) Semi-supervised, with $\kappa^{(2)} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$



(d) Test error on the κ simplex

Figure 9: The effect of varying the correlation/locality parameter κ on the classification accuracy. 9(a), 9(b), 9(c) show the top three (global or semi-supervised) eigenvectors plotted against each other as well as the ROC curve for the SGT classifier discriminating between 4s and 9s; and 9(d) shows the test error as the κ vector is varied over the unit simplex.

vector as $\gamma = [-0.0150, -0.0093, -\text{vol}(G)]$; and note that choosing these specific values correspond to the solutions visualized in Figure 9(c) when the equations are solved exactly. Figure 10(a) shows the results for $\epsilon = 0.001$. This approximation gives us sparse solutions,

and the histogram in the second row illustrates the digits that are assigned a nonzero value in the respective semi-supervised eigenvector. In particular, note that most of the mass of the eigenvector is distributed on 4s and 9s; but, for this choose of ϵ, only few digits of interest ($\approx 2.8243\%$, meaning, in particular, that not all of the 4s and 9s) have been touched by the algorithm. This results in the lack of a clean separation between the two classes as one sweeps along the leading semi-supervised eigenvector, as illustrated in the first row; the very uniform correlation distribution $\kappa = [0.8789, 0.0118, 0.1093]$; and the high classification error, as shown in the ROC curve in the bottom panel.

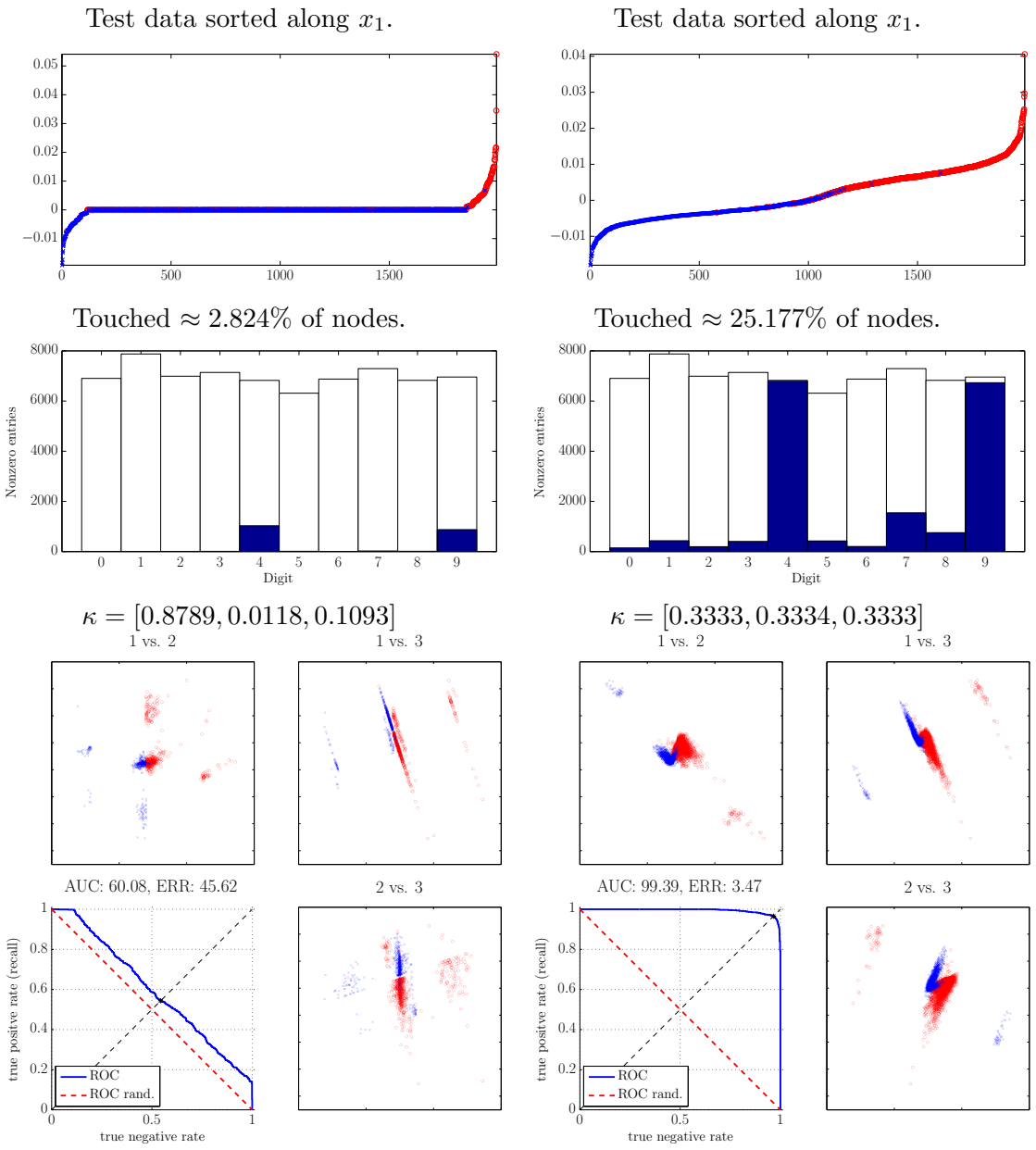
Consider, next, Figure 10(b), which shows the results for $\epsilon = 0.0001$. In this case, the algorithm reproduces the solution by touching only $\approx 25.177\%$ of the nodes in the graph, i.e., basically all of the 4s and 9s and only a few other digits. This leads to a much cleaner separation between the two classes as one sweeps over the leading semi-supervised eigenvector; a much more uniform distribution over κ; and a classification accuracy that is much better and is similar to what we saw in Figure 9(c). This example illustrates that this push-peeling approximation provides a principled manner to generalize the concept of semi-supervised eigenvectors to large-scale settings, where it will be infeasible to touch all nodes of the graph.

5.3.4 Effect of Low-Rank Nyström Approximation

Here we discuss the use of the low-rank Nyström approximation which is commonly used in large-scale kernel-based machine learning. The memory requirements for representing the explicit kernel matrix, that we here take to be our graph, scales with $\mathcal{O}(N^2)$, whereas inverting the matrix scales with $\mathcal{O}(N^3)$, which, in large-scale settings, is infeasible. The Nyström technique subsamples the data set to approximate the kernel matrix, and the memory requirements scales with $\mathcal{O}(nN)$ and runs in $\mathcal{O}(n^2N)$, where n is size of the subsample. For completeness we include the derivation of the Nyström approximation for the normalized graph Laplacian in Appendix C.

In the beginning of Section 5.3 we constructed the $70,000 \times 70,000$ k-nearest neighbor graph, with $k = 10$ and with edge weights given by $w_{ij} = \exp(-\frac{4}{\sigma_i^2}\|x_i - x_j\|^2)$. Such a sparse construction reduces the effect of "hubs", as well as being fairly insensitive to the choice of kernel parameter, as the 10 nearest neighbors are likely to be very close in the Euclidean norm. Because the Nyström method will approximate the dense kernel matrix, the choice of kernel parameter is more important, so in the following we will consider the interplay between this parameter, as well as the rank parameter n of the Nyström approximation. Moreover, to allow us to compare a rank-n Nyström approximation with the full rank-N kernel matrix, we choose to subsample the data set for all of the following experiments, due to the $\mathcal{O}(N^2)$ memory requirements. Thus, to provide a baseline, Figure 11 shows results based on a k-nearest neighbor graph constructed from 5% and 10% percent of the training data, where in both cases we used 10% for the test data. For both cases, when compared with the results of Figure 9(c), the classification quality is degraded, and so we emphasize that the goal of the following results are not to outperform the results reported in Figure 9(c), but to be comparable with this baseline.

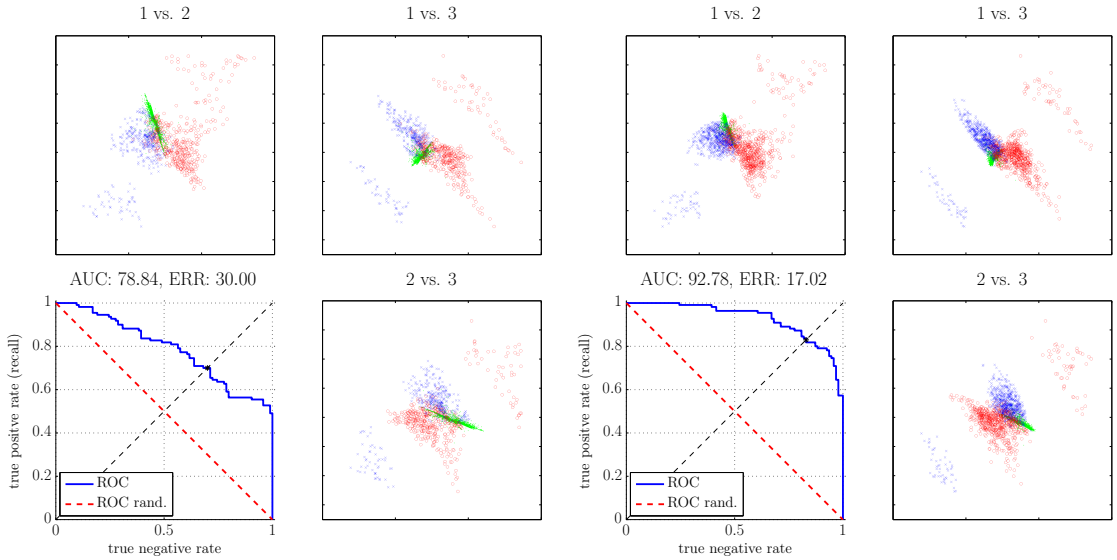
In light of this baseline, Figure 12 provides a thorough analysis for the choices of σ_i^2 that we used. Figures 12(a) and 12(b) show the classification error when using the global



(a) Locality parameter $\epsilon = 0.001$

(b) Locality parameter $\epsilon = 0.0001$

Figure 10: Illustration of the push-peeling procedure to compute 3 semi-supervised eigenvectors for $\gamma = [-0.0150, -0.0093, -\text{vol}(G)]$. 10(a) shows results for $\epsilon = 0.001$; and 10(b) shows results for $\epsilon = 0.0001$. First row shows the entries in the leading semi-supervised eigenvector corresponding to test points, color-coded and sorted according to magnitude; second row shows the distribution of digits touched in the full graph when executing the push algorithm; and bottom panels provide visualizations similar to the ones in Figure 9 (and shown above these is the correlation vector κ obtained for the fixed choice of γ.



(a) 5% of MNIST

(b) 10% of MNIST

Figure 11: Example of the impact of subsampling the data set down to 5% (in 11(a)) and 10% (in 11(b)) of the original size. Remaining parameters are the same as in Figure 9(c), which shows the result to which these two plots should be compared.

eigenvectors, for various rank approximations based on the Nyström method as well as the exact method (corresponding to rank $= n$). Interestingly, these two plots are very dissimilar in terms of their behavior as a function of the number of components. In particular, the plot in Figure 12(b) shows that the low rank approximations for a given set of components outperform the high rank approximations, and the exact representation fails to reduce the error beyond 0.4 for any of the considered set of components. This may seem counterintuitive, but the reason for this type of behavior is that the relevant global eigenvectors, for $\sigma_i^2 = 200$, are located far from the end of the spectrum (if we visualized more components for rank $= n$ the classification error would eventually drop). For the same reason, the low rank approximations improve more rapidly than the high rank approximations, as the latter approximate the lower part of the spectrum better, and these turn out to have poor discriminative properties. In contrast, the results shown in Figure 12(a) provide good class separation in the lower part of the spectrum, resulting in the high rank approximations to reduce the error most rapidly.

Finally, Figures 12(c) and 12(d) show the classification error for the SGT trained using the semi-supervised eigenvectors. (Note that the scale of the x-axis is much smaller in these subfigures.) For both kernel widths in Figures 12(c) and 12(d), the ordering of the approximations are similar, i.e., the semi-supervised eigenvectors constructed from the rank $= n$ approximation performs the best. Moreover, the gap between the rank $= 400$ and rank $= n$ is largest for $\sigma_i^2 = 200$, again suggesting this approximation is of insufficient rank to model the relevant local heterogeneities deep down in the spectrum; whereas for $\sigma_i^2 = 80$,

the rank = 400 the approximation comes very close to the exact representation, suggesting that local structures are well modeled near the end of the spectrum.

To summarize these results, the method of semi-supervised eigenvectors successfully extracts relevant local structures to perform locally-biased classification, even when they are located far from the end of the spectrum. Moreover, in both cases we considered, the classification error is reduced significantly by using only a few locally-biased components. This contrasts with the global eigenvectors, where for $\sigma_i^2 = 80$ at least 20 eigenvectors are needed in order to obtain similar performance; and for $\sigma_i^2 = 200$, the classification error remains high even for 200 eigenvectors in case of rank = n.

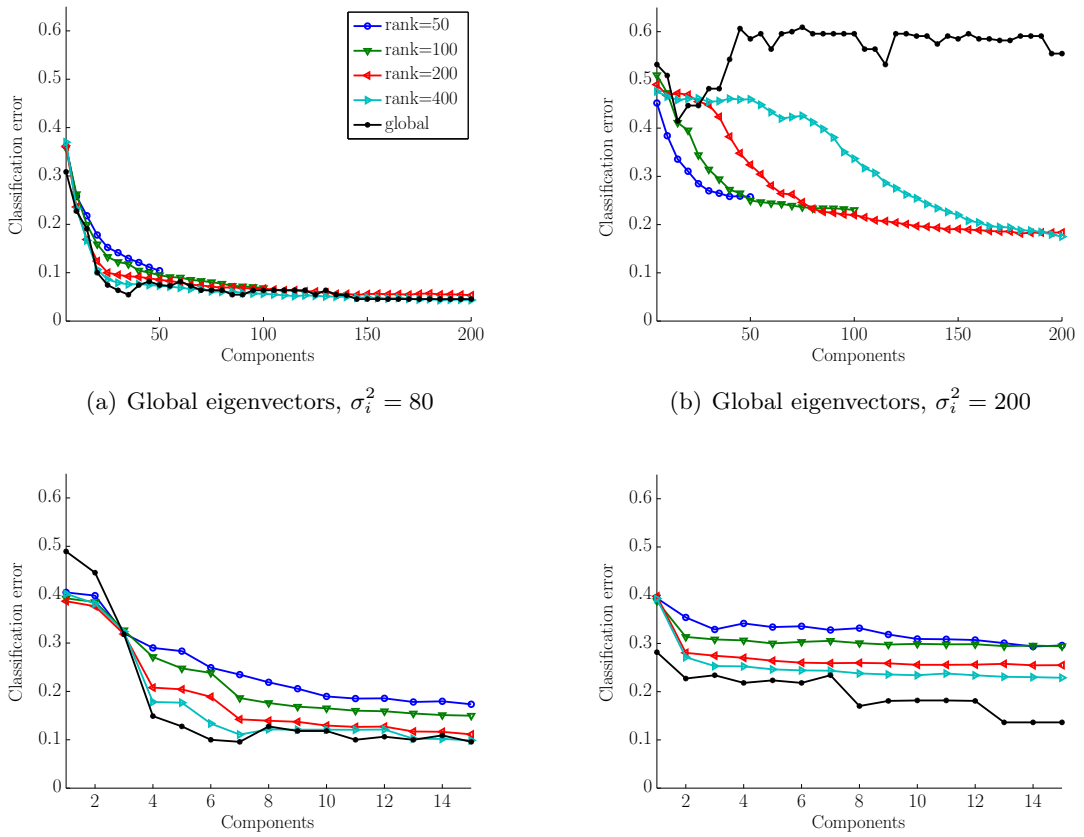
5.4 Large-scale Network Data

The final data sets we consider are from a collection of large sparse networks (Paolo et al., 2004, 2011; Paolo and Sebastiano, 2004). On these data, we demonstrate that the Push-peeling Heuristic introduced in Section 4.2 is attractive due to an improved running time, as compared to solving a system of linear equations. Moreover, we also show that the ability to obtain multiple semi-supervised eigenvectors depends on the degree heterogeneity near the seed. Finally, we empirically evaluate the influence of the ϵ parameter of the Push algorithm that implicitly determines how many nodes the algorithm will touch. This parameter can be interpreted as a regularization parameter (different from γ parameter), and setting it too large means we fail to distribute mass in the network, so that a few semi-supervised eigenvectors will consume all of the correlation. In particular, this behavior was investigated on the MNIST digits in Section 5.3.3. The basic properties for the networks considered in this section are shown in Table 2.

We start by considering the moderately sized networks from the DIMACS implementation challenge, as these networks are commonly used for the purpose of measuring realistic algorithm performance. Figure 13 shows analysis results for 6 networks from this collection, where we evaluate the performance and feasibility of the Push algorithm for approximating the leading semi-supervised eigenvector.

Network name	Number of nodes	Number of edges
DIMACS10/de2010	24,115	116,056
DIMACS10/ct2010	67,578	336,352
DIMACS10/il2010	451,554	2,164,464
DIMACS10/smallworld	100,000	999,996
DIMACS10/333SP	3,712,815	22,217,266
DIMACS10/AS365	3,799,275	22,736,152
LAW/arabic-2005	22,744,080	1,107,806,146
LAW/indochina-2004	7,414,866	301,969,638
LAW/it-2004	41,291,594	2,054,949,894
LAW/sk-2005	50,636,154	3,620,126,660
LAW/uk-2002	18,520,486	523,574,516
LAW/uk-2005	39,459,925	1,566,054,250

Table 2: Summary of the networks considered in this section. Some of these networks are directed and have been symmetrized for the purpose of this analysis, i.e., the number edges in this table refer to the number of edges in the undirected graph.



(a) Global eigenvectors, $\sigma_i^2 = 80$

(b) Global eigenvectors, $\sigma_i^2 = 200$

(c) Semi-supervised eigenvectors, $\sigma_i^2 = 80$

(d) Semi-supervised eigenvectors, $\sigma_i^2 = 200$

Figure 12: We consider 10% of the MNIST training and test data and investigate the classification accuracy of a downstream SGT classifier for various approximations of the dense similarity matrix. 12(a) and 12(b): Classification error for the SGT evaluated directly on global eigenvectors, based on various Nyström approximations and the two choices of the kernel width parameter (respectively, $\sigma_i^2 = 80$ and $\sigma_i^2 = 200$). 12(c) and 12(d): Classification error we have used the Nyström approximations as basis for computing semi-supervised eigenvectors that are then used in the downstream SGT classifier. All plots show the mean over 30 repetitions.

As stated in Section 3.3, diffusion based procedures such as the Push algorithm can be used to solve our objective for $\gamma < 0$. The impact of the reduced search range is that such procedures may not be able to produce a uniform correlation distribution for a set of semi-supervised eigenvectors. Hence, the leading solution(s) will instead pickup too much correlation, because sufficient mass cannot to diffuse away from the seed set. However, the effect of a non-uniform correlation distribution was analyzed on the MNIST data in Section 5.3, where we found that the performance of a downstream classifier is fairly robust to such

non-uniformities, as seen by the simplex in Figure 9. Consequently, we emphasize that in a large-scale setting such side effects of diffusion based procedures is offset by the advantage of a greatly improved time complexity as compared to solving the system of linear equations, that implicitly touch every node.

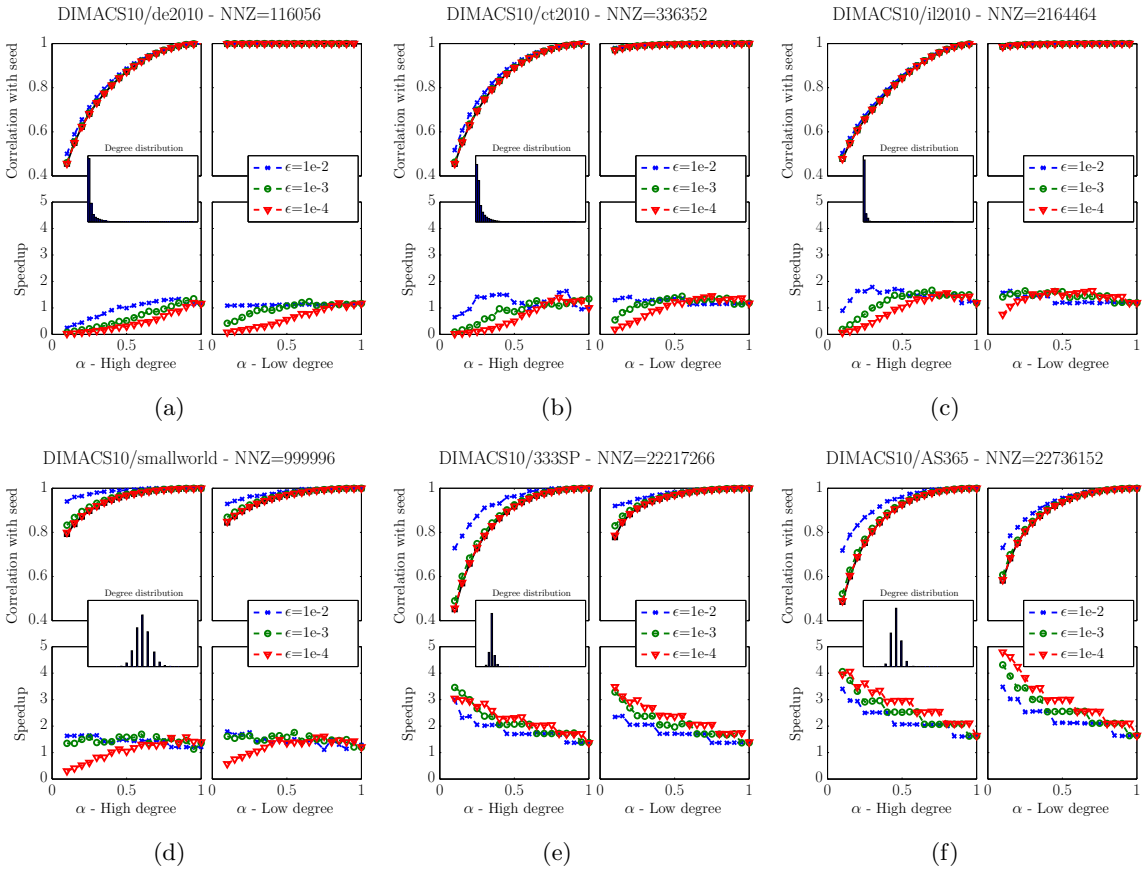


Figure 13: For each network the first row depicts how the correlation decays as α tends towards 0, whereas the bottom row shows the speedup relative to the standard approach using conjugate gradient with a tolerance of 1e-6, that is the default approach in our software distribution. Besides the three considered values of ϵ the correlation plots also illustrate the decay based on conjugate gradient (black curve), however this may be difficult to see, as the Push algorithm for $\epsilon = 1\text{e-}4$ coincides with that solution. Finally, seeds based on a high degree and low degree node are presented in respectively the first and last column, and the degree distribution for the network is visualized in a minor overlapping plot.

For each of the 6 analyzed networks in Figure 13, we run two experiments considering different seeds, using respectively a high degree and low degree single seed node. Figure 13(a)-13(c) considers census block networks characterized by heavy-tailed degree distributions, whereas Figure 13(d)-13(f) considers more densely connected synthetic networks. For

each of these 6 networks the speedup is measured by comparing with a standard conjugate gradient implementation using a tolerance of 1e-6, and we stress that this tolerance cannot be directly compared with ϵ in the Push algorithm. Moreover, we test three different settings of the ϵ parameter, and we emphasize that for $\epsilon =$ 1e-4, the Push algorithm produces a similar result as the conjugate gradient algorithm. In Figure 13 this can be seen by the red curve ($\epsilon =$ 1e-4) in the correlation decay plots (see the figure caption) being on top of the black curve (conjugate gradient).

Common for Figure 13(a)-13(c) are that low degree seed nodes yield very localized solutions for the entire range of α, opposed to the high degree nodes that all succeed in gradually reducing the correlation when α is reduced. Also, the choice of ϵ is obviously very important, i.e., choosing it too large results in a solution that correlates too much with the seed, whereas choosing it too small means that we will be touching more nodes than necessary, resulting in a performance penalty. In general the networks analyzed in Figure 13(a)-13(c) are too small to yield significant performance improvements over the conjugate gradient algorithm, and the Push algorithm is only competitive for large values of α.

For the network in Figure 13(d), we see similar performance characteristics as the networks analyzed in Figure 13(a)-13(c) due to its small size. However, the two final networks analyzed in Figure 13(e)-13(f) share similar characteristics in terms of the degree distribution, but due to a much larger size they show significant performance improvements over the conjugate gradient algorithm. Interestingly, the Push algorithm instantiated with $\epsilon =$ 1e-4 yields a greater speedup in some settings, which may be explained by faster convergence, caused by a reduced threshold for distributing mass. Hence, the running time of the Push algorithm may not always decrease monotonically as ϵ increases.

In general it seems that seeding in a sparsely connected region of a network results in a solution having a large correlation with the seed for most values of α. This is obviously a limiting factor if we are interested in using the peeling procedure to find multiple semi-supervised eigenvectors in that particular region. However, for large networks and more densely connected regions the benefit of the Push algorithm is immediate.

Finally, we scale up to demonstrate that we can adapt the notion of semi-supervised eigenvectors to large data sets, and we do so by analyzing 6 large web-crawl networks. These networks are large enough that touching all nodes is infeasible, i.e., conjugate gradient is not a feasible option, so in Figure 14 we resort to absolute timings. For the analysis results shown in Figure 14, we are solely interested in giving the reader some intuition about the running time in a large-scale setting, as well as an idea on how the parameters interplay. Hence, we only consider experiments where we seed in a high degree node, as these are likely yield the worst running times, but also succeed in reducing the correlation the most. This will make the peeling procedure described in Section 4.2 applicable, allowing us to obtain multiple semi-supervised eigenvectors. As seen for all networks analyzed in Figure 14(a)-14(f) the solution is highly sensitive to the choice of ϵ, but for all networks we are able to reduce the correlation when α tends towards 0 in case of $\epsilon =$ 1e-6. We emphasize that the reason for ϵ being smaller for these experiments, as compared to the previous is that the seed is normalized to have unit norm, implicitly requiring a lower ϵ when the network increases in size.

For diffusion based procedures to be useful with respect to the computation of semi-supervised eigenvectors, mass must be able "bleed" away from the seed set and into the

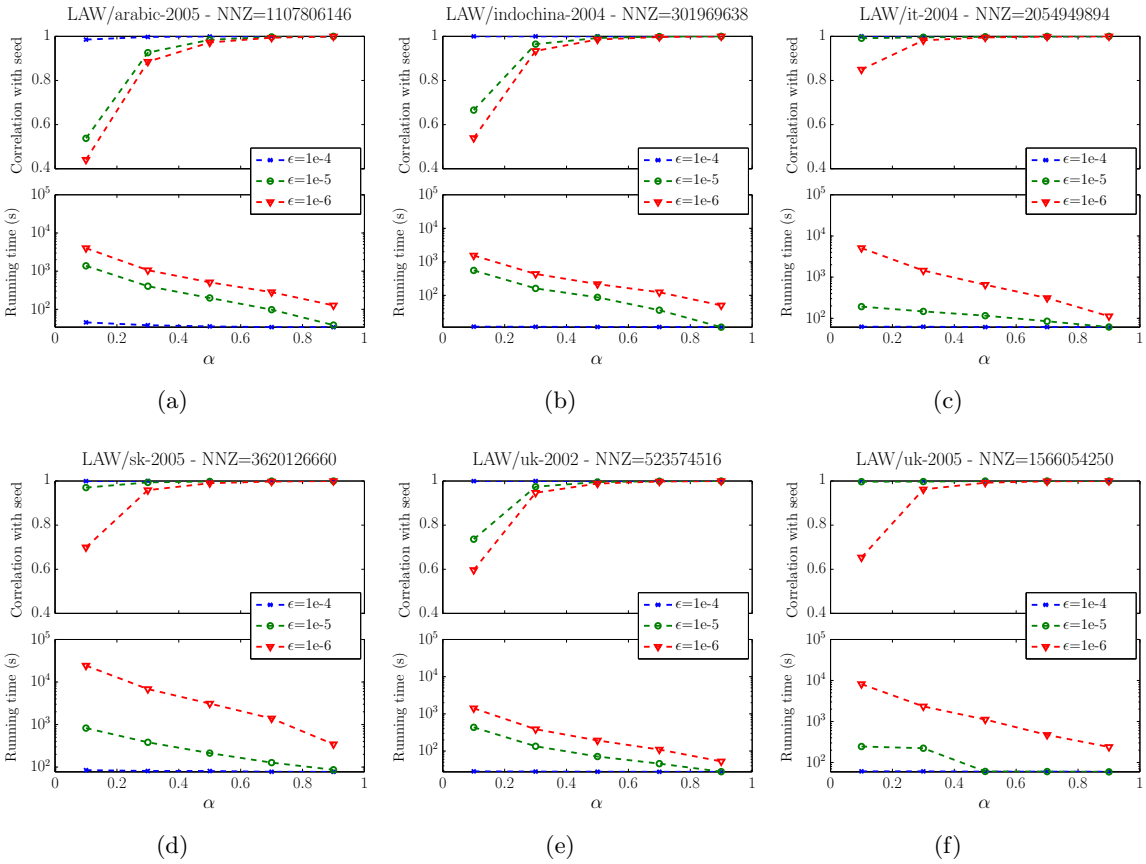


Figure 14: Visualizes results for applying the Push algorithm to 6 very large web-crawl networks. For all networks we seed in the node with the highest degree. The top plot in each subfigure shows the correlation decay as a function of α, whereas in the bottom plot we resort to absolute timings as the conjugate gradient algorithm is not feasible in this setting, as opposed to showing speedups as in Figure 13.

surrounding network. Otherwise only few semi-supervised eigenvectors can be found as the leading solution(s) become too correlated with the seed set. For moderately sized problems conjugate gradient performs very well, but in a large-scale setting, as considered here, the presented approach proves very efficient, allowing us to compute approximations to semi-supervised eigenvectors in networks consuming more than 30GB of working memory. Obtaining an improved understanding of how the method of semi-supervised eigenvectors can be used to perform common machine learning tasks on graphs of that size is an obvious direction raised by our work.

6. Conclusion

We have introduced the concept of semi-supervised eigenvectors as local analogues of the global eigenvectors of a graph Laplacian that have proven so useful in a wide range of machine learning and data analysis applications. These vectors are biased toward prespecified local regions of interest in a large data graph; and we have shown that since they inherit many of the nice properties of the usual global eigenvectors, except in a locally-biased context, they can be used to perform locally-biased machine learning. The basic method is conceptually simple and involves solving a sequence of linear equation problems; we have also presented several extensions of the basic method that have improved scaling properties; and we have illustrated the behavior of the method. Due to the speed, simplicity, stability, and intuitive appeal of the method, as well as the range of applications in which local regions of a large data set are of interest, we expect that the method of semi-supervised eigenvectors can prove useful in a wide range of machine learning and data analysis applications.

Acknowledgments

Partial support of this work has been provided by the ARO, DARPA, and NSF.

Appendix A. Supplementary Proofs

Lemma 1 *Given an SPSD matrix M and some vector x where $x^\top x = 1$, it holds that*

$$\lim_{\omega\to\infty} \left(M + \omega xx^\top\right)^+ = \left(\left(I - xx^\top\right) M \left(I - xx^\top\right)\right)^+. \tag{14}$$

Proof: Prior to applying the pseudo inverse, x is clearly an eigenvector with eigenvalue $\lambda = 0$ on the right hand side, and for left hand side x is an eigenvector with eigenvalue $\lambda = \infty$. Hence, without loss of generality we can decompose $M = \alpha xx^\top + X_\perp \Lambda X_\perp^\top$, where Λ is a diagonal matrix, such that $M^+ = \frac{1}{\alpha} xx^\top + X_\perp \Lambda^+ X_\perp^\top$. First we consider the expansion of the left hand side of Eqn. (14)

$$\lim_{\omega\to\infty} \left((\alpha+\omega)\, xx^\top + X_\perp \Lambda X_\perp^\top\right)^+ = \lim_{\omega\to\infty} \frac{1}{\alpha+\omega} xx^\top + X_\perp \Lambda^+ X_\perp^\top = X_\perp \Lambda^+ X_\perp^\top.$$

Similar, by expanding the right hand side we get

$$\left(\left(I - xx^\top\right)\left(\alpha xx^\top + X_\perp \Lambda X_\perp^\top\right)\left(I - xx^\top\right)\right)^+ = \left(X_\perp \Lambda X_\perp^\top\right)^+ = X_\perp \Lambda^+ X_\perp^\top.$$

■

Lemma 2 *For $M' = M + \omega \sum_i x_i x_i^\top$ where $\omega \geq 0$ it holds that $\lambda_k(M') \geq \lambda_k(M)$.*

Proof: All eigenvalues of the sum of rank-1 perturbations are non-negative

$$\omega \sum_i x_i x_i^\top \succeq 0 \Rightarrow M' \succeq M.$$

■

Lemma 3 *Given an orthonormal basis, $X = [x_1, \ldots, x_{n-1}]$, i.e., $X^\top D_G X = I$, and unit length seed $s^\top D_G s = 1$. Then, any unit length vector $x_n^\top D_G x_n = 1$, perpendicular to the subspace $X^\top D_G x_n = 0$, will have a correlation with the seed bounded by*

$$0 \leq (x_n^\top D_G s)^2 \leq 1 - \sum_{i=1}^{n-1} (x_i^\top D_G s)^2.$$

Proof: The proof follows directly from the Pythagorean theorem. Let $X = [x_1, \ldots, x_N]$ be the orthonormal basis of $\mathbb{R}^N$, i.e., spanning s. Then

$$\sum_{i=1}^{N} (x_i^\top D_G s)^2 = (s^\top D_G s)^2 = 1.$$

■

Lemma 4 *For the matrix $P_\gamma = \mathcal{L}_G - \gamma I$ it holds that*

$$P_\gamma^+ - P_{\hat{\gamma}}^+ = (\gamma - \hat{\gamma}) P_{\hat{\gamma}}^+ P_\gamma^+, \tag{15}$$

given that neither γ nor $\hat{\gamma}$ coincides with an eigenvalue of $\mathcal{L}_G$.

Proof: The proof follows directly by plain algebra. Simply substitute the SVD $P_\gamma = V \Lambda_\gamma V^T$, where Λ_γ is a diagonal matrix with the eigenvalues shifted by γ, into Eqn. (15)

$$\begin{aligned} V\Lambda_\gamma^+ V^T - V\Lambda_{\hat{\gamma}}^+ V^T &= (\gamma - \hat{\gamma}) V \Lambda_{\hat{\gamma}}^+ V^T V \Lambda_\gamma^+ V^T \\ V\Lambda_\gamma^+ V^T - V\Lambda_{\hat{\gamma}}^+ V^T &= (\gamma - \hat{\gamma}) V \Lambda_{\hat{\gamma}}^+ \Lambda_\gamma^+ V^T \\ \Rightarrow \Lambda_\gamma^+ - \Lambda_{\hat{\gamma}}^+ &= (\gamma - \hat{\gamma}) \Lambda_{\hat{\gamma}}^+ \Lambda_\gamma^+. \end{aligned}$$

The system is decoupled so it will be sufficient to consider a single eigenvalue

$$\begin{aligned} \frac{1}{\lambda_i - \gamma} - \frac{1}{\lambda_i - \hat{\gamma}} &= \frac{\gamma - \hat{\gamma}}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)} \\ \frac{\lambda_i - \hat{\gamma}}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)} - \frac{\lambda_i - \gamma}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)} &= \frac{\gamma - \hat{\gamma}}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)} \\ \frac{\gamma - \hat{\gamma}}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)} &= \frac{\gamma - \hat{\gamma}}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)}. \end{aligned}$$

Also, this trivially holds for the rank deficient case, i.e., $0 = 0$.

■

Lemma 5 *As pointed out in Section 3.3, it is already immediate that the initial semi-supervised eigenvector can be computed using a diffusion-based procedure, such as the Push algorithm. However, from that discussion it remains unclear how the approach can be generalized for the consecutive $k-1$ semi-supervised eigenvectors. It turns out that the k^{th} solution is approximated by*

$$x_k \approx c(I - XX^T D_G)(L_G - \gamma_k D_G)^+ D_G s, \tag{16}$$

given that $(L_G - \gamma_k D_G)^+ D_G s$ is linearly independent with respect to the previous $k-1$ solutions contained in X.

Proof: By Eqn. (9) the solution for the second semi-supervised eigenvector can be expressed as

$$y_2 = c\left(P_{\gamma_2}{}^{+} - P_{\gamma_2}{}^{+} y_1 (y_1^T P_{\gamma_2}{}^{+} y_1)^{+} y_1^T P_{\gamma_2}{}^{+}\right) D_G^{1/2} s,$$

where $(y_1^T P_{\gamma_2}{}^{+} y_1)^{+}$ is a constant. For notational convenience we start by substituting $b = D_G^{1/2} s$ together with the explicit solution $y_1 \propto P_{\gamma_1}{}^{+} b$

$$y_2 = c P_{\gamma_2}{}^{+} b - \frac{c P_{\gamma_2}{}^{+} P_{\gamma_1}{}^{+} b b^T P_{\gamma_1}{}^{+} P_{\gamma_2}{}^{+} b}{b^T P_{\gamma_1}{}^{+} P_{\gamma_2}{}^{+} P_{\gamma_1}{}^{+} b},$$

and for the same reason we also introduce $\rho_{\gamma_1\gamma_2} = b^\top P_{\gamma_1}{}^{+} P_{\gamma_2}{}^{+} b$

$$y_2 = c P_{\gamma_2}{}^{+} b - \frac{c \rho_{\gamma_1\gamma_2} P_{\gamma_2}{}^{+} P_{\gamma_1}{}^{+} b}{b^T P_{\gamma_1}{}^{+} P_{\gamma_2}{}^{+} P_{\gamma_1}{}^{+} b}.$$

We can approximate this expression by exploiting the structural result of Lemma 4, namely that $P_{\gamma_1}{}^{+} - P_{\gamma_2}{}^{+} = (\gamma_1 - \gamma_2) P_{\gamma_2}{}^{+} P_{\gamma_1}{}^{+}$

$$\begin{aligned} y_2 &\approx c P_{\gamma_2}{}^{+} b - \frac{c \rho_{\gamma_1\gamma_2} (P_{\gamma_1}{}^{+} - P_{\gamma_2}{}^{+}) b}{b^T P_{\gamma_1}{}^{+} (P_{\gamma_1}{}^{+} - P_{\gamma_2}{}^{+}) b} \\ &= c P_{\gamma_2}{}^{+} b - \frac{c \rho_{\gamma_1\gamma_2} (P_{\gamma_1}{}^{+} - P_{\gamma_2}{}^{+}) b}{\rho_{\gamma_1\gamma_1} - \rho_{\gamma_1\gamma_2}}. \end{aligned}$$

We emphasize that this approximation is exact whenever $P_{\gamma_1}{}^{+} - P_{\gamma_2}{}^{+}$ is well-conditioned, and singular for $\gamma_1 = \gamma_2$. Then, substitute $c = \frac{\rho_{\gamma_1\gamma_1} - \rho_{\gamma_1\gamma_2}}{\rho_{\gamma_1\gamma_1}}$

$$\begin{aligned} y_2 &\approx \frac{\rho_{\gamma_1\gamma_1} P_{\gamma_2}{}^{+} b - \rho_{\gamma_1\gamma_2} P_{\gamma_2}{}^{+} b}{\rho_{\gamma_1\gamma_1}} - \frac{\rho_{\gamma_1\gamma_2} (P_{\gamma_1}{}^{+} - P_{\gamma_2}{}^{+}) b}{\rho_{\gamma_1\gamma_1}} \\ &= \frac{\rho_{\gamma_1\gamma_1} P_{\gamma_2}{}^{+} b - \rho_{\gamma_1\gamma_2} P_{\gamma_2}{}^{+} b - \rho_{\gamma_1\gamma_2} P_{\gamma_1}{}^{+} b + \rho_{\gamma_1\gamma_2} P_{\gamma_2}{}^{+} b}{\rho_{\gamma_1\gamma_1}} \\ &= \frac{\rho_{\gamma_1\gamma_1} P_{\gamma_2}{}^{+} b - \rho_{\gamma_1\gamma_2} P_{\gamma_1}{}^{+} b}{\rho_{\gamma_1\gamma_1}} \\ &= P_{\gamma_2}{}^{+} b - \frac{\rho_{\gamma_1\gamma_2} P_{\gamma_1}{}^{+} b}{\rho_{\gamma_1\gamma_1}}. \end{aligned}$$

By resubstituting for the auxiliary variables we obtain the desired result

$$y_2 \approx c(I - y_1 y_1^T)(\mathcal{L}_G - \gamma I)^{+} D_G^{1/2} s,$$

and by applying this procedure recursively it follows that

$$y_k \approx c(I - Y Y^T)(\mathcal{L}_G - \gamma_k I)^{+} D_G^{1/2} s.$$

Finally, we can relate this result to the combinatorial graph Laplacian as follows

$$\begin{aligned} y_k &\approx c(I - D_G^{1/2} X X^T D_G^{1/2}) D_G^{1/2} (L_G - \gamma_k D_G)^+ D_G s \\ &= c(D_G^{1/2} - D_G^{1/2} X X^T D_G)(L_G - \gamma_k D_G)^+ D_G s \\ &= c D_G^{1/2} (I - X X^T D_G)(L_G - \gamma_k D_G)^+ D_G s, \end{aligned}$$

and due to the relationship $x_k = D_G^{-1/2} y_k$ it follows that

$$x_k \approx c(I - X X^T D_G)(L_G - \gamma_k D_G)^+ D_G s.$$

■

Appendix B. Derivation of Sparse Graph Diffusions

To allow efficient computation of semi-supervised eigenvectors by graph diffusions, we must make the relationship with the sparse seed vector explicit. Here we specifically consider the derivation of Eqn. (13). Given a sparse seed indicator s_0, we can write the seed vector s as $s \propto D_G^{-1/2}(I - v_0 v_0^T) s_0$, where $v_0 \propto \mathrm{diag}(D^{1/2})$ is the leading eigenvector of the normalized graph Laplacian (corresponding to the all-one vector of the combinatorial graph Laplacian). Using this explicit form of s we can rewrite the leading solution as

$$\begin{aligned} x_1 &= c(L_G - \gamma D_G)^+ D_G s \\ &= c D_G^{-1/2} (\mathcal{L}_G - \gamma I)^+ D_G^{1/2} s \\ &= c D_G^{-1/2} (\mathcal{L}_G - \gamma I)^+ D_G^{1/2} D_G^{-1/2} (I - v_0 v_0^T) s_0 \\ &= c D_G^{-1/2} \left((\mathcal{L}_G - \gamma I)^+ s_0 - (\mathcal{L}_G - \gamma I)^+ v_0 v_0^T s_0 \right). \end{aligned}$$

Since $\mathcal{L}_G - \gamma I$ simply shifts the eigenvalues of $\mathcal{L}_G$ by $-\gamma$, the latter term simplifies to

$$\begin{aligned} x_1 &= c D_G^{-1/2} \left((\mathcal{L}_G - \gamma I)^+ s_0 - \left(\frac{1}{-\gamma} v_0 v_0^T \right) v_0 v_0^T s_0 \right) \\ &= c D_G^{-1/2} \left((\mathcal{L}_G - \gamma I)^+ s_0 + \frac{1}{\gamma} v_0 v_0^T s_0 \right) \\ &= c D_G^{-1/2} \left(\frac{1}{-\gamma} D_G^{-1/2} \mathrm{pr}_\epsilon \left(\frac{\gamma}{\gamma - 2}, D_G^{1/2} s_0 \right) + \frac{1}{\gamma} v_0 v_0^T s_0 \right). \end{aligned}$$

Finally, by exploiting the peeling result in Eqn. (16), we can use the Push algorithm to approximate the sequence of semi-supervised eigenvectors in an extremely efficient manner

$$x_t^* \approx c \left(I - X X^T D_G \right) \left(D_G^{-1} \mathrm{pr}_\epsilon \left(\frac{\gamma_t}{\gamma_t - 2}, D_G^{1/2} s_0 \right) - D_G^{-1/2} v_0 v_0^T s_0 \right),$$

as the Push algorithm is only applied on the sparse seed set.

Appendix C. Nyström Approximation For The Normalized Graph Laplacian

The vanilla procedure is as follows; we choose m samples at random from the full data set, and for notational simplicity we reorder the samples so that these m samples are followed by the remaining $n = N - m$ samples, i.e., we can partition the adjacency matrix as

$$A_G = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix},$$

where $A \in \mathbb{R}^{m \times m}$, $B \in \mathbb{R}^{m \times n}$, and $C \in \mathbb{R}^{n \times n}$, with $N = m + n$ and $m \ll n$. The Nyström extension then approximates the huge C matrix in terms of A and B, so the resulting approximation to weight matrix becomes

$$A_G \approx \hat{A}_G = \begin{pmatrix} A & B \\ B^T & B^T A^{-1} B \end{pmatrix}.$$

Hence, rather than encoding only each nodes k-nearest-neighbors into the weight matrix, the Nyström methods provides a low-rank approximation to the entire dense weight matrix. Since the leading eigenvectors of $D_G^{-1/2} A_G D_G^{-1/2}$ correspond to the smallest of $\mathcal{L}_G$, our goal is to diagonalize $D_G^{-1/2} A_G D_G^{-1/2}$. At the risk of washing out the local heterogeneities the Nyström procedure approximates the largest eigenvectors of $D_G^{-1/2} A_G D_G^{-1/2}$ using the normalized matrices $\tilde{A}$ and $\tilde{B}$

$$\tilde{A}_{ij} = \frac{A_{ij}}{\sqrt{\hat{d}_i \hat{d}_j}}, \quad i, j = 1, \ldots m$$
$$\tilde{B}_{ij} = \frac{B_{ij}}{\sqrt{\hat{d}_i \hat{d}_{j+m}}}, \quad i = 1, \ldots m, \quad j = 1, \ldots, n.$$

Finally, let $U \Lambda U^T$ be the SVD of $\tilde{A} + \tilde{A}^{-1/2} \tilde{B} \tilde{B}^T \tilde{A}^{-1/2}$, then the m leading eigenvectors are approximated by

$$V = \begin{pmatrix} \tilde{A} \\ \tilde{B}^T \end{pmatrix} \tilde{A}^{-1/2} U \Lambda^{-1/2},$$

and the normalized graph Laplacian by $\mathcal{L}_G \approx I - V \Lambda V^T$.

References

R. Andersen and K. Lang. Communities from seed sets. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, pages 223–232, 2006.

R. Andersen, F.R.K. Chung, and K. Lang. Local graph partitioning using PageRank vectors. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 475–486, 2006.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

M. Belkin and P. Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning*, 56:209–239, 2004.

M. Belkin and P. Niyogi. Towards a theoretical foundation for Laplacian-based manifold methods. *Journal of Computer and System Sciences*, 74(8):1289–1308, 2008.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

P. Boldi and S. Vigna. The push algorithm for spectral ranking. Technical report, 2011. Preprint: arXiv:1109.4680 (2011).

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

O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems 15: Proceedings of the 2002 Conference*, pages 585–592, 2003.

F.R.K. Chung. *Spectral graph theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 1997.

R.R. Coifman, S. Lafon, A.B. Lee, M. Maggioni, B. Nadler, F. Warner, and S.W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition in data: Diffusion maps. *Proceedings of the National Academy of Sciences of the United States of America*, 102(21):7426–7431, 2005.

M. Cucuringu and M. W. Mahoney. Localization on low-order eigenvectors of data matrices. Technical report, 2011. Preprint: arXiv:1109.1355 (2011).

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.

A. P. Eriksson, C. Olsson, and F. Kahl. Normalized cuts revisited: A reformulation for segmentation with linear grouping constraints. In *Proceedings of the 11th International Conference on Computer Vision*, pages 1–8, 2007.

W. Gander, G. H. Golub, and U. von Matt. A constrained eigenvalue problem. *Linear Algebra and its Applications*, 114/115:815–839, 1989.

A. Gittens and M. W. Mahoney. Revisiting the Nyström method for improved large-scale machine learning. Technical report, 2012. Preprint arXiv:1303.1849 (2012).

J. Ham, D.D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the 21st International Conference on Machine Learning*, pages 000–000, 2004.

T.H. Haveliwala. Topic-sensitive PageRank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):784–796, 2003.

G. Jeh and J. Widom. Scaling personalized web search. In *WWW '03: Proceedings of the 12th International Conference on World Wide Web*, pages 271–279, 2003.

T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the 20th International Conference on Machine Learning*, pages 290–297, 2003.

R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th International Conference on Machine Learning*, pages 315–322, 2002.

Y. Lecun and C. Cortes. The MNIST database of handwritten digits. URL `http://yann.lecun.com/exdb/mnist/`. `http://yann.lecun.com/exdb/mnist/`.

J. Leskovec, K.J. Lang, A. Dasgupta, and M.W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW '08: Proceedings of the 17th International Conference on World Wide Web*, pages 695–704, 2008.

M. W. Mahoney and L. Orecchia. Implementing regularization implicitly via approximate eigenvector computation. In *Proceedings of the 28th International Conference on Machine Learning*, pages 121–128, 2011.

M. W. Mahoney, L. Orecchia, and N. K. Vishnoi. A local spectral method for graphs: with applications to improving graph partitions and exploring data graphs locally. *Journal of Machine Learning Research*, 13:2339–2365, 2012.

S. Maji, N. K. Vishnoi, and J. Malik. Biased normalized cuts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2057–2064, 2011.

P.J. Mucha, T. Richardson, K. Macon, M.A. Porter, and J.P. Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878, 2010.

A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 15: Proceedings of the 2001 Conference*, 2001.

K. A. Norman, S. M. Polyn, G. J. Detre, and J. V. Haxby. Beyond mind-reading: multi-voxel pattern analysis of fMRI data. *Trends in Cognitive Sciences*, 10(9):424–430, 2006.

L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

B. Paolo and V. Sebastiano. The WebGraph framework I: Compression techniques. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*, pages 595–601, Manhattan, USA, 2004. ACM Press.

B. Paolo, C. Bruno, S. Massimo, and V. Sebastiano. Ubicrawler: A scalable fully distributed web crawler. *Software: Practice & Experience*, 34(8):711–726, 2004.

B. Paolo, R. Marco, S. Massimo, and V. Sebastiano. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *Proceedings of the 20th International Conference on World Wide Web*. ACM Press, 2011.

P. O. Perry and M. W. Mahoney. Regularized Laplacian estimation and fast eigenvector approximation. In *Advances in Neural Information Processing Systems 25: Proceedings of the 2011 Conference*, 2011.

K.T. Poole. The decline and rise of party polarization in congress during the twentieth century. *Extensions: A Journal of the Carl Albert Congressional Research and Studies Center*, Fall 2005.

K.T. Poole and H. Rosenthal. Patterns of congressional voting. *American Journal of Political Science*, 35:228–278, 1991.

A. Pothen, H.D. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, 1990.

S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by local linear embedding. *Science*, 290:2323–2326, 2000.

L. K. Saul, K. Q. Weinberger, J. H. Ham, F. Sha, and D. D. Lee. Spectral methods for dimensionality reduction. In O. Chapelle, B. Schoelkopf, and A. Zien, editors, *Semisupervised Learning*, pages 293–308. MIT Press, 2006.

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

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transcations of Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

D.A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC '04: Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 81–90, 2004.

M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In *Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Conference*, pages 945–952, 2002.

A. Talwalkar and A. Rostamizadeh. Matrix coherence and the Nyström method. In *Proceedings of the 26th Conference in Uncertainty in Artificial Intelligence*, 2010.

J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

S.-H. Teng. The Laplacian paradigm: Emerging algorithms for massive graphs. In *Proceedings of the 7th Annual Conference on Theory and Applications of Models of Computation*, pages 2–14, 2010.

S. Vigna. Spectral ranking. Technical report, 2009. Preprint: arXiv:0912.0238 (2009).

D.J. Watts and S.H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393: 440–442, 1998.

A. S. Waugh, L. Pei, J. H. Fowler, P. J. Mucha, and M. A. Porter. Party polarization in congress: A network science approach. Technical report, 2009. Preprint: arXiv:0907.3509 (2009).

C.K.I. Williams and M. Seeger. The effect of the input density distribution on kernel-based classifiers. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1159–1166, 2000.

C.K.I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, pages 682–688, 2001.

S. X. Yu and J. Shi. Grouping with bias. In *Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Conference*, pages 1327–1334, 2002.

D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*, pages 321–328, 2004.

Journal of Machine Learning Research 15 (2014) 3915-3919 Submitted 7/13; Revised 5/14; Published 11/14

BayesOpt: A Bayesian Optimization Library for Nonlinear Optimization, Experimental Design and Bandits

Ruben Martinez-Cantin RMCANTIN@UNIZAR.ES
Centro Universitario de la Defensa
Zaragoza, 50090, Spain

Editor: Balázs Kégl

Abstract

BayesOpt is a library with state-of-the-art Bayesian optimization methods to solve nonlinear optimization, stochastic bandits or sequential experimental design problems. Bayesian optimization characterized for being sample efficient as it builds a posterior distribution to capture the evidence and prior knowledge of the target function. Built in standard C++, the library is extremely efficient while being portable and flexible. It includes a common interface for C, C++, Python, Matlab and Octave.

Keywords: Bayesian optimization, efficient global optimization, sequential model-based optimization, sequential experimental design, Gaussian processes

1. Introduction

Bayesian optimization (Mockus, 1989; Brochu et al., 2010) is a special case of nonlinear optimization where the algorithm *decides* which point to explore next based on the analysis of a distribution over functions $P(f)$, for example a Gaussian process or other *surrogate model*. The decision is taken based on a certain criterion $\mathcal{C}(\cdot)$ called *acquisition function*. Bayesian optimization has the advantage of having a *memory* of all the observations, encoded in the posterior of the surrogate model $P(f|\mathcal{D})$ (see Figure 1). Usually, this posterior distribution is sequentially updated using a nonparametric model. In this setup, each observation improves the knowledge of the function in all the input space, thanks to the spatial correlation (kernel) of the model. Consequently, it requires a lower number of iterations compared to other nonlinear optimization algorithms. However, updating the posterior distribution and maximizing the acquisition function increases the cost per sample. Thus, Bayesian optimization is normally used to optimize *expensive* target functions $f(\cdot)$, which can be multimodal or without closed form. The quality of the prior and posterior information about the surrogate model is of paramount importance for Bayesian optimization, because it can reduce considerably the number of evaluations to achieve the same performance.

2. BayesOpt Library

BayesOpt uses a surrogate model of the form: $f(\mathbf{x}) = \phi(\mathbf{x})^T\mathbf{w} + \epsilon(\mathbf{x})$, where we have $\epsilon(\mathbf{x}) \sim \mathcal{TP}\left(0, \sigma_s^2(\mathbf{K}(\theta) + \sigma_n^2\mathbf{I})\right)$. Here, $\mathcal{TP}()$ means a Student-t process or a mixture of Student-t processes, with the Gaussian process as a special case. This model can be considered as a linear regression model $\phi(\mathbf{x})^T\mathbf{w}$ with heteroscedastic perturbation $\epsilon(\mathbf{x})$, as a nonparametric

Input: target $f(\cdot)$, priors $P(f)$, $P(\theta)$, criterion $\mathcal{C}(\cdot)$, budget N Output: $\mathbf{x}^*$

Build a data set $\mathcal{D}$ of points $\mathbf{X} = \mathbf{x}_1 \dots \mathbf{x}_l$ and its response $\mathbf{y} = y_1 \dots y_l$ using an initial design.
While $i < N$

- Update the distribution with all data available, for example, $P(f|\mathcal{D}) \propto \int P(\mathcal{D}|f,\theta)P(f)P(\theta)\, d\theta$
- Select the point $\mathbf{x}_i$ which maximizes the criterion: $\mathbf{x}_i = \arg\max \mathcal{C}(\mathbf{x}|P(f|\mathcal{D}))$. Observe $y_i = f(\mathbf{x}_i)$
- Augment the data with the new point and response: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathbf{x}_i, y_i\}$ $i \leftarrow i+1$

Figure 1: General algorithm for Bayesian optimization.

process with nonzero mean function or as a semiparametric model. The library allows to define priors on $\mathbf{w}$, σ_s^2 and θ. The marginal posterior $P(f|\mathcal{D})$ can be computed in closed form, except for the kernel parameters θ. BayesOpt allows to use different approximations based on empirical Bayes (Santner et al., 2003) or MCMC (Snoek et al., 2012) on $P(\theta|\mathcal{D})$.

2.1 Implementation

Efficiency has been one of the main objectives during development. We found evidence that updating θ every iteration might be unnecessary or even counterproductive (Bull, 2011). For empirical Bayes (ML or MAP of θ), we found that a combination of global and local derivative-free methods such as DIRECT (Jones et al., 1993) and BOBYQA (Powell, 2009) marginally outperforms gradient-based method, in CPU time, for optimizing θ by avoiding the overhead of computing the marginal likelihood derivative.

One of the most critical components, in terms of computational cost, is the computation of the inverse of the kernel matrix $\mathbf{K}^{-1}$ (Rasmussen and Nickisch, 2010). We compared different numerical solutions and we found that the *Cholesky decomposition* method outperforms any other method in terms of performance and numerical stability. Furthermore, we can exploit the structure of the Bayesian optimization algorithm in two ways. First, *points arrive sequentially*. Thus, we can do incremental computations of matrices and vectors, except when the distribution of the kernel parameters $P(\theta|\mathcal{D})$ is updated. For example, at each iteration, we know that only n new elements will appear in the correlation matrix $\mathbf{K}$, that is, the correlation of the new point with each of the existing points. The rest of the matrix remains invariant. Thus, instead of computing the whole *Cholesky* decomposition of $\mathbf{K}$, being $\mathcal{O}(n^3)$, we just add a new row of elements to the *Cholesky* decomposition, which is $\mathcal{O}(n^2)$. Second, finding the optimal decision at each iteration $\mathbf{x}_i$ requires *multiple queries of the acquisition function from the same posterior* $\mathcal{C}(\mathbf{x}|P(f|\mathcal{D}))$ (see Figure 1). Thus, we can precompute some terms of the posterior and criterion functions that are independent of the query point $\mathbf{x}$. To our knowledge, this is the first software for Gaussian processes/Bayesian optimization that exploits the idea of precomputing terms for multiple queries.

Table 1 compares CPU time (single thread) and accuracy of two different configurations of BayesOpt with respect to other open source libraries. SMAC (Hutter et al., 2011), HyperOpt (Bergstra et al., 2011) and Spearmint (Snoek et al., 2012) used the HPOlib (Eggensperger et al., 2013) timing system, based on `runsolver`. DiceOptim (Roustant et al., 2012) used R timing system (`proc.time`). For BayesOpt, standard `ctime` was used.

Another main objective has been *flexibility*. The user can easily select among different algorithms, hyperpriors, kernels or mean functions. Currently, the library supports contin-

	Branin (2D)				Camelback (2D)			
	Gap 50 samp.	Gap 200 samp.	t 200 sam.	ID	Gap 50 samp.	Gap 100 samp.	t 100 sam.	ID
SMAC	0.19444 (0.195)	0.06780 (0.059)	147.3 (1.3)	0	0.08534 (0.103)	0.03772 (0.034)	70.5 (0.9)	0
HyperOpt	0.69499 (0.414)	0.07507 (0.059)	23.5 (0.2)	0	0.10941 (0.050)	0.03383 (0.025)	8.0 (0.09)	0
Spearmint	1.48953 (1.468)	**0.00000** (0.000)	7530 (30)	2	0.00005 (0.000)	0.00004 (0.000)	1674 (8)	2
DiceOptim	**0.00004** (0.000)	0.00003 (0.000)	624 (35)	5	0.80861 (0.417)	0.35811 (0.350)	215 (10)	5
BayesOpt1	1.16844 (1.745)	**0.00000** (0.000)	**8.6** (0.07)	5	0.00852 (0.021)	**0.00000** (0.000)	**2.2** (0.2)	5
BayesOpt2	0.04742 (0.116)	**0.00000** (0.000)	1802 (78)	2	**0.00000** (0.000)	**0.00000** (0.000)	147 (1.3)	2

	Hartmann (6D)				Configuration - θ learning
	Gap 50 samp.	Gap 200 samp.	t 200 sam.	ID	
SMAC	1.23130 (0.645)	0.31628 (0.249)	155.9 (1.3)	0	Default HPOlib
HyperOpt	1.21979 (0.496)	0.39065 (0.208)	**33.3** (0.3)	0	Default HPOlib
Spearmint	2.13990 (0.659)	0.59980 (0.866)	8244 (106)	2	Def. HPOlib, MCMC (10 particles, 100 burn-in)
DiceOptim	**0.06008** (0.063)	0.06004 (0.063)	1267 (316)	10	ML, Genoud 50 pop., 20 gen., 5 wait, 5 burn-in
BayesOpt1	0.06476 (0.047)	**0.02385** (0.048)	39.0 (0.04)	10	MAP, DIRECT+BOBYQA every 20 iterations.
BayesOpt2	1.05608 (0.831)	0.04769 (0.058)	4093 (56)	2	MCMC (10 particles, 100 burn-in)

Table 1: Mean (and standard deviation) of the optimization gap, $f(\mathbf{x}_{best}) - f(\mathbf{x}_{opt})$, and time (in seconds) for 10 runs for different number of samples (including initial design) to illustrate the convergence of each method. *ID* represent the number of samples of the initial design for each algorithm and problem.

uous, discrete and categorical optimization. We also provide a method for optimization in high-dimensional spaces (Wang et al., 2013). The initial set of points (*initial design*, see Figure 1) can be selected using well-known methods such as latin hypercube sampling or Sobol sequences. BayesOpt relies on a factory-like design for the components of the optimization process. They can be selected and combined at runtime while maintaining a simple structure. This has two major advantages. First, it simplifies creating new components. For example, a new kernel can be defined by inheriting the abstract kernel or one of the existing kernels. Then, the new kernel is automatically integrated in the library. Second, inspired by the *GPML* toolbox by Rasmussen and Nickisch (2010), we can easily combine different components, like a linear combination of kernels or multiple criteria. This can be used to optimize a function considering an additional cost for each sample, for example, the cost of moving a sensor while maximizing the information (Marchant and Ramos, 2012). BayesOpt also implements *metacriteria algorithms*, like the bandit algorithm GP-Hedge by Hoffman et al. (2011) that can be used to automatically select the most suitable criteria during the optimization. Examples of these combinations can be found in Section 2.3.2.

The third objective is *correctness*. For example, the library is thread and exception safe, allowing parallelized calls. Parts that are sensible to numerical issues, such as the GP-Hedge algorithm, have been implemented with variation of the actual algorithm to avoid over- or underflow issues. The library internally uses NLOPT by Johnson (2014) for the inner optimization loops (optimize criteria, learn kernel parameters, etc.).

The library can be found at: `https://bitbucket.org/rmcantin/bayesopt/`

2.2 Compatibility

BayesOpt has been designed to be highly compatible in many platforms and setups. It has been tested and compiled in different operating systems (Linux, Mac OS, Windows), with

different compilers (Visual Studio, GCC, Clang, MinGW). The core of the library is written in C++, however, it provides interfaces for C, Python and Matlab/Octave.

2.3 Using the Library

There is a common API implemented for several languages and programming paradigms. Before running the optimization we need to follow two simple steps:

2.3.1 Target Function Definition

Defining the function that we want to optimize can be achieved in two ways. We can directly send the function (or a pointer) to the optimizer based on a *function template*. For example, in C/C++:

```
double my_function (unsigned int n_query, const double *query,
                    double *gradient, void *func_data);
```

The gradient has been included for future compatibility. Python, Matlab and Octave interfaces define a similar template function.

For a more object-oriented approach, we can inherit the abstract module and define the virtual methods. Using this approach, we can also include nonlinear constraints in the *checkReachability* method. This is available for C++ and Python. For example, in C++:

```
class MyOptimization: public bayesopt::ContinuousModel  {
 public:
  MyOptimization(size_t dim, bopt_params param): ContinousModel(dim,param) {}
  double evaluateSample(const boost::numeric::ublas::vector<double> &query)
  {  // My function here   };
  bool checkReachability(const boost::numeric::ublas::vector<double> &query)
  {  // My constraints here   };
};
```

2.3.2 BayesOpt Parameters

The parameters are defined in the `bopt_params` struct or a dictionary in Python. The details of each parameter can be found in the included documentation. The user can define expressions to combine different functions (kernels, criteria, etc.). All the parameters have a default value, so it is not necessary to define all of them. For example, in Matlab:

```
par.surr_name = 'sStudentTProcessNIG';        % Surrogate model and hyperpriors
% We combine Expected Improvement, Lower Confidence Bound and Thompson sampling
par.crit_name = 'cHedge(cEI,cLCB,cThompsonSampling)';
par.kernel_name = 'kSum(kMaternISO3,kRQISO)';                    % Sum of kernels
par.kernel_hp_mean = [1, 1]; par.kernel_hp_std = [5, 5]; % Hyperprior on kernel
par.l_type = 'L_MCMC';              % Method for learning the kernel parameters
par.sc_type = 'SC_MAP';     % Score function for learning the kernel parameters
par.n_iterations = 200;                      % Number of iterations <=> Budget
par.epsilon = 0.1;          % Add an epsilon-greedy step for better exploration
```

Acknowledgments

We would like to thank Luis Montesano for his insightful comments. This work has been supported by Spanish government grant DPI2011-25892 and CUD grant CUD2013-5.

References

James Bergstra, Remi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *NIPS*, pages 2546–2554, 2011.

Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. eprint arXiv:1012.2599, arXiv.org, December 2010.

Adam D. Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12:2879–2904, 2011.

Katharina Eggensperger, Matthias Feurer, Frank Hutter, James Bergstra, Jasper Snoek, Holger Hoos, and Kevin Leyton-Brown. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *BayesOpt workshop (NIPS)*, 2013.

Matthew Hoffman, Eric Brochu, and Nando de Freitas. Portfolio allocation for Bayesian optimization. In *UAI*, 2011.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION-5*, page 507523, 2011.

Steven G. Johnson. The NLopt nonlinear-optimization package. `http://ab-initio.mit.edu/nlopt`, 2014.

Donald R. Jones, Cary D. Perttunen, and Bruce E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1): 157–181, October 1993.

Roman Marchant and Fabio Ramos. Bayesian optimisation for intelligent environmental monitoring. In *IEEE/RSJ IROS*, pages 2242–2249, 2012.

Jonas Mockus. *Bayesian Approach to Global Optimization*, volume 37 of *Mathematics and Its Applications*. Kluwer Academic Publishers, 1989.

Michael J. D. Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. Technical Report NA2009/06, Department of Applied Mathematics and Theoretical Physics, Cambridge England, 2009.

Carl E. Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (GPML) toolbox. *Journal of Machine Learning Research*, 11:3011–3015, 2010.

Olivier Roustant, David Ginsbourger, and Yves Deville. DiceKriging, DiceOptim: two R packages for the analysis of computer experiments by kriging-based metamodelling and optimization. *Journal of Statistical Software*, 51(1):1–55, 2012.

Thomas J. Santner, Brian J. Williams, and William I. Notz. *The Design and Analysis of Computer Experiments*. Springer-Verlag, 2003.

Jasper Snoek, Hugo Larochelle, and Ryan Adams. Practical Bayesian optimization of machine learning algorithms. In *NIPS*, pages 2960–2968, 2012.

Ziyu Wang, Masrour Zoghi, David Matheson, Frank Hutter, and Nando de Freitas. Bayesian optimization in a billion dimensions via random embeddings. In *IJCAI*, 2013.

Journal of Machine Learning Research 15 (2014) 3921-3962 Submitted 9/13; Revised 7/14; Published 11/14

Order-Independent Constraint-Based Causal Structure Learning

Diego Colombo COLOMBO@STAT.MATH.ETHZ.CH
Marloes H. Maathuis MAATHUIS@STAT.MATH.ETHZ.CH
Seminar for Statistics
ETH Zurich
8092 Zurich, Switzerland

Editor: Peter Spirtes

Abstract

We consider constraint-based methods for causal structure learning, such as the PC-, FCI-, RFCI- and CCD- algorithms (Spirtes et al., 1993, 2000; Richardson, 1996; Colombo et al., 2012; Claassen et al., 2013). The first step of all these algorithms consists of the adjacency search of the PC-algorithm. The PC-algorithm is known to be order-dependent, in the sense that the output can depend on the order in which the variables are given. This order-dependence is a minor issue in low-dimensional settings. We show, however, that it can be very pronounced in high-dimensional settings, where it can lead to highly variable results. We propose several modifications of the PC-algorithm (and hence also of the other algorithms) that remove part or all of this order-dependence. All proposed modifications are consistent in high-dimensional settings under the same conditions as their original counterparts. We compare the PC-, FCI-, and RFCI-algorithms and their modifications in simulation studies and on a yeast gene expression data set. We show that our modifications yield similar performance in low-dimensional settings and improved performance in high-dimensional settings. All software is implemented in the R-package `pcalg`.

Keywords: directed acyclic graph, PC-algorithm, FCI-algorithm, CCD-algorithm, order-dependence, consistency, high-dimensional data

1. Introduction

Constraint-based methods for causal structure learning use conditional independence tests to obtain information about the underlying causal structure. We start by discussing several prominent examples of such algorithms, designed for different settings.

The PC-algorithm (Spirtes et al., 1993, 2000) was designed for learning directed *acyclic* graphs (DAGs) under the assumption of *causal sufficiency*, i.e., no unmeasured common causes and no selection variables. It learns a Markov equivalence class of DAGs that can be uniquely described by a so-called completed partially directed acyclic graph (CPDAG) (see Section 2 for a precise definition). The PC-algorithm is widely used in high-dimensional settings (Kalisch et al., 2010; Nagarajan et al., 2010; Stekhoven et al., 2012; Zhang et al., 2012), since it is computationally feasible for sparse graphs with up to thousands of variables, and open-source software is available, for example in TETRAD IV (Spirtes et al., 2000) and the R-package `pcalg` (Kalisch et al., 2012). Moreover, the PC-algorithm has been shown

to be consistent for high-dimensional sparse graphs (Kalisch and Bühlmann, 2007; Harris and Drton, 2013).

The FCI- and RFCI-algorithms and their modifications (Spirtes et al., 1993, 2000, 1999; Spirtes, 2001; Zhang, 2008; Colombo et al., 2012; Claassen et al., 2013) were designed for learning directed *acyclic* graphs when *allowing for latent and selection variables*. Thus, these algorithms learn a Markov equivalence class of DAGs with latent and selection variables, which can be uniquely represented by a partial ancestral graph (PAG). These algorithms first employ the adjacency search of the PC-algorithm, and then perform additional conditional independence tests because of the latent variables.

Finally, the CCD-algorithm (Richardson, 1996) was designed for learning Markov equivalence classes of directed (*not necessarily acyclic*) graphs under the assumption of *causal sufficiency*. Again, the first step of this algorithm consists of the adjacency search of the PC-algorithm.

Hence, all these algorithms share the adjacency search of the PC-algorithm as a common first step. We will therefore focus our analysis on this algorithm, since any improvements to the PC-algorithm can be directly carried over to the other algorithms. When the PC-algorithm is applied to data, it is generally order-dependent, in the sense that its output depends on the order in which the variables are given. Dash and Druzdzel (1999) exploit the order-dependence to obtain candidate graphs for a score-based approach. Cano et al. (2008) resolve the order-dependence via a rather involved method based on measuring edge strengths. Spirtes et al. (2000) (Section 5.4.2.4) propose a method that removes the "weakest" edges as early as possible. Overall, however, the order-dependence of the PC-algorithm has received relatively little attention in the literature, suggesting that it seems to be regarded as a minor issue. We found, however, that the order-dependence can become very problematic for high-dimensional data, leading to highly variable results and conclusions for different variable orderings.

In particular, we analyzed the yeast gene expression data set of Hughes et al. (2000). We chose these data, despite the existence of larger and newer yeast gene expression data sets, since these data contain both observational and experimental data, obtained under similar conditions. The observational data consist of gene expression levels of 5361 genes for 63 wild-type yeast organisms, and the experimental data consist of gene expression levels of the same 5361 genes for 234 single-gene knockout strains. Hence, these data form a nice test bed for causal inference: algorithms can be applied to the observational data, and their output can be compared to the "gold standard" experimental data. (Please see Section 6 for more detailed information about the data.)

First, we considered estimating the skeleton of the CPDAG from the observational data, that is, the undirected graph obtained by discarding all arrowheads in the CPDAG. Figure 1(a) shows the large variability in the estimated skeletons for 25 random orderings of the variables. Each estimated skeleton consists of roughly 5000 edges which can be divided into three groups: about 1500 are highly stable and occur in all orderings, about 1500 are moderately stable and occur in at least 50% of the orderings, and about 2000 are unstable and occur in at most 50% of the orderings. Since the FCI- and CCD-algorithms employ the adjacency search of the PC-algorithm as a first step, their resulting skeletons for these data are also highly order-dependent.

An important motivation for learning DAGs lies in their causal interpretation. We therefore also investigated the effect of different variable orderings on causal inference that is based on the PC-algorithm. In particular, we applied the IDA algorithm (Maathuis et al., 2010, 2009) to the observational yeast gene expression data, for 25 random variable orderings. The IDA algorithm conceptually consists of two-steps: one first estimates the Markov equivalence class of DAGs using the PC-algorithm, and one then applies Pearl's do-calculus (Pearl, 2000) to each DAG in the Markov equivalence class. (The algorithm uses a fast local implementation that does not require listing all DAGs in the equivalence class.) One can then obtain estimated lower bounds on the sizes of the causal effects between all pairs of genes. For each of the 25 random variable orderings, we ranked the gene pairs according to these lower bounds, and compared these rankings to a gold standard set of large causal effects computed from the experimental single gene knock-out data, as in Maathuis et al. (2010). Figure 1(b) shows the large variability in the resulting receiver operating characteristic (ROC) curves. The ROC curve that was published in Maathuis et al. (2010) was significantly better than random guessing with $p < 0.001$, and is somewhere in the middle. Some of the other curves are much better, while there are also curves that are indistinguishable from random guessing.

The remainder of the paper is organized as follows. In Section 2 we discuss some background and terminology. Section 3 explains the original PC-algorithm. Section 4 introduces modifications of the PC-algorithm (and hence also of the (R)FCI- and CCD-algorithms) that remove part or all of the order-dependence. These modifications are identical to their original counterparts when perfect conditional independence information is used. When applied to data, the modified algorithms are partly or fully order-independent. Moreover, they are consistent in high-dimensional settings under the same conditions as the original algorithms. Section 5 compares all algorithms in simulations, and Section 6 compares them on the yeast gene expression data discussed above. We close with a discussion in Section 7.

2. Preliminaries

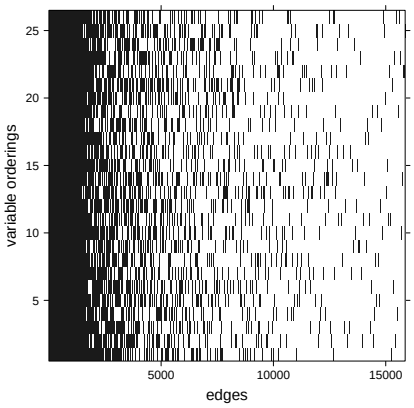
In this section, we introduce some necessary terminology and background information.

2.1 Graph Terminology

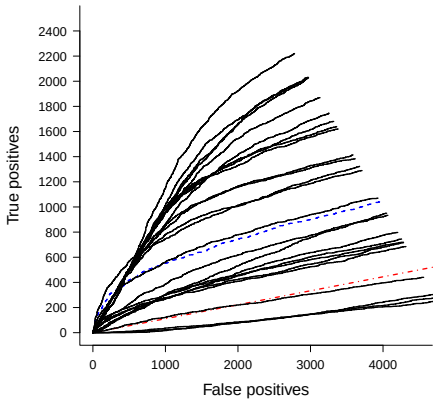
A graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ consists of a vertex set $\mathbf{V} = \{X_1, \ldots, X_p\}$ and an edge set $\mathbf{E}$. The vertices represent random variables and the edges represent relationships between pairs of variables.

A graph containing only directed edges ($\rightarrow$) is *directed*, one containing only undirected edges ($-$) is *undirected*, and one containing directed and/or undirected edges is *partially directed*. The *skeleton* of a partially directed graph is the undirected graph that results when all directed edges are replaced by undirected edges.

All graphs we consider are *simple*, meaning that there is at most one edge between any pair of vertices. If an edge is present, the vertices are said to be *adjacent*. If all pairs of vertices in a graph are adjacent, the graph is called *complete*. The *adjacency set* of a vertex X_i in a graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, denoted by $\text{adj}(\mathcal{G}, X_i)$, is the set of all vertices in $\mathbf{V}$ that are adjacent to X_i in $\mathcal{G}$. A vertex X_j in $\text{adj}(\mathcal{G}, X_i)$ is called a *parent* of X_i if $X_j \rightarrow X_i$. The corresponding set of parents is denoted by $\text{pa}(\mathcal{G}, X_i)$.



(a) Edges occurring in the estimated skeletons for 25 random variable orderings, as well as for the original ordering (shown as variable ordering 26). A black entry for an edge i and a variable ordering j means that edge i occurred in the estimated skeleton for the jth variable ordering. The edges along the x-axis are ordered according to their frequency of occurrence in the estimated skeletons, from edges that occurred always to edges that occurred only once. Edges that did not occur for any of the variable orderings were omitted. For technical reasons, only every 10th edge is actually plotted.



(b) ROC curves corresponding to the 25 random orderings of the variables (solid black), where the curves are generated exactly as in Maathuis et al. (2010). The ROC curve for the original ordering of the variables (dashed blue) was published in Maathuis et al. (2010). The dashed-dotted red curve represents random guessing.

Figure 1: Analysis of the yeast gene expression data (Hughes et al., 2000) for 25 random orderings of the variables, using tuning parameter $\alpha = 0.01$. The estimated graphs and resulting causal rankings are highly order-dependent.

A *path* is a sequence of distinct adjacent vertices. A *directed path* is a path along directed edges that follows the direction of the arrowheads. A *directed cycle* is formed by a directed path from X_i to X_j together with the edge $X_j \to X_i$. A (partially) directed graph is called a *(partially) directed acyclic graph* if it does not contain directed cycles.

A triple (X_i, X_j, X_k) in a graph $\mathcal{G}$ is *unshielded* if X_i and X_j as well as X_j and X_k are adjacent, but X_i and X_k are not adjacent in $\mathcal{G}$. A *v-structure* (X_i, X_j, X_k) is an unshielded triple in a graph $\mathcal{G}$ where the edges are oriented as $X_i \to X_j \leftarrow X_k$.

2.2 Probabilistic and Causal Interpretation of DAGs

We use the notation $X_i \perp\!\!\!\perp X_j | \mathbf{S}$ to indicate that X_i is independent of X_j given $\mathbf{S}$, where $\mathbf{S}$ is a set of variables not containing X_i and X_j (Dawid, 1980). If $\mathbf{S}$ is the empty set, we simply write $X_i \perp\!\!\!\perp X_j$. If $X_i \perp\!\!\!\perp X_j | \mathbf{S}$, we refer to $\mathbf{S}$ as a *separating set* for (X_i, X_j). A

separating set $\mathbf{S}$ for (X_i, X_j) is called *minimal* if there is no proper subset $\mathbf{S}'$ of $\mathbf{S}$ such that $X_i \perp\!\!\!\perp X_j | \mathbf{S}'$.

A distribution Q is said to *factorize* according to a DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ if the joint density of $\mathbf{V} = (X_1, \ldots, X_p)$ can be written as the product of the conditional densities of each variable given its parents in $\mathcal{G}$: $q(X_1, \ldots, X_p) = \prod_{i=1}^{p} q(X_i | \text{pa}(\mathcal{G}, X_i))$.

A DAG entails conditional independence relationships via a graphical criterion called *d-separation* (Pearl, 2000). If two vertices X_i and X_j are not adjacent in a DAG $\mathcal{G}$, then they are d-separated in $\mathcal{G}$ by a subset $\mathbf{S}$ of the remaining vertices. If X_i and X_j are d-separated by $\mathbf{S}$, then $X_i \perp\!\!\!\perp X_j | \mathbf{S}$ in any distribution Q that factorizes according to $\mathcal{G}$. A distribution Q is said to be *faithful* to a DAG $\mathcal{G}$ if the reverse implication also holds, that is, if the conditional independence relationships in Q are exactly the same as those that can be inferred from $\mathcal{G}$ using d-separation.

Several DAGs can describe exactly the same conditional independence information. Such DAGs are called Markov equivalent and form a Markov equivalence class. Markov equivalent DAGs have the same skeleton and the same v-structures, and a Markov equivalence class can be described uniquely by a completed partially directed acyclic graph (CPDAG) (Andersson et al., 1997; Chickering, 2002). A CPDAG is a partially directed acyclic graph with the following properties: every directed edge exists in every DAG in the Markov equivalence class, and for every undirected edge $X_i - X_j$ there exists a DAG with $X_i \to X_j$ and a DAG with $X_i \leftarrow X_j$ in the Markov equivalence class. A CPDAG $\mathcal{C}$ is said to *represent* a DAG $\mathcal{G}$ if $\mathcal{G}$ belongs to the Markov equivalence class described by $\mathcal{C}$.

A DAG can be interpreted causally in the following way (Pearl, 2000, 2009; Spirtes et al., 2000): X_1 is a direct cause of X_2 only if $X_1 \to X_2$, and X_1 is a possibly indirect cause of X_2 only if there is a directed path from X_1 to X_2.

3. The PC-Algorithm

We now describe the PC-algorithm in detail. In Section 3.1, we discuss the algorithm under the assumption that we have perfect conditional independence information between all variables in $\mathbf{V}$. We refer to this as the *oracle version*. In Section 3.2 we discuss the more realistic situation where conditional independence relationships have to be estimated from data. We refer to this as the *sample version*.

3.1 Oracle Version

A sketch of the PC-algorithm is given in Algorithm 3.1. We see that the algorithm consists of three steps. Step 1 (also called *adjacency search*) finds the skeleton and separation sets, while Steps 2 and 3 determine the orientations of the edges.

Step 1 is given in pseudo-code in Algorithm 3.2. We start with a complete undirected graph $\mathcal{C}$. This graph is subsequently thinned out in the loop on lines 3-15 in Algorithm 3.2, where an edge $X_i - X_j$ is deleted if $X_i \perp\!\!\!\perp X_j | \mathbf{S}$ for some subset $\mathbf{S}$ of the remaining variables. These conditional independence queries are organized in a way that makes the algorithm computationally efficient for high-dimensional sparse graphs, since we only need to query conditional independencies up to order $q - 1$, where q is the maximum size of the adjacency sets of the nodes in the underlying DAG.

Algorithm 3.1 The PC-algorithm (oracle version)

Require: Conditional independence information among all variables in $\mathbf{V}$, and an ordering order($\mathbf{V}$) on the variables

Adjacency search: Find the skeleton $\mathcal{C}$ and separation sets using Algorithm 3.2;
Orient unshielded triples in the skeleton $\mathcal{C}$ based on the separation sets;
In $\mathcal{C}$ orient as many of the remaining undirected edges as possible by repeated application of rules R1-R3 (see text);
return Output graph ($\mathcal{C}$) and separation sets (sepset).

Algorithm 3.2 Adjacency search / Step 1 of the PC-algorithm (oracle version)

Require: Conditional independence information among all variables in $\mathbf{V}$, and an ordering order($\mathbf{V}$) on the variables

Form the complete undirected graph $\mathcal{C}$ on the vertex set $\mathbf{V}$
Let $\ell = -1$;
repeat
  Let $\ell = \ell + 1$;
  repeat
    Select a (new) ordered pair of vertices (X_i, X_j) that are adjacent in $\mathcal{C}$ and satisfy $|\text{adj}(\mathcal{C}, X_i) \setminus \{X_j\}| \geq \ell$, using order($\mathbf{V}$);
    repeat
      Choose a (new) set $\mathbf{S} \subseteq \text{adj}(\mathcal{C}, X_i) \setminus \{X_j\}$ with $|\mathbf{S}| = \ell$, using order($\mathbf{V}$);
      if X_i and X_j are conditionally independent given $\mathbf{S}$ **then**
        Delete edge $X_i - X_j$ from $\mathcal{C}$;
        Let $\text{sepset}(X_i, X_j) = \text{sepset}(X_j, X_i) = \mathbf{S}$;
      end if
    until X_i and X_j are no longer adjacent in $\mathcal{C}$ or all $\mathbf{S} \subseteq \text{adj}(\mathcal{C}, X_i) \setminus \{X_j\}$ with $|\mathbf{S}| = \ell$ have been considered
  until all ordered pairs of adjacent vertices (X_i, X_j) in $\mathcal{C}$ with $|\text{adj}(\mathcal{C}, X_i) \setminus \{X_j\}| \geq \ell$ have been considered
until all pairs of adjacent vertices (X_i, X_j) in $\mathcal{C}$ satisfy $|\text{adj}(\mathcal{C}, X_i) \setminus \{X_j\}| \leq \ell$
return $\mathcal{C}$, sepset.

First, when $\ell = 0$, all pairs of vertices are tested for marginal independence. If $X_i \perp\!\!\!\perp X_j$, then the edge $X_i - X_j$ is deleted and the empty set is saved as separation set in $\text{sepset}(X_i, X_j)$ and $\text{sepset}(X_j, X_i)$. After all pairs of vertices have been considered (and many edges might have been deleted), the algorithm proceeds to the next step with $\ell = 1$.

When $\ell = 1$, the algorithm chooses an ordered pair of vertices (X_i, X_j) still adjacent in $\mathcal{C}$, and checks $X_i \perp\!\!\!\perp X_j | \mathbf{S}$ for subsets $\mathbf{S}$ of size $\ell = 1$ of $\text{adj}(\mathcal{C}, X_i) \setminus \{X_j\}$. If such a conditional independence is found, the edge $X_i - X_j$ is removed, and the corresponding conditioning set $\mathbf{S}$ is saved in $\text{sepset}(X_i, X_j)$ and $\text{sepset}(X_j, X_i)$. If all ordered pairs of adjacent vertices have been considered for conditional independence given all subsets of size ℓ of their adjacency sets, the algorithm again increases ℓ by one. This process continues until all adjacency sets in the current graph are smaller than ℓ. At this point the skeleton and the separation sets have been determined.

We see that this procedure indeed ensures that we only query conditional independencies up to order $q - 1$, where q is the maximum size of the adjacency sets of the nodes in the underlying DAG. This makes the algorithm particularly efficient for large sparse graphs.

Step 2 determines the v-structures. In particular, it considers all unshielded triples in $\mathcal{C}$, and orients an unshielded triple (X_i, X_j, X_k) as a v-structure if and only if $X_j \notin \text{sepset}(X_i, X_k)$.

Finally, Step 3 orients as many of the remaining undirected edges as possible by repeated application of the following three rules:

R1: orient $X_j - X_k$ into $X_j \to X_k$ whenever there is a directed edge $X_i \to X_j$ such that X_i and X_k are not adjacent (otherwise a new v-structure is created);

R2: orient $X_i - X_j$ into $X_i \to X_j$ whenever there is a chain $X_i \to X_k \to X_j$ (otherwise a directed cycle is created);

R3: orient $X_i - X_j$ into $X_i \to X_j$ whenever there are two chains $X_i - X_k \to X_j$ and $X_i - X_l \to X_j$ such that X_k and X_l are not adjacent (otherwise a new v-structure or a directed cycle is created).

The PC-algorithm was shown to be sound and complete.

Theorem 1 *(Theorem 5.1 on p.410 of Spirtes et al., 2000) Let the distribution of $\mathbf{V}$ be faithful to a DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, and assume that we are given perfect conditional independence information about all pairs of variables (X_i, X_j) in $\mathbf{V}$ given subsets $\mathbf{S} \subseteq \mathbf{V} \setminus \{X_i, X_j\}$. Then the output of the PC-algorithm is the CPDAG that represents $\mathcal{G}$.*

We briefly discuss the main ingredients of the proof, as these will be useful for understanding our modifications in Section 4. The faithfulness assumption implies that conditional independence in the distribution of $\mathbf{V}$ is equivalent to d-separation in the graph $\mathcal{G}$. The skeleton of $\mathcal{G}$ can then be determined as follows: X_i and X_j are adjacent in $\mathcal{G}$ if and only if they are conditionally dependent given any subset $\mathbf{S}$ of the remaining nodes. Naively, one could therefore check all these conditional dependencies, which is known as the SGS-algorithm (Spirtes et al., 2000). The PC-algorithm obtains the same result with fewer tests, by using the following fact about DAGs: two variables X_i and X_j in a DAG $\mathcal{G}$ are d-separated by some subset $\mathbf{S}$ of the remaining variables if and only if they are d-separated

by $\text{pa}(\mathcal{G}, X_i)$ or $\text{pa}(\mathcal{G}, X_j)$. The PC-algorithm is guaranteed to check these conditional independencies: at all stages of the algorithm, the graph $\mathcal{C}$ is a supergraph of the true CPDAG, and the algorithm checks conditional dependencies given all subsets of the adjacency sets, which obviously include the parent sets.

The v-structures are determined based on Lemmas 5.1.2 and 5.1.3 of Spirtes et al. (2000). The soundness and completeness of the orientation rules in Step 3 was shown in Meek (1995) and Andersson et al. (1997).

3.2 Sample Version

In applications, we of course do not have perfect conditional independence information. Instead, we assume that we have an i.i.d. sample of size n of $\mathbf{V} = (X_1, \ldots, X_p)$. A sample version of the PC-algorithm can then be obtained by replacing all conditional independence queries by statistical conditional independence tests at some pre-specified significance level α. For example, if the distribution of $\mathbf{V}$ is multivariate Gaussian, one can test for zero partial correlation, see, e.g., Kalisch and Bühlmann (2007). This is the test we used throughout this paper.

We note that the PC-algorithm performs many tests. Hence, α should not be interpreted as an overall significance level. Rather, it plays the role of a tuning parameter, where smaller values of α tend to lead to sparser graphs.

3.3 Order-Dependence in the Sample Version

Let $\text{order}(\mathbf{V})$ denote an ordering on the variables in $\mathbf{V}$. We now consider the role of $\text{order}(\mathbf{V})$ in every step of the algorithm. Throughout, we assume that all tasks are performed according to the lexicographical ordering of $\text{order}(\mathbf{V})$, which is the standard implementation in `pcalg` (Kalisch et al., 2012) and TETRAD IV (Spirtes et al., 2000), and is called "PC-1" in Spirtes et al. (2000) (Section 5.4.2.4).

In Step 1, $\text{order}(\mathbf{V})$ affects the estimation of the *skeleton* and the *separating sets*. In particular, at each level of ℓ, $\text{order}(\mathbf{V})$ determines the order in which pairs of adjacent vertices and subsets $\mathbf{S}$ of their adjacency sets are considered (see lines 6 and 8 in Algorithm 3.2). The skeleton $\mathcal{C}$ is updated after each edge removal. Hence, the adjacency sets typically change within one level of ℓ, and this affects which other conditional independencies are checked, since the algorithm only conditions on subsets of the adjacency sets. In the oracle version, we have perfect conditional independence information, and all orderings on the variables lead to the same output. In the sample version, however, we typically make mistakes in keeping or removing edges. In such cases, the resulting changes in the adjacency sets can lead to different skeletons, as illustrated in Example 1.

Moreover, different variable orderings can lead to different separating sets in Step 1. In the oracle version, this is not important, because any valid separating set leads to the correct v-structure decision in Step 2. In the sample version, however, different separating sets in Step 1 of the algorithm may yield different decisions about v-structures in Step 2. This is illustrated in Example 2.

Finally, we consider the role of $\text{order}(\mathbf{V})$ on the *orientation rules* in Steps 2 and 3 of the sample version of the PC-algorithm. Example 3 illustrates that different variable

orderings can lead to different orientations, even if the skeleton and separating sets are order-independent.

Example 1 *(Order-dependent skeleton in the sample version of the PC-algorithm.) Suppose that the distribution of* $\mathbf{V} = \{X_1, X_2, X_3, X_4, X_5\}$ *is faithful to the DAG in Figure 2(a). This DAG encodes the following conditional independencies with minimal separating sets:* $X_1 \perp\!\!\!\perp X_2$ *and* $X_2 \perp\!\!\!\perp X_4 | \{X_1, X_3\}$.

Suppose that we have an i.i.d. sample of $(X_1, X_2, X_3, X_4, X_5)$*, and that the following conditional independencies with minimal separating sets are judged to hold at some significance level* α*:* $X_1 \perp\!\!\!\perp X_2$*,* $X_2 \perp\!\!\!\perp X_4 | \{X_1, X_3\}$*, and* $X_3 \perp\!\!\!\perp X_4 | \{X_1, X_5\}$*. Thus, the first two are correct, while the third is false.*

We now apply the PC-algorithm with two different orderings: $order_1(\mathbf{V}) = (X_1, X_4, X_2, X_3, X_5)$ *and* $order_2(\mathbf{V}) = (X_1, X_3, X_4, X_2, X_5)$*. The resulting skeletons are shown in Figures 2(b) and 2(c), respectively. We see that the skeletons are different, and that both are incorrect as the edge* $X_3 - X_4$ *is missing. The skeleton for* $order_2(\mathbf{V})$ *contains an additional error, as there is an additional edge* $X_2 - X_4$*.*

We now go through Algorithm 3.2 to see what happened. We start with a complete undirected graph on $\mathbf{V}$*. When* $\ell = 0$*, variables are tested for marginal independence, and the algorithm correctly removes the edge between* X_1 *and* X_2*. No other conditional independencies are found when* $\ell = 0$ *or* $\ell = 1$*. When* $\ell = 2$*, there are two pairs of vertices that are thought to be conditionally independent given a subset of size 2, namely the pairs* (X_2, X_4) *and* (X_3, X_4)*.*

In $order_1(\mathbf{V})$*, the pair* (X_4, X_2) *is considered first. The corresponding edge is removed, as* $X_4 \perp\!\!\!\perp X_2 | \{X_1, X_3\}$ *and* $\{X_1, X_3\}$ *is a subset of* $adj(\mathcal{C}, X_4) = \{X_1, X_2, X_3, X_5\}$*. Next, the pair* (X_4, X_3) *is considered and the corresponding edges is erroneously removed, because of the wrong decision that* $X_4 \perp\!\!\!\perp X_3 | \{X_1, X_5\}$ *and the fact that* $\{X_1, X_5\}$ *is a subset of* $adj(\mathcal{C}, X_4) = \{X_1, X_3, X_5\}$*.*

In $order_2(\mathbf{V})$*, the pair* (X_3, X_4) *is considered first, and the corresponding edge is erroneously removed. Next, the algorithm considers the pair* (X_4, X_2)*. The corresponding separating set* $\{X_1, X_3\}$ *is not a subset of* $adj(\mathcal{C}, X_4) = \{X_1, X_2, X_5\}$*, so that the edge* $X_2 - X_4$ *remains. Next, the algorithm considers the pair* (X_2, X_4)*. Again, the separating set* $\{X_1, X_3\}$ *is not a subset of* $adj(\mathcal{C}, X_2) = \{X_3, X_4, X_5\}$*, so that the edge* $X_2 - X_4$ *again remains. In other words, since* (X_3, X_4) *was considered first in* $order_2(\mathbf{V})$*, the adjacency set of* X_4 *was affected and no longer contained* X_3*, so that the algorithm "forgot" to check the conditional independence* $X_2 \perp\!\!\!\perp X_4 | \{X_1, X_3\}$*.*

Example 2 *(Order-dependent separating sets and v-structures in the sample version of the PC-algorithm.) Suppose that the distribution of* $\mathbf{V} = \{X_1, X_2, X_3, X_4, X_5\}$ *is faithful to the DAG in Figure 3(a). This DAG encodes the following conditional independencies with minimal separating sets:* $X_1 \perp\!\!\!\perp X_3 | \{X_2\}$*,* $X_1 \perp\!\!\!\perp X_4 | \{X_2\}$*,* $X_1 \perp\!\!\!\perp X_4 | \{X_3\}$*,* $X_2 \perp\!\!\!\perp X_4 | \{X_3\}$*,* $X_2 \perp\!\!\!\perp X_5 | \{X_1, X_3\}$*,* $X_2 \perp\!\!\!\perp X_5 | \{X_1, X_4\}$*,* $X_3 \perp\!\!\!\perp X_5 | \{X_1, X_4\}$ *and* $X_3 \perp\!\!\!\perp X_5 | \{X_2, X_4\}$*.*

We consider the oracle version of the PC-algorithm with two different orderings on the variables: $order_3(\mathbf{V}) = (X_1, X_4, X_2, X_3, X_5)$ *and* $order_4(\mathbf{V}) = (X_1, X_4, X_3, X_2, X_5)$*. For* $order_3(\mathbf{V})$*, we obtain* $sepset(X_1, X_4) = \{X_2\}$*, while for* $order_4(\mathbf{V})$ *we get* $sepset(X_1, X_4) =$

$\{X_3\}$. *Thus, the separating sets are order-dependent. However, we obtain the same v-structure* $X_1 \to X_5 \leftarrow X_4$ *for both orderings, since* X_5 *is not in the sepset*(X_1, X_4)*, regardless of the ordering. In fact, this holds in general, since in the oracle version of the PC-algorithm, a vertex is either in all possible separating sets or in none of them (Spirtes et al., 2000, Lemma 5.1.3).*

Now suppose that we have an i.i.d. sample of $(X_1, X_2, X_3, X_4, X_5)$*. Suppose that at some significance level* α*, all true conditional independencies are judged to hold, and* $X_1 \perp\!\!\!\perp X_3 | \{X_4\}$ *is thought to hold by mistake. We again consider two different orderings:* $order_5(\mathbf{V}) = (X_1, X_3, X_4, X_2, X_5)$ *and* $order_6(\mathbf{V}) = (X_3, X_1, X_2, X_4, X_5)$*. With* $order_5(\mathbf{V})$ *we obtain the incorrect* $sepset(X_1, X_3) = \{X_4\}$*. This also leads to an incorrect v-structure* $X_1 \to X_2 \leftarrow X_3$ *in Step 2 of the algorithm. With* $order_6(\mathbf{V})$*, we obtain the correct* $sepset(X_1, X_3) = \{X_2\}$*, and hence correctly find that* $X_1 - X_2 - X_3$ *is not a v-structure in Step 2. This illustrates that order-dependent separating sets in Step 1 of the sample version of the PC-algorithm can lead to order-dependent v-structures in Step 2 of the algorithm.*

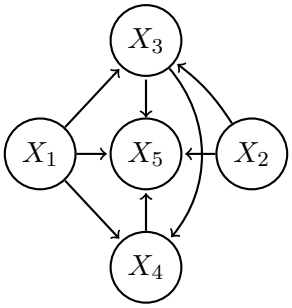
Example 3 *(Order-dependent orientation rules in Steps 2 and 3 of the sample version of the PC-algorithm.) Consider the graph in Figure 4(a) with unshielded triples* (X_1, X_2, X_3) *and* (X_2, X_3, X_4)*, and assume this is the skeleton after Step 1 of the sample version of the PC-algorithm. Moreover, assume that we found* $sepset(X_1, X_3) = sepset(X_2, X_4) = sepset(X_1, X_4) = \emptyset$*. Then in Step 2 of the algorithm, we obtain two v-structures* $X_1 \to X_2 \leftarrow X_3$ *and* $X_2 \to X_3 \leftarrow X_4$*. Of course this means that at least one of the statistical tests is wrong, but this can happen in the sample version. We now have conflicting information about the orientation of the edge* $X_2 - X_3$*. In the current implementation of* `pcalg`*, where conflicting edges are simply overwritten, this means that the orientation of* $X_2 - X_3$ *is determined by the v-structure that is last considered. Thus, we obtain* $X_1 \to X_2 \to X_3 \leftarrow X_4$ *if* (X_2, X_3, X_4) *is considered last, while we get* $X_1 \to X_2 \leftarrow X_3 \leftarrow X_4$ *if* (X_1, X_2, X_3) *is considered last.*

Next, consider the graph in Figure 4(b), and assume that this is the output of the sample version of the PC-algorithm after Step 2. Thus, we have two v-structures, namely $X_1 \to X_2 \leftarrow X_3$ *and* $X_4 \to X_5 \leftarrow X_6$*, and four unshielded triples, namely* (X_1, X_2, X_5)*,* (X_3, X_2, X_5)*,* (X_4, X_5, X_2)*, and* (X_6, X_5, X_2)*. Thus, we then apply the orientation rules in Step 3 of the algorithm, starting with rule R1. If one of the two unshielded triples* (X_1, X_2, X_5) *or* (X_3, X_2, X_5) *is considered first, we obtain* $X_2 \to X_5$*. On the other hand, if one of the unshielded triples* (X_4, X_5, X_2) *or* (X_6, X_5, X_2) *is considered first, then we obtain* $X_2 \leftarrow X_5$*. Note that we have no issues with overwriting of edges here, since as soon as the edge* $X_2 - X_5$ *is oriented, all edges are oriented and no further orientation rules are applied.*

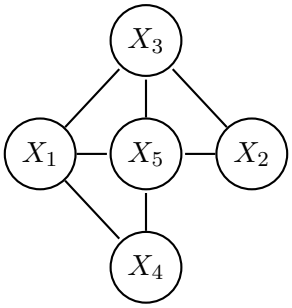
These examples illustrate that Steps 2 and 3 of the PC-algorithm can be order-dependent regardless of the output of the previous steps.

4. Modified Algorithms

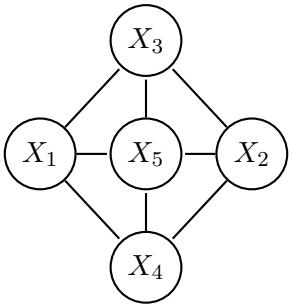
We now propose several modifications of the PC-algorithm (and hence also of the related algorithms) that remove the order-dependence in the various stages of the algorithm. Sections 4.1, 4.2, and 4.3 discuss the skeleton, the v-structures and the orientation rules, respectively. In each of these sections, we first describe the oracle version of the modifications, and then



(a) True DAG.

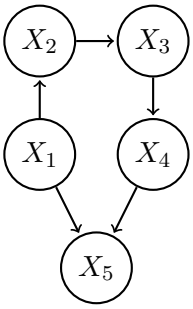


(b) Skeleton returned by the oracle version of Algorithm 3.2 with any ordering, and by the sample version of Algorithm 3.2 with $\text{order}_1(\mathbf{V})$.

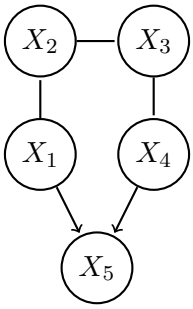


(c) Skeleton returned by the sample version of Algorithm 3.2 with $\text{order}_2(\mathbf{V})$.

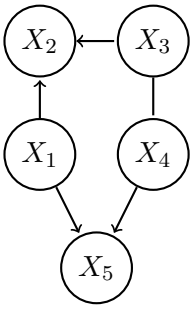
Figure 2: Graphs corresponding to Examples 1 and 4.



(a) True DAG.

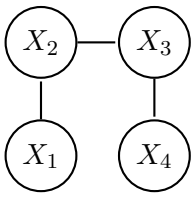


(b) Output of the oracle version of the PC-algorithm with any ordering, and of the sample version with $\text{order}_6(\mathbf{V})$.

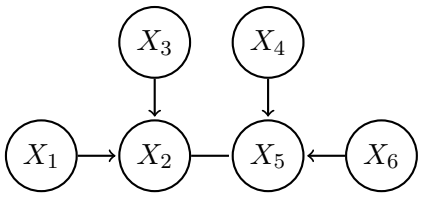


(c) Output of the sample version of the PC-algorithm with $\text{order}_5(\mathbf{V})$.

Figure 3: Graphs corresponding to Examples 2 and 5.



(a) Possible skeleton after Step 1 of the sample version of the PC-algorithm.



(b) Possible partially directed graph after Step 2 of the sample version of the PC-algorithm.

Figure 4: Graphs corresponding to Examples 3 and 6.

results and examples about order-dependence in the corresponding sample version (obtained by replacing conditional independence queries by conditional independence tests, as in Section 3.3). Finally, Section 4.4 discusses order-independent versions of related algorithms like RFCI and FCI, and Section 4.5 presents high-dimensional consistency results for the sample versions of all modifications.

4.1 The Skeleton

We first consider estimation of the skeleton in the adjacency search (Step 1) of the PC-algorithm. The pseudocode for our modification is given in Algorithm 4.1. The resulting PC-algorithm, where Step 1 in Algorithm 3.1 is replaced by Algorithm 4.1, is called "PC-stable".

The main difference between Algorithms 3.2 and 4.1 is given by the for-loop on lines 6-8 in the latter one, which computes and stores the adjacency sets $a(X_i)$ of all variables after each new size ℓ of the conditioning sets. These stored adjacency sets $a(X_i)$ are used whenever we search for conditioning sets of this given size ℓ. Consequently, an edge deletion on line 13 no longer affects which conditional independencies are checked for other pairs of variables at this level of ℓ.

In other words, at each level of ℓ, Algorithm 4.1 records which edges should be removed, but for the purpose of the adjacency sets it removes these edges only when it goes to the next value of ℓ. Besides resolving the order-dependence in the estimation of the skeleton, our algorithm has the advantage that it is easily parallelizable at each level of ℓ.

The PC-stable algorithm is sound and complete in the oracle version (Theorem 2), and yields order-independent skeletons in the sample version (Theorem 3). We illustrate the algorithm in Example 4.

Theorem 2 *Let the distribution of* $\mathbf{V}$ *be faithful to a DAG* $\mathcal{G} = (\mathbf{V}, \mathbf{E})$*, and assume that we are given perfect conditional independence information about all pairs of variables* (X_i, X_j) *in* $\mathbf{V}$ *given subsets* $\mathbf{S} \subseteq \mathbf{V} \setminus \{X_i, X_j\}$*. Then the output of the PC-stable algorithm is the CPDAG that represents* $\mathcal{G}$*.*

Proof The proof of Theorem 2 is completely analogous to the proof of Theorem 1 for the original PC-algorithm, as discussed in Section 3.1. ■

Theorem 3 *The skeleton resulting from the sample version of the PC-stable algorithm is order-independent.*

Proof We consider the removal or retention of an arbitrary edge $X_i - X_j$ at some level ℓ. The ordering of the variables determines the order in which the edges (line 9 of Algorithm 4.1) and the subsets $\mathbf{S}$ of $a(X_i)$ and $a(X_j)$ (line 11 of Algorithm 4.1) are considered. By construction, however, the order in which edges are considered does not affect the sets $a(X_i)$ and $a(X_j)$.

If there is at least one subset $\mathbf{S}$ of $a(X_i)$ or $a(X_j)$ such that $X_i \perp\!\!\!\perp X_j | \mathbf{S}$, then any ordering of the variables will find a separating set for X_i and X_j (but different orderings may lead to different separating sets as illustrated in Example 2). Conversely, if there is no

Algorithm 4.1 Step 1 of the PC-stable algorithm (oracle version)

Require: Conditional independence information among all variables in $\mathbf{V}$, and an ordering order($\mathbf{V}$) on the variables

Form the complete undirected graph $\mathcal{C}$ on the vertex set $\mathbf{V}$
Let $\ell = -1$;
repeat
  Let $\ell = \ell + 1$;
  for all vertices X_i in $\mathcal{C}$ **do**
    Let $a(X_i) = \text{adj}(\mathcal{C}, X_i)$
  end for
  repeat
    Select a (new) ordered pair of vertices (X_i, X_j) that are adjacent in $\mathcal{C}$ and satisfy $|a(X_i) \setminus \{X_j\}| \geq \ell$, using order($\mathbf{V}$);
    repeat
      Choose a (new) set $\mathbf{S} \subseteq a(X_i) \setminus \{X_j\}$ with $|\mathbf{S}| = \ell$, using order($\mathbf{V}$);
      if X_i and X_j are conditionally independent given $\mathbf{S}$ **then**
        Delete edge $X_i - X_j$ from $\mathcal{C}$;
        Let sepset(X_i, X_j) = sepset$(X_j, X_i) = \mathbf{S}$;
      end if
    until X_i and X_j are no longer adjacent in $\mathcal{C}$ or all $\mathbf{S} \subseteq a(X_i) \setminus \{X_j\}$ with $|\mathbf{S}| = \ell$ have been considered
  until all ordered pairs of adjacent vertices (X_i, X_j) in $\mathcal{C}$ with $|a(X_i) \setminus \{X_j\}| \geq \ell$ have been considered
until all pairs of adjacent vertices (X_i, X_j) in $\mathcal{C}$ satisfy $|a(X_i) \setminus \{X_j\}| \leq \ell$
return $\mathcal{C}$, sepset.

subset $\mathbf{S}'$ of $a(X_i)$ or $a(X_j)$ such that $X_i \perp\!\!\!\perp X_j | \mathbf{S}'$, then no ordering will find a separating set.

Hence, any ordering of the variables leads to the same edge deletions, and therefore to the same skeleton. ■

Example 4 *(Order-independent skeletons) We go back to Example 1, and consider the sample version of Algorithm 4.1. The algorithm now outputs the skeleton shown in Figure 2(b) for both orderings* $\text{order}_1(\mathbf{V})$ *and* $\text{order}_2(\mathbf{V})$.

We again go through the algorithm step by step. We start with a complete undirected graph on $\mathbf{V}$. *The only conditional independence found when* $\ell = 0$ *or* $\ell = 1$ *is* $X_1 \perp\!\!\!\perp X_2$, *and the corresponding edge is removed. When* $\ell = 2$, *the algorithm first computes the new adjacency sets:* $a(X_1) = a(X_2) = \{X_3, X_4, X_5\}$ *and* $a(X_i) = \mathbf{V} \setminus \{X_i\}$ *for* $i = 3, 4, 5$. *There are two pairs of variables that are thought to be conditionally independent given a subset of size 2, namely* (X_2, X_4) *and* (X_3, X_4). *Since the sets* $a(X_i)$ *are not updated after edge removals, it does not matter in which order we consider the ordered pairs* (X_2, X_4), (X_4, X_2), (X_3, X_4) *and* (X_4, X_3). *Any ordering leads to the removal of both edges, as the*

separating set $\{X_1, X_3\}$ *for* (X_4, X_2) *is contained in* $a(X_4)$*, and the separating set* $\{X_1, X_5\}$ *for* (X_3, X_4) *is contained in* $a(X_3)$ *(and in* $a(X_4)$*).*

4.2 Determination of the V-structures

We propose two methods to resolve the order-dependence in the determination of the v-structures, using the conservative PC-algorithm (CPC) of Ramsey et al. (2006) and a variation thereof.

The CPC-algorithm works as follows. Let $\mathcal{C}$ be the graph resulting from Step 1 of the PC-algorithm (Algorithm 3.1). For all unshielded triples (X_i, X_j, X_k) in $\mathcal{C}$, determine all subsets $\mathbf{Y}$ of $\text{adj}(\mathcal{C}, X_i)$ and of $\text{adj}(\mathcal{C}, X_k)$ that make X_i and X_k conditionally independent, i.e., that satisfy $X_i \perp\!\!\!\perp X_k | \mathbf{Y}$. We refer to such sets as separating sets. The triple (X_i, X_j, X_k) is labelled as *unambiguous* if at least one such separating set is found and either X_j is in all separating sets or in none of them; otherwise it is labelled as *ambiguous*. If the triple is unambiguous, it is oriented as v-structure if and only if X_j is in none of the separating sets. Moreover, in Step 3 of the PC-algorithm (Algorithm 3.1), the orientation rules are adapted so that only unambiguous triples are oriented. The output of the CPC-algorithm is a partially directed graph in which ambiguous triples are marked.

We found that the CPC-algorithm can be very conservative, in the sense that very few unshielded triples are unambiguous in the sample version. We therefore propose a minor modification of this approach, called majority rule PC-algorithm (MPC). As in CPC, we first determine all subsets $\mathbf{Y}$ of $\text{adj}(\mathcal{C}, X_i)$ and of $\text{adj}(\mathcal{C}, X_k)$ satisfying $X_i \perp\!\!\!\perp X_k | \mathbf{Y}$. We then label the triple (X_i, X_j, X_k) as *unambiguous* if at least one such separating set is found and X_j is not in exactly 50% of the separating sets. Otherwise it is labelled as *ambiguous*. (Of course, one can also use different cut-offs to declare ambiguous and non-ambiguous triples.) If a triple is unambiguous, it is oriented as v-structure if and only if X_j is in less than half of the separating sets. As in CPC, the orientation rules in Step 3 are adapted so that only unambiguous triples are oriented, and the output is a partially directed graph in which ambiguous triples are marked.

We refer to the combination of PC-stable and CPC/MPC as the CPC/MPC-stable algorithms. Theorem 4 states that the oracle versions of the CPC- and MPC-stable algorithms are sound and complete. When looking at the sample versions of the algorithms, we note that any unshielded triple that is judged to be unambiguous in CPC-stable is also unambiguous in MPC-stable, and any unambiguous v-structure in CPC-stable is an unambiguous v-structure in MPC-stable. In this sense, CPC-stable is more conservative than MPC-stable, although the difference appears to be small in simulations and for the yeast data (see Sections 5 and 6). Both CPC-stable and MPC-stable share the property that the determination of v-structures no longer depends on the (order-dependent) separating sets that were found in Step 1 of the algorithm. Therefore, both CPC-stable and MPC-stable yield order-independent decisions about v-structures in the sample version, as stated in Theorem 5. Example 5 illustrates both algorithms.

We note that the CPC/MPC-stable algorithms may yield a lot fewer directed edges than PC-stable. On the other hand, we can put more trust in those edges that were oriented.

Theorem 4 *Let the distribution of* $\mathbf{V}$ *be faithful to a DAG* $\mathcal{G} = (\mathbf{V}, \mathbf{E})$*, and assume that we are given perfect conditional independence information about all pairs of variables* (X_i, X_j)

in $\mathbf{V}$ *given subsets* $\mathbf{S} \subseteq \mathbf{V} \setminus \{X_i, X_j\}$. *Then the output of the CPC/MPC(-stable) algorithms is the CPDAG that represents* $\mathcal{G}$.

Proof The skeleton of the CPDAG is correct by Theorems 1 and 2. The unshielded triples are all unambiguous (in the conservative and the majority rule versions), since for any unshielded triple (X_i, X_j, X_k) in a DAG, X_j is either in all sets that d-separate X_i and X_k or in none of them (Spirtes et al., 2000, Lemma 5.1.3). In particular, this also means that all v-structures are determined correctly. Finally, since all unshielded triples are unambiguous, the orientation rules are as in the original oracle PC-algorithm, and soundness and completeness of these rules follows from Meek (1995) and Andersson et al. (1997). ■

Theorem 5 *The decisions about v-structures in the sample versions of the CPC/MPC-stable algorithms are order-independent.*

Proof The CPC/MPC-stable algorithms have order-independent skeletons in Step 1, by Theorem 3. In particular, this means that their unshielded triples and adjacency sets are order-independent. The decision about whether an unshielded triple is unambiguous and/or a v-structure is based on the adjacency sets of nodes in the triple, which are order-independent. ■

Example 5 *(Order-independent decisions about v-structures) We consider the sample versions of the CPC/MPC-stable algorithms, using the same input as in Example 2. In particular, we assume that all conditional independencies induced by the DAG in Figure 3(a) are judged to hold, plus the additional (erroneous) conditional independency* $X_1 \perp\!\!\!\perp X_3 | X_4$.

Denote the skeleton after Step 1 by $\mathcal{C}$. *We consider the unshielded triple* (X_1, X_2, X_3). *First, we compute* $adj(\mathcal{C}, X_1) = \{X_2, X_5\}$ *and* $adj(\mathcal{C}, X_3) = \{X_2, X_4\}$. *We now consider all subsets* $\mathbf{Y}$ *of these adjacency sets, and check whether* $X_1 \perp\!\!\!\perp X_3 | \mathbf{Y}$. *The following separating sets are found:* $\{X_2\}$, $\{X_4\}$, *and* $\{X_2, X_4\}$.

Since X_2 *is in some but not all of these separating sets, CPC-stable determines that the triple is ambiguous, and no orientations are performed. Since* X_2 *is in more than half of the separating sets, MPC-stable determines that the triple is unambiguous and not a v-structure. The output of both algorithms is given in Figure 3(b).*

4.3 Orientation Rules

Even when the skeleton and the determination of the v-structures are order-independent, Example 3 showed that there might be some order-dependence left in the sample-version. This can be resolved by allowing bi-directed edges ($\leftrightarrow$) and working with lists containing the candidate edges for the v-structures in Step 2 and the orientation rules R1-R3 in Step 3.

In particular, in Step 2 we generate a list of all (unambiguous) v-structures, and then orient all of these, creating a bi-directed edge in case of a conflict between two v-structures. In Step 3, we first generate a list of all edges that can be oriented by rule R1. We orient all

these edges, again creating bi-directed edges if there are conflicts. We do the same for rules R2 and R3, and iterate this procedure until no more edges can be oriented.

When using this procedure, we add the letter L (standing for lists), e.g., LCPC-stable and LMPC-stable. The LCPC-stable and LMPC-stable algorithms are correct in the oracle version (Theorem 6) and fully order-independent in the sample versions (Theorem 7). The procedure is illustrated in Example 6.

We note that the bi-directed edges cannot be interpreted causally. They simply indicate that there was some conflicting information in the algorithm.

Theorem 6 *Let the distribution of* $\mathbf{V}$ *be faithful to a DAG* $\mathcal{G} = (\mathbf{V}, \mathbf{E})$*, and assume that we are given perfect conditional independence information about all pairs of variables* (X_i, X_j) *in* $\mathbf{V}$ *given subsets* $\mathbf{S} \subseteq \mathbf{V} \setminus \{X_i, X_j\}$*. Then the (L)CPC(-stable) and (L)MPC(-stable) algorithms output the CPDAG that represents* $\mathcal{G}$*.*

Proof By Theorem 4, we know that the CPC(-stable) and MPC(-stable) algorithms are correct. With perfect conditional independence information, there are no conflicts between v-structures in Step 2 of the algorithms, nor between orientation rules in Step 3 of the algorithms. Therefore, the (L)CPC(-stable) and (L)MPC(-stable) algorithms are identical to the CPC(-stable) and MPC(-stable) algorithms. ■

Theorem 7 *The sample versions of LCPC-stable and LMPC-stable are fully order-independent.*

Proof This follows straightforwardly from Theorems 3 and 5 and the procedure with lists and bi-directed edges discussed above. ■

Table 1 summarizes the three order-dependence issues explained above and the corresponding modifications of the PC-algorithm that removes the given order-dependence problem.

	skeleton	v-structures decisions	edges orientations
PC	-	-	-
PC-stable	√	-	-
CPC/MPC-stable	√	√	-
BCPC/BMPC-stable	√	√	√

Table 1: Order-dependence issues and corresponding modifications of the PC-algorithm that remove the problem. A tick mark indicates that the corresponding aspect of the graph is estimated order-independently in the sample version. For example, with PC-stable the skeleton is estimated order-independently but not the v-structures and the edge orientations.

Example 6 *First, we consider the two unshielded triples* (X_1, X_2, X_3) *and* (X_2, X_3, X_4) *as shown in Figure 4(a). The version of the algorithm that uses lists for the orientation rules, orients these edges as* $X_1 \to X_2 \leftrightarrow X_3 \leftarrow X_4$*, regardless of the ordering of the variables.*

Next, we consider the structure shown in Figure 4(b). As a first step, we construct a list containing all candidate structures eligible for orientation rule R1 in Step 3. The list contains the unshielded triples (X_1, X_2, X_5)*,* (X_3, X_2, X_5)*,* (X_4, X_5, X_2)*, and* (X_6, X_5, X_2)*. Now, we go through each element in the list and we orient the edges accordingly, allowing bi-directed edges. This yields the edge orientation* $X_2 \leftrightarrow X_5$*, regardless of the ordering of the variables.*

4.4 Related Algorithms

If there are unmeasured common causes or unmeasured selection variables, which is often the case in practice, then causal inference based on the PC-algorithm may be incorrect. In such cases, one needs a generalization of a DAG, called a maximal ancestral graph (MAG) (Richardson and Spirtes, 2002). A MAG describes causal information in its edge marks, and entails conditional independence relationships via *m-separation* (Richardson and Spirtes, 2002), a generalization of d-separation. Several MAGs can describe exactly the same conditional independence information. Such MAGs are called Markov equivalent and form a Markov equivalence class, which can be represented by a partial ancestral graph (PAG) (Richardson and Spirtes, 2002; Ali et al., 2009). PAGs describe causal features common to every MAG in the Markov equivalence class, and hence to every DAG (possibly with latent and selection variables) compatible with the observable independence structure under the assumption of faithfulness. More information on the interpretation of MAGs and PAGs can be found in, e.g., Colombo et al. (2012) (Sections 1.2 and 2.2).

PAGs can be learned by the FCI-algorithm (Spirtes et al., 2000, 1999). As a first step, this algorithm runs Steps 1 and 2 of the PC-algorithm (Algorithm 3.1). Based on the resulting graph, it then computes certain sets, called "Possible-D-SEP" sets, and conducts more conditional independence tests given subsets of the Possible-D-SEP sets. This can lead to additional edge removals and corresponding separating sets. After this, the v-structures are newly determined. Finally, there are ten orientation rules as defined by Zhang (2008). The output of the FCI-algorithm is an (estimated) PAG (Colombo et al., 2012, Definition 3.1).

From our results, it immediately follows that FCI with any of our modifications of the PC-algorithm is sound and complete in the oracle version. Moreover, we can easily construct partially or fully order-independent sample versions as follows. To solve the order-dependence in the skeleton we can use the following three step approach. First, we use PC-stable to find an initial order-independent skeleton. Next, since Possible-D-SEP sets are determined from the orientations of the v-structures, we need order-independent v-structures. Therefore, in Step 2 we can determine the v-structures using CPC. Finally, we compute the Possible-D-SEP sets for all pairs of nodes at once, and do not update these after possible edge removals. The modification that uses these three steps returns an order-independent skeleton, and we call it FCI-stable. To assess order-independent v-structures in the final output, one should again use an order-independent procedure, as in CPC or MPC for the second time that v-structures are determined. We call these modifications

CFCI-stable and MFCI-stable, respectively. Regarding the orientation rules, we note that the FCI-algorithm does not suffer from conflicting v-structures (as shown in Figure 4(a) for the PC-algorithm), because it orients edge *marks* and because bi-directed edges are allowed. However, the ten orientation rules still suffer from order-dependence issues as in the PC-algorithm (see Example 3 and Figure 4(b)). To solve this problem, we can again use lists of candidate edges for each orientation rule as explained in the previous section about the PC-algorithm. We refer to these modifications as LCFCI-stable and LMFCI-stable, and they are fully order-independent in the sample version. However, since these ten orientation rules are more involved than the three for PC, using lists can be very slow for some rules, for example the one for discriminating paths.

Table 2 summarizes the three order-dependence issues for FCI and the corresponding modifications that remove them.

	skeleton	v-structures decisions	edges orientations
FCI	-	-	-
FCI-stable	$\surd$	-	-
CFCI/MFCI-stable	$\surd$	$\surd$	-
LCFCI/LMFCI-stable	$\surd$	$\surd$	$\surd$

Table 2: Order-dependence issues and corresponding modifications of the FCI-algorithm that remove the problem. A tick mark indicates that the corresponding aspect of the graph is estimated order-independently in the sample version. For example, with FCI-stable the skeleton is estimated order-independently but not the v-structures and the edge orientations.

In the presences of latent and selection variables, one can also use the RFCI-algorithm (Colombo et al., 2012). This algorithm can be viewed as a compromise between PC and FCI, in the sense that its computational complexity is of the same order as PC, but its output can be interpreted causally without assuming causal sufficiency (but is slightly less informative than the output from FCI).

RFCI works as follows. It starts with the first step of PC. It then has a more involved Step 2 to determine the v-structures (Colombo et al., 2012, Lemma 3.1). In particular, for any unshielded triple (X_i, X_j, X_k), it conducts additional tests to check if both X_i and X_j and X_j and X_k are conditionally dependent given $\text{sepset}(X_i, X_j) \setminus \{X_j\}$ found in Step 1. If a conditional independence relationship is detected, the corresponding edge is removed and a minimal separating set is stored. The removal of an edge can create new unshielded triples or destroy some of them. Therefore, the algorithm works with lists to make sure that these actions are order-independent. On the other hand, if both conditional dependencies hold and X_j is not in the separating set for (X_i, X_k), the triple is oriented as a v-structure. Finally, in Step 3 it uses the ten orientation rules of Zhang (2008) with a modified orientation rule for the discriminating paths, that also involves some additional conditional independence tests. The output of the RFCI-algorithm is an (estimated) RFCI-PAG (Colombo et al., 2012, Definition 3.2).

From our results, it immediately follows that RFCI with any of our modifications of the PC-algorithm is correct in the oracle version, in the sense it outputs the true RFCI-PAG. Because of its more involved rules for v-structures and discriminating paths, one needs to make several adaptations to create a fully order-independent algorithm. For example, the additional conditional independence tests conducted for the v-structures are based on the separating sets found in Step 1. As already mentioned before (see Example 2) these separating sets are order-dependent, and therefore also the possible edge deletions based on them are order-dependent, leading to an order-dependent skeleton. To produce an order-independent skeleton one should use a similar approach to the conservative one for the v-structures to make the additional edge removals order-independent. Nevertheless, we can remove a large amount of the order-dependence in the skeleton by using the stable version for the skeleton as a first step. We refer to this modification as RFCI-stable. Note that this procedure does not produce a fully order-independent skeleton, but as shown in Section 5.2, it reduces the order-dependence considerably. Moreover, we can combine this modification with CPC or MPC on the final skeleton to reduce the order-dependence of the v-structures. We refer to these modifications as CRFCI-stable and MRFCI-stable. Finally, we can again use lists for the orientation rules as in the FCI-algorithm to reduce the order-dependence caused by the orientation rules.

Finally, in the presence of directed cycles, one can use the CCD-algorithm (Richardson, 1996). This algorithm can also be made order-independent using a similar approach.

4.5 High-Dimensional Consistency

The original PC-algorithm has been shown to be consistent for certain sparse high-dimensional graphs. In particular, Kalisch and Bühlmann (2007) proved consistency for multivariate Gaussian distributions. More recently, Harris and Drton (2013) showed consistency for the broader class of Gaussian copulas when using rank correlations, under slightly different conditions.

These high-dimensional consistency results allow the DAG $\mathcal{G}$ and the number of observed variables p in $\mathbf{V}$ to grow as a function of the sample size, so that $p = p_n$, $\mathbf{V} = \mathbf{V}_n = (X_{n,1}, \ldots, X_{n,p_n})$ and $\mathcal{G} = \mathcal{G}_n$. The corresponding CPDAGs that represent $\mathcal{G}_n$ are denoted by $\mathcal{C}_n$, and the estimated CPDAGs using tuning parameter α_n are denoted by $\hat{\mathcal{C}}_n(\alpha_n)$. Then the consistency results say that, under some conditions, there exists a sequence α_n such that $P(\hat{\mathcal{C}}_n(\alpha_n) = \mathcal{C}_n) \to 1$ as $n \to \infty$.

These consistency results rely on the fact that the PC-algorithm only performs conditional independence tests between pairs of variables given subsets $\mathbf{S}$ of size less than or equal to the degree of the graph (when no errors are made). We made sure that our modifications still obey this property, and therefore the consistency results of Kalisch and Bühlmann (2007) and Harris and Drton (2013) remain valid for the (L)CPC(-stable) and (L)MPC(-stable) algorithms, under exactly the same conditions as for the original PC-algorithm.

Finally, also the consistency results of Colombo et al. (2012) for the FCI- and RFCI-algorithms remain valid for the (L)CFCI(-stable), (L)MFCI(-stable), CRFCI(-stable), and MRFCI(-stable) algorithms, under exactly the same conditions as for the original FCI- and RFCI-algorithms.

5. Simulations

We compared all algorithms on simulated data from low-dimensional and high-dimensional systems with and without latent variables. In the low-dimensional setting, we compared the modifications of PC, FCI and RFCI. All algorithms performed similarly in this setting, and the results are presented in Appendix A.1. The remainder of this section therefore focuses on the high-dimensional setting, where we compared (L)PC(-stable), (L)CPC(-stable) and (L)MPC(-stable) in systems without latent variables, and RFCI(-stable), CRFCI(-stable) and MRFCI(-stable) in systems with latent variables. We omitted the FCI-algorithm and the modifications with lists for the orientation rules of RFCI because of their computational complexity. Our results show that our modified algorithms perform better than the original algorithms in the high-dimensional settings we considered.

In Section 5.1 we describe the simulation setup. Section 5.2 evaluates the estimation of the skeleton of the CPDAG or PAG (i.e., only looking at the presence or absence of edges), and Section 5.3 evaluates the estimation of the CPDAG or PAG (i.e., also including the edge marks). Appendix A.2 compares the computing time and the number of conditional independence tests performed by PC and PC-stable, showing that PC-stable generally performs more conditional independence tests, and is slightly slower than PC. Finally, Appendix A.3 compares the modifications of FCI and RFCI in two medium-dimensional settings with latent variables, where the number of nodes in the graph is roughly equal to the sample size and we allow somewhat denser graphs. The results indicate that also in this setting our modified versions perform better than the original ones.

5.1 Simulation Setup

We used the following procedure to generate a random weighted DAG with a given number of vertices p and an expected neighborhood size $E(N)$. First, we generated a random adjacency matrix A with independent realizations of Bernoulli$(E(N)/(p-1))$ random variables in the lower triangle of the matrix and zeroes in the remaining entries. Next, we replaced the ones in A by independent realizations of a Uniform$([0.1, 1])$ random variable, where a nonzero entry A_{ij} can be interpreted as an edge from X_j to X_i with weight A_{ij}. (We bounded the edge weights away from zero to avoid problems with near-unfaithfulness.)

We related a multivariate Gaussian distribution to each DAG by letting $X_1 = \epsilon_1$ and $X_i = \sum_{r=1}^{i-1} A_{ir} X_r + \epsilon_i$ for $i = 2, \ldots, p$, where $\epsilon_1, \ldots, \epsilon_p$ are mutually independent $\mathcal{N}(0, 1)$ random variables. The variables $X_1, \ldots, X_p$ then have a multivariate Gaussian distribution with mean zero and covariance matrix $\Sigma = (\mathbf{1} - A)^{-1}(\mathbf{1} - A)^{-T}$, where $\mathbf{1}$ is the $p \times p$ identity matrix.

We generated 250 random weighted DAGs with $p = 1000$ and $E(N) = 2$, and for each weighted DAG we generated an i.i.d. sample of size $n = 50$. The settings were chosen to somewhat resemble the observational yeast gene expression data (see Section 6). In the setting without latents, we simply used all variables. In the setting with latents, we removed half of the variables that had no parents and at least two children, chosen at random.

We estimated each graph for 20 random variable orderings, using the sample versions of (L)PC(-stable), (L)CPC(-stable), and (L)MPC(-stable) in the setting without latents, and the sample versions of RFCI(-stable), CRFCI(-stable), and MRFCI(-stable) in the setting with latents, with tuning parameter $\alpha \in \{0.000625, 0.00125, 0.0025, 0.005, 0.01, 0.02, 0.04\}$.

Thus, from each randomly generated DAG, we obtained 20 estimated CPDAGs or RFCI-PAGs from each algorithm, for each value of α.

5.2 Estimation of the Skeleton

Figure 5 shows the number of edges, the number of errors, and the true discovery rate for the estimated skeletons, when compared to the true CPDAG or true PAG. The figure only compares PC and PC-stable in the setting without latent variables, and RFCI and RFCI-stable in the setting with latent variables, since the modifications for the v-structures and the orientation rules do not affect the estimation of the skeleton.

We first consider the number of estimated errors in the skeleton, shown in the first row of Figure 5. We see that PC-stable and RFCI-stable return estimated skeletons with fewer edges than PC and RFCI, for all values of α. This can be explained by the fact that PC-stable and RFCI-stable tend to perform more tests than PC and RFCI (see also Appendix A.2). Moreover, for all algorithms smaller values of α lead to sparser outputs, as expected. When interpreting these plots, it is useful to know that the average number of edges in the true CPDAGs and PAGs is about 1000. Thus, for all algorithms and almost all values of α, the estimated graphs are too sparse.

The second row of Figure 5 shows that PC-stable and RFCI-stable make fewer errors in the estimation of the skeletons than PC and RFCI, for all values of α. This may be somewhat surprising given the observations above: for most values of α the output of PC and RFCI is too sparse, and the output of PC-stable and RFCI-stable is even sparser. Thus, it must be that PC-stable and RFCI-stable yield a large decrease in the number of false positive edges that outweighs any increase in false negative edges.

This conclusion is also supported by the last row of Figure 5, which shows that PC-stable and RFCI-stable have a better True Discovery Rate (TDR) for all values of α, where the TDR is defined as the proportion of edges in the estimated skeleton that are also present in the true skeleton.

Figure 6 shows more detailed results for the estimated skeletons of PC and PC-stable for one of the 250 graphs (randomly chosen), for four different values of α. For each value of α shown, PC yielded a certain number of stable edges that were present for all 20 variable orderings, but also a large number of extra edges that seem to pop in or out randomly for different orderings. The PC-stable algorithm yielded far fewer edges (shown in red), and roughly captured the edges that were stable among the different variable orderings for PC. The results for RFCI and RFCI-stable show an equivalent picture.

5.3 Estimation of the CPDAGs and PAGs

We now consider estimation of the CPDAG or PAG, that is, also taking into account the edge orientations. For CPDAGs, we summarize the number of estimation errors using the Structural Hamming Distance (SHD), which is defined as the minimum number of edge insertions, deletions, and flips that are needed in order to transform the estimated graph into the true one. For PAGs, we summarize the number of estimation errors by counting the number of errors in the edge marks, which we call "SHD edge marks". For example, if an edge $X_i \to X_j$ is present in the estimated PAG but the true PAG contains $X_i \leftrightarrow X_j$,

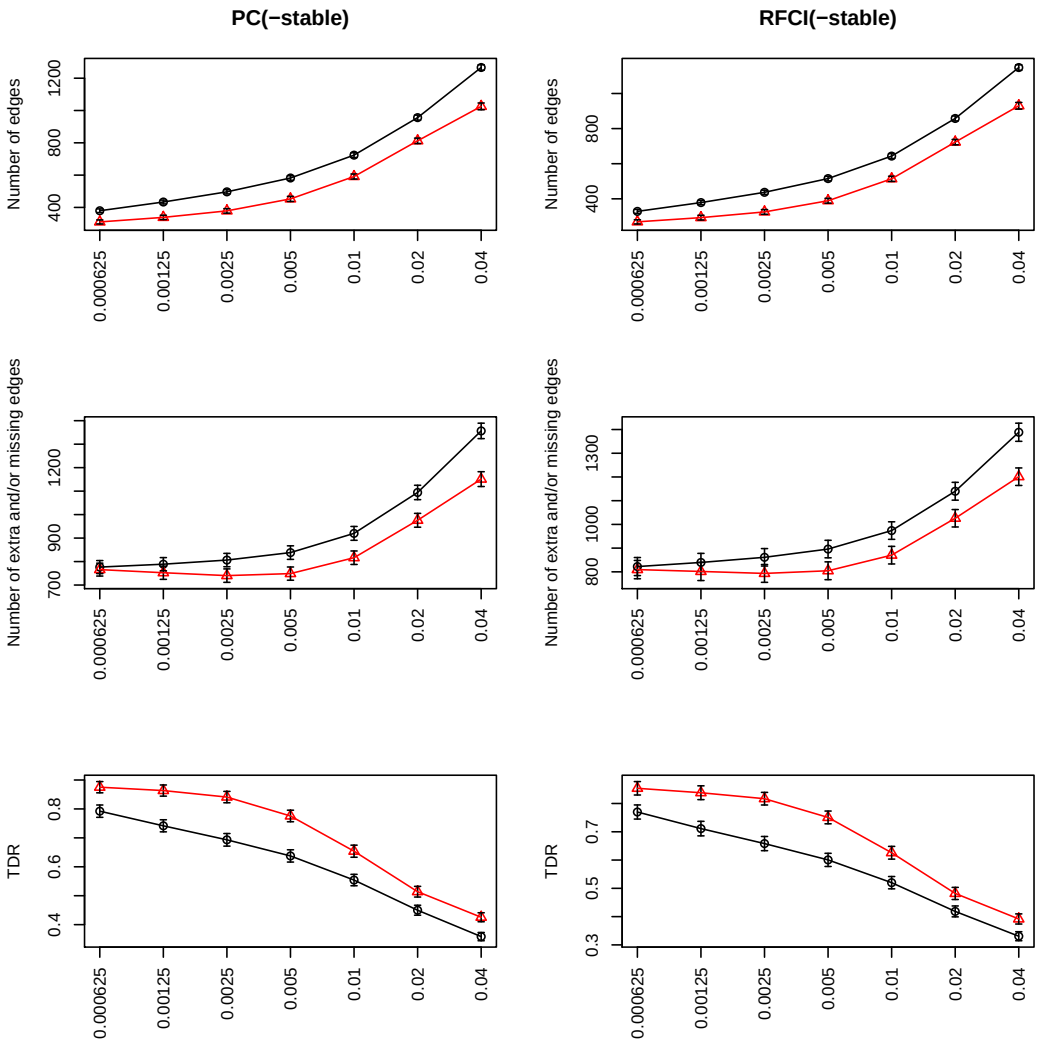
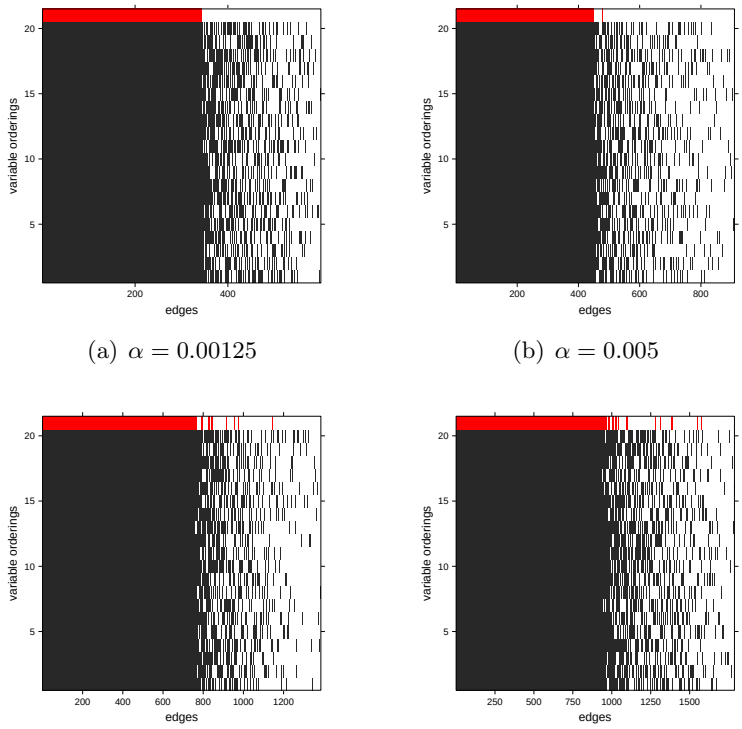


Figure 5: Estimation performance of PC (circles; black line) and PC-stable (triangles; red line) for the skeleton of the CPDAGs (first column of plots), and of RFCI (circles; black line) and RFCI-stable (triangles; red line) for the skeleton of the PAGs (second column of plots), for different values of α (x-axis displayed in log scale). The results are shown as averages plus or minus one standard deviation, computed over 250 randomly generated graphs and 20 random variable orderings per graph. The average number of edges in the true underlying CPDAGs and PAGs is about 1000.



(a) $\alpha = 0.00125$ (b) $\alpha = 0.005$

(c) $\alpha = 0.02$ (d) $\alpha = 0.04$

Figure 6: Estimated edges with the PC-algorithm (black) for 20 random orderings on the variables, as well as with the PC-stable algorithm (red, shown as variable ordering 21), for a random graph from the high-dimensional setting. The edges along the x-axes are ordered according to their presence in the 20 random orderings using the original PC-algorithm. Edges that did not occur for any of the orderings were omitted.

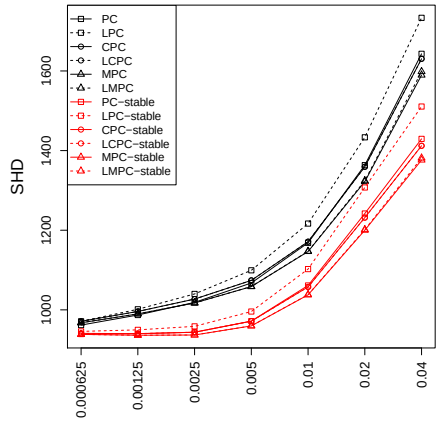
then that counts as one error, while it counts as two errors if the true PAG contains, for example, $X_i \leftarrow X_j$ or X_i and X_j are not adjacent.

Figure 7 shows that the PC-stable and RFCI-stable versions have significantly better estimation performance than the versions with the original skeleton, for all values of α. Moreover, MPC(-stable) and CPC(-stable) perform better than PC(-stable), as do MRFCI(-stable) and CRFCI(-stable) with respect to RFCI(-stable). Finally, for PC the idea to introduce bi-directed edges and lists in LCPC(-stable) and LMPC(-stable) seems to make little difference.

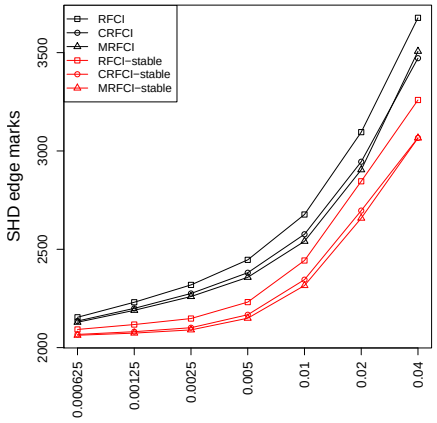
Figure 8 shows the variance in SHD for the CPDAGs, see Figure 8(a), and the variance in SHD edge marks for the PAGs, see Figure 8(b), both computed over the 20 random variable orderings per graph, and then plotted as averages over the 250 randomly generated graphs for the different values of α. The PC-stable and RFCI-stable versions yield significantly smaller variances than their counterparts with unstabilized skeletons. More-

over, the variance is further reduced for (L)CPC-stable and (L)MPC-stable, as well as for CRFCI-stable and MRFCI-stable, as expected.

Figure 9 shows receiver operating characteristic curves (ROC) for the directed edges in the estimated CPDAGs (Figure 9(a)) and PAGs (Figure 9(b)). We see that finding directed edges is much harder in settings that allow for hidden variables, as shown by the lower true positive rates (TPR) and higher false positive rates (FPR) in Figure 9(b). Within each figure, the different versions of the algorithms perform roughly similar, and MPC-stable and MRFCI-stable yield the best ROC-curves.



(a) Estimation performance of (L)PC(-stable), (L)CPC(-stable), and (L)MPC(-stable) for the CPDAGs in terms of SHD.

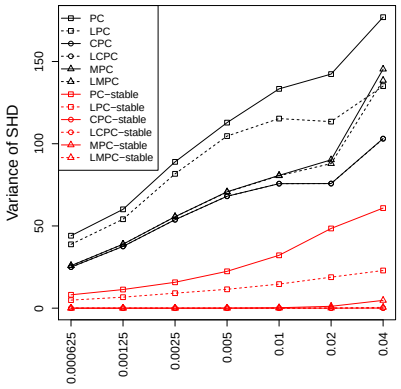


(b) Estimation performance of RFCI(-stable), CRFCI(-stable), and MRFCI(-stable) for the PAGs in terms of SHD edge marks.

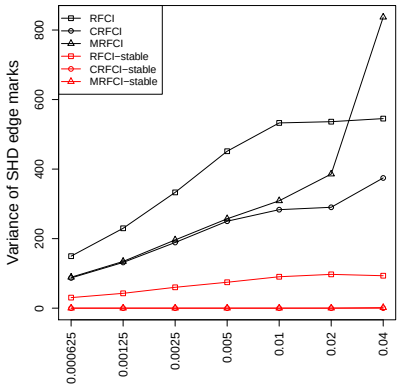
Figure 7: Estimation performance in terms of SHD for the CPDAGs and SHD edge marks for the PAGs, shown as averages over 250 randomly generated graphs and 20 random variable orderings per graph, for different values of α (x-axis displayed in log scale).

6. Yeast Gene Expression Data

We also compared the PC and PC-stable algorithms on the yeast gene expression data (Hughes et al., 2000) that were already briefly discussed in Section 1. We recall that we chose these data since they contain both observational and experimental data, obtained under similar conditions. The observational data consist of gene expression measurements of 5361 genes for 63 wild-type cultures (observational data of size 63×5361). The experimental data consist of gene expression measurements of the same 5361 genes for 234 single-gene deletion mutant strains (experimental data of size 234×5361).



(a) Estimation performance of (L)PC(-stable), (L)CPC(-stable), and (L)MPC(-stable) for the CPDAGs in terms of the variance of SHD.



(b) Estimation performance of RFCI(-stable), CRFCI(-stable), and MRFCI(-stable) for the PAGs in terms of the variance SHD edge marks.

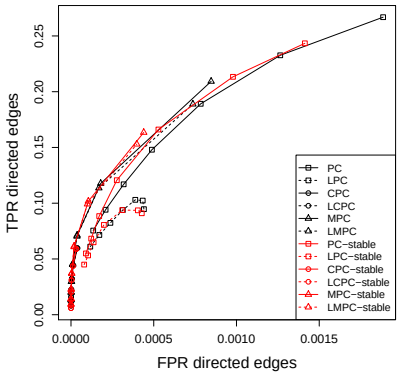
Figure 8: Estimation performance in terms of the variance of the SHD for the CPDAGs and SHD edge marks for the PAGs over the 20 random variable orderings per graph, shown as averages over 250 randomly generated graphs, for different values of α (x-axis displayed in log scale).

In Section 6.1 we consider estimation of the skeleton of the CPDAG, and in Section 6.2 we consider estimation of bounds on causal effects. We used the same data pre-processing as in Maathuis et al. (2010).

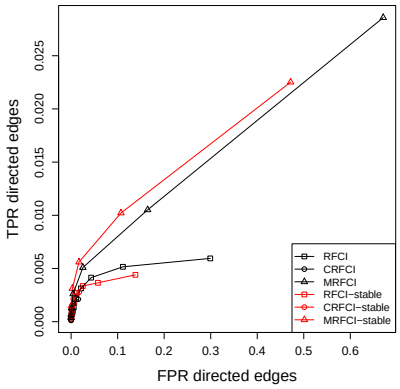
6.1 Estimation of the Skeleton

We applied PC and PC-stable to the (pre-processed) observational data. We saw in Section 1 that the PC-algorithm yielded estimated skeletons that were highly dependent on the variable ordering, as shown in black in Figure 10 for the 26 variable orderings (the original ordering and 25 random orderings of the variables). The PC-stable algorithm does not suffer from this order-dependence, and consequently all these 26 random orderings over the variables produce the same skeleton which is shown in the figure in red. We see that the PC-stable algorithm yielded a far sparser skeleton (2086 edges for PC-stable versus 5015-5159 edges for the PC-algorithm, depending on the variable ordering). Just as in the simulations in Section 5 the order-independent skeleton from the PC-stable algorithm roughly captured the edges that were stable among the different order-dependent skeletons estimated from different variable orderings for the original PC-algorithm.

To make "captured the edges that were stable" somewhat more precise, we defined the following two sets: Set 1 contained all edges (directed edges) that were present for all 26 variable orderings using the original PC-algorithm, and Set 2 contained all edges (directed edges) that were present for at least 90% of the 26 variable orderings using the original



(a) Estimation performance of (L)PC(-stable), (L)CPC(-stable), and (L)MPC(-stable) for the CPDAGs in terms of TPR and FPR for the directed edges.



(b) Estimation performance of RFCI(-stable), CRFCI(-stable), and MRFCI(-stable) for the PAGs in terms of TPR and FPR for the directed edges.

Figure 9: Estimation performance in terms of TPR and FPR for the directed edges in CPDAGs and PAGs, shown as averages over 250 randomly generated graphs and 20 random variable orderings per graph, where every curve is plotted with respect to the different values of α.

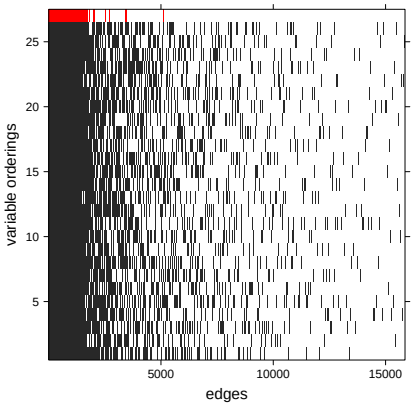
PC-algorithm. Set 1 contained 1478 edges (7 directed edges), while Set 2 contained 1700 edges (20 directed edges).

Table 3 shows how well the PC and PC-stable algorithms could find these stable edges in terms of number of edges in the estimated graphs that are present in Sets 1 and 2 (IN), and the number of edges in the estimated graphs that are not present in Sets 1 and 2 (OUT). We see that the number of estimated edges present in Sets 1 and 2 is about the same for both algorithms, while the output of the PC-stable algorithm has far fewer edges which are not present in the two specified sets.

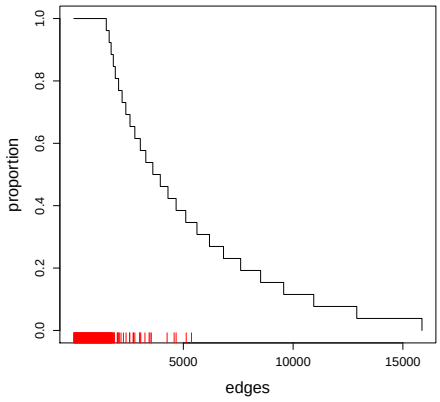
Throughout our analyses of the yeast data, we used tuning parameter $\alpha = 0.01$, as in Maathuis et al. (2010). We are not aware of any fully satisfactory method to choose α in practice. In Appendix B, we briefly mention two possibilities that were described in Maathuis et al. (2009): optimizing a Bayesian Information Criterion (BIC) and stability selection (Meinshausen and Bühlmann, 2010).

6.2 Estimation of Causal Effects

We used the experimental data as the gold standard for estimating the total causal effects of the 234 deleted genes on the remaining 5361 (Maathuis et al., 2010). We then defined the top 10% of the largest effects in absolute value as the target set of effects, and we evaluated how well IDA (Maathuis et al., 2009, 2010) identified these effects from the observational data.



(a) As Figure 1(a), plus the edges occurring in the unique estimated skeleton using the PC-stable algorithm over the same 26 variable orderings (red, shown as variable ordering 27).



(b) The step function shows the proportion of the 26 variable orderings in which the edges were present for the original PC-algorithm, where the edges are ordered as in Figure 10(a). The red bars show the edges present in the estimated skeleton using the PC-stable algorithm.

Figure 10: Analysis of estimated skeletons of the CPDAGs for the yeast gene expression data (Hughes et al., 2000), using the PC and PC-stable algorithms with tuning parameter $\alpha = 0.01$. The PC-stable algorithm yields an order-independent skeleton that roughly captures the edges that were stable among the different variable orderings for the original PC-algorithm.

		Edges		Directed edges	
		PC	PC-stable	PC	PC-stable
Set 1	IN	1478 (0)	1478 (0)	7 (0)	7 (0)
	OUT	3606 (38)	607 (0)	4786 (47)	1433 (7)
Set 2	IN	1688 (3)	1688 (0)	19 (1)	20 (0)
	OUT	3396 (39)	397 (0)	4774 (47)	1420 (7)

Table 3: Number of edges in the estimated graphs that are present in Sets 1 and 2 (IN), and the number of edges in the estimated graphs that are not present in Sets 1 and 2 (OUT). The results are shown as averages (standard deviations) over the 26 variable orderings.

Figure 1(b) showed that IDA with the original PC-algorithm is highly order-dependent. Figure 11 shows the same analysis with PC-stable (solid black lines). We see that using PC-stable generally yielded better and more stable results than the original PC-algorithm. Note that some of the curves for PC-stable are worse than the reference curve of Maathuis et al. (2010) towards the beginning of the curves. This can be explained by the fact that the original variable ordering seems to be especially "lucky" for this part of the curve (see Figure 1(b)). There is still variability in the ROC curves in Figure 11 due to the order-dependent v-structures (because of order-dependent separating sets) and orientations in the PC-stable algorithm, but this variability is less prominent than in Figure 1(b). Finally, we see that there are 3 curves that produce a very poor fit.

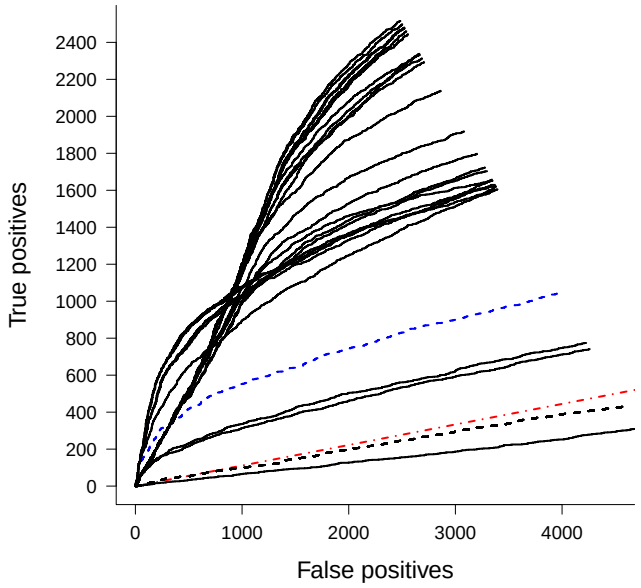


Figure 11: ROC curves corresponding to the 25 random orderings of the variables for the analysis of yeast gene expression data (Hughes et al., 2000), where the curves are generated as in Maathuis et al. (2010) but using PC-stable (solid black lines) and MPC-stable and CPC-stable (dashed black lines) with $\alpha = 0.01$. The ROC curves from Maathuis et al. (2010) (dashed blue) and the one for random guessing (dashed-dotted red) are shown as references. The resulting causal rankings are less order-dependent.

Using CPC-stable and MPC-stable helps in stabilizing the outputs, and in fact all the 25 random variable orderings produce almost the same CPDAGs for both modifications. Unfortunately, these estimated CPDAGs are almost entirely undirected (around 90 directed edges among the 2086 edges) which leads to a large equivalence class and consequently to a poor performance in IDA, see the dashed black line in Figure 11 which corresponds to the 25 random variable orderings for both CPC-stable and MPC-stable algorithms.

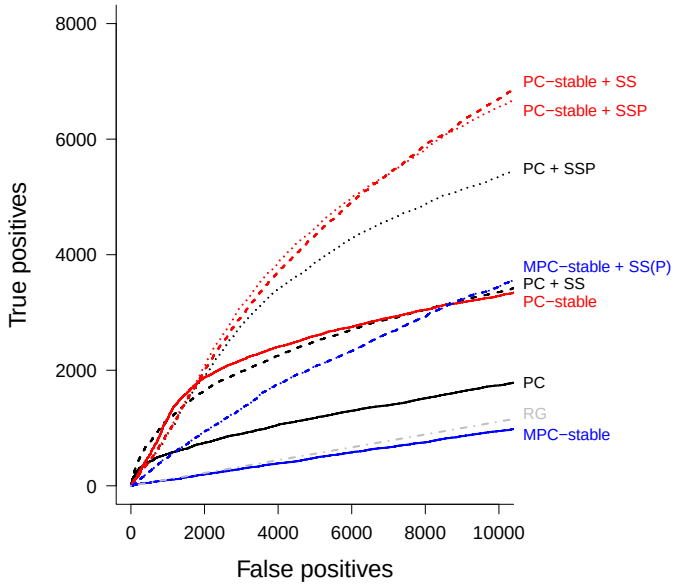


Figure 12: Analysis of the yeast gene expression data (Hughes et al., 2000) for PC, PC-stable, and MPC-stable algorithms using the original ordering over the variables (solid lines), using 100 runs stability selection without permuting the variable orderings, labelled as + SS (dashed lines), and using 100 runs stability selection with permuting the variable orderings, labelled as + SSP (dotted lines). The grey line labelled as RG represents the random guessing.

Another possible solution for the order-dependence orientation issues would be to use stability selection (Meinshausen and Bühlmann, 2010) to find the most stable orientations among the runs. In fact, Stekhoven et al. (2012) already proposed a combination of IDA and stability selection which led to improved performance when compared to IDA alone, but they used the original PC-algorithm and did not permute the variable ordering. We present here a more extensive analysis, where we consider the PC-algorithm (black lines), the PC-stable algorithm (red lines), and the MPC-stable algorithm (blue lines). Moreover, for each one of these algorithms we propose three different methods to estimate the CPDAGs and the causal effects: (1) use the original ordering of the variables (solid lines); (2) use the same methodology used in Stekhoven et al. (2012) with 100 stability selection runs but without permuting the variable orderings (labelled as + SS; dashed lines); and (3) use the same methodology used in Stekhoven et al. (2012) with 100 stability selection runs but permuting the variable orderings in each run (labelled as + SSP; dotted lines). The results are shown in Figure 12 where we investigate the performance for the top 20000 effects instead of the 5000 as in Figures 1(b) and 11.

We see that PC with stability selection and permuted variable orderings (PC + SSP) loses some performance at the beginning of the curve when compared to PC with standard

stability selection (PC + SS), but it has much better performance afterwards. The PC-stable algorithm with the original variable ordering performs very similar to PC plus stability selection (PC + SS) along the whole curve. Moreover, PC-stable plus stability selection (PC-stable + SS and PC-stable + SSP), loses a bit at the beginning of the curves but picks up much more signal later on in the curve. It is interesting to note that for PC-stable with stability selection, it makes little difference if the variable orderings are further permuted or not, even though PC-stable is not fully order-independent (see Figure 11). In fact, PC-stable plus stability selection (with or without permuted variable orderings) produces the best fit over all results.

7. Discussion

Due to their computational efficiency, constraint-based causal structure learning algorithms are often used in sparse high-dimensional settings. We have seen, however, that especially in these settings the order-dependence in these algorithms is highly problematic.

In this paper, we investigated this issue systematically, and resolved the various sources of order-dependence. There are of course many ways in which the order-dependence issues could be resolved, and we designed our modifications to be as simple as possible. Moreover, we made sure that existing high-dimensional consistency results for PC-, FCI- and RFCI-algorithms remain valid for their modifications under the same conditions. We showed that our proposed modifications yield improved and more stable estimation in sparse high-dimensional settings for simulated data, while their performances are similar to the performances of the original algorithms in low-dimensional settings.

Additionally to the order-dependence discussed in this paper, there is another minor type of order-dependence in the sense that the output of these algorithms also depends on the order in which the final orientation rules for the edges are applied. The reason is that an edge(mark) could be eligible for orientation by several orientation rules, and might be oriented differently depending on which rule is applied first. In our analyses, we have always used the original orderings in which the rules were given.

Compared to the adaptation of Cano et al. (2008), the modifications we propose are much simpler and we made sure that they preserve existing soundness, completeness, and high-dimensional consistency results. Finally, our modifications can be easily used together with other adaptations of constraint-based algorithms, for example hybrid versions of PC with score-based methods (Singh and Valtorta, 1993; Spirtes and Meek, 1995; van Dijk et al., 2003) or the PC* algorithm (Spirtes et al., 2000, Section 5.4.2.3).

All software is implemented in the R-package `pcalg` (Kalisch et al., 2012).

Acknowledgments

This research was supported in part by Swiss NSF grant 200021-129972. We thank Richard Fox, Markus Kalisch and Thomas Richardson for their valuable comments.

Appendix A. Additional Simulation Results

We now present additional simulation results for low-dimensional settings (Appendix A.1), high-dimensional settings (Appendix A.2) and medium-dimensional settings (Appendix A.3).

A.1 Estimation Performance in Low-Dimensional Settings

We considered the estimation performance in low-dimensional settings with less sparse graphs.

For the scenario without latent variables, we generated 250 random weighted DAGs with $p = 50$ and $E(N) = \{2, 4\}$, as described in Section 5.1. For each weighted DAG we generated an i.i.d. sample of size $n = 1000$. We then estimated each graph for 50 random orderings of the variables, using the sample versions of (L)PC(-stable), (L)CPC(-stable), and (L)MPC(-stable) with tuning parameter $\alpha \in \{0.000625, 0.00125, 0.0025, 0.005, 0.01, 0.02, 0.04\}$ for $E(N) = 2$ and $\alpha \in \{0.005, 0.01, 0.02, 0.04, 0.08, 0.16, 0.32\}$ for $E(N) = 4$ for the partial correlation tests. Thus, for each randomly generated graph, we obtained 50 estimated CPDAGs from each algorithm, for each value of α. Figure 13 shows the estimation performance of PC (circle; black line) and PC-stable (triangles; red line) for the skeleton. Figure 14 shows the estimation performance of all modifications of PC and PC-stable with respect to the CPDAGs in terms of SHD, and in terms of the variance of the SHD over the 50 random variable orderings per graph.

For the scenario with latent variables, we generated 120 random weighted DAGs with $p = 50$ and $E(N) = 2$, as described in Section 5.1. For each DAG we generated an i.i.d. sample size of $n = 1000$. To assess the impact of latent variables, we randomly defined in each DAG half of the variables that have no parents and at least two children to be latent. We then estimated each graph for 20 random orderings of the observed variables, using the sample versions of FCI(-stable), CFCI(-stable), MFCI(-stable), RFCI(-stable), CRFCI(-stable), and MRFCI(-stable) with tuning parameter $\alpha \in \{0.0025, 0.005, 0.01, 0.02, 0.04, 0.08\}$ for the partial correlation tests. Thus, for each randomly generated graph, we obtained 20 estimated PAGs from each algorithm, for each value of α. Figure 15 shows the estimation performance of FCI (circles; black dashed line), FCI-stable (triangles; red dashed line), RFCI (circles; black solid line), and RFCI-stable (triangles; red solid line) for the skeleton. Figure 16 shows the estimation performance for the PAGs in terms of SHD edge marks, and in terms of the variance of the SHD edge marks over the 20 random variable orderings per graph.

Regarding the skeletons of the CPDAGs and PAGs, the estimation performances between PC and PC-stable, as well as between (R)FCI and (R)FCI-stable are basically indistinguishable for all values of α. However, Figure 15 shows that FCI(-stable) returns graphs with slightly fewer edges than RFCI(-stable), for all values of α. This is related to the fact that FCI(-stable) tends to perform more tests than RFCI(-stable).

Regarding the CPDAGs and PAGs, the performance of the modifications of PC and (R)FCI (black lines) are very close to the performance of PC-stable and (R)FCI-stable (red lines). Moreover, CPC(-stable) and MPC(-stable) as well as C(R)FCI(-stable) and M(R)FCI(-stable) perform better in particular in reducing the variance of the SHD and SHD edge marks, respectively. This indicates that most of the order-dependence in the low-dimensional setting is in the orientation of the edges.

We also note that in all proposed measures there are only small differences between modifications of FCI and of RFCI.

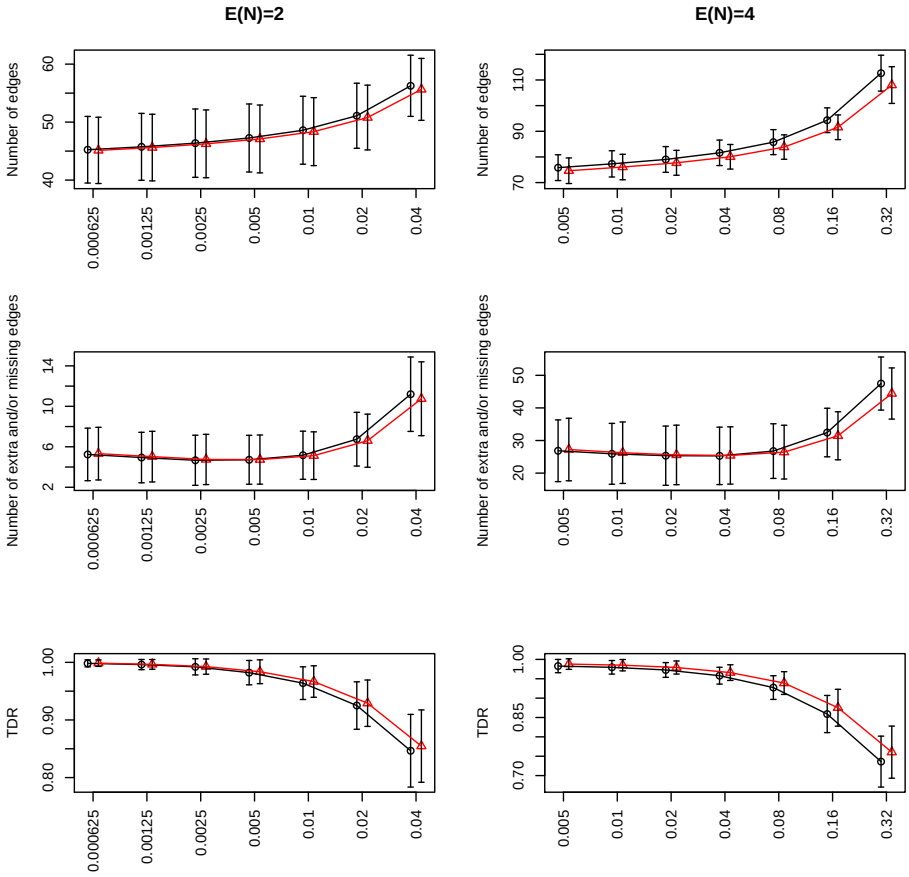


Figure 13: Estimation performance of PC (circles; black line) and PC-stable (triangles; red line) for the skeleton of the CPDAGs, for different values of α (x-axis displayed in log scale) in both low-dimensional settings. The results are shown as averages plus or minus one standard deviation, computed over 250 randomly generated graphs and 50 random variable orderings per graph, and slightly shifted up and down from the real values of α for a better visualization.

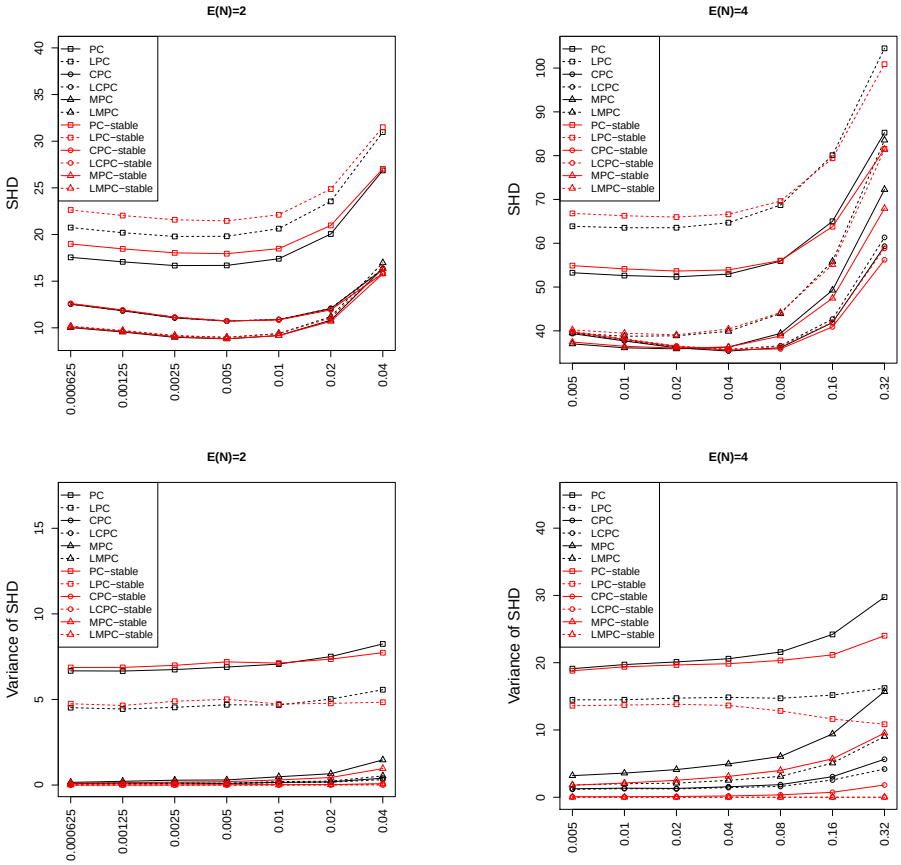


Figure 14: Estimation performance of (L)PC(-stable), (L)CPC(-stable), and (L)MPC(-stable) for the CPDAGs in the low-dimensional settings, for different values of α. The first row of plots shows the performance in terms of SHD, shown as averages over 250 randomly generated graphs and 50 random variable orderings per graph. The second row of plots shows the performance in terms of the variance of the SHD over the 50 random variable orderings per graph, shown as averages over 250 randomly generated graphs.

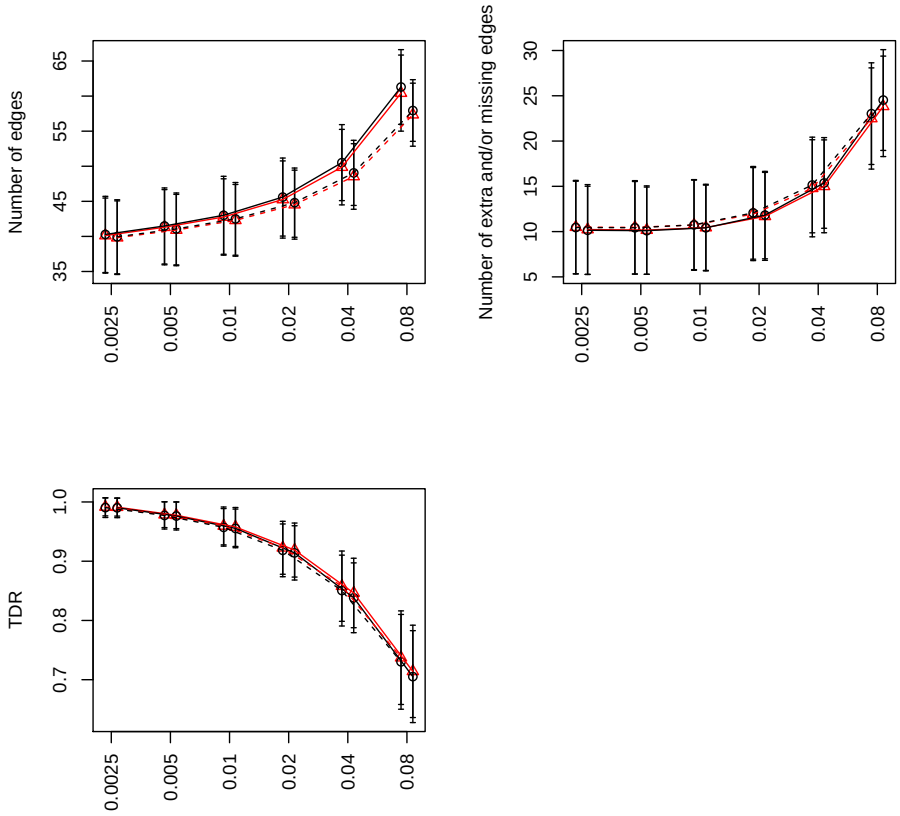


Figure 15: Estimation performance of FCI (circles; black dashed line), FCI-stable (triangles; red dashed line), RFCI (circles; black solid line), and RFCI-stable (triangles; red solid line), for the skeleton of the PAGs for different values of α (x-axis displayed in log scale) in the low-dimensional setting. The results are shown as averages plus or minus one standard deviation, computed over 120 randomly generated graphs and 20 random variable orderings per graph, and slightly shifted up and down from the real values of α for a better visualization.

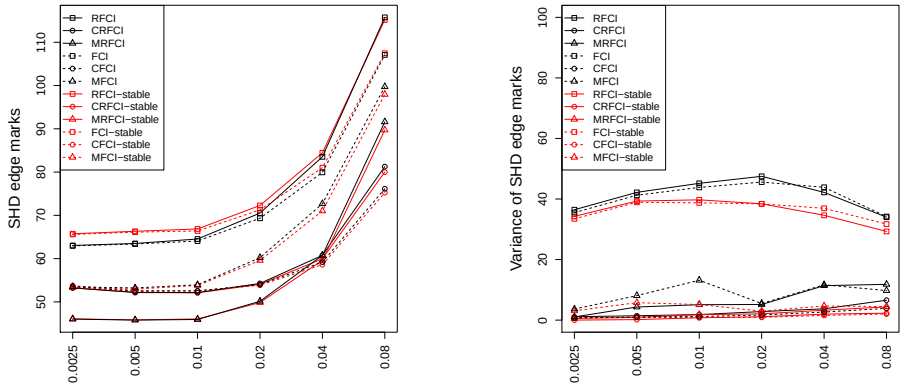


Figure 16: Estimation performance of the modifications of FCI(-stable) and RFCI(-stable) for the PAGs in the low-dimensional setting, for different values of α. The left panel shows the performance in terms of SHD edge marks, shown as averages over 120 randomly generated graphs and 20 random variable orderings per graph. The right panel shows the performance in terms of the variance of the SHD edge marks over the 20 random variable orderings per graph, shown as averages over 120 randomly generated graphs.

A.2 Number of Tests and Computing Time

We consider the number of tests and the computing time of PC and PC-stable in the high-dimensional setting described in Section 5.1.

One can easily deduce that Step 1 of the PC- and PC-stable algorithms perform the same number of tests for $\ell = 0$, because the adjacency sets do not play a role at this stage. Moreover, for $\ell = 1$ PC-stable performs at least as many tests as PC, since the adjacency sets $a(X_i)$ (see Algorithm 4.1) are always supersets of the adjacency sets $\text{adj}(X_i)$ (see Algorithm 3.2). For larger values of ℓ, however, it is difficult to analyze the number of tests analytically.

Table 4 therefore shows the average number of tests that were performed by Step 1 of the two algorithms, separated by size of the conditioning set, where we considered the high-dimensional setting with $\alpha = 0.04$ (see Section 5.2) since this was most computationally intensive. As expected the number of marginal correlation tests was identical for both algorithms. For $\ell = 1$, PC-stable performed slightly more than twice as many tests as PC, amounting to about 1.36×10^5 additional tests. For $\ell = 2$, PC-stable performed more tests than PC, amounting to 3.4×10^3. For larger values of ℓ, PC-stable performed fewer tests than PC, since the additional tests for $\ell = 1$ and $\ell = 2$ lead to a sparser skeleton. However, since PC also performed relatively few tests for larger values of ℓ, the absolute difference in the number of tests for large ℓ is rather small. In total, PC-stable performed about 1.39×10^5 more tests than PC.

Table 5 shows the average runtime of the PC- and PC-stable algorithms. We see that PC-stable is somewhat slower than PC for all values of α, which can be explained by the fact that PC-stable tends to perform a larger number of tests (cf. Table 4).

	PC-algorithm	PC-stable algorithm
$\ell = 0$	5.21×10^5 (1.95×10^2)	5.21×10^5 (1.95×10^2)
$\ell = 1$	1.29×10^5 (2.19×10^3)	2.65×10^5 (4.68×10^3)
$\ell = 2$	1.10×10^4 (5.93×10^2)	1.44×10^4 (8.90×10^2)
$\ell = 3$	1.12×10^3 (1.21×10^2)	5.05×10^2 (8.54×10^1)
$\ell = 4$	9.38×10^1 (2.86×10^1)	3.08×10^1 (1.78×10^1)
$\ell = 5$	2.78×10^0 (4.53×10^0)	0.65×10^0 (1.94×10^0)
$\ell = 6$	0.02×10^0 (0.38×10^0)	-
Total	6.62×10^5 (2.78×10^3)	8.01×10^5 (5.47×10^3)

Table 4: Number of tests performed by Step 1 of the PC and PC-stable algorithms for each size of the conditioning sets ℓ, in the high-dimensional setting with $p = 1000$, $n = 50$ and $\alpha = 0.04$. The results are shown as averages (standard deviations) over 250 random graphs and 20 random variable orderings per graph.

	PC-algorithm	PC-stable algorithm	PC / PC-stable
$\alpha = 0.000625$	111.79 (7.54)	115.46 (7.47)	0.97
$\alpha = 0.00125$	110.13 (6.91)	113.77 (7.07)	0.97
$\alpha = 0.025$	115.90 (12.18)	119.67 (12.03)	0.97
$\alpha = 0.05$	116.14 (9.50)	119.91 (9.57)	0.97
$\alpha = 0.01$	121.02 (8.61)	125.81 (8.94)	0.96
$\alpha = 0.02$	131.42 (13.98)	139.54 (14.72)	0.94
$\alpha = 0.04$	148.72 (14.98)	170.49 (16.31)	0.87

Table 5: Run time in seconds (computed on an AMD Opteron(tm) Processor 6174 using R 2.15.1.) of PC and PC-stable for the high-dimensional setting with $p = 1000$ and $n = 50$. The results are shown as averages (standard deviations) over 250 random graphs and 20 random variable orderings per graph.

A.3 Estimation Performance in Settings where $p = n$

Finally, we consider two settings for the scenario with latent variables, where we generated 250 random weighted DAGs with $p = 50$ and $E(N) = \{2, 4\}$, as described in Section 5.1. For each DAG we generated an i.i.d. sample size of $n = 50$. We again randomly defined in each

DAG half of the variables that have no parents and at least two children to be latent. We then estimated each graph for 50 random orderings of the observed variables, using the sample versions of FCI(-stable), CFCI(-stable), MFCI(-stable), RFCI(-stable), CRFCI(-stable), and MRFCI(-stable) with tuning parameter $\alpha \in \{0.0025, 0.005, 0.01, 0.02, 0.04, 0.08, 0.16\}$ for $E(N) = 2$ and $\alpha \in \{0.005, 0.01, 0.02, 0.04, 0.08, 0.16, 0.32\}$ for $E(N) = 4$. Thus, for each randomly generated graph, we obtained 50 estimated PAGs from each algorithm, for each value of α.

Figure 17 shows the estimation performance for the skeleton. The (R)FCI-stable versions (red lines) lead to slightly sparser graphs and slightly better performance in TDR than (R)FCI versions (black lines) in both settings.

Figures 18 shows the estimation performance of all modifications of (R)FCI with respect to the PAGs in terms of SHD edge marks, and in terms of the variance of the SHD edge marks over the 50 random variable orderings per graph. The (R)FCI-stable versions produce a better fit than the (R)FCI versions. Moreover, C(R)FCI(-stable) and M(R)FCI(-stable) perform similarly for sparse graphs and they improve the fit, while in denser graphs M(R)FCI(-stable) still improves the fit and it performs much better than C(R)FCI(-stable) for the SHD edge marks. Again we see little difference between modifications of RFCI and FCI with respect to all measures.

Appendix B. Choice of the Tuning Parameter in the PC-Algorithm

We now discuss two possible methods for the choice of α in the PC-algorithm: one based on optimizing a Bayesian Information Criterion (BIC), and another based on stability selection.

In order to optimize the BIC, we take a grid of α's. For each α, we compute the estimated CPDAG $\hat{G}(\alpha)$. Based on a DAG $\hat{G}'(\alpha)$ in the Markov equivalence class described $\hat{G}(\alpha)$, we then compute the maximum likelihood estimators $\hat{\Sigma}_{\hat{G}'(\alpha)}$ and $\hat{\mu}$ for the covariance matrix and mean vector of the Gaussian distribution of our variables $X_1, \ldots, X_p$ (Marchetti et al., 2012). Finally, we choose α to minimize

$$-2\ell\left(\hat{\Sigma}_{\hat{G}'(\alpha)}, \hat{\mu}\right) + \log n\left(\sum_{i \leq j} 1_{(\hat{\Sigma}_{\hat{G}'(\alpha)})_{ij} \neq 0} + p\right),$$

where $\ell(\cdot)$ denotes the log-likelihood of a p-dimensional multivariate Gaussian distribution. We point out, however, that one carefully needs to consider the behavior of BIC in the high-dimensional setting.

Another approach to tune the PC-algorithm is based on stability selection (Meinshausen and Bühlmann, 2010), which we also used in Section 6 (with permutations of the variable orderings) to solve order-dependence issues. For a rather large α, or for a range of α's, one can investigate which edges are stable under a subsampling procedure, where stability is measured in terms of the relative frequency of occurrence of (directed or undirected) edges under the sub-sampling scheme. An edge is kept if it is stable, i.e., if the corresponding subsampling frequency is larger than a certain cut-off. One can also apply a stability selection procedure to IDA, where one considers the stability of the ranking of causal effects. The latter approach was taken in Stekhoven et al. (2012).

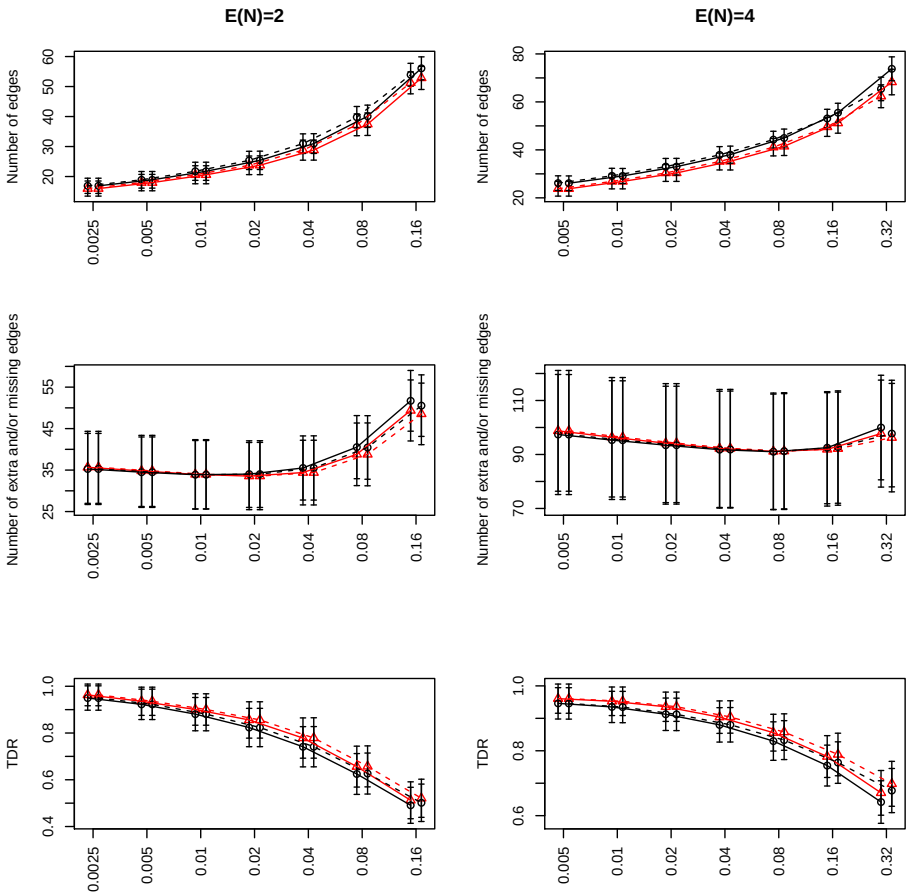


Figure 17: Estimation performance of FCI (circles; black dashed line), FCI-stable (triangles; red dashed line), RFCI (circles; black solid line), and RFCI-stable (triangles; red solid line), for the skeleton of the PAGs for different values of α (x-axis displayed in log scale) in two settings where $p = n$. The results are shown as averages plus or minus one standard deviation, computed over 250 randomly generated graphs and 50 random variable orderings per graph, and slightly shifted up and down from the real values of α for a better visualization.

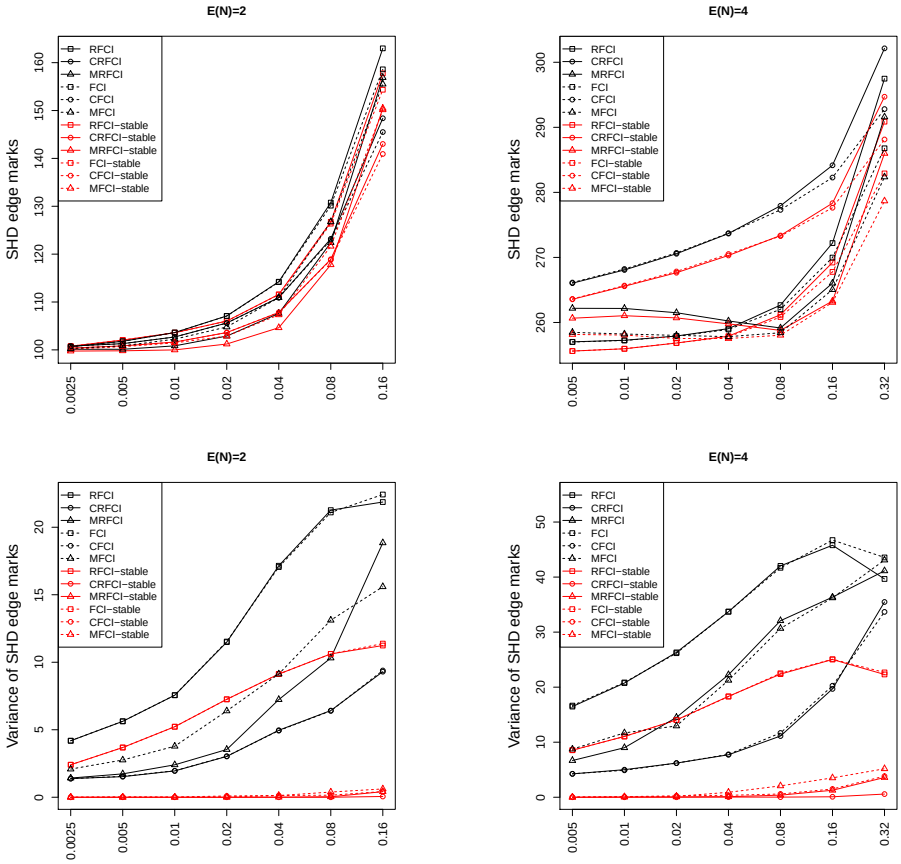


Figure 18: Estimation performance of the modifications of FCI(-stable) and RFCI(-stable) for the PAGs in settings where $p = n$, for different values of α. The first row of plots shows the performance in terms of SHD edge marks, shown as averages over 250 randomly generated graphs and 50 random variable orderings per graph. The second row of plots shows the performance in terms of the variance of the SHD edge marks over the 50 random variable orderings per graph, shown as averages over 250 randomly generated graphs.

References

R.A. Ali, T.S. Richardson, and P. Spirtes. Markov equivalence for ancestral graphs. *Ann. Statist.*, 37:2808–2837, 2009.

S. A. Andersson, D. Madigan, and M. D. Perlman. A characterization of Markov equivalence classes for acyclic digraphs. *Ann. Statist.*, 25:505–541, 1997.

A. Cano, M. Gómez-Olmedo, and S. Moral. A score based ranking of the edges for the PC algorithm. In *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models (PGM 2008)*, pages 41–48, 2008. Available at http://pgm08.cs.aau.dk/online_proc.html.

D.M. Chickering. Learning equivalence classes of Bayesian-network structures. *J. Mach. Learn. Res.*, 2:445–498, 2002.

T. Claassen, J. Mooij, and T. Heskes. Learning sparse causal models is not NP-hard. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI-2013)*, pages 172–181. AUAI Press, Corvallis, 2013.

D. Colombo, M.H. Maathuis, M. Kalisch, and T.S. Richardson. Learning high-dimensional directed acyclic graphs with latent and selection variables. *Ann. Statist.*, 40:294–321, 2012.

D. Dash and M.J. Druzdzel. A hybrid anytime algorithm for the construction of causal models from sparse data. In *Proceedings of the Fifteenth Conference on Uncertainty on Artificial Intelligence (UAI-1999)*, pages 142–149. Morgan Kaufmann, San Francisco, 1999.

A.P. Dawid. Conditional independence for statistical operations. *Ann. Statist.*, 8:598–617, 1980.

N. Harris and M. Drton. PC algorithm for nonparanormal graphical models. *J. Mach. Learn. Res.*, 14:3365–3383, 2013.

T.R. Hughes, M.J. Marton, A.R. Jones, C.J. Roberts, R. Stoughton, C.D. Armour, H.A. Bennett, E. Coffey, H. Dai, Y.D. He, et al. Functional discovery via a compendium of expression profiles. *Cell*, 102:109–126, 2000.

M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *J. Mach. Learn. Res.*, 8:613–636, 2007.

M. Kalisch, B.A.G. Fellinghauer, E. Grill, M.H. Maathuis, U. Mansmann, P. Bühlmann, and G. Stucki. Understanding human functioning using graphical models. *BMC Med. Res. Methodol.*, 10(14), 2010.

M. Kalisch, M. Mächler, D. Colombo, M.H. Maathuis, and P. Bühlmann. Causal inference using graphical models with the R package pcalg. *J. Stat. Softw.*, 47(11), 2012.

M.H. Maathuis, M. Kalisch, and P. Bühlmann. Estimating high-dimensional intervention effects from observational data. *Ann. Statist.*, 37:3133–3164, 2009.

M.H. Maathuis, D. Colombo, M. Kalisch, and P. Bühlmann. Predicting causal effects in large-scale systems from observational data. *Nature Methods*, 7:247–248, 2010.

G.M. Marchetti, M. Drton, and K. Sadeghi. R-package `ggm`: Graphical Gaussian Models. Available at http://cran.r-project.org, 2012.

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

N. Meinshausen and P. Bühlmann. Stability selection. *J. R. Stat. Soc. Series B*, 72:417–473, 2010.

R. Nagarajan, S. Datta, M. Scutari, M. Beggs, G. Nolen, and C. Peterson. Functional relationships between genes associated with differentiation potential of aged myogenic progenitors. *Front. Physiol.*, 1(21), 2010.

J. Pearl. *Causality: Models, reasoning, and inference*. Cambridge University Press, Cambridge, 2000.

J. Pearl. Causal inference in statistics: An overview. *Statistics Surveys*, 3:96–146, 2009.

J. Ramsey, J. Zhang, and P. Spirtes. Adjacency-faithfulness and conservative causal inference. In *Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence (UAI-2006)*. AUAI Press, Arlington, 2006.

T.S. Richardson. A discovery algorithm for directed cyclic graphs. In *Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence (UAI-1996)*, pages 454–461. Morgan Kaufmann, San Francisco, 1996.

T.S. Richardson and P. Spirtes. Ancestral graph Markov models. *Ann. Statist.*, 30:962–1030, 2002.

M. Singh and M. Valtorta. An algorithm for the construction of Bayesian network structures from data. In *Proceedings of the 9th Annual Conference on Uncertainty in Artificial Intelligence (UAI-1993)*, pages 259–265. Morgan Kaufmann, San Francisco, 1993.

P. Spirtes. An anytime algorithm for causal inference. In *Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics*, pages 213–221. Morgan Kaufmann, San Francisco, 2001.

P. Spirtes and C. Meek. Learning Bayesian networks with discrete variables from data. In *Proceeding of the First International Conference on Knowledge Discovery and Data Mining (KDD-1995)*, pages 294–299, AAAI, Menlo Park, 1995.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, prediction, and search*. Springer-Verlag, New York, 1993.

P. Spirtes, C. Meek, and T.S. Richardson. An algorithm for causal inference in the presence of latent variables and selection bias. In C. Glymour and G.F. Cooper, editors, *Computation, Causation and Discovery*, pages 211–252. MIT Press, Cambridge, 1999.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, Cambridge, second edition, 2000. With additional material by David Heckerman, Christopher Meek, Gregory F. Cooper and Thomas Richardson.

D.J. Stekhoven, I. Moraes, G. Sveinbjörnsson, L. Hennig, M.H. Maathuis, and P. Bühlmann. Causal stability ranking. *Bioinformatics*, 28:2819–2823, 2012.

S. van Dijk, L.C. van der Gaag, and D. Thierens. A skeleton-based approach to learning Bayesian networks from data. In *Proceedings of the 7th Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-2003)*, pages 132–143. Springer, Berlin, 2003.

J. Zhang. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artif. Intell.*, 172:1873–1896, 2008.

X. Zhang, X.M. Zhao, K. He, L. Lu, Y. Cao, J. Liu, J.K. Hao, Z.P. Liu, and L. Chen. Inferring gene regulatory networks from gene expression data by path consistency algorithm based on conditional mutual information. *Bioinformatics*, 28:98–104, 2012.

Journal of Machine Learning Research 15 (2014) 3963-4009 Submitted 5/14; Published 12/14

Effective Sampling and Learning for Mallows Models with Pairwise-Preference Data

Tyler Lu TL@CS.TORONTO.EDU
Craig Boutilier CEBLY@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
6 King's College Rd.
Toronto, ON, Canada M5S 3G4

Editor: Guy Lebanon

Abstract

Learning preference distributions is a critical problem in many areas (e.g., recommender systems, IR, social choice). However, many existing learning and inference methods impose restrictive assumptions on the form of user preferences that can be admitted as evidence. We relax these restrictions by considering as data arbitrary *pairwise comparisons* of alternatives, which represent the fundamental building blocks of ordinal rankings. We develop the first algorithms for learning Mallows models (and mixtures thereof) from pairwise comparison data. At the heart of our technique is a new algorithm, the *generalized repeated insertion model (GRIM)*, which allows sampling from arbitrary ranking distributions, and conditional Mallows models in particular. While we show that sampling from a Mallows model with pairwise evidence is computationally difficult in general, we develop approximate samplers that are exact for many important special cases—and have provable bounds with pairwise evidence—and derive algorithms for evaluating log-likelihood, learning Mallows mixtures, and non-parametric estimation. Experiments on real-world data sets demonstrate the effectiveness of our approach.[1]

Keywords: preference learning, ranking, incomplete data, Mallows models, mixture models

1. Introduction

With the abundance of preference data from search engines, review sites, etc., there is tremendous demand for learning detailed models of user preferences to support personalized recommendation, information retrieval, social choice, and other applications. Much work has focused on ordinal preference models and learning user or group *rankings* of alternatives or items. Within this setting, we can distinguish two classes of models. First, we may wish to learn an underlying *objective* (or "correct") ranking from noisy data or noisy expressions of user preferences (e.g., as in web search, where user selection suggests relevance), a view adopted frequently in IR and "learning to rank" (Burges, 2010) and occasionally in social choice (Young, 1995). Second, we might assume that users have different

1. Some parts of this paper appeared in: T. Lu and C. Boutilier, Learning Mallows Models with Pairwise Preferences, *Proceedings of the Twenty-Eighth International Conference on Machine Learning (ICML 2011)*, pp.145-152, Bellevue, WA (2011).

types with inherently distinct preferences, and learn a population model that explains this diversity. Learning preference types (e.g., by segmenting or clustering the population) is key to effective personalization and preference elicitation in recommender systems, social choice, and numerous other domains. For example, with a learned population preference distribution, choice data obtained from a specific user allows inferences to be drawn about her preferences. In this work, we focus on the latter setting, learning preference distributions when users have genuinely distinct preferences.

Considerable work in machine learning has exploited ranking models developed in the statistics and psychometrics literature, such as the Mallows model (Mallows, 1957), the Plackett-Luce model (Plackett, 1975; Luce, 1959), and others (Marden, 1995), as well as their non-parametric representations (Lebanon and Mao, 2008). However, most research to date provides methods for learning preference distributions using very restricted forms of evidence about individual user preferences, whether passively observed or actively elicited, ranging from complete rankings, to top-t/bottom-t alternatives, to *partitioned preferences* (Lebanon and Mao, 2008). Missing from this list are arbitrary *pairwise comparisons* of the form "alternative a is preferred to alternative b." Such pairwise preferences form the building blocks of almost all reasonable evidence about preferences, and subsumes the most general evidential models proposed in the literature. Furthermore, preferences in this form naturally arise in active elicitation of user preferences and choice contexts (e.g., web search, product comparison, advertisement clicks), where a user selects one alternative over others in some set (Louviere et al., 2000). In general, data about a user's preferences will often take the form of arbitrary choice sets as is common in web search, online advertising, product comparison, etc. But none of the techniques and algorithms developed to date can learn from such choice sets. These preferences can be as simple as a single paired comparison: "I like alternative a better than b," or as complex as a set of comparisons: "I like a better than $b, c, \ldots$, *and* I like z better than $y, x, \ldots$" In this sense, pairwise comparisons should be viewed as the fundamental building block and universal language of ordinal preference ranking.[2]

While learning with pairwise preferences is clearly of great importance, it is widely believed that learning probabilistic models of ordinal preference using paired comparison data is impractically difficult (indeed, we show this formally below). As a consequence, the Mallows model is often shunned in favor of more inference-friendly models (e.g., Plackett-Luce, which accommodates more general, but still restrictive, preferences; see Cheng et al., 2010; Guiver and Snelson, 2009). To date, no methods have been proposed for learning from arbitrary pairwise preferences in any of the commonly used ranking models in machine learning. We tackle this problem directly by developing techniques for learning Mallows models, and mixtures thereof, from pairwise preference data.

Our core contribution is the *generalized repeated insertion model (GRIM)*, a new method for sampling from *arbitrary ranking distributions*—including conditional Mallows—that generalizes the repeated insertion method for unconditional sampling of Mallows models (Doignon et al., 2004). We show that even evaluating the log-likelihood under a Mallows model with respect to arbitrary ordinal data is #P-hard, implying that learning will be at

2. Of course, ordinal preferences do not capture strength of preference; but real-valued or scaled preferences (e.g., movie or book ratings) can be converted to pairwise preferences readily, albeit with some loss of information.

least as difficult. However, we derive another method, which we call AMP, which efficiently, though approximately, samples from any conditional Mallows distribution given arbitrary pairwise evidence. Moreover, we show that AMP is *exact* for important classes of evidence (including partitioned preferences), and that empirically it provides very close approximations given general pairwise evidence. We use this sampler as the core of a Monte Carlo EM algorithm to learn Mallows mixtures, evaluate log-likelihood, and make predictions about missing preferences. We also extend the non-parametric framework of Lebanon and Mao (2008) to handle unrestricted ordinal preference data. Experiments show our algorithms can effectively learn Mallows mixtures, with reasonable running time, on data sets with hundreds of alternatives and thousands of users. Our sampling algorithm can be adapted rather easily to other models as well (e.g., we show how a simple modification allows sampling from Mallows models with a weighted Kendall-tau metric).

The remainder of the paper is organized as follows. In Section 2 we describe the necessary background on ordinal preferences, Mallows models, and the *repeated insertion method* (Doignon et al., 2004) for Mallows distributions, which we extend later in the paper. We also discuss related work on learning probabilistic preference models. We introduce our main technical tool, the *generalized repeated insertion method (GRIM)*, in Section 3. We show how it can be used to sample from Mallows mixtures conditioned on incomplete preferences by first defining an approximate, but *direct* sampler AMP that is exact for important special cases, and analyzing its computational and statistical properties. We then develop Metropolis and Gibbs sampling methods that exploit AMP to soundly sample any Mallows or Mallows mixture posterior. In Section 4 we develop an EM algorithm for learning a Mallows mixture from arbitrary pairwise comparison data that leverages our sampling algorithms, and provide experimental results of this procedure on several real-world data sets in Section 5. Section 6 extends the framework of Lebanon and Mao (2008) for non-parametric estimation to handle evidence in the form of arbitrary ordinal preferences. We conclude in Section 7 with a discussion of future directions.

2. Preliminaries

We begin by describing the ordinal preferences (rankings) used in the work, providing a brief overview of several common probabilistic preference models, with an emphasis on the Mallows ϕ-model (and mixtures) and models of partial preference data. We then outline Doignon et al. (2004) repeated insertion model for sampling preferences from a Mallows distribution (and draw connections to older models for sampling rankings proposed by Condorcet, Kemeny and Young). We also briefly discuss related work on learning probabilistic preference models.

2.1 Ordinal Preferences

We assume a set of m *alternatives* $A = \{a_1, \ldots, a_m\}$ and n *agents* $N = \{1, \ldots, n\}$. Each agent ℓ has *preferences* over the set of alternatives represented by a total ordering or *ranking* $\succ_\ell$ over A. We write $x \succ_\ell y$ to mean ℓ prefers x to y. Rankings can be represented as permutations of A. For any positive integer b, let $[b] = \{1, \ldots, b\}$. We often represent a ranking as a permutation or bijection $\sigma : A \to [m]$, where $\sigma(a)$ is the *rank* or *position* of a in the ranking. Thus, for $i \in [m]$, $\sigma^{-1}(i)$ is the alternative with rank i. We write

$\sigma = \sigma_1\sigma_2\cdots\sigma_m$ for a ranking with i-th ranked alternative $\sigma_i \in A$, and $\succ_\sigma$ for the induced preference relation. For any $X \subseteq A$, let $\sigma|_X$ denote the ranking obtained by restricting σ to alternatives in X. Let $\mathbf{1}[\cdot]$ be the indicator function.

Generally, we do not have access to the complete preferences of agents, but only partial information about their rankings (e.g., based on choice behavior, query responses, etc.). We assume this data has a very general form: for each agent ℓ we have a set of *revealed pairwise preference comparisons* over A, or simply *preferences*:

$$v_\ell = \{x_1^\ell \succ_\ell y_1^\ell, \ldots, x_{k_\ell}^\ell \succ_\ell y_{k_\ell}^\ell\}.$$

Intuitively, these reflect information about ℓ's preferences revealed by some process. For example, this could represent product-ratings data; preference revealed by selection or purchase of certain items (e.g., web links, products) over others, or responses to survey data.

Let $\mathsf{tc}(v_\ell)$ denote the transitive closure of v_ℓ, i.e., the smallest transitive relation containing v_ℓ. We write $\{x, y\} \in v_\ell$ if there is a comparison between x and y in v_ℓ and, similarly, $\{x, y\} \in \mathsf{tc}(v_\ell)$ if x, y are comparable in its transitive closure. Since preferences are strict, $\mathsf{tc}(v_\ell)$ is a strict partial order on A. We assume each v_ℓ is *consistent*, i.e., $\mathsf{tc}(v_\ell)$ contains no cycles.[3] Preferences v_ℓ are *complete* if and only if $\mathsf{tc}(v)$ is a total order on A. Let $\Omega(v)$ denote the set of *linear extensions* of v, i.e., those rankings consistent with v. Let $\Omega = \Omega(\emptyset)$ be the set of all $m!$ complete preferences. A collection $V = (v_1, \ldots, v_n)$ is a *(partial) preference profile*—we assume that the observed data used for inference and learning purposes in our work takes this form. Given ranking $\sigma = \sigma_1\sigma_2\cdots\sigma_m$ and preference v, we define the *dissimilarity* or *disagreement* $d(v, \sigma)$ between the two to be:

$$d(v, \sigma) = \sum_{i<j\leq m} \mathbf{1}[\sigma_j \succ \sigma_i \in \mathsf{tc}(v)]. \tag{1}$$

Dissimilarity between a partial preference and a ranking is the number of pairwise disagreements among the relative ranking of alternatives, i.e., those pairs in v that are misordered relative to σ. If v is a complete ranking, $d(v, \sigma)$ is the classic *Kendall-tau* metric on rankings. Likewise, define $s(v, \sigma)$ to be the number of pairwise comparisons in $\mathsf{tc}(v)$ that are consistent with σ. We have that $d(v, \sigma) + s(v, \sigma)$ is the number of comparisons in $\mathsf{tc}(v)$. If v is complete, then $d(v, \sigma) + s(v, \sigma) = \binom{m}{2}$.

Arbitrary sets v of pairwise comparisons can be used to model a wide range of realistic revealed preferences:[4]

- Complete rankings require $m - 1$ paired comparisons (e.g, $a \succ b \succ c \ldots$), and can be elicited with at most $m(m-1)/2$ paired comparison queries.
- Top-t preferences (Busse et al., 2007) require that users provide a complete ranking of their top t most preferred alternatives. These can be represented using $m-1$ pairs:

3. Many of the concepts for probabilistic modeling, inference and learning developed in this paper can be applied *mutatis mutandis* to models where revealed preferences are noisy; however, we leave this topic to future research.
4. One exception to this is information about preferences that involve "disjunctive" constraints. For instance, a response to the question "What alternative is ranked t^{th}?" cannot be mapped to a set of pairwise preferences unless the positions t are queried in ascending or descending order (hence inducing top-t or bottom-t preferences).

$t-1$ comparisons to order the top t alternatives, and $m-t$ pairs to ensure the t-th alternative is ranked above the remaining $m-t$. Bottom-t preferences are similar.

- Complete rankings of *subsets* $X \subseteq A$ (Guiver and Snelson, 2009; Cheng et al., 2010) are also representable in the obvious fashion (requiring $k-1$ comparisons if $|X| = k$).
- Preferences revealed by the choice of an alternative a from $X \subseteq A$ (Louviere et al., 2000) can also be represented using $k-1$ pairs of the form $a \succ b$ for each $b \in X \setminus \{a\}$ (where $|X| = k$). Sets of such choices are captured in the obvious way.
- Ordinal ratings data: if alternatives are scored on an ordinal scale s (e.g., a scale of 1–5 where 1 is most preferred), we simply include $a \succ b$ whenever $s(a) < s(b)$, assuming that alternatives with the same rating cannot be compared using the level of granularity provided.

Much of the existing work in learning or modelling distributions over ordinal preferences restricts the class of representable preferences. Much work has focused on top-t preferences (Busse et al., 2007; Meila and Chen, 2010; Gormley and Murphy, 2007; Fligner and Verducci, 1986, 1993), and its generalizations (Lebanon and Mao, 2008); other papers have worked with rankings of a subset of alternatives (Guiver and Snelson, 2009; Cheng et al., 2010). The main issue in allowing arbitrary consistent collections of paired preferences, which can represent all of the above special cases, is the difficult inference problem that results. The primary aim of this work is to develop tractable inference algorithms for a much broader and realistic class of preferences. Before closing our discussion of ordinal preferences, we define a recently studied and relatively expressive class of preferences

Definition 1 (Lebanon and Mao 2008) *A partial preference v is a* partitioned preference *if A can be partitioned into subsets $A_1, \ldots, A_q$ s.t.: (a) for all $i < j \leq q$, if $x \in A_i$ and $y \in A_j$ then $x \succ_{\mathsf{tc}(v)} y$; and (b) for each $i \leq q$, alternatives in A_i are incomparable under $\mathsf{tc}(v)$.*

Partitioned preferences are quite general, subsuming some of the special cases above, including top-t or bottom-t preferences, or ratings data. However, they cannot represent many naturally occurring preferences, including those as simple as a single pairwise comparison $a \succ b$. We demonstrate below that our techniques can be applied effectively to such preferences.

2.2 Mallows Models and Sampling Procedures

There are many distributional models of rankings that have been developed in psychometrics, statistics and econometrics to explain choice behavior (Marden, 1995 provides a good overview). Two of the more popular in the machine learning community are the *Mallows model* (Mallows, 1957) and the *Plackett-Luce model* (Plackett, 1975; Luce, 1959). We focus on Mallows in this work, though we believe our methods can be extended to other models.

2.2.1 The Mallows Model

The *Mallows ϕ-model* (which we simply call the Mallows model hereafter) is typical of a wide-range of distance-based ranking models (Mallows, 1957; Marden, 1995). As above, let

d be the Kendall-tau distance. The Mallows model is parameterized by a modal or *reference ranking* σ and a *dispersion parameter* $\phi \in (0, 1]$. For any ranking r, the Mallows model specifies:

$$P(r) = P(r \mid \sigma, \phi) = \frac{1}{Z}\phi^{d(r,\sigma)} , \tag{2}$$

where $Z = \sum_{r' \in \Omega} \phi^{d(r',\sigma)}$ is the normalization constant. It can be shown that

$$Z = 1 \cdot (1 + \phi) \cdot (1 + \phi + \phi^2) \cdots (1 + \cdots + \phi^{m-1}). \tag{3}$$

When $\phi = 1$ we obtain the uniform distribution over Ω (in the social choice literature, this model is known as *impartial culture*). As $\phi \to 0$, the distribution concentrates all mass on σ. The model can also be expressed as $P(r|\sigma, \lambda) = \frac{1}{Z}e^{-\lambda d(r,\sigma)}$, where $\lambda = -\ln \phi \geq 0$. Various extensions and generalizations of this model have been developed (e.g., using other distance measures) (Marden, 1995).

2.2.2 Condorcet's Decision Problem

We describe a simple sampling procedure proposed by Mallows, Condorcet and further analyzed by Young, since this will motivate the RIM sampler discussed in Section 2.2.3. Mallows (1957) explained his model using process in which a judge assesses alternatives by repeatedly making pairwise comparisons. The outcome of such a comparison is stochastic and depends on the reference ranking σ. If x and y are compared and x is preferred to y in σ, then the judge "correctly" assesses $x \succ y$ with probability $1 - p_{xy}$, and erroneously assesses $y \succ x$ with probability $p_{xy} < 1/2$. Each assessment is independent of other comparisons. Mallows' process generated a pairwise comparison for each pair of alternatives as described: after all paired comparisons are made, if the result is consistent (i.e., corresponds to a ranking), it is accepted; otherwise the process is repeated. While the error probability p_{xy} can depend in a fairly general way on their positions in σ, if $p_{xy} = p$ for all x, y then we obtain the Mallows model.

Such a probabilistic view of rankings was studied two centuries earlier by Nicolas de Condorcet in the context of collective political decision making (Condorcet, 1785). He modeled his view of the role of government, that of making the "right decisions," by considering the selection from a set of choices (e.g., policies), one that maximizes benefit to society. Members of society, or *voters*, express their opinion in the form of a ranking over choices. He assumed that some (latent) *objective ranking* orders choices from most to least beneficial to society and that each voter is able to provide an independent, random assessment of relative rank of any pair of choices: if $a \succ b$, in the objective ranking a voter will assess that to be the case with probability $1 - p$, with a error probability less than $1/2$. Instead of studying the probabilistic model *per se*, Condorcet addressed the *decision problem*: how to find the ranking most likely to be correct. For the case of three alternatives, he proved that the ranking which minimized the total number pairwise preference disagreements (i.e., Kendall-tau distance) with respect to the stated voter rankings was the most likely to be correct.

In modern parlance, Condorcet showed how to compute the maximum likelihood estimator (MLE) of the objective or reference ranking. Kemeny (1959) proposed the *Kemeny ranking* as a general method for aggregating noisy voter rankings, extending Condorcet's

approach to accommodate any number of alternatives. The Kemeny ranking is that which minimizes total number of pairwise preference disagreements with the set of voter rankings, which Kemeny justified axiomatically (showing it to be the only aggregate ranking that satisfies certain intuitive axioms). A statistical rationale for Kemeny's approach was provided by Young (1995), who extended Condorcet's analysis, showing that, for *any* number of alternatives, under Condorcet's noise model, the MLE of the reference ranking is in fact the Kemeny ranking. These two independent threads (Condorcet-Kemeny-Young and Mallows) can both be viewed as statistical estimation of a noisy ranking model. We tie these threads together, showing that Condorcet's noise model for any number of alternatives corresponds to the Mallows models (which implies, by Young's result, that the Kemeny ranking is the MLE for the Mallows model). The Condorcet-Mallows noisy ranking process can be formalized as follows:

Pairwise Comparison Sampling of Mallows

1. Let σ be the reference ranking and $0 \leq p \leq 1/2$.
2. Initialize $v \leftarrow \emptyset$.
3. For each pair of items x, y in A, such that $x \succ_\sigma y$,
   (a) with probability $1 - p$ add $x \succ y$ to v,
   (b) otherwise add $y \succ x$ to v.
4. If v is intransitive, go back to step 1 and start over.
5. v is transitive and corresponds to a ranking.

This pairwise comparison process generates rankings in accordance with the Mallows model (Equation 2), a fact shown by Mallows (1957), but which we derive here (since it will be instructive below). Consider the following distribution over rankings v:

$$P'(v \mid \sigma, p) = \frac{1}{Z'} \prod_{\{x,y\} \subseteq A} \begin{cases} p & \text{if } v \text{ and } \sigma \text{ disagree on } x, y \\ 1 - p & \text{otherwise,} \end{cases} \tag{4}$$

where Z' is the normalization constant (i.e., the sum of the probabilities generated by the above procedure, over all transitive, complete preferences). The form of this distribution corresponds exactly to the rankings generated. This can be seen by noticing that the generating procedure independently decides for each pair of alternatives x, y, with a flip of p-biased coin, whether to order them according to σ. Since intransitive preferences v are discarded by the procedure, the generating procedure corresponds to P'. We can simplify the expression for P' to:

$$\begin{aligned} P'(v \mid \sigma, p) &= \frac{1}{Z'} p^{d(v,\sigma)} (1-p)^{s(v,\sigma)} \\ &= \frac{1}{Z'} p^{d(v,\sigma)} (1-p)^{\binom{m}{2} - d(v,\sigma)} \\ &= \frac{1}{Z'} (1-p)^{\binom{m}{2}} \left(\frac{p}{1-p} \right)^{d(v,\sigma)} . \end{aligned} \tag{5}$$

By setting $\phi = \frac{p}{1-p}$, recalling the definition of Z (Equation 3), and noticing that

$$Z' = (1-p)^{\binom{m}{2}} Z \quad (6)$$

$$= (1-p)^{\binom{m}{2}} \left(1 + \frac{p}{1-p}\right)\left(1 + \frac{p}{1-p} + \left(\frac{p}{1-p}\right)^2\right) \cdots \left(1 + \cdots + \left(\frac{p}{1-p}\right)^{m-1}\right), \quad (7)$$

we obtain Equation 2. The log-likelihood, given observed complete rankings $r_1, \ldots, r_n$, is

$$\sum_{\ell=1}^{n} \left[d(r_\ell, \sigma) \ln \phi - \ln Z \right].$$

Hence, the MLE ranking is the minimizer of $\sum_{\ell=1}^{n} d(r_\ell, \sigma)$, namely, the Kemeny ranking.

2.2.3 The Repeated Insertion Model

The Condorcet/Mallows sampling procedure for drawing rankings from the Mallows distribution can be very inefficient, since it relies on rejection of partially constructed rankings as soon as a single circular, or non-transitive, triad $(a \succ b \succ c \succ a)$ is drawn. While the original motivation for these models was not computational, efficient sampling is important for a variety of inference and learning tasks. Doignon et al. (2004) introduce the *repeated insertion model (RIM)* for the analysis of probabilistic models of approval voting, but which also provides a much more effective means of sampling from a Mallows distribution.

RIM is a generative process that gives rise to a family of distributions over rankings and provides a practical way to sample rankings from a Mallows model. The model assumes some reference ranking $\sigma = \sigma_1 \sigma_2 \cdots \sigma_m$, and *insertion probabilities* p_{ij} for each $i \leq m, j \leq i$. RIM generates a new *output ranking* using the following process, proceeding in m steps. At step 1, σ_1 is added to the output ranking. At step 2, σ_2 is inserted above σ_1 with probability $p_{2,1}$ and inserted below with probability $p_{2,2} = 1 - p_{2,1}$. More generally, at the i-th step, the output ranking will be an ordering of $\sigma_1, \ldots, \sigma_{i-1}$ and σ_i will be inserted at rank $j \leq i$ with probability p_{ij}. Critically, the insertion probabilities are *independent of the ordering of the previously inserted alternatives.*

It is easy to see that one can generate any ranking with the appropriate insertion positions. As we describe below, Doignon et al. (2004) show that one can sample from a Mallows distribution using RIM with appropriate insertion probabilities. We now introduce several concepts that can be used to more easily formalize and analyze RIM, and our subsequent extensions of it.

Definition 2 *Let $\sigma = \sigma_1 \cdots \sigma_m$ be a reference ranking. Let an* insertion vector *be any positive integer vector $\mathbf{j} = (j_1, \ldots, j_m)$ satisfying $j_i \leq i, \forall i \leq m$; and let I be the set of such insertion vectors. A* repeated insertion function $\Phi_\sigma : I \to \Omega$ *maps an insertion vector $\mathbf{j}$ into a ranking $\Phi_\sigma(\mathbf{j})$ by placing each σ_i, in turn, into rank j_i, for all $i \leq m$.*

This definition is best illustrated with an example. Consider the insertion vector $(1, 1, 2, 3)$ and reference ranking $\sigma = abcd$. In this case, $\Phi_\sigma(1, 1, 2, 3) = bcda$ because: we first insert a into rank 1; we then insert b into rank 1, shifting a down to obtain partial ranking ba; we then insert c into rank 2, leaving b in place, but moving a down, obtaining bca; finally,

we insert d at rank 3, giving *bcda*. By the same process we obtain $\Phi_\sigma(1,2,3,4) = abcd$, and $\Phi_\sigma(1,1,1,1) = dcba$. Given reference ranking σ, there is a one-to-one correspondence between rankings and insertion vectors.

Observation 3 *For any reference ranking σ, the repeated insertion function Φ_σ is a bijection between I and Ω.*

Sampling using RIM can characterized as follows:

Definition 4 *The* repeated insertion model *is a probabilistic model over rankings defined by a reference ranking σ, the repeated insertion function $\Phi_\sigma(j_1, \ldots, j_m)$ and a sequence of* insertion probabilities *p_{ij_i} for $i \leq m$, $j_i \leq i$, such that $\sum_{j=1}^{i} p_{ij} = 1$, $\forall i \leq m$. A ranking is generated at random by first drawing an insertion vector $\mathbf{j} = (j_1, \ldots, j_m) \in I$, where each j_i is drawn independently with probability p_{ij_i}, and then applying the insertion function $\Phi_\sigma(\mathbf{j})$.*

Let $\Phi_\sigma^{-1}(r) = (j'_1, \ldots, j'_m)$. Then the probability of generating a particular ranking r under RIM is $\prod_{i \leq m} p_{ij'_i}$. It is easy to see that the Kendall-tau distance between the reference ranking and the ranking induced by an insertion vector is the sum of the number "insertion misorderings" over all alternatives:

Proposition 5 *For any insertion vector $\mathbf{j} = (j_1, \ldots, j_m) \in I$, we have that*

$$\sum_{i=1}^{m} i - j_i = d(\Phi_\sigma(\mathbf{j}), \sigma). \tag{8}$$

Proof Observe that whenever σ_i is inserted at the j_i-th position, it creates $i - j_i$ pairwise misorderings with respect to alternatives $\sigma_1, \ldots, \sigma_{i-1}$. All pairwise misorderings can be accounted for this way. Summing over all $i \leq m$ gives the Kendall-tau distance. ■

Doignon et al. (2004) show that by setting the insertion probabilities p_{ij} appropriately, the resulting generative process corresponds to the Mallows model. We reprove their Theorem here, since the proof will be instructive later.

Theorem 6 (Doignon et al. 2004) *By setting insertion probabilities $p_{ij} = \phi^{i-j}/(1 + \phi + \cdots + \phi^{i-1})$ for $j \leq i \leq m$, the distribution induced by RIM with insertion function Φ_σ is identical to that of the Mallows model with reference ranking σ and dispersion parameter ϕ.*

Proof We reprove the Doignon et al. (2004) theorem. Let r be any ranking and σ the reference ranking of the Mallows model. Let $\Phi_\sigma^{-1}(r) = (j_1, \ldots, j_m)$ be the insertion ranks. If we multiply the factors ϕ^{i-j_i} across $i \leq m$ this gives $\phi^{\sum_{i=1}^{m} i - j_i} = \phi^{d(r,\sigma)}$ by Proposition 5. This term $\phi^{d(r,\sigma)}$ is exactly the proportional probability of r in Mallows. The denominator of $\prod_{i=1}^{m} p_{ij_i}$ is $(1+\phi)(1+\phi+\phi^2)\cdots(1+\phi+\cdots+\phi^{m-1})$ regardless of r—this is exactly the normalizing constant in Mallows model. Interestingly, this gives an alternate proof of the normalization constant in the Mallows model. ■

Thus RIM offers a simple, useful way to sample rankings from the Mallows model while maintaining consistent partial rankings at each stage. In contrast to the rejection sampling approach of Condorcet/Mallows, RIM can be much more effective since it does not require the rejection of intransitive triads (which may occur with high probability if ϕ is large). We summarize the RIM approach from Mallows model:

RIM Sampling of Mallows

1. Let $\sigma = \sigma_1 \cdots \sigma_m$ be the reference ranking and ϕ the dispersion.
2. Start with an empty ranking r.
3. For $i = 1..m$:
   - Insert σ_i into r at rank position $j \leq i$ with probability $\phi^{i-j}/(1 + \phi + \cdots + \phi^{i-1})$.

RIM has worst-case quadratic running time (required number of draws from a Bernoulli distribution) when sampling from a Mallows model (this can be explained in much the same way as the complexity of insertion sort). However, the average-case time complexity can be much smaller, since insertions at each stage of the algorithm are likely to occur near the bottom of the partial ranking.

Proposition 7 *The expected time complexity of repeated insertion sampling for a Mallows model* (σ, ϕ) *is*

$$O\left(\min\left\{\frac{m(1+\phi^{m+1})}{1-\phi} - \frac{\phi(1-\phi^m)}{(1-\phi)^2}, m^2\right\}\right).$$

Proof Suppose we have $O(1)$ access to biased coin flips. The implementation will be as follows. Place σ_1 in the first rank. Then loop for $i = 2$ to m. Let $p_{ij} = \phi^{i-j}/\sum_{j'=0}^{i-1}\phi^{j'}$. Sample a rank position j to insert σ_i: start with $j = i$, flip a coin with probability p_{ij}, if success insert at rank j. Otherwise decrease j by 1, flip a coin with probability $p_{ij}/(1 - \sum_{j'>j} p_{ij'})$, if success, insert at rank j, otherwise decrease j by 1 and repeat this process until $j = 1$. By the chain rule, the probability of insertion at rank j is exactly what Mallows model requires. For each σ_i, when the sampled insertion rank position is j, it would require at most $i - j + 1$ coin flips. The expected running time, i.e., total number of coin flips, if $\phi < 1$, is proportional to

$$\begin{aligned}\sum_{i=1}^{m} \frac{\sum_{j=0}^{i-1}(j+1)\phi^j}{\sum_{j=0}^{i-1}\phi^j} &= \sum_{i=1}^{m} \frac{1}{1-\phi} - i\phi^i \\ &\leq \frac{m(1+\phi^{m+1})}{1-\phi} - \frac{\phi(1-\phi^m)}{(1-\phi)^2}.\end{aligned}$$

This means one can effectively sample in linear time if ϕ is not too close to 1. If $\phi = 1$, the expected running time is $O(m^2)$. ■

Sampling with Weighted Kendall-tau. To illustrate the flexibility of RIM, we show it can be used to sample from a Mallows model using a weighted Kendall-tau distance. For two rankings r and σ and insertion vector $\mathbf{j} = (j_1, \ldots, j_m)$ such that $\Phi_\sigma(\mathbf{j}) = r$, one can define a *weighted Kendall-tau distance* (Shieh, 1998) with respect to positive weights $\mathbf{w} = (w_1, \ldots, w_m)$ as follows

$$d_{\mathbf{w}}(r, \sigma) = \sum_{i=1}^{m} w_i(i - j_i).$$

Recall that by Proposition 5, if $\mathbf{w} = \mathbf{1}$, then $d_{\mathbf{w}}$ is the standard Kendall-tau distance. Otherwise, this weighted Kendall-tau is sensitive to the pairwise misorderings of top-ranked alternatives in σ.

One can sample from a Mallows model defined by $P_{\mathbf{w}}(r) \propto e^{-d_{\mathbf{w}}(r,\sigma)}$ using RIM as follows. Let $\phi_i = e^{-w_i}$ for $i \leq m$. If we define the insertion probability of σ_i at position $j_i \leq i$ to be $\phi_i^{i-j_i}/(1+\phi_i+\cdots+\phi_i^{i-1})$, then the probability of generating r is proportional to $e^{\sum_{i=1}^{m}(i-j_i)\ln\phi_i} = e^{-d_{\mathbf{w}}(r,\sigma)}$.

2.3 A Mallows Mixture Model for Incomplete Preferences

While distributions such as Mallows or its mixture formulation (Murphy and Martin, 2003) give rise to complete rankings, there is relatively little work on generative models for partial rankings, and in particular, models that generate arbitrary (consistent) sets of pairwise comparisons. We introduce such a generative model in this section upon which to base our subsequent learning and inference procedures given such pairwise evidence.

A *Mallows mixture distribution* with K components is parameterized by mixing proportions $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$, reference rankings $\boldsymbol{\sigma} = (\sigma^{(1)}, \ldots, \sigma^{(K)})$, and dispersion parameters $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_K)$. Rankings are generated randomly by selecting one of the K components according to the multinomial distribution with parameters $\boldsymbol{\pi}$. We sometimes represent this with a unit *component indicator* vector $\mathbf{z} = (z_1, \ldots, z_K) \in \{0,1\}^K$ in which the only entry of $\mathbf{z}$ set to 1 is that of the selected component. If $z_k = 1$, then ranking r is drawn from the Mallows distribution with parameters $\sigma^{(k)}, \phi_k$.

In our model for partial preferences, we assume that each agent ℓ possesses a latent ranking r, where r is drawn from a mixture of Mallows distributions. We obtain the set of pairwise comparisons for ℓ by assuming a single additional parameter α which generates random pairs of alternatives. Intuitively, this reflects a process in which, given ℓ's latent ranking r, each pair of alternatives is selected independently with probability α, and ℓ's preference for that pair, as dictated by r, is revealed. That is,

$$P(v \mid r, \alpha) = \begin{cases} \alpha^{|v|}(1-\alpha)^{\binom{m}{2}-|v|} & \text{if } r \in \Omega(v), \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

This model reflects the relatively straightforward *missing at random assumption* (Ghahramani and Jordan, 1995), in which there is no correlation among those pairwise preferences that are missing/observed, nor any between observed pairs and the underlying ranking (e.g., the positions of the observed pairs). The missing at random assumption is not always realistic (Marlin and Zemel, 2007). We also note that this model assumes a single global parameter α that indicates the expected degree of completeness of each agent ℓ's partial preferences. Allowing agent-specific completeness parameters α_ℓ and moving beyond "missing at random" are important directions. However, this model serves as a reasonable starting point for investigation.

Figure 1 illustrates a graphical model for the entire process. The resulting joint distribution is

$$P(v, r, \mathbf{z} \mid \boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi}, \alpha) = P(v \mid r, \alpha) P(r \mid \mathbf{z}, \boldsymbol{\sigma}, \boldsymbol{\phi}) P(\mathbf{z} \mid \boldsymbol{\pi}). \tag{10}$$

In our basic inference and learning problem, we take the observed data to be a preference profile $V = (v_1, \ldots, v_n)$ of n agents, and we let $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n)$ denote the corresponding latent component memberships (i.e., $\mathbf{z}_i$ indicates the mixture component where v_i is generated from).

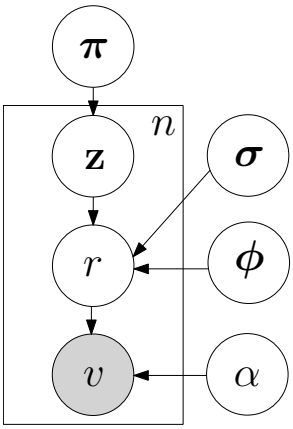


Figure 1: The generative model of incomplete preferences. Observed data v, a set of pairwise comparisons, is shaded.

2.4 Related Work

There is a large literature on ranking in the machine learning, statistics, economics, and theory of computation communities. It includes a variety of approaches, evaluation criteria, heuristics and applications, driven by several distinct motivations. In this section we briefly review two somewhat distinct lines of research.

The first body of work is that on *rank aggregation*. Roughly speaking, the aim is to find the best *objective ranking* given complete or partial observations generated by some noisy process involving the (latent) objective ranking. For example, such a ranking may be a ranking of web pages expressing a typical user's (relative) degree of satisfaction with the pages. Observed information may consist of feedback, in the form of expert ratings or user preferences expressed implicitly via web page clicks on a search results page. In other applications, observed data may include partial rankings (e.g., in political elections), or pairwise comparisons (e.g., in sports leagues). Given such feedback, the ranking system will aggregate and optimize some objective function that attempts to capture user or population satisfaction such as NDCG—common in the IR field—(Burges et al., 2005; Volkovs and Zemel, 2009), misordered pairs (Cohen et al., 1999; Freund et al., 2003; Joachims, 2002; R. Herbrich and Obermayer, 2000), binary relevance (Agarwal and Roth, 2005; Rudin, 2009), and objectives from social choice theory (e.g., Kemeny, Borda rankings). For example, in machine learning, the area of *learning to rank (LETOR)* has been a topic of much research since the late 1990s, starting with the work of Cohen et al. (1999). Research into ranking systems often seeks strong *generalization* capabilities, in the sense that it can produce an objective ranking given a previously unencountered ranking problem using new attributes (e.g., rank web pages given a new search query). Much of this research has indeed been focused on web ranking applications (e.g., the Yahoo! Learning to Rank Challenge; see Burges, 2010). More recently, Busa-Fekete et al. (2014) have developed active learning algorithms for inferring certain distributional properties of the Mallows model.

There are also communities in statistics and computational social choice that are concerned with estimating the maximum likelihood ranking under some distributional assumptions. Often such models—for example, the Mallows and Plackett-Luce models discussed above—assume a central, modal or reference objective ranking at which the distribution is peaked. A fundamental problem is estimation of this objective ranking from a collection of ordinal preference data. For example, the Kemeny ranking can be interpreted as a maximum likelihood estimate of the modal ranking in a Mallows model (Young, 1995). Other such interpretations of common rank aggregation rules also exist (Conitzer and Sandholm, 2005; Conitzer et al., 2009).

The above perspective, that of computing an *objective* ranking, applies to many situations (e.g., one would expect the ranking of web pages for a search query in "norovirus symptoms" to be objectively stable, since users will largely agree the informativeness of retrieved web pages). However, in many settings this is entirely inappropriate. When a group of individuals plans an activity together, such as going to a restaurant for dinner, the ranking of restaurants should clearly depend on the personal tastes and preferences of the individuals involved. In such cases, a distribution over a population's *subjective preferences* better reflects reality. A second, growing, body of work aims to assess (individual or aggregate/group) rankings of options, or decisions, by explicitly using, modelling or reasoning about the diversity of user preferences. This is a more general problem than that of objective rank aggregation. For example, the Netflix collaborative filtering competition has initiated much research on predicting a user's movie ratings given the ratings for other movies, including their own and those of other users. Other relevant research on such ranking work includes *label ranking* (Hüllermeier et al., 2008), which seeks to aggregate sparse preference data of "similar users" into personalized preferences.

In recent years there has been growing interest in applying probabilistic models of preferences from statistics, psychometrics, and econometrics to model a population's preferences. This is the context in which our work is situated. We focus on learning such preference distributions, including multimodal distributions over preferences where each mode (cluster) corresponds to a "sub-type" within the population. Much recent research has focused on using the single-peaked Mallows model as a basis for multimodal mixture distributions. One of the first papers to propose an algorithm for learning Mallows mixtures is that of Murphy and Martin (2003). Their method assumes that training data takes the form of *complete* preference rankings (individual preferences), and has a running time that is factorial in the number of alternatives. Busse et al. (2007) develop a tractable EM algorithm for Mallows mixtures where preferences are restricted to be of the top-t type. A recent extension by Meila and Chen (2010) of Mallows mixtures allows for a Bayesian treatment in choosing the number of components using Dirichlet process mixtures, and offers experiments on considerably larger data sets. Recent work has also studied fitting temporal mixture models (a variation on the Bradley-Terry model) using EM (Francis et al., 2014).

Aside from mixture models, Lebanon and Mao (2008) propose a non-parametric kernel density estimator for rankings, which places a "smooth Mallows bump" on each training preference. They derive an efficiently computable, closed-form formula for the evaluation of the estimator. However, they restrict their training data to partitioned preferences (see above), a more general concept than top-t rankings, but significantly less expressive than arbitrary pairwise comparisons. In contrast to our work, they do not address how to learn

the kernel bandwidth parameter (see Section 6 for further discussion). There has been recent work on sampling algorithms for rankings that shares some similarities with the GRIM algorithm we develop here. This includes a sampling algorithm based on a generalization of the Plackett-Luce model (Volkovs and Zemel, 2012), inspired by bipartite matching problems that occur in certain application domains. Biernacki and Jacques (2013) propose a noisy insertion-sort model of rankings and develop EM algorithms for estimating its parameters. This is related to RIM but with some minor differences. However, none of this work addresses the question of sampling from a posterior distribution given partial preferences as evidence.

Apart from the the Mallows model, the Plackett-Luce model has also been popular as a representation of preferences. Recent work on learning and inference with this model includes: an approach to Bayesian inference of the modal ranking (Guiver and Snelson, 2009), but where training preferences are limited to ranking of all of alternatives in some subset of alternatives; and a method for learning a mixture model given top-k preferences (Gormley and Murphy, 2008), with application to political voting data.

Huang and Guestrin (2009) develop the *riffle independence model*, which partitions a set of alternatives into two sets: a ranking of each set is generated stochastically (and independently); then a stochastic process is used to interleave or "riffle" the two resulting rankings to produce a combined ranking. The model can applied hierarchically, with the same process used to generate the required subrankings. Huang et al. (2012) show that inference in this model is tractable for certain classes of observations. Of particular note is that fact that conditioning on partitioned preferences (which they term "partial ranking observations") can be accomplished efficiently . Interestingly, Mallows models can be represented using the riffle independence model.

3. Generalized Repeated Insertion Model

Our ultimate goal is to support effective learning and inference with Mallows models (and by extension, Mallows mixtures) given observed data or evidence in the form of partial preference profiles consisting of arbitrary pairwise comparisons. Sampling is, of course, an important aspect of this. The rejection sampling models discussed above can obviously be extended to accommodate pairwise observations, but are likely to be extremely inefficient. By contrast, while RIM provides a powerful tool for sampling from Mallows models (and mixtures), it samples unconditionally, without allowing for (direct) conditioning on evidence. In this section, we describe and analyze a generalized version of the RIM technique that permits conditioning at each insertion step. In fact, our *generalized repeated insertion model (GRIM)* can be used to sample from *arbitrary* rank distributions. We begin in Section 3.1 by describing GRIM in this general, abstract fashion. The primary focus of our theoretical and computational analysis in Section 3.2, however, will be on its use for Mallows distributions.

3.1 Sampling from Arbitrary Ranking Distributions

We first present the *generalized repeated insertion model (GRIM)* abstractly as a means of sampling from *any* distribution over rankings. GRIM is based on a relatively simple insight, namely, that the chain rule allows us to represent any distribution over rankings in a concise

way, as long as we admit *dependencies* in our insertion probabilities. Specifically, we allow the insertion probabilities for any alternative σ_i in the reference ranking to be *conditioned on the ordering* of the previously inserted alternatives $(\sigma_1, \ldots, \sigma_{i-1})$.

Let Q be any distribution over rankings and σ an (arbitrary) reference ranking. Recall that we can (uniquely) represent any ranking $r \in \Omega$ using σ and an insertion vector $\mathbf{j}^r = (j_1^r, \ldots, j_m^r) \in I$, where $r = \Phi_\sigma(\mathbf{j}^r)$. Thus Q can be represented by a distribution Q' over the space I of insertion vectors, i.e., $Q'(\mathbf{j}^r) = Q(r)$. Similarly, for $k < m$, any partial ranking $r[k] = (r_1, \ldots, r_k)$ of the alternatives $\{\sigma_1, \ldots, \sigma_k\}$, can be represented by a partial insertion vector $\mathbf{j}[k] = (j_1^r, \ldots, j_k^r)$. Letting

$$Q(r[k]) = \sum\{Q(r) : r_1 \succ r_2 \succ \cdots \succ r_k\} \quad \text{and} \quad Q'(\mathbf{j}[k]) = \sum\{Q'(\mathbf{j}') : \mathbf{j}'[k] = \mathbf{j}[k]\},$$

we have $Q'(\mathbf{j}[k]) = Q(r[k])$. We define *conditional insertion probabilities*:

$$p_{ij\,|\,\mathbf{j}[i-1]} = Q'(j_i = j\,|\,\mathbf{j}[i-1]). \tag{11}$$

This denotes the probability with which the ith alternative σ_i in the reference ranking is inserted at position $j \le i$, *conditioned on the specific insertions* $(j_1^r, \ldots, j_{i-1}^r)$ of all previous alternatives. By the chain rule, we have

$$Q'(\mathbf{j}) = Q'(j_m|\mathbf{j}[m-1])Q'(j_{m-1}|\mathbf{j}[m-2]) \cdots Q'(\mathbf{j}[1]).$$

Suppose we apply RIM with conditional insertion probabilities $p_{ij|\mathbf{j}[i-1]}$ defined above; that is, we draw random insertion vectors $\mathbf{j}$ by sampling j_1 through j_m, in turn, but with each conditioned on the previously sampled components. The chain rule ensures that the resulting insertion vector is sampled from the distribution Q'. Hence the induced distribution over rankings $r = \Phi_\sigma(\mathbf{j})$ is Q. We call the aforementioned procedure the *generalized repeated insertion model (GRIM)*. Based on the arguments above, we have:

Theorem 8 *Let Q be any ranking distribution and σ a reference ranking. For any $r \in \Omega$, with insertion vector $\mathbf{j}^r$ (i.e., $r = \Phi_\sigma(\mathbf{j}^r)$), GRIM, using the insertion probabilities in Equation 11, generates insertion vector $\mathbf{j}^r$ with probability $Q'(\mathbf{j}^r) = Q(r)$.*

For instance, GRIM can be used to sample from a (conditional) Mallows model given evidence in the form of pairwise comparisons, as shown in the following example.

Example 1 *We illustrate GRIM using a simple example, sampling from a (conditional) Mallows model over $A = \{a, b, c\}$, with dispersion ϕ, given evidence $v = \{a \succ c\}$. The following table describes the steps in the process:*

Insert a, b		*Insert c given ab*		*Insert c given ba*	
r	*Insertion Prob.*	r	*Insertion Prob.*	r	*Insertion Prob.*
a	$P(j_a{=}1){=}1$	cab	$P(j_c{=}1){=}\,0$	cba	$P(j_c{=}1){=}0$
ab	$P(j_b{=}1){=}\frac{1}{1+\phi}$	acb	$P(j_c{=}2){=}\frac{\phi}{1+\phi}$	bca	$P(j_c{=}2){=}0$
ba	$P(j_b{=}2){=}\frac{\phi}{1+\phi}$	abc	$P(j_c{=}3){=}\frac{1}{1+\phi}$	bac	$P(j_c{=}3){=}1$

The resulting ranking distribution Q is given by the product of the conditional insertion probabilities: $Q(abc) = 1/(1+\phi)^2$; $Q(acb) = \phi/(1+\phi)^2$; and $Q(bac) = \phi/(1+\phi)$. As required, $Q(r) = 0$ iff r is inconsistent with evidence v.

3.2 Sampling from Mallows Posteriors

We now develop and analyze several techniques for sampling from (mixtures of) Mallows models given partial preference profiles as evidence. We use the term *Mallows posterior* to refer to the conditional distribution that arises from incorporating evidence—in the form of a set of pairwise comparisons—into a known Mallows model. This is the primary inference task facing a system making predictions about a specific user's preferences given pairwise evidence from that user, assuming a reasonably stable population model. This stands in contrast to the more general problem of learning the parameters of a Mallows model (a problem we address in Section 4).

3.2.1 Intractability of Sampling

One key difficulty with enabling inference conditioned on pairwise comparisons is the intractability of the posterior. In the above model (Equation 10), where agent ℓ's incomplete preference v_ℓ is observed, it is intractable to work with the posterior $P(r, \mathbf{z}|v_\ell, \boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi}, \alpha)$ even when the mixture model has a single component, a fact we prove below. One typical approach is to rely on sampling to estimate the posterior. To this end, we develop a polynomial-time posterior sampling algorithm based on GRIM, but relying on approximation of the relevant conditional insertion probabilities.

While GRIM allows sampling from arbitrary distributions over rankings, as presented above it is largely a theoretical device, since it requires inference to compute the required conditional probabilities. Thus to use GRIM to sample from a Mallows posterior, given arbitrary pairwise comparisons v, we must first derive these required terms. The Mallows posterior is given by

$$P_v(r) = P(r \mid v) = \frac{\phi^{d(r,\sigma)}}{\sum_{r' \in \Omega(v)} \phi^{d(r',\sigma)}} \mathbf{1}[r \in \Omega(v)], \tag{12}$$

which requires summing over an intractable number of rankings to compute the normalization constant.

We could use RIM for rejection sampling: sample unconditional insertion ranks, and reject a ranking at any stage if it is inconsistent with v. However, this is impractical because of the high probability of rejection. One can also modify the pairwise comparison sampling model (see Section 2.2.2) to reject inconsistent pairwise comparisons. However, if $|v|$ is small relative to m, then for values of ϕ that are not too small, the probability of rejection is very high. For instance, if ϕ is close to 1, $m = 120$ and 30 alternatives appear in v, any three alternatives the probability of a cyclic triad for any triple (e.g., $a \succ b, b \succ c, c \succ a$) is $\approx 1/4$. The 90 alternatives unconstrained by v can be divided into 30 groups of 3 alternatives, hence the probability that a cycle occurs among at least one triad is at least $1 - (3/4)^{30} = 0.9998$. This is a lower bound on the probability of rejection, showing rejection sampling to be impractical in many settings.

The main obstacle to using GRIM for sampling is computation of the insertion probabilities of a specific alternatives given the inserted positions all previous alternatives, as given by Equation 11, when Q' (more precisely, the corresponding Q) is the Mallows posterior. This essentially involves computing a high-order marginal over rankings, and turns

out to be #P-hard, even with a uniform distribution over $\Omega(v)$. The following result on the complexity of counting linear extensions of a partial order will be useful below:

Theorem 9 (Brightwell and Winkler 1991) *Given a partial order v, computing the number of linear extensions of v, that is $|\Omega(v)|$, is #P-complete.*

To show that computing a function $f(x)$ is #P-hard for input x, it is sufficient to show that a #P-complete problem can be reduced to it in polynomial time.

Proposition 10 *Given v, a reference ordering σ, a partial ranking $r_1 \cdots r_{i-1}$ over $\sigma_1, \ldots, \sigma_{i-1}$, and $j \leq i$, computing the probability of inserting σ_i at rank j with respect to the uniform Mallows posterior P (i.e., computing $P(r) \propto \mathbf{1}[r \in \Omega(v)]$) is #P-hard.*

Proof We reduce the problem of counting the number of linear extensions of incomplete preferences v, which is a #P-complete problem, to that of computing the desired insertion probabilities, showing the problem to be #P-hard. Given v, notice that any $r = r_1 \ldots r_m \in \Omega(v)$ has a uniform posterior probability of $1/|\Omega(v)|$. Let $\Phi_\sigma^{-1}(r) = (j_1, \ldots, j_m)$. Assume the existence of an algorithm f to compute the required insertion probabilities. We can use it to solve the counting problem as follows: we use f to compute $p_{ij_i} = \Pr(\text{insert } \sigma_i \text{ at rank } j_i \mid r|_{\{\sigma_1,\ldots,\sigma_{i-1}\}})$ with partial order v for each $i \in \{2, \ldots, m\}$ (i.e., $m-1$ applications of f). By Theorem 8, we know the posterior probability of r is $1/|\Omega(v)| = \prod_i p_{ij_i}$; thus we can compute $|\Omega(v)|$ by inverting the product of the insertion probabilities. Note that this reduction can be computed in polynomial time: we can construct any $r \in \Omega(v)$ by using a topological sort algorithm, and we require only $m-1$ calls to the algorithm insertion algorithm f. ■

This result shows that it is hard to sample exactly in general, and suggests that computing the normalization constant in a Mallows posterior is difficult. This would also imply a computational complexity obstacle in the work on non-parametric estimators with a Mallows kernel (Lebanon and Mao, 2008) for an *arbitrary* set of pairwise comparisons. Nevertheless we develop an *approximate sampler* AMP that is computationally very efficient. While its approximation quality can be quite poor in the worst case, we see below that, empirically, it produces excellent posterior approximations. We also derive bounds that delineate circumstances under which it will provide approximations with low error.

3.2.2 AMP: An Approximate Sampler

AMP is based on the same intuitions as those illustrated in Example 1, where instead of computing the correct insertion probabilities, we use the (unconditional) insertion probabilities used by RIM, but subject to constraints imposed by v. First, we compute the transitive closure $\mathsf{tc}(v)$ of v. Then we use a modified repeated insertion procedure where at each step, the alternative being inserted can only be placed in positions that do not contradict $\mathsf{tc}(v)$. We can show that the valid insertion positions for any alternative, given v, form a *contiguous region* of the ranking (see Figure 2 for an illustration).

Proposition 11 *Given partial preference v, let the insertion of $i-1$ alternatives $\sigma_1, \ldots, \sigma_{i-1}$ induce a ranking $r_1 \cdots r_{i-1}$ that is consistent with $\mathsf{tc}(v)$. Let $L_i = \{i' < i | r_{i'} \succ_{\mathsf{tc}(v)} \sigma_i\}$ and*

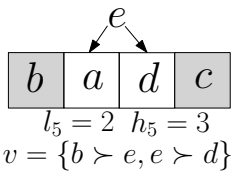


Figure 2: Valid insertion ranks for e are $\{l_5, \ldots, h_5\} = \{2, 3\}$ given previous insertions and constraints v.

Algorithm 1 AMP Approximate Mallows Posterior

Input: v, σ, ϕ

1: $r \leftarrow \sigma_1$
2: **for** $i = 2..m$ **do**
3: Calculate l_i and h_i from Equations 13 and 14.
4: Insert σ_i in r at rank $j \in \{l_i, \ldots, h_i\}$ with probability $\frac{\phi^{i-j}}{\sum_{l_i \leq j' \leq h_i} \phi^{i-j'}}$.
5: **end for**

Output: r

$H_i = \{i' < i | r_{i'} \prec_{\mathsf{tc}(v)} \sigma_i\}$. *Then inserting* σ_i *at rank* j *is consistent with* $\mathsf{tc}(v)$ *if and only if* $l_i \leq j \leq h_i$, *where*

$$l_i = \begin{cases} 1 & \text{if } L_i = \emptyset \\ \max(i' \in L_i) + 1 & \text{otherwise,} \end{cases} \tag{13}$$

$$h_i = \begin{cases} i & \text{if } H_i = \emptyset \\ \min(i' \in H_i) & \text{otherwise.} \end{cases} \tag{14}$$

Proof Inserting σ_i at any rank position less than l_i is impossible since either $l_i = 1$ (we can't insert in rank 0) or σ_i lies above r_{l_i}, which contradicts the requirement imposed by $\mathsf{tc}(v)$ that r_{l_i} must be ranked higher. A similar argument can be made for inserting in rank below h_i since r_{h_i} needs to be below σ_i. Finally, inserting into any rank in $\{l_i, \ldots, h_i\}$ does not violate $\mathsf{tc}(v)$ since the alternative will be inserted below *all* alternatives that must precede it in $\mathsf{tc}(v)$ and *all* alternatives that must succeed it. ■

Proposition 11 immediately suggests an implementation of the GRIM algorithm, AMP, for approximate sampling of the Mallows posterior—AMP is outlined in Algorithm 1. It first initializes ranking r with σ_1 at rank 1. Then for each $i = 2 \ldots m$, it computes l_i, h_i and inserts σ_i at rank $j \in \{l_i, \ldots, h_i\}$ with probability proportional to ϕ^{i-j}. Note that $\mathsf{tc}(v)$, which is required as part of the algorithm, can be computed via a modified depth-first search. AMP induces a sampling distribution $\hat{P}_v$ that does not match the posterior P_v exactly: indeed the KL-divergence between the two can be severe, as the following example shows.

Example 2 *Let* $A = \{a_1, \ldots a_m\}$ *and* $v = a_2 \succ a_3 \succ \cdots \succ a_m$. *Let* P *be the uniform Mallows prior (*$\phi = 1$*) with* $\sigma = a_1 \cdots a_m$. *There are* m *rankings in* $\Omega(v)$, *one ranking*

r_i for each placement of a_1 into rank position $1 \leq i \leq m$. That is, $r_1 = a_1 a_2 \cdots a_m$ and $r_i = a_2 \cdots a_i a_1 a_{i+1} \cdots a_m$ for $i \geq 2$. The true Mallows posterior P_v is uniform over $\Omega(v)$. But AMP *induces an approximation with $\hat{P}_v(r_i) = 2^{-i}$ for $i \leq m-1$ and $\hat{P}_v(r_m) = 2^{-m-1}$. To see this, note that to construct r_i,* AMP *would need to insert alternatives $a_2, \ldots, a_i$ successively, each with probability $1/2$, above a_1. Then a_{i+1} must be inserted below a_1 with probability $1/2$, and finally the remaining alternatives $a_{i+2}, \ldots, a_m$ can only be inserted at the bottom (with probability 1). Hence, the KL-divergence between P_v and $\hat{P}_v$ is*

$$\begin{aligned} KL(P_v || \hat{P}_v) &= \sum_{i=1}^{m} P_v(r_i) \log_2 \left(\frac{P_v(r_i)}{\hat{P}_v(r_i)} \right) \\ &= \left[\sum_{i=1}^{m-1} \frac{1}{m} \log_2 \frac{1/m}{2^{-i}} \right] + \frac{1}{m} \log_2 \frac{1/m}{2^{-m+1}} \\ &= 1 - \frac{1}{m} + \frac{m-1}{2} - \log_2 m \, . \end{aligned}$$

3.2.3 Statistical Properties of AMP

Example 2 shows that AMP may provide poor approximations in the worst case; however we will see below (Section 5) that it performs very well in practice. We can also prove interesting properties, and provide theoretical guarantees of exact sampling in important special cases.

We first observe that AMP always produces a valid ranking; in other words, valid insertion positions always exist given any consistent v.

Proposition 12 *For all $i \geq 2$ and all rankings of alternatives $\sigma_1, \ldots, \sigma_{i-1}$ that is consistent with v, we have that $l_i \leq h_i$, where l_i and h_i are defined in Equation 13 and 14, respectively. That is,* AMP *always has a position at which to insert alternative σ_i.*

Proof Let r be a ranking of $\sigma_1, \ldots, \sigma_{i-1}$ consistent with v. Let x be the lowest ranking alternative in r such that $x \succ_{\mathsf{tc}(v)} \sigma_i$ and y the highest-ranked alternative in r with $y \prec_{\mathsf{tc}(v)} \sigma_i$. By transitivity, $x \succ_{\mathsf{tc}(v)} y$. Now if $h_i < l_i$ (as defined in terms of r) this implies $y \succ_r x$, but this contradicts the assumption that r is consistent with v. ■

Furthermore, the approximate posterior has the same support as the true posterior:

Proposition 13 *The support of the distribution over rankings as defined by* AMP *is equal to $\Omega(v)$ which is equal to the support of the Mallows posterior as given in Equation 12.*

Proof By Proposition 11, the algorithm never violates the constraints in $\mathsf{tc}(v)$, and by Proposition 12, it will always have at least one valid insertion position. Hence the algorithm always outputs a ranking consistent with v. Now, let $r \in \Omega(v)$ and $\Phi_\sigma^{-1}(r) = (j_1, \ldots, j_m)$ be the its corresponding insertion vector. We show that for all $i \leq m$, $j_i \in \{l_i, \ldots, h_i\}$. If this is not true, then there exists a smallest $i' \leq m$ such that $j_{i'} \notin \{l_{i'}, \ldots, h_{i'}\}$ (note $i' \geq 2$ since the first alternative is always inserted at the first position). However, Proposition 11 asserts that this insertion rank would lead to a ranking inconsistent with v—so this is not possible. Since AMP places positive probability on any insertion position in $\{l_i, \ldots, h_i\}$ then r has positive probability under AMP. ■

Proposition 14 *For any $r \in \Omega(v)$, the probability* AMP *will output r is*

$$\hat{P}_v(r) = \frac{\phi^{d(r,\sigma)}}{\prod_{i=1}^{m}(\phi^{i-h_i} + \phi^{i-h_i+1} + \cdots + \phi^{i-l_i})}. \tag{15}$$

Proof Let $\Phi_\sigma^{-1}(r) = (j_1, \ldots, j_m)$ be the insertion ranks. We have already established in Proposition 13 that AMP puts positive probability on these valid insertion ranks. In fact the probability of r under the algorithm (see Algorithm 1) is

$$\begin{aligned}\prod_{i=1}^{m} \frac{\phi^{i-j_i}}{(\phi^{i-l_i} + \phi^{i-l_i-1} + \cdots + \phi^{i-h_i})} &= \frac{\phi^{\sum_{i=1}^{m} i-j_i}}{\prod_{i=1}^{m}(\phi^{i-l_i} + \phi^{i-l_i-1} + \cdots + \phi^{i-h_i})} \\ &= \frac{\phi^{d(r,\sigma)}}{\prod_{i=1}^{m}(\phi^{i-l_i} + \phi^{i-l_i-1} + \cdots + \phi^{i-h_i})},\end{aligned}$$

where the last equality comes from Proposition 5. ■

Using this result we can show that if v lies in the class of partitioned preferences, AMP's induced distribution is exactly the Mallows posterior:

Proposition 15 (Lebanon and Mao 2008) *Let σ be a reference ranking. Let v be a partitioned preference (see Definition 1) with partition $A_1, \ldots, A_q$ of A. Let $\delta = |\{(x,y)|y \succ_\sigma x, x \in A_i, y \in A_j, i,j \in [q], i < j\}|$, which is the number of pairs of alternatives, that span different subsets of the partition, that are misordered with respect to σ. Then*

$$\delta = \sum_{i=1}^{q-1} \sum_{x \in A_i} \sum_{j=i+1}^{q} \sum_{y \in A_j} \mathbf{1}[y \succ_\sigma x], \tag{16}$$

$$\sum_{r \in \Omega(v)} \phi^{d(r,\sigma)} = \phi^\delta \prod_{i=1}^{q} \prod_{j=1}^{|A_i|} (1 + \phi + \phi^2 + \cdots + \phi^{j-1}). \tag{17}$$

Notice that Equation 17 represents the normalization constant in Mallows posterior. The intuition underlying Equation 17 is that, for any $r \in \Omega(v)$, the misorderings contributed by alternatives that span two subsets, as given by δ, are the same (hence the leading factor) whereas within a subset A_i alternatives can be ordered arbitrarily (hence the product of normalization constants for $|A_i|$).

Proposition 16 *Given a partitioned preference v, the distribution induced by* AMP*, $\hat{P}_v$, is equal to the true Mallows posterior P_v.*

Proof Since the numerator in Equation 15 (which denotes the probability that AMP outputs r) is the same as the proportional probability of the Mallows posterior, it is sufficient to show that the denominator in Equation 15 equals the Mallows posterior normalization constant given by Equation 17. Suppose $\sigma = \sigma_1 \cdots \sigma_m$. Let v be a partitioned preference $A_1, \ldots, A_q$. Consider alternatives in A_i such that $\sigma|_{A_i} = \sigma_{t_1}\sigma_{t_2}\cdots\sigma_{t_{|A_i|}}$ (i.e., the ranking of alternatives in A_i according to σ). For any $k \in \{1, \ldots, |A_i|\}$, suppose alternatives $A' = \{\sigma_1, \ldots, \sigma_{t_k-1}\}$ are inserted. The structure of the resulting ranking is as follows: the alternatives $(A_1 \cup A_2 \cup \cdots \cup A_{i-1}) \cap A'$ must lie at the top of the ranking; the alternatives

$A_i \cap A' = \{\sigma_{t_1}, \ldots, \sigma_{t_{k-1}}\}$ are in the middle; and $B_{t_k} = (A_{i+1} \cup \cdots \cup A_q) \cap A'$ are at bottom. When inserting σ_{t_k} at rank j, we have $j \in \{l_{t_k}, \ldots, h_{t_k}\}$, where $h_{t_k} = t_k - |B_{t_k}|$ and $l_{t_k} = h_{t_k} - |A_i \cap A'| = t_k - (k-1) - |B_{t_k}|$. Hence σ_{t_k} is inserted at rank j with probability

$$\frac{\phi^{t_k - j}}{\phi^{t_k - h_{t_k}} + \cdots + \phi^{t_k - l_{t_k}}} = \frac{\phi^{t_k - j}}{\phi^{|B_{t_k}|} + \cdots \phi^{k-1+|B_{t_k}|}}.$$

The denominator can be written $\phi^{|B_{t_k}|}(1 + \cdots + \phi^{k-1})$. Observe that B_{t_k} consists of all alternatives from A' that are above σ_{t_k} in σ, but are below it in v (since all such alternatives belong to $A_{i+1} \cup \cdots \cup A_q$). So $\sum_{k=1}^{|A_i|} |B_{t_k}|$ is the total number of pairs (x, y), where $x \in A_i$ and $y \in A_{i+1} \cup \cdots \cup A_q$, that are misordered with respect to σ. Thus inserting alternatives in A_i contributes a factor of

$$\prod_{k=1}^{|A_i|} \phi^{|B_{t_k}|}(1 + \cdots + \phi^{k-1}) = \phi^{\sum_{x \in A_i} \sum_{j=i+1}^{q} \sum_{y \in A_j} \mathbf{1}[y \succ_\sigma x]} \prod_{k=1}^{|A_i|} (1 + \cdots + \phi^{k-1})$$

to the denominator in Equation 15. Once all alternatives have been inserted, the denominator becomes

$$\phi^{\sum_{i=1}^{q} \sum_{x \in A_i} \sum_{j=i+1}^{q} \sum_{y \in A_j} \mathbf{1}[y \succ_\sigma x]} \prod_{i=1}^{q} \prod_{k=1}^{|A_i|} (1 + \cdots + \phi^{k-1}).$$

This is exactly the Mallows posterior normalization constant in Equation 17. ■

As a consequence, AMP provides exact sampling in the case of partitioned preferences,

In general, this is not the case with arbitrary partial preferences (pairwise comparisons). We now derive bounds on the relative error of AMP's posterior, bounding the ratio between the sample probability of an arbitrary ranking r for AMP and the true posterior probability. The main technical challenge is deriving a bound on the Mallows posterior normalization constant. We can obtain an upper bound by exploiting the pairwise comparison interpretation of Mallows model (see Section 2.2.2).

Theorem 17 (Upper Bound on Normalization Constant) *Let σ be a reference ranking, $\phi \in (0, 1]$ and v a preference. The Mallows posterior normalization constant is upper bounded by*

$$\sum_{r \in \Omega(v)} \phi^{d(r,\sigma)} \leq \phi^{d(v,\sigma)}(1 + \phi)^{\binom{m}{2} - d(v,\sigma) - s(v,\sigma)}. \tag{18}$$

Proof The LHS of Equation 18 can be written in terms Equation 4, by setting $\phi = p/(1-p)$ (see Section 2.2.2 for derivations of the pairwise comparison interpretation of Mallows) as follows:

$$\begin{aligned} \sum_{r \in \Omega(v)} \phi^{d(r,\sigma)} &= Z \cdot \sum_{r \in \Omega(v)} P(r|\sigma, p) \qquad (19) \\ &= Z \cdot \frac{1}{Z'} \sum_{r \in \Omega(v)} \prod_{\{x,y\} \subseteq A} \begin{cases} p & \text{if } r \text{ and } \sigma \text{ disagree on } x, y \\ 1 - p & \text{otherwise,} \end{cases} \end{aligned}$$

where $p = \phi/(1+\phi)$, Z' is given by Equation 6 and Z is given by Equation 3, thus the constant in front simplifies to $1/(1-p)^{\binom{m}{2}}$. Since r must be consistent with v, if x and y are comparable under v, then r must be agree with v on (x, y), i.e., if $x \succ_{\mathsf{tc}(v)} y$ then $x \succ_r y$. So

$$P(r|\sigma, p) = \frac{1}{Z'} p^{d(v,\sigma)} (1-p)^{s(v,\sigma)} \prod_{\{x,y\} \notin \mathsf{tc}(v)} \begin{cases} p & \text{if } r \text{ and } \sigma \text{ disagree on } x, y \\ 1-p & \text{otherwise.} \end{cases}$$

Hence, since $\Omega(v)$ is contained in the set of all *intransitive* relations on A that is consistent with comparisons in $\mathsf{tc}(v)$, we must have (for $k = \binom{m}{2} - d(v,\sigma) - s(v,\sigma)$)

$$\begin{aligned} \sum_{r \in \Omega(v)} P(r|\sigma, p) &\leq \frac{1}{Z'} p^{d(v,\sigma)} (1-p)^{s(v,\sigma)} \sum_{z \in \{0,1\}^k} \prod_{i=1}^{k} p^{z_i} (1-p)^{1-z_i}, \\ &= \frac{1}{Z'} p^{d(v,\sigma)} (1-p)^{s(v,\sigma)}. \\ Z \cdot \sum_{r \in \Omega(v)} P(r|\sigma, p) &\leq \frac{1}{(1-p)^{\binom{m}{2}}} p^{d(v,\sigma)} (1-p)^{s(v,\sigma)}. \end{aligned} \tag{20}$$

Combining Equation 20 with Equation 19, and noting that $p = \phi/(1+\phi)$, we obtain Equation 18. ■

Equation 18 tells us if $d(v, \sigma)$ increases (i.e., v increasingly disagrees with σ), then the first factor dominates and upper bound gets smaller—this reflects our natural intuitions since the set $\Omega(v)$ gets "further away" from reference ranking σ and hence its probability mass is small. We also see that if $|\mathsf{tc}(v)|$ is small, then $d(v, \sigma) + s(v, \sigma)$ is small and the upper bound increases since the second factor dominates. This too makes sense because $\Omega(v)$ is large and has greater probability mass. If $s(v, \sigma)$ is large, more constraints are placed on v, hence $\Pr(\Omega(v))$ is smaller, and likewise the upper bound decreases. The following example illustrates that this bound may be quite loose in some cases, but tight in others.

Example 3 *Consider again the partial ranking evidence from Example 2, where $v = a_2 \succ \cdots \succ a_m$, the alternatives are $\{a_1, \ldots, a_m\}$, and our reference ranking is $\sigma = a_1 a_2 \cdots a_m$. Recall that there are m rankings in $\Omega(v)$, one ranking r_i for each placement of a_1 into rank position i. Now the term on the LHS of Equation 18, i.e., the true value of the normalization constant, is*

$$\sum_{i=1}^{m} \phi^{d(r_i,\sigma)} = 1 + \phi + \phi^2 + \cdots + \phi^m.$$

Note that $d(v, \sigma) = 0$ and $s(v, \sigma) = \binom{m-1}{2}$ since all pairwise comparisons in $\mathsf{tc}(v)$ agree with σ. Thus, the term on the RHS of Equation 18, i.e., the upper bound is

$$\phi^0 (1+\phi)^{\binom{m}{2} - 0 - \binom{m-1}{2}} = (1+\phi)^{m-1}.$$

This upper bound on the normalization constant gets tight as $\phi \to 0$, but becomes exponentially loose in m as $\phi \to 1$.

Before we derive a lower bound, we introduce some notions from order theory.

Definition 18 *Let v be a partial preference. An* anti-chain *of v is a subset X of A such that for every $x, y \in X$ they are incomparable under $\mathsf{tc}(v)$. A* maximum anti-chain *is an anti-chain whose size is at least the size of any anti-chain. The* width *of v, $\mathsf{w}(v)$ is the size of a maximum anti-chain of v.*

Theorem 19 (Lower Bound on Normalization Constant) *Let σ be a reference ranking, and $\phi \in (0, 1]$. Let X be a maximum anti-chain of v, $Y = \{a \in A \backslash X \mid \exists x \in X, a \succ_{\mathsf{tc}(v)} x\}$ and $Z = A \backslash (X \cup Y)$. Let $\delta = |\{(x, y) | x \in X, y \in Y, x \succ_\sigma y\}| + |\{(y, z) | y \in Y, z \in Z, z \succ_\sigma y\}| + |\{(x, z) | x \in X, z \in Z, z \succ_\sigma x\}|$. Denote by $\mathsf{tc}(v)|_Y$ and $\mathsf{tc}(v)|_Z$ the transitive closure of v restricted to the subsets Y and Z, respectively. Also let $\Omega(\mathsf{tc}(v)|_Y)$ denote those rankings over Y that are consistent with $\mathsf{tc}(v)|_Y$, and similarly for $\Omega(\mathsf{tc}(v)|_Z)$. We have*

$$\sum_{r \in \Omega(v)} \phi^{d(r,\sigma)} \geq \phi^\delta \left[\sum_{r \in \Omega(\mathsf{tc}(v)|_Y)} \phi^{d(r,\sigma|_Y)} \right] \left[\sum_{r \in \Omega(\mathsf{tc}(v)|_Z)} \phi^{d(r,\sigma|_Z)} \right] \prod_{i=1}^{\mathsf{w}(v)} \sum_{j=0}^{i-1} \phi^j. \tag{21}$$

Proof We first show that $Z' = \{a \in A \backslash X \mid \exists x \in X, x \succ_{\mathsf{tc}(v)} a\} = Z$. If $a \in A \backslash X$ does not belong to Y then it must be comparable to at least one element in $x \in X$ otherwise we can add it to Y and obtain a larger anti-chain. Hence, since a is not in Y, then $x \succ_{\mathsf{tc}(v)} a$. Also, note that if $a \in Y$ then $a \notin Z'$. This is because if a belonged to both Y and Z, then there exists $x_1, x_2 \in X$ such that $x_1 \succ_{\mathsf{tc}(v)} a$ and $a \succ_{\mathsf{tc}(v)} x_2$ this would mean $x_1 \succ_{\mathsf{tc}(v)} x_2$ which contradicts the anti-chain property of X. For a particular alternative in X, alternatives in Y are either incomparable to it or must be preferred to it, similarly alternatives in Z are either incomparable or must be dis-preferred to it.

This also implies no alternative in Z can be preferred over alternatives in Y since if this were to happen, i.e., if $z \succ_{\mathsf{tc}(v)} y$ where $z \in Z, y \in Y$, then $\exists x \in X$ such that $y \succ_{\mathsf{tc}(v)} x$, this implies $z \succ_{\mathsf{tc}(v)} x$ which is impossible from the above observation that $Z \cap Y = \emptyset$.

Consider all rankings $\widetilde{\Omega}(v)$ where we place alternatives of Y at the top, X in the middle and Z at the bottom. Within Y and Z we rank alternatives respecting $\mathsf{tc}(v)$ and since X is an anti-chain, rank these alternatives without restrictions. That is

$$\widetilde{\Omega}(v) = \{r | \forall y \in Y, x \in X, z \in Z, y \succ_r x, x \succ_r z, r|_Y \in \Omega(\mathsf{tc}(v)|_Y), r|_Z \in \Omega(\mathsf{tc}(v)|_Z)\}.$$

Now we argue $\widetilde{\Omega}(v) \subseteq \Omega(v)$. Note that we satisfy preference constraints when ranking within Y, X and Z. Also as we showed above, alternatives in Y are never dis-preferred to alternatives in X or Z and alternatives in X are never dis-preferred to alternatives in Z.

For the lower bound, first observe if $r \in \widetilde{\Omega}(v)$ then $d(r, \sigma) = d(r|_Y, \sigma|_Y) + d(r|_X, \sigma|_X) + d(r|_Z, \sigma|_Z) + \delta$ where δ is defined in the theorem as the number of misorderings of alternatives across X, Y, Z, which is independent of r. Hence,

$$\sum_{r \in \Omega(v)} \phi^{d(r,\sigma)} \geq \sum_{r \in \widetilde{\Omega}(v)} \phi^{d(r,\sigma)} = \phi^\delta \left[\sum_{r \in \Omega(\mathsf{tc}(v)|_Y)} \phi^{d(r,\sigma|_Y)} \right] \left[\sum_{r \in \Omega(\mathsf{tc}(v)|_X)} \phi^{d(r,\sigma|_X)} \right] \left[\sum_{r \in \Omega(\mathsf{tc}(v)|_Z)} \phi^{d(r,\sigma|_Z)} \right].$$

Finally, it can be seen that the sum inside the third factor is exactly the normalization constant of an unconstrained Mallows model with $|X| = \mathsf{w}(v)$ alternatives, and hence equal to $\prod_{i=1}^{\mathsf{w}(v)} \sum_{j=0}^{i-1} \phi^j$, the second and fourth factors involve sums over rankings of Y and Z consistent with $\mathsf{tc}(v)$. This proves the lower bound. ■

While the lower bound is not presented in a convenient closed-form, it is useful nonetheless if $\mathsf{w}(v)$ is large: if there are few preference constraints in v (e.g., v involves only a small subset of alternatives) we expect $\Omega(v)$ to be large and hence have higher probability mass. We recover the true Mallows normalization constant if $v = \emptyset$ since $\mathsf{w}(v) = m$. If v is highly constrained—$\Omega(v)$ has smaller probability mass—then $\mathsf{w}(v)$ is small, but so are the factors involving summations in Equation 21. Note that ϕ^δ decreases as the number of comparisons in v that disagree with σ increases; this again corresponds to intuition.

With these bounds in hand, we can bound the quality of the posterior estimate $\hat{P}_v(r)$ produced by AMP:

Corollary 20 *Let L and U be the lower and upper bound as in Theorems 19 and 17, respectively. Then for $r \in \Omega(v)$, where l_i and h_i are defined in Proposition 11, we have*

$$\frac{L}{\prod_{i=1}^{m} \sum_{j=l_i}^{h_i} \phi^{i-j}} \leq \frac{\hat{P}_v(r)}{P_v(r)} \leq \frac{U}{\prod_{i=1}^{m} \sum_{j=l_i}^{h_i} \phi^{i-j}}. \tag{22}$$

Proof $\hat{P}_v(r)$ has the form given in Proposition 14 while $P_v(r) \propto \phi^{d(r,\sigma)}$. Then apply upper and lower bounds on the normalizing constant of $P_v(r)$. ■

3.2.4 MMP: An MCMC Sampler Based on AMP

While AMP may have (theoretically) poor worst-case performance, we use it as the basis for a *statistically sound* sampler MMP, by exploiting AMP to propose new rankings for the Metropolis algorithm. With Equation 15, we can derive the acceptance ratio for Metropolis. At step $t+1$ of Metropolis, let $r^{(t)}$ be the previous sampled ranking. Ranking r, proposed by AMP independently of $r^{(t)}$, will be accepted as the $t+1$st sample $r^{(t+1)}$ with probability $a^*\left(r, r^{(t)}\right)$, where:

$$\begin{aligned} a^*\left(r, r^{(t)}\right) &= \min\left(1, \frac{\phi^{d(r,\sigma)}/Z_v}{\phi^{d(r^{(t)},\sigma)}/Z_v} \frac{\frac{\phi^{d(r^{(t)},\sigma)}}{\prod_{i=1}^{m} \phi^{i-h_i^t} + \phi^{i-h_i^t+1} + \cdots + \phi^{i-l_i^t}}}{\frac{\phi^{d(r,\sigma)}}{\prod_{i=1}^{m} \phi^{i-h_i} + \phi^{i-h_i+1} + \cdots + \phi^{i-l_i}}}\right) \\ &= \min\left(1, \prod_{i=1}^{m} \begin{cases} \frac{h_i - l_i + 1}{h_i^t - l_i^t + 1} & \text{if } \phi = 1 \\ \frac{\phi^{h_i^t - h_i}(1 - \phi^{h_i - l_i + 1})}{1 - \phi^{h_i^t - l_i^t + 1}} & \text{otherwise} \end{cases}\right). \end{aligned} \tag{23}$$

Here the l_is and h_is are defined as in Equations 13 and 14, respectively (with respect to r; and l_i^t), and h_i^t are defined similarly, but with respect to $r^{(t)}$. The term $Z_v = \sum_{r' \in \Omega(v)} \phi^{d(r',\sigma)}$ is the normalization constant of the Mallows posterior (given partial evidence v). The algorithm is specified in detail in Algorithm 2.

Exploiting Proposition 13, we can show:

Algorithm 2 MMP Sample Mallows Posterior using Metropolis

Input: v, σ, ϕ, number of steps T

1: **for** $t = 1..T$ **do**

2: $r \leftarrow \mathsf{AMP}(v, \sigma, \phi)$

3: $a \sim$ Uniform[0,1]

4: $r^{(t)} \leftarrow \begin{cases} r & \text{if } t = 1 \text{ or } a \leq a^*(r, r^{(t-1)}) \\ r^{(t-1)} & \text{otherwise} \end{cases}$

5: **end for**

Output: $r^{(T)}$

Theorem 21 *The Markov chain induced by* MMP *is ergodic on the class of states (rankings)* $\Omega(v)$.

Proof Note that the acceptance ratio as given in Equation 23 is always positive. The proposal distribution AMP draws rankings that are independent of previous rankings and by Proposition 13, its support is $\Omega(v)$. Hence, for any $r' \in \Omega(v)$, MMP has positive probability of making a transition to any ranking in $\Omega(v)$—thus establishing that $\Omega(v)$ is a recurrent class—including itself—implying aperiodicity. ■

Thus, along with the detailed balance property of Metropolis, we have that the steady state distribution of MMP is exactly the Mallows posterior $P_v(r)$.

3.3 Sampling Mallows Mixture Posterior

Extending the GRIM, AMP and MMP algorithms to sampling from a mixture of Mallows models is straightforward. Recall the mixture posterior:

$$P(r, \mathbf{z}|v, \boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi}) = \frac{P(v|r, \alpha)P(r|\mathbf{z}, \boldsymbol{\sigma}, \boldsymbol{\phi})P(\mathbf{z}|\boldsymbol{\pi})}{\sum_{\mathbf{z}} \sum_{r \in \Omega} P(v|r, \alpha)P(r|\mathbf{z}, \boldsymbol{\sigma}, \boldsymbol{\phi})P(\mathbf{z}|\boldsymbol{\pi})}.$$

We use Gibbs sampling to alternate between r and $\mathbf{z}$, since the posterior does not factor in a way that permits us to draw samples exactly by sampling one variable, then conditionally sampling another. We initialize the process with some $\mathbf{z}^{(0)}$ and $r^{(0)}$, then repeatedly sample $\mathbf{z}$ conditional on r, and r conditional on $\mathbf{z}$. For the tth sample, $\mathbf{z}^{(t)}$ is drawn from a multinomial with K outcomes:

$$\begin{aligned} P(\mathbf{z} : z_k = 1|r^{(t-1)}, \boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi}) &= \frac{P(r^{(t-1)}|\mathbf{z}, \boldsymbol{\sigma}, \boldsymbol{\phi})P(\mathbf{z}|\boldsymbol{\pi})}{\sum_{\mathbf{z}'} P(r^{(t-1)}|\mathbf{z}', \boldsymbol{\sigma}, \boldsymbol{\phi})P(\mathbf{z}'|\boldsymbol{\pi})} \\ &= \frac{\phi_k^{d(r^{(t-1)}, \sigma^{(k)})}\pi_k}{\sum_{k'=1}^{K} \phi_{k'}^{d(r^{(t-1)}, \sigma^{(k')})}\pi_{k'}}. \end{aligned}$$

To sample $r^{(t)}$ given $\mathbf{z}^t$, we use:

$$P(r|\mathbf{z}^{(t)}, v, \boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi}) = \frac{P(v|r)P(r|\mathbf{z}^{(t)}, \boldsymbol{\sigma}, \boldsymbol{\phi})P(\mathbf{z}^{(t)}|\boldsymbol{\pi})}{\sum_{r' \in \Omega} P(v|r')P(r'|\mathbf{z}^{(t)}, \boldsymbol{\sigma}, \boldsymbol{\phi})P(\mathbf{z}^{(t)}|\boldsymbol{\pi})}. \quad (24)$$

Algorithm 3 SP: Sample Mallows Mixture Posterior using Gibbs

Input: $v, \boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi}$, number of steps T

1: Initialize $r^{(0)}$ (e.g., topological sort on v)
2: **for** $t = 1..T$ **do**
3: $\quad \mathbf{z}^{(t)} \sim P(\cdot|r^{(t-1)}, \boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi}) \propto \phi_k^{d(r^{(t-1)}, \sigma^{(k)})} \pi_k$
4: $\quad$ Suppose $\mathbf{z}^{(t)}$ is indicator for kth component.
5: $\quad r^{(t)} \leftarrow$ AMP or MMP$(v, \sigma^{(k)}, \phi_k)$
6: **end for**

Output: $(\mathbf{z}^{(T)}, r^{(T)})$

Note that the term $P(\mathbf{z}^{(t)}|\boldsymbol{\pi})$ in the numerator and denominator cancels, and the missing completely at random assumption (see Equation 9) implies $P(v|r) = \mathbf{1}[r \in \Omega(v)]f(v)$, where f is a function independent of r. Thus Equation 24 becomes Equation 12 (conditioned on parameters $\sigma^{(k)}$, ϕ_k). This is exactly the Mallows posterior sampling problem addressed in the previous section. Combining Gibbs sampling with sampling from a single component gives the overall SP algorithm, which is detailed in Algorithm 3. We note that this sampler is described using either MMP to exactly sample rankings (given the sampled mixture component) or AMP to allow more tractable, but approximate, sampling of rankings (see Line 5). In our experiments, we find that AMP works well within this Gibbs sampler.

4. EM Learning Algorithm for Mallows Mixtures

Armed with the sampling algorithms derived from GRIM, we now turn to maximum likelihood learning of the parameters $\boldsymbol{\pi}$, $\boldsymbol{\sigma}$, and $\boldsymbol{\phi}$ of a Mallows mixture using the *expectation maximization (EM)* algorithm. Before detailing our EM algorithm, we first consider the evaluation of the Mallows mixture log-likelihood in Section 4.1, which can be used to select the number of mixture components, or to test EM learning convergence. We then review the EM algorithm in Section 4.2 before detailing the steps of our EM learning procedure for Mallows mixture models in Section 4.3. In Section 4.4 we analyze the running time of our learning algorithm and suggest several ways to improve its performance.

4.1 Evaluating Log-Likelihood

The log-likelihood in our mixture model is

$$\mathcal{L}_\alpha(\boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi}|V) = \sum_{\ell \in N} \ln \left[\sum_{\mathbf{z}_\ell} \sum_{r_\ell \in \Omega} P(v_\ell|r_\ell) P(r_\ell|\mathbf{z}_\ell, \boldsymbol{\sigma}, \boldsymbol{\phi}) P(\mathbf{z}_\ell|\boldsymbol{\pi}) \right]. \tag{25}$$

This can be rewritten as

$$\begin{aligned} \mathcal{L}_\alpha(\boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi} \mid V) &= \sum_{\ell \in N} \ln \left[\sum_{k=1}^{K} \sum_{r_\ell \in \Omega(v_\ell)} \pi_k P(r_\ell|\sigma^{(k)}, \phi_k) \alpha^{|v_\ell|} (1-\alpha)^{\binom{m}{2} - |v_\ell|} \right] \\ &= \sum_{\ell \in N} \ln \left[\sum_{k=1}^{K} \sum_{r_\ell \in \Omega(v_\ell)} \pi_k P(r_\ell|\sigma^{(k)}, \phi_k) \right] + \ln \left[\alpha^{|v_\ell|} (1-\alpha)^{\binom{m}{2} - |v_\ell|} \right]. \end{aligned}$$

Note that the latter term involving α is decoupled from the other parameters, and in fact its maximum likelihood estimate is $\alpha^* = \sum_{\ell \in N} 2|v_\ell|/(nm(m-1))$. Since we are only interested in the log-likelihood as a function of the other parameters, we can ignore this additive constant and focus on

$$\mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi} \mid V) = \sum_{\ell \in N} \ln \left[\sum_{k=1}^{K} \sum_{r_\ell \in \Omega(v_\ell)} \frac{\pi_k \phi_k^{d(r_\ell, \sigma^{(k)})}}{Z_k} \right], \tag{26}$$

where Z_k is the Mallows normalization constant. Unfortunately, evaluating this term is provably hard.

Theorem 22 *Let $V = (v_1, \ldots, v_n)$ be a profile of partial preferences. Computing the log-likelihood $\mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi}|V)$ is #P-hard.*

Proof We reduce the problem of counting the number of linear extensions of a partial order to this problem (see Theorem 9). Let v be a partial order for which we wish to count its linear extensions. We encode the input to log-likelihood computation as follows: let $V = (v)$, $K = 1$ with $\phi = 1$, and let σ be an arbitrary ranking. We have $\mathcal{L} = \mathcal{L}(\pi, \sigma, \phi|V) = \ln \sum_{r \in \Omega(v)} 1/m!$. Thus we can recover the number of linear extensions by computing $\exp(\mathcal{L}) \cdot m!$. That this can be accomplished in polynomial time can be seen by noting that $\mathcal{L}$ is polynomial in m and we can use the power series expansion $\sum_{i \geq 0} \mathcal{L}^i m!/i!$, where we can truncating the series after a polynomial number of steps, after which the terms in the expansion no longer impact the integer portion of the solution (number of extensions). ■

Given the computational difficulty of evaluating the log-likelihood exactly, we consider approximations. We can rewrite the log-likelihood as

$$\mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi}|V) = \sum_{\ell \in N} \ln \left[\sum_{k=1}^{K} \pi_k \mathop{\mathbb{E}}_{P(r|\sigma^{(k)}, \phi_k)} \mathbf{1}[r \in \Omega(v_\ell)] \right],$$

and estimate the inner expectations by sampling from the Mallows model $P(r|\sigma^{(k)}, \phi_k)$. However, this can require exponential sample complexity in the worst case (e.g., if $K = 1$ and v is far from σ, i.e., $d(v, \sigma)$ is large, then to ensure v is in the sample requires a sample set of exponential size in expectation). But we can rewrite the summation inside the log as

$$\mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi}|V) \;\; = \;\; \sum_{\ell \in N} \ln \left[\sum_{k=1}^{K} \frac{\pi_k}{Z_k} \sum_{r \in \Omega(v_\ell)} \phi_k^{d(r, \sigma^{(k)})} \right],$$

and evaluate $\sum_{r \in \Omega(v_\ell)} \phi_k^{d(r,\sigma^{(k)})}$ using importance sampling:

$$\sum_{r \in \Omega(v_\ell)} \phi_k^{d(r,\sigma^{(k)})} = \mathop{\mathbb{E}}_{r \sim \hat{P}_{v_\ell}} \left[\frac{\phi_k^{d(r,\sigma^{(k)})}}{\hat{P}_{v_\ell}(r|\sigma^{(k)}, \phi_k)} \right]. \tag{27}$$

We generate samples $r_{\ell k}^{(1)}, \ldots, r_{\ell k}^{(T)}$ with $\mathsf{AMP}(v_\ell, \sigma^{(k)}, \phi_k)$ for $\ell \leq n$ and $k \leq K$, then substitute $\hat{P}_v$ from Equation 15 into Equation 27 to obtain:

$$\sum_{\ell \in N} \ln \left[\sum_{k=1}^{K} \frac{\pi_k}{Z_k} \frac{1}{T} \sum_{t=1}^{T} \prod_{i=1}^{m} \sum_{j=i-h_i^{(\ell kt)}}^{i-l_i^{(\ell kt)}} \phi_k^j \right],$$

where $h_i^{(\ell kt)}$ and $l_i^{(\ell kt)}$ are defined in Equations 14 and 13, and defined with respect to $r_{\ell k}^{(t)}$, $\sigma^{(k)}$, and v_ℓ. We can simplify the expression inside the log and derive the estimate:

$$\hat{\mathcal{L}}(\boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi} | V) = \sum_{\ell \in N} \ln \left[\frac{1}{T} \sum_{k=1}^{K} \sum_{t=1}^{T} \pi_k \cdot \begin{cases} \frac{1}{m!} \prod_{i=1}^{m} (h_i^{(\ell kt)} - l_i^{(\ell kt)} + 1) & \text{if } \phi_k = 1 \\ \phi_k^{\sum_{i=1}^{m} i - h_i^{(\ell kt)}} \prod_{i=1}^{m} \frac{1 - \phi_k^{h_i^{(\ell kt)} - l_i^{(\ell kt)} + 1}}{1 - \phi_k^i} & \text{otherwise} \end{cases} \right]. \quad (28)$$

As a matter of practical implementation, to ensure the sum of terms inside the log do not evaluate to zero (as it may be too small to be represented using common floating point standards), we observe that given numbers a and b with $a > b > 0$, $\ln(a + b) = \ln(a) + \ln(1 + b/a)$. Thus even if a and b are too small to be represented as floating point data types, we still obtain good approximations if $\ln(a)$ can be readily evaluated. This same technique can be used to ensure numerical stability.

4.2 The EM Algorithm

A popular approach to maximum likelihood estimation is the *expectation maximization (EM)* algorithm (Dempster et al., 1977). It is applied to probabilistic models in which a set of parameters θ determine the values of random variables, but observed data is available for only some of these variables. Let v denote the observed variables, and h the remaining unobserved (hidden or latent). In our model, we have $\theta = (\boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi}, \alpha)$, while v consists of a set of pairwise comparisons and $h = (\mathbf{z}, r)$ consist of the mixture-component assignment and its underlying complete preference ranking. EM is effectively a local search algorithm, which alternates between two steps. The *E-step* computes a posterior distribution over the hidden variables given the observed variables and a current estimate $\tilde{\theta}$ of the model parameters:

$$\textbf{E-Step:} \quad P(h | v, \tilde{\theta}).$$

The *M-step* computes, as its new estimate, those model parameters θ that maximize the expected value (w.r.t. $\tilde{\theta}$) of the log-likelihood (using the posterior computed in the E-step):

$$\textbf{M-step:} \quad \max_{\theta} \mathop{\mathbb{E}}_{P(h|v,\tilde{\theta})} \ln P(h, v | \theta).$$

These steps are iterated until convergence. Indeed, EM converges and gives a locally optimal solution, since each iteration of EM will increase the log-likelihood. In general one does not need to maximize the log-likelihood in the M-step, but simply increase it. An important

variation of EM called *Monte Carlo EM* is used when the posterior in the E-step is hard to compute (e.g., when dealing with large discrete event spaces, such as rankings). In Monte Carlo EM, ones samples from the posterior in the E-step, and in the M-step simply optimizes the choice of parameters with respect to the empirical (sample) expectation.

4.3 Monte Carlo EM for Mallows Mixtures

Learning a Mallows mixture is challenging, since even evaluating its log-likelihood is #P-hard. A straightforward application of EM yields the following algorithm:

Initialization. Initialize values for $\boldsymbol{\pi}^{\text{old}}$, $\boldsymbol{\sigma}^{\text{old}}$, and $\boldsymbol{\phi}^{\text{old}}$.
E-step. Compute/estimate the posterior $P(\mathbf{z}_\ell, r_\ell | v_\ell, \boldsymbol{\pi}^{\text{old}}, \boldsymbol{\sigma}^{\text{old}}, \boldsymbol{\phi}^{\text{old}})$ for all $\ell \in N$.
M-step. Compute model parameters that maximize expected log-likelihood:

$$\begin{aligned}\boldsymbol{\pi}^{\text{new}}, \boldsymbol{\sigma}^{\text{new}}, \boldsymbol{\phi}^{\text{new}} &= \operatorname*{argmax}_{\boldsymbol{\pi},\boldsymbol{\sigma},\boldsymbol{\phi}} \sum_{\ell \in N} \mathop{\mathbb{E}}_{P(r_\ell, \mathbf{z}_\ell | v_\ell, \boldsymbol{\pi}^{\text{old}}, \boldsymbol{\sigma}^{\text{old}}, \boldsymbol{\phi}^{\text{old}})} [\ln P(v_\ell, r_\ell, \mathbf{z}_\ell | \boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi})] \\ &= \operatorname*{argmax}_{\boldsymbol{\pi},\boldsymbol{\sigma},\boldsymbol{\phi}} \sum_{\ell \in N} \sum_{\mathbf{z}_\ell} \sum_{r_\ell \in \Omega} P(r_\ell, \mathbf{z}_\ell | v_\ell, \boldsymbol{\pi}^{\text{old}}, \boldsymbol{\sigma}^{\text{old}}, \boldsymbol{\phi}^{\text{old}}) \ln P(v_\ell, r_\ell, \mathbf{z}_\ell | \boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi}).\end{aligned}$$

Exact estimation in the E-step and optimization in the M-step is of course difficult due to the intractability of the Mallows posterior. Hence we resort to *Monte Carlo EM* and exploit our sampling methods to render EM tractable as follows. We initialize the parameters with values $\boldsymbol{\pi}^{\text{old}}$, $\boldsymbol{\sigma}^{\text{old}}$, and $\boldsymbol{\phi}^{\text{old}}$. For the E-step, instead of working directly with the posterior, we use GRIM-based Gibbs sampling (see Section 3.3) to obtain samples $(\mathbf{z}_\ell^{(t)}, r_\ell^{(t)})_{t=1}^T$ from the posteriors $P(r_\ell, \mathbf{z}_\ell | v_\ell, \boldsymbol{\pi}^{\text{old}}, \boldsymbol{\sigma}^{\text{old}}, \boldsymbol{\phi}^{\text{old}})$ of each agent $\ell \leq n$. We note once again that Gibbs sampling may use either approximate AMP or the full-fledged MCMC MMP to generate rankings.

In the M-step, we maximize the expected log-likelihood using the *empirical expectation* with respect to the generated samples:

$$\boldsymbol{\pi}^{\text{new}}, \boldsymbol{\sigma}^{\text{new}}, \boldsymbol{\phi}^{\text{new}} = \operatorname*{argmax}_{\boldsymbol{\pi},\boldsymbol{\sigma},\boldsymbol{\phi}} \sum_{\ell=1}^{n} \frac{1}{T} \sum_{t=1}^{T} \ln P(v_\ell, r_\ell^{(t)}, \mathbf{z}_\ell^{(t)} | \boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi}). \tag{29}$$

We show below in Theorem 23 that we can perform this maximization by adjusting the three (sets of) parameters in sequence—specifically, if the parameters are maximized in the order $\boldsymbol{\pi}$, $\boldsymbol{\sigma}$ and $\boldsymbol{\phi}$ (and the first two can be maximized independently), this provides a globally optimal solution for the M-step (i.e., the solution obtained by optimizing parameters simultaneously). However, optimization of $\boldsymbol{\sigma}$, in particular, is NP-hard (as we discuss below), so we use a local search heuristic to approximate the choice of reference rankings in the M-step. We now detail the steps involved in the M-step optimization.

Somewhat abusing notation, let indicator vector $\mathbf{z}_\ell^{(t)}$ denote the mixture component to which the tth sample derived from preference ℓ belongs. We partition the collection of *all agent samples* (over all ℓ) into such classes: let $S_k = (\rho_{k1}, \ldots, \rho_{kj_k})$ be the sub-sample of the rankings $r_\ell^{(t)}$, over all $\ell \in N, t \in [T]$, that are drawn from the kth component of the mixture model, i.e., where $\mathbf{z}_\ell^{(t)} = k$. Note that $j_1 + \cdots + j_K = nT$. We can rewrite the

objective in the M-step as

$$\frac{1}{T}\sum_{k=1}^{K}\sum_{i=1}^{j_k}\ln P(v_{\ell(k,i)}|\rho_{ki})P(\rho_{ki}|\sigma^{(k)},\phi_k)P(k|\pi_k),$$

where $\ell(k,i)$ is the agent in sample $\rho_{k,i}$. We ignore $\ln P(v_{\ell(k,i)}|\rho_{ki})$, which only impacts α; and we know $\rho_{ki} \in \Omega(v_{\ell(k,i)})$. Thus, we can rewrite the objective as

$$\sum_{k=1}^{K}\sum_{i=1}^{j_k}\left[\ln\pi_k + d(\rho_{ki},\sigma^{(k)})\ln\phi_k - \sum_{w=1}^{m}\ln\frac{1-\phi_k^w}{1-\phi_k}\right]. \tag{30}$$

where the last summation is the log of the Mallows normalization term.

Optimizing $\boldsymbol{\pi}$. We apply the method of Lagrange multipliers. The Lagrangian $L = (\sum_{k=1}^{K}\sum_{i=1}^{j_k}\ln\pi_k) + \lambda(\pi_1 + \cdots + \pi_K - 1)$, where we have removed irrelevant terms of the objective not involving $\boldsymbol{\pi}$. Taking the gradient, setting to zero and solving the system of equations $\nabla_{\boldsymbol{\pi},\lambda}L = 0$, we obtain:

$$\pi_k = \frac{j_k}{nT},\ \forall k \le K. \tag{31}$$

Optimizing $\boldsymbol{\sigma}$. The only term involving $\boldsymbol{\sigma}$ in Equation 30 is $\sum_{k=1}^{K}\sum_{i=1}^{j_k} d(\rho_{ki},\sigma^{(k)})\ln\phi_k$. Since $\ln\phi_k$ is a negative scaling factor, and we can optimize the reference rankings $\sigma^{(k)}$ for each mixture component independently, we obtain:

$$\sigma^{(k)*} = \operatorname*{argmin}_{\sigma^{(k)}}\sum_{i=1}^{j_k} d(\rho_{ki},\sigma^{(k)}). \tag{32}$$

Optimizing the choice of reference ranking $\sigma^{(k)}$ within a mixture component requires computation of the Kemeny ranking with respect to the rankings in S_k. This is, unfortunately, an NP-hard problem (Bartholdi III et al., 1989). To maintain tractability, we exploit the notion of *local Kemenization* (Dwork et al., 2001): instead of optimizing the ranking, we compute a locally optimal $\sigma^{(k)}$, in which swapping any two *adjacent* alternatives in $\sigma^{(k)}$ does not reduce the sum of distances in the Kemeny objective. While this may not result in optimal rankings, it has been shown to be extremely effective experimentally (Dwork et al., 2001; Busse et al., 2007).

We detail our local Kemenization algorithm in Algorithm 4. It works by first initializing the new ranking $\sigma^{(k)}$ to that from the previous EM iteration, $\sigma^{\text{old},(k)}$. Then, for each alternative x, starting with those at the top of the ranking and moving downwards, we evaluate swaps of x with the element above it, say y, and proceeding with the swap if the majority of rankings in S_k prefer x over y. This proceeds until the first potential swap of x fails (at which point we move on to the next alternative). This results in a locally optimal ranking (Dwork et al., 2001). Note we need not store all rankings in S_k; we require only its *pairwise tournament graph*, which is a complete directed graph with vertices corresponding to the alternatives A and the weight of each edge $x \to y$ set to be $c_{xy} = |\{\rho \in S_k : y \succ_\rho x\}|$. Here c_{xy} is the "cost" of placing x above y.

Algorithm 4 LocalKemeny

Input: $S_k = (\rho_{k1}, \ldots, \rho_{kj_k})$

$\sigma \leftarrow \sigma_k^{\text{old}}$
Compute pairwise tournament graph:
for all pair $(x, y) : x, y \in A$ and $x \neq y$ **do**
  $c_{xy} = |\{\rho \in S_k : y \succ_\rho x\}|$.
end for
$d \leftarrow \sum_{\{x,y\} \,:\, x \succ_{\sigma^{(k)}} y} c_{xy}$
for $i = 2..m$ **do**
  $x \leftarrow$ alternative in ith rank of σ
  for $j = i - 1..1$ **do**
    $y \leftarrow$ alternative in jth rank of σ
    if $c_{xy} < c_{yx}$ **then**
      Swap x with y
      $d \leftarrow d - c_{xy} + c_{yx}$
    else
      quit this loop
    end if
  end for
end for

Output: σ, Kemeny cost d

Optimizing $\boldsymbol{\phi}$. When optimizing $\boldsymbol{\phi}$ in Equation 30, the objective decomposes into a sum that permits independent optimization of each ϕ_k. Exact optimization of ϕ_k is difficult; however, we can use gradient ascent with:

$$\frac{\partial \text{ (Equation 30)}}{\partial \phi_k} = \frac{d(S_k, \sigma^{(k)})}{\phi_k} - j_k \sum_{i=1}^{m} \frac{[(i-1)\phi_k - i]\phi_k^{i-1} + 1}{(1-\phi_k^i)(1-\phi_k)},$$

where $d(S_k, \sigma^{(k)}) = \sum_{i=1}^{j_k} d(\rho_{ki}, \sigma^{(k)})$ is the Kemeny objective, which we obtain after running LocalKemeny.

Theorem 23 *Let $\boldsymbol{\pi}^*$ be given by Equation 31, $\boldsymbol{\sigma}^*$ be given by Equation 32, and $\boldsymbol{\phi}^*$ be the optimal $\boldsymbol{\phi}$ in Equation 30 where $\boldsymbol{\pi}$ is replaced with $\boldsymbol{\pi}^*$ and $\boldsymbol{\sigma}$ is replaced with $\boldsymbol{\sigma}^*$. Then $\boldsymbol{\pi}^*$, $\boldsymbol{\sigma}^*$ and $\boldsymbol{\phi}^*$ is a globally optimal solution to Equation 29.*

Proof Regardless of the values of $\boldsymbol{\sigma}$ and $\boldsymbol{\phi}$, $\boldsymbol{\pi}$ is optimized by Equation 31 (see our analysis above), giving the optimal solution. It is also easy to see that the optimal reference rankings $\boldsymbol{\sigma}$ are the Kemeny rankings corresponding to ranking sets $S_1, \ldots, S_K$, respectively, independent of the value of $\boldsymbol{\phi}$. Finally, if we substitute the optimal values $\boldsymbol{\pi}^*$ and $\boldsymbol{\sigma}^*$ into Equation 30, its optimal solution $\boldsymbol{\phi}^*$ forms part of the optimal solution $(\boldsymbol{\pi}^*, \boldsymbol{\sigma}^*, \boldsymbol{\phi}^*)$ to Equation 29. ■

It isn't difficult to see that a "locally optimal" pair $(\boldsymbol{\sigma}, \boldsymbol{\phi})$ obtained by optimizing $\boldsymbol{\sigma}$ first, then $\boldsymbol{\phi}$ is a locally optimal pair for Equation 29. Hence the resulting EM estimates

are also locally optimal with respect to the likelihood (Neal and Hinton, 1999). While no approximation bounds can be given, this lends some support to the optimization approach we adopt. To test the convergence of EM, one can test the convergence of the parameters (use Kendall-tau distance to measure $\boldsymbol{\sigma}$ against that of the previous iteration). One can also measure whether the log-likelihood is converging.

To reduce problems with local maxima, we initialize the mixture parameters using a K-means clustering approach where distances are measured using Kendall-tau rather than the usual squared Euclidean distance. One can use a modified version of Lloyd's 1982 method for K-means, where the "centroid" (pertaining to Lloyd's method) of a set of rankings can is simply its Kemeny ranking.

4.4 Complexity of EM Steps

We analyze the running time of one iteration of our EM approach. In the E-step, we sample variables $(\mathbf{z}, r)$. We need not store the ranking r for the component corresponding to $\mathbf{z}$, since in the M-step we do not need the actual rankings in S_k, but only its pairwise tournament graph. Hence we need only update the tournament graph corresponding to component $\mathbf{z}$ with sample r, which takes $O(m^2)$ time. When sampling r, let T_{Metro} be the number of Metropolis steps before using the next sample. Each draw of r from AMP requires $O(m^2)$ time. Sampling $\mathbf{z}$ requires $O(Km \log m)$ time since Kendall-tau distance can be computed in $O(m \log m)$ time. Let T_{Gibbs} be the number of Gibbs sampling steps run Gibbs before outputting a sample and suppose we restart Gibbs after each such sample. Suppose also we draw T_P posterior samples for each data point v_ℓ. Then the E-step takes $O(nT_P T_{\text{Gibbs}}(T_{\text{Metro}} m^2 + Km \log m))$ time. In practice, one can chose a very small number of samples, and run relatively few steps, when running the MCMC methods. Indeed, in our experiments below, we don't use MMP within the Gibbs sampler, but instead use AMP directly (this can be viewed as running Metropolis for a single step); we discuss this further below. In principle, posterior sampling can be executed in parallel, with multiple processors handling the sampling and tournament graph updates for disjoint subsets of the data v_ℓ, with the results from different processors merged into the K tournament graphs.

For the M-step, updating $\boldsymbol{\pi}$ takes constant time, while updating the component reference rankings $\boldsymbol{\sigma}$ takes $O(Km^2)$ time. Optimizing $\boldsymbol{\phi}$ can also be realized effectively, for instance, by using gradient ascent and bounding number of iterations. Hence the M-step requires $O(Km^2)$ time. Space complexity is dominated by the size of the K tournament graphs, hence is $O(Km^2)$.

Various techniques can be used to speed up computation from a practical perspective. Instead of storing the tournament graphs, which require quadratic memory, one can instead approximate the Kemeny ranking for any component using the *Borda count* to rank alternatives, which is a 5-approximation to Kemeny (Coppersmith et al., 2006), and often provides much better approximations in practice. If using Borda, when generating a complete ranking r in the posterior-sampling step (E-step) belonging to component k, one need only to update the Borda scores of all alternatives within component k; in the M-step we simply rank alternatives (within each component) according to their sampled Borda scores. We still need the Kemeny distance between the resulting Borda ranking and the sampled rankings, but this can be approximated by re-running the E-step and evaluating the Kendall-tau

distance in an online fashion. One might also consider using Spearman footrule distance, which can be computed in $O(m)$ time rather than $O(m \log m)$ as in Kendall-tau, since it is 2-approximation to the Kemeny distance (Diaconis, 1988).

5. Experiments

We perform a series of experiments to validate the efficacy of our sampling and learning algorithms, to discover interesting properties of the learned mixture models on several popular data sets, and to evaluate the predictive power of our learned models to help predict missing preferences. We first assess the quality of our GRIM-based posterior sampling method AMP, measuring its accuracy relative to the true Mallows posterior. We then measure the approximation quality of our Monte Carlo algorithm for evaluating the Mallows mixture log-likelihood. Next we apply our EM algorithm to learn mixture models using several data sets: synthetically generated data sets, a Movielens ratings data set (with large m); and a sushi preference data set. The synthetic data experiments confirm the effectiveness of our EM algorithm while also revealing insights on how the size of preference data (either n or α) impacts learning. We also remark on some of its connections to crowdsourcing. Finally we assess the predictive accuracy of the learned models by conditioning on partial preference information and inferring the probability of the missing pairwise comparison preferences. In all experiments, we use Equation 26 to measure log-likelihood.

5.1 Sampling Quality

We first assess how well AMP approximates the true Mallows posterior P_v using randomly generated (synthetic) data. We vary parameters m, ϕ and α, while fixing a canonical reference ranking $\sigma = (1, 2, \cdots m)$. For each parameter setting, we generate 20 preferences v (e.g., the partial preferences of 20 agents) using our mixture model (see Section 2.3 and Equations 9 and 10), and evaluate the exact KL-divergence of P_v with respect to $\hat{P}_v$.[5] This divergence is normalized by the entropy of P_v, since, when increasing m, KL-divergence and entropy both increase. Results are shown in Figure 3, with fixed and varying parameters for all three plots described in the caption. These results indicate that AMP approximates the posterior very well, with average normalized KL error ranging from 1–5%, across the parameter ranges tested.

5.2 Evaluating Log-Likelihood

In Section 4.1 we showed the #P-hardness of evaluating the log-likelihood and derived a Monte Carlo estimator that uses the AMP sampler. We evaluate the quality of the approximation produced by this estimator in this section. We vary three parameters to generate three experiments: (a) the number of alternatives m; (b) the number of mixture components K; and (c) the number of samples T per agent and per component (Equation 28). In all experiments, we fix the number of agents (i.e., the number of input preferences) at $n = 50$.

5. To compute KL-divergence, we need only consider consistent completions of our partial preferences. This set of rankings usually has size much smaller than $m!$, and can be enumerated by modifying the topological sort algorithm.

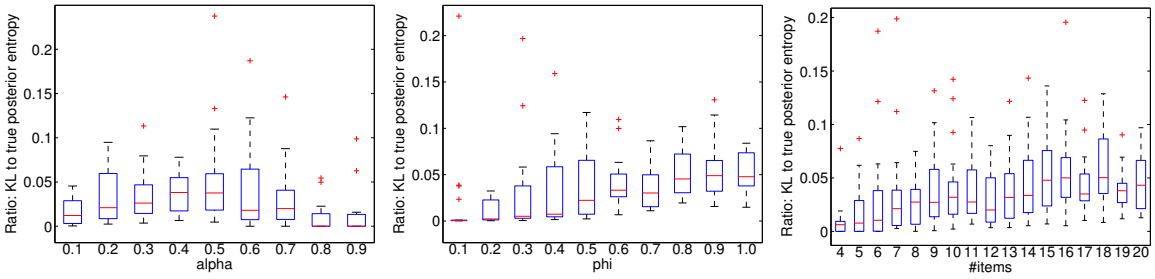


Figure 3: Comparing the posterior generated by AMP to the true Mallows posterior: normalized KL-divergence. The box-and-whisker plots have boxes shown the 25-75 percentile range over 20 runs, with the line inside box indicating the median, and the '+' symbols outliers. From left to right: **Plot 1**: Varying α, while fixing $\phi = 0.5$, $m = 10$. **Plot 2**: Varying ϕ, while fixing $\alpha = 0.2$, $m = 10$. **Plot 3**: Varying m, fixing $\phi = 0.5$ and for $m \leq 13$, $\alpha = 0.2$, for $m > 13$, $\alpha = 0.5$.

In setting (a) (varying m), we generate v from a mixture model with $K = 3$ and $\boldsymbol{\pi} = (1/3, 1/3, 1/3)$, $\boldsymbol{\phi} = (1/2, 1/2, 1/2)$ and $\alpha = 0.2$. Each $\boldsymbol{\sigma}^k$ $(k \leq K)$ is drawn uniformly at random from Ω.

In setting (b) (varying K), we generate v from a mixture model with K components, where $m = 8$, $\boldsymbol{\pi} = (1/K, \ldots, 1/K)$, $\boldsymbol{\phi} = (1/2, \ldots, 1/2)$ and $\alpha = 0.2$. Again $\boldsymbol{\sigma}$ drawn uniformly at random as in setting (a).

In setting (c) (varying T), parameters are $K = 1$, $m = 8$, σ chosen uniformly at random, $\phi = 0.5$ and $\alpha = 0.2$.

The parameters for which we evaluated the log-likelihood are generated as follows: mixture weights $\boldsymbol{\pi}$ are sampled from a "uniform" Dirichlet distribution with a parameter vector (i.e., equivalent sample size counts) consisting of K 5s. The reference rankings $\boldsymbol{\sigma}$ were drawn uniformly at random, and ϕ is drawn uniformly at random from interval $(0, 1)$.

The results for all three settings are shown in Figure 4. Overall we see that the Monte Carlo approximation is very good, and improves significantly while reducing variance as we increase the sample size for each agent's log-likelihood (as captured by $K \cdot T$). Increasing m slightly degrades approximation quality, although it offers excellent estimates across the entire range of tested values.

5.3 EM Mixture Learning

We now evaluate our EM mixture-learning algorithms on the synthetic, Sushi and Movielens data sets.

5.3.1 Synthetic Data

Having empirically established that AMP provides good approximations to the true posterior, and that the log-likelihood can be closely approximated by importance sampling, we now evaluate how effective our EM algorithm is at recovering parameters in a controlled

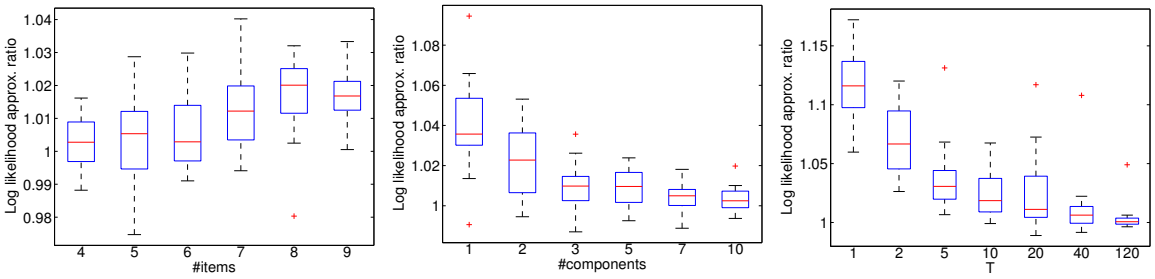


Figure 4: Comparing the ratio of the true log-likelihood to its Monte Carlo approximation. 20 instances are run per parameter setting. From left to right: **Plot 1**: Varying m, while fixing $T = 5$. **Plot 2**: Varying K while fixing $T = 5$. **Plot 3**: Varying T. Other parameter values are described in the text.

setting, using synthetic data generated from models with known parameters. We emphasize that the following experiments all use AMP within the Gibbs sampler in the E-step of Monte Carlo EM, rather than the MCMC algorithm MMP, given the approximation quality of AMP as well as its much better tractability.

We perform four experiments in which we vary: (a) α, the (expected) fraction of pairwise comparisons revealed from each preference; (b) the number of alternatives m; (c) the number of mixture components K; and (d) the number of agent preferences (data set size). In each experiment, we generate random model parameters as follows: $\boldsymbol{\pi}$ is drawn from a Dirichlet distribution with a uniform parameter vector of 5s; $\boldsymbol{\sigma}$ is drawn uniformly at random; and $\boldsymbol{\phi}$ values are drawn uniformly at random from $[0.2, 0.8]$. Training data is generated using our probabilistic model with these parameters. When varying the single parameter for each experiment, we fix the other three, with fixed values: $\alpha = 0.2$, $m = 20$, $K = 3$ and $n = 50 \times K$. We analyze the performance of EM by (approximately) evaluating the ratio of the log-likelihood of the learned parameters to that of the true model parameters $(\boldsymbol{\pi}, \boldsymbol{\sigma}, \boldsymbol{\phi})$ on test data (preferences) generated from the true model—we set $n_{test} = n$ and $\alpha_{test} = 1$.

Results are shown in Figure 5 and provide some interesting insights. First they suggest that learning is more effective when either of α or n is larger (i.e., when we have more preference data for training). We also see that learning performance degrades when we increase the number of mixture components—this is hardly surprising, since there is less data per component as we increase K. Finally, learning improves as m increases for fixed values of α. This holds because the transitive closure for larger m tends to offer more preference information. For instance, $a_1 \succ a_2 \succ a_3 \succ a_4 \succ a_5 \succ a_6$ provides 5 comparisons, and corresponds to 1/9 of all comparisons when $m = 10$, while leaving many comparisons unavailable, even after taking its transitive closure. By contrast, $a_1 \succ a_2 \succ \cdots \succ a_{100}$ has 99 comparisons which is only 1/50 of all comparisons when $m = 100$; but its transitive closure is a complete ranking.

These observations have interesting implications when considering information elicitation via "wisdom of the crowds." When estimating a single objective ranking (i.e., $K = 1$), the amount of data needed for reliable estimation can be obtained by either increasing α (the

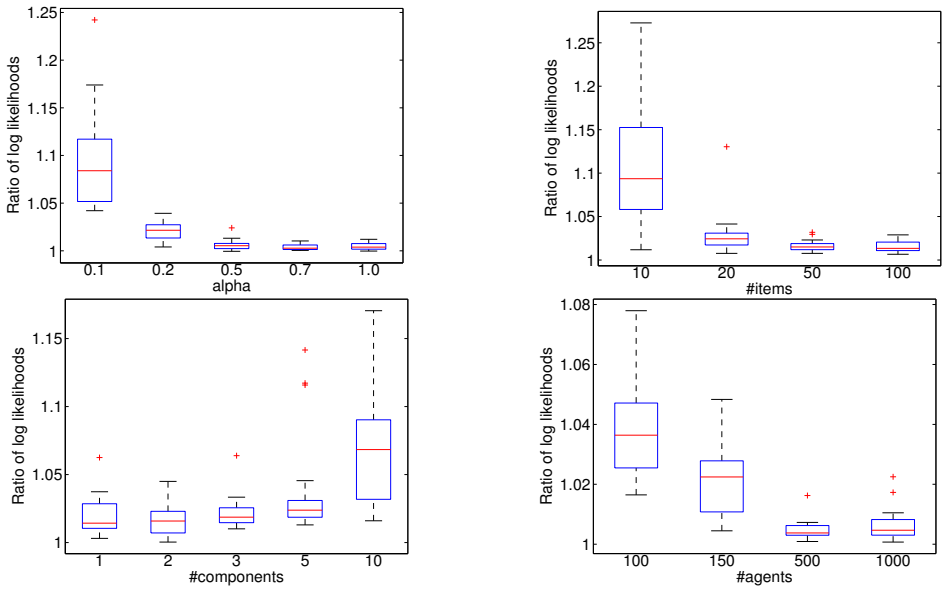


Figure 5: Performance of EM on synthetic data. Each plot shows the ratio of the log-likelihood of learned parameters to those of the true model parameters. Each parameter setting averages results of 20 instances. Log-likelihoods are approximated as in Section 4.1 with $T = 10$. Other parameter settings are described in the text.

average number of pairwise comparisons revealed per agent) and decreasing n (the number of agents queried) or by increasing n and decreasing α. In other words, one can obtain the same "effective" data by either asking more agents about their objective assessments while decreasing the number of questions per agent, as asking fewer agents to respond, but demanding more pairwise assessments per agent.

5.3.2 Sushi Data

The Sushi data set consists of 5000 complete rankings over 10 varieties of sushi indicating sushi preferences (Kamishima et al., 2005). We used 3500 preferences for training and 1500 for validation. We ran EM experiments by generating revealed pairwise comparisons for training with various probabilities α. To mitigate issues with local maxima, we ran EM ten times (more than is necessary) for each instance. Figure 6 shows that, even without complete preferences, EM learns well even with only 30-50% of all paired comparisons, though it degrades significantly at 20%, in part because only 10 alternatives are ranked (still performance at 20% is good when $K = 1, 2$). With $K = 6$ components, a good fit is found when training on complete preferences: Table 1 shows the learned clusters (all with reasonably low dispersion), illustrating interesting patterns (e.g., fatty tuna is strongly preferred by all but one group; a strong correlation exists across groups in preference/dispreference for salmon roe and sea urchin, which are "atypical fish"; and cucumber roll is consistently dispreferred).

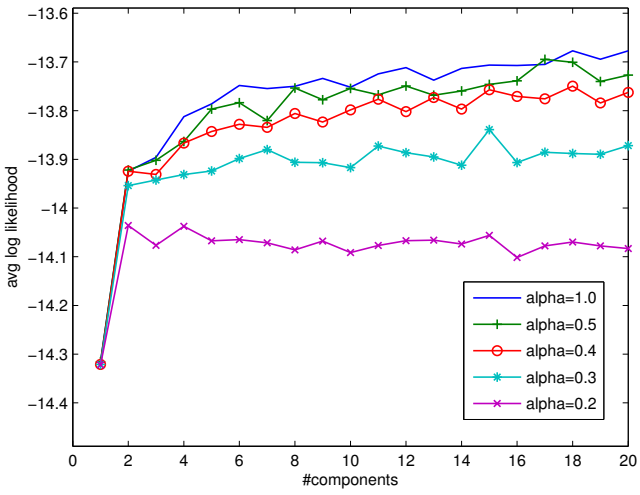


Figure 6: Sushi data set. Plots of average validation log-likelihood when the training data, pairwise comparisons, are revealed with probabilities $\alpha \in \{0.2, 0.3, 0.4, 0.5, 1.0\}$. Learning degrades as α gets closer to 0.2, that is, as more pairwise comparisons are removed.

$\pi_0 = 0.17$ $\phi_0 = 0.66$	$\pi_1 = 0.15$ $\phi_1 = 0.74$	$\pi_2 = 0.17$ $\phi_2 = 0.61$	$\pi_3 = 0.18$ $\phi_3 = 0.64$	$\pi_4 = 0.16$ $\phi_4 = 0.61$	$\pi_5 = 0.18$ $\phi_5 = 0.62$
fatty tuna	shrimp	sea urchin	fatty tuna	fatty tuna	fatty tuna
salmon roe	sea eel	fatty tuna	tuna	sea urchin	sea urchin
tuna	squid	sea eel	shrimp	tuna	salmon roe
sea eel	egg	salmon roe	tuna roll	salmon roe	shrimp
tuna roll	fatty tuna	shrimp	squid	sea eel	tuna
shrimp	tuna	tuna	sea eel	tuna roll	squid
egg	tuna roll	squid	egg	shrimp	tuna roll
squid	cucumber roll	tuna roll	cucumber roll	squid	sea eel
cucumber roll	salmon roe	egg	salmon roe	egg	egg
sea urchin	sea urchin	cucumber roll	sea urchin	cucumber roll	cucumber roll

Table 1: Learned model for $K = 6$ on the sushi data set with complete preferences.

5.3.3 Movielens Data

We apply our EM algorithm to a subset of the Movielens data set (see `www.grouplens.org`) to find "preference types" across users. We use the 200 (out of roughly 3900) most frequently rated movies, and the ratings of the 5980 users (out of roughly 6000) who rated at least one of these. Integer ratings from 1 to 5 are converted to pairwise preferences in the obvious way (for ties, no preference was added to v). For example, if A and B had rating 5, C had rating 3 and D rating 1 then the user preference becomes $v = \{A \succ C, A \succ D, B \succ C, B \succ D, C \succ D\}$. We discard preferences that are empty when restricted to the top 200 movies, and use 3986 preferences for training and 1994 for validation. We run EM with number of components $K = 1, \ldots, 20$; for each K we ran EM 20 times to mitigate the impact of local maxima. For each K, we evaluate average log-likelihood of the best run on the validation set to select the number of mixture components K. Log-likelihoods are approximated using our Monte Carlo estimator (with $K \cdot T = 120$).[6]

Log-likelihood results are shown in Figure 7 as a function of the number of mixture components. These results suggest that the best component sizes are $K = 10$ and $K = 5$ on the validation set. The learned model with $K = 5$ is displayed in Table 2, with each component ranking truncated to the top-20 movies. The five references rankings in this case are have some intuitive interpretation, but do not seem to exhibit the same separation as in the Sushi data set, in part due to the non-trivial overlap involving a number of "universally popular" movies (e.g., two movies, *The Shawshank Redemption* and *The Usual Suspects*, occur in all five components; two more occur in four, and more than 30 occur in three). Note also that the dispersion of each component is extremely high, approaching 1.

Despite this, certain patterns can be discerned. especially by focusing on reasonably *unique* movies, those than occur in only one or two components. For example, the second component contains the following "unique" movies: *Monty Python and the Holy Grail*, *The Maltese Falcon*, *Blade Runner*, *One Flew Over the Cuckoo's Nest*, *A Clockwork Orange*, *2001: A Space Odyssey*, *North by Northwest*, *Pulp Fiction*, *Chinatown*, and *Apocalypse Now*. Themes within this cluster of unique movies include "older" science fiction, ultra-violence, actor Jack Nicholson and director Stanley Kubrick. The average date of the (top) twenty movies within this component is 1970, which is significantly lower than those of other components.

The same analysis of the fifth component shows the following "unique" movies: *A Christmas Story*, *This is Spinal Tap*, *American Beauty*, *Pulp Fiction*, *The Princess Bride*, *Forrest Gump*, *Fight Club*, *Fargo*, *Ferris Bueller's Day Off*, *Raising Arizona*, *Good Will Hunting*, and *The Matrix*. Many of the movies here would commonly be characterized as "quirky," including five "quirky comedies," and several that tend toward extreme violence. The movies in this component also have a significantly later average date, 1992, than the others.

6. The C++ implementation of our algorithms have EM wall clock times of 15–20 minutes (Intel Xeon dual-core, 3GHz), certainly practical for a data set of this size. In other data sets, given the smaller number of alternatives, run times are much faster.

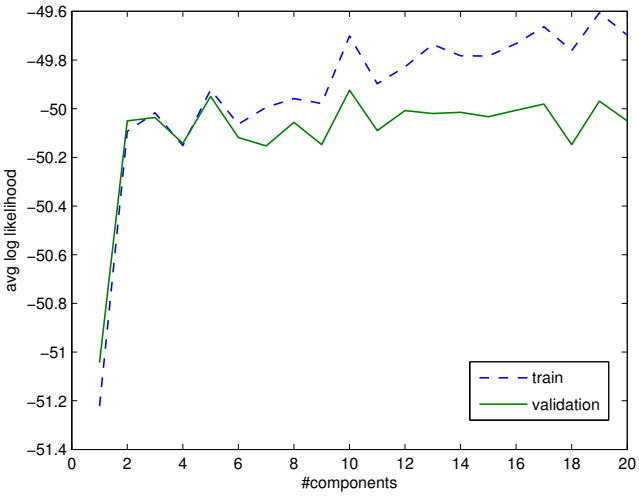


Figure 7: Movielens data set: average training and validation log likelihoods on the learned model parameters of different component sizes.

$\pi_1 = 0.24$, $\phi_1 = 0.98$	$\pi_2 = 0.23$, $\phi_2 = 0.98$	$\pi_3 = 0.21$, $\phi_3 = 0.98$	$\pi_4 = 0.19$, $\phi_4 = 0.98$	$\pi_5 = 0.13$, $\phi_5 = 0.97$
Citizen Kane (1941)	Godfather, The (1972)	Raiders of the Lost Ark (1981)	Shawshank Redemption, The (1994)	Usual Suspects, The (1995)
Godfather, The (1972)	Dr. Strangelove (1963)	Godfather, The (1972)	Life Is Beautiful (1997)	Shawshank Redemption, The (1994)
Dr. Strangelove (1963)	Citizen Kane (1941)	Schindler's List (1993)	Raiders of the Lost Ark (1981)	Schindler's List (1993)
Schindler's List (1993)	Casablanca (1942)	Rear Window (1954)	Schindler's List (1993)	Life Is Beautiful (1997)
Rear Window (1954)	Star Wars: Episode IV - A New Hope (1977)	Star Wars: Episode IV - A New Hope (1977)	Star Wars: Episode IV - A New Hope (1977)	Christmas Story, A (1983)
Shawshank Redemption, The (1994)	Usual Suspects, The (1995)	Shawshank Redemption, The (1994)	Matrix, The (1999)	This Is Spinal Tap (1984)
American Beauty (1999)	Raiders of the Lost Ark (1981)	Casablanca (1942)	Sixth Sense, The (1999)	American Beauty (1999)
Godfather: Part II, The (1974)	Monty Python and the Holy Grail (1974)	Sixth Sense, The (1999)	Sting, The (1973)	Sixth Sense, The (1999)
One Flew Over the Cuckoo's Nest (1975)	Rear Window (1954)	Psycho (1960)	Forrest Gump (1994)	Pulp Fiction (1994)
Casablanca (1942)	Maltese Falcon, The (1941)	Citizen Kane (1941)	Usual Suspects, The (1995)	Princess Bride, The (1987)
Usual Suspects, The (1995)	Blade Runner (1982)	Sting, The (1973)	Braveheart (1995)	Silence of the Lambs, The (1991)
Pulp Fiction (1994)	One Flew Over the Cuckoo's Nest (1975)	Usual Suspects, The (1995)	Green Mile, The (1999)	Godfather, The (1972)
Monty Python and the Holy Grail (1974)	Clockwork Orange, A (1971)	Saving Private Ryan (1998)	Indiana Jones and the Last Crusade (1989)	Forrest Gump (1994)
Fargo (1996)	2001: A Space Odyssey (1968)	Godfather: Part II, The (1974)	Saving Private Ryan (1998)	Fight Club (1999)
Life Is Beautiful (1997)	North by Northwest (1959)	Silence of the Lambs, The (1991)	Princess Bride, The (1987)	Fargo (1996)
Graduate, The (1967)	Pulp Fiction (1994)	Wizard of Oz, The (1939)	Star Wars: Episode V - The Empire Strikes Back (1980)	Ferris Bueller's Day Off (1986)
North by Northwest (1959)	Godfather: Part II, The (1974)	Dr. Strangelove (1963)	Silence of the Lambs, The (1991)	Raising Arizona (1987)
GoodFellas (1990)	Chinatown (1974)	Jaws (1975)	Good Will Hunting (1997)	Saving Private Ryan (1998)
Chinatown (1974)	Apocalypse Now (1979)	Braveheart (1995)	Ferris Bueller's Day Off (1986)	Good Will Hunting (1997)
Raiders of the Lost Ark (1981)	Shawshank Redemption, The (1994)	Aliens (1986)	When Harry Met Sally (1989)	Matrix, The (1999)

Table 2: Learned model for $K = 5$ on Movielens. Shows the top 20 (out of 200) movies.

5.4 Predicting Missing Pairwise Preferences

In our prediction experiments, we seek to evaluate the performance of the learned models in predicting unseen pairwise comparisons. In particular, we use the complete sushi data set, train our mixture model on the first 3500 complete rankings (we train for all $K = 1, \ldots, 20$), and select the best K by evaluating the log-likelihood on the validation data set, which consists of 500 complete rankings. It turns out that a mixture model with $K = 6$ was most suitable.

To test posterior prediction performance, we use 1000 complete rankings, distinct from both the training and validation sets, and randomly remove a fraction $1 - \alpha$ of the pairwise comparisons from each ranking, then compute the transitive closure of the remaining comparisons to obtain partial preferences. We generate preferences for four different values of α. With $\alpha = 0$, all preferences are removed; with $\alpha = 0.25$, 42% of the pairwise comparisons are left after computing transitive closures; with $\alpha = 0.5$, 76% of the all pairwise comparisons remain; and with $\alpha = 0.75$, 83% of the pairwise comparisons are left.

We conditioned the learned model on the partial preferences of each agent in turn, to obtain posterior distributions over which we can infer each agent's missing pairwise comparisons. In making predictions, we use our posterior sampling algorithm SP to sample complete rankings, which we then use to update a tournament graph—recall, this is a set of counts c_{ab} to count the number of rankings for which $a \succ b$, for all $a, b \in A$. Then we estimate the posterior probability $P(a \succ b \mid v)$ by $\frac{c_{ab}}{c_{ab}+c_{ba}}$.

We define our prediction loss as follows. Suppose we have a complete ranking r with its corresponding partial preference v obtained as described above. For a given $a \succ_r b$ that is unobserved in $\mathsf{tc}(v)$, we define the posterior prediction loss to be $\hat{P}(a \prec b \mid v) = \frac{c_{ba}}{c_{ab}+c_{ba}}$. Let $M(v) = \{(a, b) : a \succ_r b, a \succ b \notin \mathsf{tc}(v)\}$ be the set of missing pairwise comparisons in v. We define the *average loss of* v as

$$\varepsilon_v = \frac{\sum_{(a,b)\in M(v_\ell)} \hat{P}(a \prec b|v)}{|M(v_\ell)|}.$$

We next define the *average loss per preference* to be

$$\overline{\varepsilon} = \frac{1}{n} \sum_{\ell=1}^{n} \varepsilon_{v_\ell},$$

where n is the number of distinct agents or preferences (in this case $n = 1000$). For $a \succ_r b$, let $\mathcal{D}(a, b) = r(b) - r(a)$ be the difference in their rank positions and $M_D(v) = \{(a, b) \in M(v) : \mathcal{D}(a, b) = D\}$. We also measure the *average loss at distance* D as follows:

$$\overline{\varepsilon}_D = \frac{\sum_{\ell=1}^{n} \sum_{(a,b)\in M_D(v_\ell)} \hat{P}(a \prec b|v)}{\sum_{\ell=1}^{n} |M_D(v_\ell)|}.$$

The results for average loss per preference are as follows:

- $\overline{\varepsilon} = 0.43$ for preferences generated with $\alpha = 0$;
- $\overline{\varepsilon} = 0.35$ for $\alpha = 0.25$,
- $\overline{\varepsilon} = 0.39$ for $\alpha = 0.5$, and

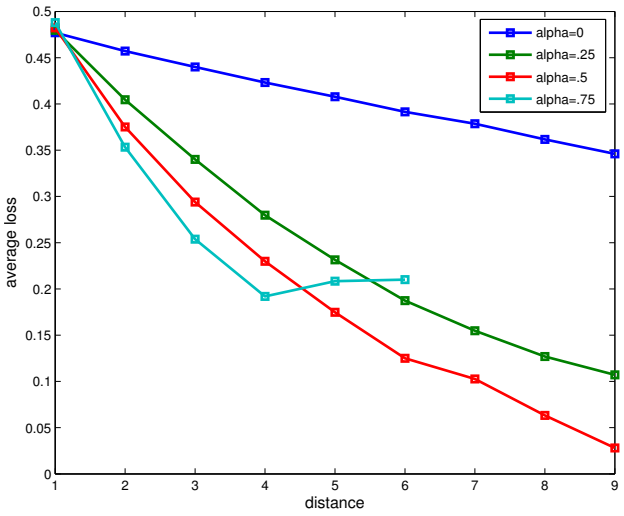


Figure 8: Sushi prediction results: average prediction loss for missing pairwise comparisons for pairs at different distances in the underlying ranking.

- $\overline{\varepsilon} = 0.44$ for $\alpha = 0.75$.

(We interpret these results below). Results for $\overline{\varepsilon}_D$ at various distances are plotted in Figure 8. Since these results are extremely sensitive to the number of pairwise comparisons available in the data at different distances, we show the number of such comparisons, per distance, in Table 3.

The results indicate that predictive performance is weakly accurate when pairs are close in distance, but improves as the distance between the predicted alternatives increases in the underlying ranking. For $\alpha = 0.75$, the average loss at distances 5 and 6 is higher than expected, but this is due to the small number of comparisons missing available for testing (and in general) at those distances. We also observe that the number of comparisons of a particular distance decreases as a function of the distance—this is more pronounced for smaller values of α. This can be attributed to the use of transitive closure: the further apart a pair of alternatives are in the underlying ranking, the less likely it is that we will remove all of the pairwise comparisons required render the two alternatives incomparable *after* taking the transitive closure of the preferences that remain. As a consequence of the skewed distribution of missing pairs available for prediction at specific distances, the average loss per preference does not in fact decrease as α increases. For example, it is 0.39 for $\alpha = 0.5$, and 0.44 for $\alpha = 0.75$; this is because the *relative* number of missing comparisons at smaller distances (which are more difficult to predict) is much greater when $\alpha = 0.75$ than when $\alpha = 0.5$ (as shown in Table 3).

6. Applications to Non-Parametric Estimators

Lebanon and Mao (2008) propose a non-parametric estimator for Mallows models when observations form partitioned preferences. This estimator is an analogue of typical kernel

	$D=1$	$D=2$	$D=3$	$D=4$	$D=5$	$D=6$	$D=7$	$D=8$	$D=9$
$\alpha=0$	9000	8000	7000	6000	5000	4000	3000	2000	1000
$\alpha=.25$	6610	5487	4393	3459	2492	1744	1092	606	256
$\alpha=.5$	4429	2911	1683	898	443	191	65	22	1
$\alpha=.75$	2230	824	244	62	9	2	0	0	0

Table 3: The number of missing pairwise comparisons (over all agents) among pairs that are distance D from each other, with preferences generated by randomly deleting fraction $1-\alpha$ of preferences, then taking the transitive closure of the remaining comparisons.

density estimators, but over the space of rankings. Their purpose, similar to mixture models, is to model the distribution of real ranking data. The idea is to place "smooth unimodal bumps," formulated as a single Mallows model, at every input (training) preference. This is much like a mixture model with the number of components equal to the number of preferences in the training data. They offer closed-form solutions by exploiting the existence of the closed-form for the Mallows normalization constant when partitioned preferences are observed. Unfortunately, with general pairwise comparisons, computing this normalization constant is intractable unless #P=P. In contrast to our contributions above, they do not address the question of how to find a maximum likelihood estimate of the Mallows dispersion parameter, also known as the kernel width, which they suggest as being "extremely difficult."

It turns out we can use AMP for approximate marginalization to support non-parametric estimation with general preference data. This shows the potential applicability of our sampling algorithm to a wider range of problems where observations consist of pairwise comparisons. We illustrate its application by defining a non-parametric estimator and deriving a Monte Carlo evaluation formula suitable for incomplete preferences.

Define a joint distribution q_ℓ over the probability space $\Omega(v_\ell) \times \Omega$:

$$q_\ell(s,r) = \frac{\phi^{d(r,s)}}{|\Omega(v_\ell)|Z_\phi}, \tag{33}$$

where Z_ϕ is the Mallows normalization constant with respect to dispersion ϕ. This distribution corresponds to drawing a ranking s uniformly at random from $\Omega(v_\ell)$ and then drawing r from a Mallows distribution with reference ranking s and dispersion ϕ. The estimator, extended in the style of Lebanon and Mao (2008) to any set of paired comparisons, is simply:

$$\begin{aligned} p(v) &= \frac{1}{n}\sum_{\ell\in N} q_\ell(s\in\Omega(v_\ell), r\in\Omega(v)) \\ &= \frac{1}{n}\sum_{\ell\in N}\sum_{s\in\Omega(v_\ell)}\sum_{r\in\Omega(v)}\frac{\phi^{d(r,s)}}{|\Omega(v_\ell)|Z_\phi}. \end{aligned} \tag{34}$$

Note that this is a distribution over *rankings* and not incomplete preferences, that is, a marginal over $\Omega(v)$. A special case arises when V consists entirely of complete rankings,

which simplifies to a mixture of Mallows models with n equally weighted components, each with one of the observed rankings v_ℓ as its reference ranking, and dispersion ϕ. This estimator can be useful for inference over the posterior $p(r|v) = p(r)\mathbf{1}[r \in \Omega(v)]/p(v)$ for $r \in \Omega(v)$. For any fixed v, let $f(s) = \sum_{r \in \Omega(v)} \phi^{d(r,s)}$. Then we have

$$\begin{aligned} p(v) &= \frac{1}{nZ_\phi} \sum_{\ell \in N} \sum_{s \in \Omega(v_\ell)} \frac{1}{|\Omega(v_\ell)|} f(s) \\ &= \frac{1}{nZ_\phi} \sum_{\ell \in N} \mathop{\mathbb{E}}_{s \sim \Omega(v_\ell)} f(s), \end{aligned}$$

where s is drawn uniformly from $\Omega(v_\ell)$. One can estimate the expectation by importance sampling. Suppose we draw, for each ℓ, rankings $s_\ell^{(1)}, \ldots, s_\ell^{(T)}$ using $\mathsf{AMP}(v_\ell, \sigma, \phi = 1)$ to approximate uniform sampling (e.g., choose some ranking σ from $\Omega(v_\ell)$). Let $w_{\ell t} = 1/\hat{P}_{v_\ell}(s_\ell^{(t)})$, which has a closed-form given by Equation 15. Then the estimate is

$$\hat{p}(v) = \frac{1}{nZ_\phi} \sum_{\ell=1}^{n} \frac{\sum_{t=1}^{T} w_{\ell t} f(s_\ell^{(t)})}{\sum_{t=1}^{T} w_{\ell t}}.$$

Evaluating $f(s_\ell^{(t)})$ is generally intractable, but again, it can be approximated using our earlier techniques, as given by Equation 27. In summary, we can realize non-parametric estimation using a nested sampling procedure to first approximate the outer expectation over s, followed by the inner summation $f(s)$.

7. Conclusion and Future Work

We have developed a set of algorithms to support the efficient and effective learning of ranking or preference distributions when the observed data comprise a set of unrestricted pairwise comparisons of alternatives. Given the fundamental nature of pairwise comparisons in revealed preference, our methods extend the reach of rank learning in a vital way. One of our main technical contributions, the GRIM algorithm, allows sampling of arbitrary distributions, including Mallows models conditioned on pairwise data. It supports a tractable approximation to the #P-hard problem of log-likelihood evaluation of Mallows mixtures; and it forms the heart of an EM algorithm that was shown to be quite effective in our experiments. GRIM can also be used for non-parametric estimation.

We are pursuing a number of interesting directions, including various extensions and applications of the model we have developed here. One of the weaknesses with Mallows is its lack of flexibility in various dimensions, such as allowing different dispersion "rates" in different regions of the ranking. Models that allow more flexibility while controlling for overfitting could lead to more realistic ranking models for real-world settings. Other extensions include exploration of other probabilistic models of incomplete preferences that employ different distributions over rankings, such as Plackett-Luce or weighted Mallows; that account for noisy comparison data from users; and that account for data that is not missing at random—this may occur, say, in settings in which a bias exists towards observing preferences for higher ranked alternatives.

In another vein, we are interested in exploiting learned preference models of the type developed here for decision-theoretic tasks in social choice or personalized recommendation. Learned preferences can be leveraged in both active *preference elicitation* (e.g., in social choice or group decision making (Lu and Boutilier, 2011)), or in passive (purely observational) settings. It would also be interesting to apply GRIM to other posterior distributions such as energy models, and to compare it to different MCMC techniques like chain flipping (Dellaert et al., 2003).

Acknowledgments

This research was supported by the the Natural Sciences and Engineering Research Council (NSERC) of Canada. Tyler Lu was supported by a Google Ph.D. Fellowship.

References

Shivani Agarwal and Dan Roth. Learnability of bipartite ranking functions. In *Proceedings of the Eighteenth Annual Conference on Learning Theory (COLT-05)*, pages 16–31, 2005.

John Bartholdi III, Craig Tovey, and Michael Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6(2):157–165, 1989.

Christophe Biernacki and Julien Jacques. A generative model for rank data based on insertion sort algorithm. *Computational Statistics and Data Analysis*, 58:162–176, 2013.

Graham Brightwell and Peter Winkler. Counting linear extensions is #P-complete. In *ACM Symposium on the Theory of Computing*, pages 175–181, 1991.

Chris J.C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. Technical Report MSR-TR-2010-82, Microsoft Research, July 2010.

Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. In *Proceedings of the Twenty-Second International Conference on Machine Learning (ICML-05)*, pages 89–96, Bonn, Germany, 2005.

Róbert Busa-Fekete, Eyke Hüllermeier, and Balázs Szörényi. Preference-based rank elicitation using statistical models: The case of Mallows. In *Proceedings of the Thirty-first International Conference on Machine Learning (ICML-14)*, pages 1071–1079, Beijing, China, 2014.

Ludwig M. Busse, Peter Orbanz, and Joachim M. Buhmann. Cluster analysis of heterogeneous rank data. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning (ICML-07)*, pages 113–120, Corvallis, OR, 2007.

Weiwei Cheng, Krzysztof Dembczynski, and Eyke Hüllermeier. Label ranking methods based on the Plackett-Luce model. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning (ICML-10)*, pages 215–222, Haifa, Israel, 2010.

William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

marquis de (Marie Jean Antoine Nicolas de Caritat) Condorcet. *Essai sur l'Application de l'Analyse a la Probabilite des Decisions rendues a la Probabilite des Voix.* Paris: L'Imprimerie Royale, 1785.

Vincent Conitzer and Tuomas Sandholm. Common voting rules as maximum likelihood estimators. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 145–152, Edinburgh, 2005.

Vincent Conitzer, Matthew Rognlie, and Lirong Xia. Preference functions that score rankings and maximum likelihood estimation. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 109–115, Pasadena, CA, 2009.

Don Coppersmith, Lisa Fleischer, and Atri Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournaments. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-06)*, pages 776–782, Miami, FL, 2006.

Frank Dellaert, Steven M. Seitz, Charles E. Thorpe, and Sebastian Thrun. EM, MCMC, and chain flipping for structure from motion with unknown correspondence. *Machine Learning*, 50(1–2):45–71, 2003.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

Persi Diaconis. *Group Representations in Probability and Statistics*, volume 11 of *Lecture Notes–Monograph Series*. Institute of Mathematical Statistics, Hayward, CA, 1988.

Jean-Paul Doignon, Aleksandar Pekec, and Michel Regenwetter. The repeated insertion model for rankings: Missing link between two subset choice models. *Psychometrika*, 69 (1):33–54, 2004.

Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the Tenth International World Wide Web Conference (WWW-01)*, pages 613–622, Hong Kong, 2001. ACM.

M. A. Fligner and J. S. Verducci. Distance based ranking models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(3):359–369, 1986. ISSN 00359246.

Michael A. Fligner and Joseph S. Verducci, editors. *Probability Models and Statistical Analysis for Ranking Data*, volume 80 of *Lecture Notes in Statistics*. New York: Springer-Verlag, 1993.

Brian Francis, Regina Dittrich, Reinhold Hatzinger, and Les Humphreys. A mixture model for longitudinal partially ranked data. *Communications in Statistics — Theory and Methods*, 43(4):722–734, 2014.

Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

Zoubin Ghahramani and Mike I. Jordan. Learning from incomplete data. Technical report, Lab Memo No. 1509, CBCL Paper No. 108, MIT AI Lab, 1995.

Isobel Claire Gormley and Thomas Brendan Murphy. A latent space model for rank data. In Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, Anna Goldenberg, Eric P. Xing, and Alice X. Zheng, editors, *Statistical Network Analysis: Models, Issues, and New Directions*, pages 90–102. Springer, 2007.

Isobel Claire Gormley and Thomas Brendan Murphy. Exploring voting blocs within the Irish electorate. *Journal of the American Statistical Association*, 103(483):1014–1027, 2008.

John Guiver and Edward Snelson. Bayesian inference for Plackett-Luce ranking models. In *Proceedings of the Twenty-Sixth International Conference on Machine Learning (ICML-09)*, pages 377–384, Montreal, 2009. ISBN 978-1-60558-516-1.

Jonathan Huang and Carlos Guestrin. Riffled independence for ranked data. In *Advances in Neural Information Processing Systems 21*, pages 799–807, Vancouver, 2009.

Jonathan Huang, Ashish Kapoor, and Carlos Guestrin. Riffled independence for efficient inference with partial ranking. *Journal of Artificial Intelligence Research*, 44:491–532, 2012.

Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16–17):1897–1916, 2008.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.

Toshihiro Kamishima, Hideto Kazawa, and Shotaro Akaho. Supervised ordering: An empirical survey. In *IEEE International Conference on Data Mining (ICDM-05)*, pages 673–676, Houston, TX, 2005.

John G. Kemeny. Mathematics without numbers. *Daedalus*, 88(4):577–591, 1959.

Guy Lebanon and Yi Mao. Non-parametric modeling of partially ranked data. *Journal of Machine Learning Research*, 9:2401–2429, 2008.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982.

Jordan J. Louviere, David A. Hensher, and Joffre D. Swait. *Stated Choice Methods: Analysis and Application*. Cambridge University Press, Cambridge, 2000.

Tyler Lu and Craig Boutilier. Robust approximation and incremental elicitation in voting protocols. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-11)*, pages 287–293, Barcelona, 2011.

R. Duncan Luce. *Individual Choice Behavior: A Theoretical Analysis*. Wiley, 1959.

Colin L. Mallows. Non-null ranking models. In *Biometrika*, volume 44, pages 114–130, 1957.

John I. Marden. *Analyzing and Modeling Rank Data*. Chapman and Hall, London, 1995.

Benjamin M. Marlin and Richard S. Zemel. Collaborative filtering and the missing at random assumption. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence (UAI-07)*, pages 50–54, Vancouver, 2007.

Marina Meila and Harr Chen. Dirichlet process mixtures of generalized Mallows models. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 358–367, Catalina Island, CA, USA, 2010.

Thomas Brendan Murphy and Donal Martin. Mixtures of distance-based models for ranking data. *Computational Statistics and Data Analysis*, 41:645–655, January 2003.

Radford Neal and Geoff Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. Jordan, editor, *Learning in Graphical Models*, pages 355–368. MIT Press, Cambridge, MA, 1999.

R. Plackett. The analysis of permutations. *Applied Statistics*, 24:193–202, 1975.

T. Graepel R. Herbrich and K. Obermayer. Large margin rank boundaries for ordinal regression. In Peter J. Bartlett, Bernhard Schölkopf, Dale Schuurmans, and Alex J. Smola, editors, *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, 2000.

Cynthia Rudin. The p-norm push: A simple convex ranking algorithm that concentrates at the top of the list. *Journal of Machine Learning Research*, 10:2233–2271, 2009.

Grace S. Shieh. A weighted Kendall's tau statistic. *Statistics and Probability Letters*, 39 (1):17–24, 1998.

Maksims Volkovs and Richard S. Zemel. Boltzrank: Learning to maximize expected ranking gain. In *Proceedings of the Twenty-Sixth International Conference on Machine Learning (ICML-09)*, page 137, Montreal, 2009.

Maksims Volkovs and Richard S. Zemel. Efficient sampling for bipartite matching problems. In *Advances in Neural Information Processing Systems 25 (NIPS-12)*, pages 1322–1330, Lake Tahoe, NV, 2012.

Peyton Young. Optimal voting rules. *Journal of Economic Perspectives*, 9:51–64, 1995.

Journal of Machine Learning Research 15 (2014) 4011-4051 Submitted 12/13; Published 12/14

Robust Hierarchical Clustering*

Maria-Florina Balcan NINAMF@CS.CMU.EDU
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Yingyu Liang YINGYUL@CS.PRINCETON.EDU
Department of Computer Science
Princeton University
Princeton, NJ 08540, USA

Pramod Gupta PRAMODG@GOOGLE.COM
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043, USA

Editor: Sanjoy Dasgupta

Abstract

One of the most widely used techniques for data clustering is agglomerative clustering. Such algorithms have been long used across many different fields ranging from computational biology to social sciences to computer vision in part because their output is easy to interpret. Unfortunately, it is well known, however, that many of the classic agglomerative clustering algorithms are not robust to noise. In this paper we propose and analyze a new robust algorithm for bottom-up agglomerative clustering. We show that our algorithm can be used to cluster accurately in cases where the data satisfies a number of natural properties and where the traditional agglomerative algorithms fail. We also show how to adapt our algorithm to the inductive setting where our given data is only a small random sample of the entire data set. Experimental evaluations on synthetic and real world data sets show that our algorithm achieves better performance than other hierarchical algorithms in the presence of noise.

Keywords: unsupervised learning, clustering, agglomerative algorithms, robustness

1. Introduction

Many data mining and machine learning applications ranging from computer vision to biology problems have recently faced an explosion of data. As a consequence it has become increasingly important to develop effective, accurate, robust to noise, fast, and general clustering algorithms, accessible to developers and researchers in a diverse range of areas.

One of the oldest and most commonly used methods for clustering data, widely used in many scientific applications, is hierarchical clustering (Gower, 1967; Bryant and Berry, 2001; Cheng et al., 2006; Dasgupta and Long, 2005; Duda et al., 2000; Gollapudi et al., 2006; Jain and Dubes, 1981; Jain et al., 1999; Johnson, 1967; Narasimhan et al., 2006).

*. A preliminary version of this article appeared under the title *Robust Hierarchical Clustering* in the Proceedings of the Twenty-Third Conference on Learning Theory, 2010.

In hierarchical clustering the goal is not to find a single partitioning of the data, but a hierarchy (generally represented by a tree) of partitions which may reveal interesting structure in the data at multiple levels of granularity. The most widely used hierarchical methods are the agglomerative clustering techniques; most of these techniques start with a separate cluster for each point and then progressively merge the two closest clusters until only a single cluster remains. In all cases, we assume that we have a measure of similarity between pairs of objects, but the different schemes are distinguished by how they convert this into a measure of similarity between two clusters. For example, in single linkage the similarity between two clusters is the maximum similarity between points in these two different clusters. In complete linkage, the similarity between two clusters is the minimum similarity between points in these two different clusters. Average linkage defines the similarity between two clusters as the average similarity between points in these two different clusters (Dasgupta and Long, 2005; Jain et al., 1999).

Such algorithms have been used in a wide range of application domains ranging from biology to social sciences to computer vision mainly because they are quite fast and the output is quite easy to interpret. It is well known, however, that one of the main limitations of the agglomerative clustering algorithms is that they are not robust to noise (Narasimhan et al., 2006). In this paper we propose and analyze a robust algorithm for bottom-up agglomerative clustering. We show that our algorithm satisfies formal robustness guarantees and with proper parameter values, it will be successful in several natural cases where the traditional agglomerative algorithms fail.

In order to formally analyze correctness of our algorithm we use the discriminative framework (Balcan et al., 2008). In this framework, we assume there is some target clustering (much like a k-class target function in the multi-class learning setting) and we say that an algorithm correctly clusters data satisfying property P if on any data set having property P, the algorithm produces a tree such that the target is some pruning of the tree. For example if all points are more similar to points in their own target cluster than to points in any other cluster (this is called the strict separation property), then any of the standard single linkage, complete linkage, and average linkage agglomerative algorithms will succeed.[1] See Figure 1 for an example. However, with just tiny bit of noise, for example if each point has even just one point from a different cluster that it is similar to, then these standard algorithms will all fail (we elaborate on this in Section 2.2). See Figure 2 for an example. This brings up the question: is it possible to design an agglomerative algorithm that is robust to these types of situations and more generally can tolerate a substantial degree of noise? The contribution of our paper is to provide a positive answer to this question; we develop a robust, linkage based algorithm that will succeed in many interesting cases where standard agglomerative algorithms will fail. At a high level, our new algorithm is robust to noise in two different and important ways. First, it uses more global information for determining the similarities between clusters; second, it uses a robust linkage procedure involving a median test for linking the clusters, eliminating the influence of the noisy similarities.

1. We note however that the Ward's minimum variance method, another classic linkage based procedure, might fail under the strict separation property in the presence of unbalanced clusters. We provide a concrete example in Appendix C.

1.1 Our Results

In particular, in Section 3 we show that if the data satisfies a natural good neighborhood property, then our algorithm can be used to cluster well in the tree model, that is, to output a hierarchy such that the target clustering is (close to) a pruning of that hierarchy. The good neighborhood property relaxes the strict separation property, and only requires that after a small number of bad points (which could be extremely malicious) have been removed, for the remaining good points in the data set, in the neighborhood of their target cluster's size, most of their nearest neighbors are from their target cluster. We show that our algorithm produces a hierarchy with a pruning that assigns all good points correctly. In Section 4, we further generalize this to allow for a good fraction of "boundary" points that do not fully satisfy the good neighborhood property. Unlike the good points, these points may have many nearest neighbors outside their target cluster in the neighborhood of their target cluster's size; but also unlike the bad points, they have additional structure: they fall into a sufficiently large subset of their target cluster, such that all points in this subset have most of their nearest neighbors from this subset. As long as the fraction of boundary points in such subsets is not too large, our algorithm can produce a hierarchy with a pruning that assigns all good and boundary points correctly.

We further show how to adapt our algorithm to the inductive setting with formal correctness guarantees in Section 5. In this setting, the clustering algorithm only uses a small random sample over the data set and generates a hierarchy over this sample, which also implicitly represents a hierarchy over the entire data set. This is especially useful when the amount of data is enormous such as in astrophysics and biology. We prove that our algorithm requires only a small random sample whose size is independent of that of the entire data set and depends only on the noise and the confidence parameters.

We then perform experimental evaluations of our algorithm on synthetic data and real-world data sets. In controlled experiments on synthetic data presented in Section 6.1, our algorithm achieves results consistent with our theoretical analysis, outperforming several other hierarchical algorithms. We also show in Section 6.2 that our algorithm performs consistently better than other hierarchical algorithms in experiments on several real-world data. These experimental results suggest that the properties and the algorithm we propose can handle noise in real-world data as well. To obtain good performance, however, our algorithm requires tuning the noise parameters which roughly speaking quantify the extent to which the good neighborhood property is satisfied.

1.2 Related Work

In agglomerative hierarchical clustering (Dasgupta and Long, 2005; Duda et al., 2000; Jain and Dubes, 1981; Jain et al., 1999), the goal is not to find a single partitioning of the data, but a hierarchy (generally represented by a tree) of partitionings which may reveal interesting structure in the data at multiple levels of granularity. Traditionally, only clusterings at a certain level are considered, but as we argue in Section 2 it is more desirable to consider all the prunings of the tree, since this way we can then handle much more general situations.

As mentioned above, it is well known that standard agglomerative hierarchical clustering techniques are not tolerant to noise (Nagy, 1968; Narasimhan et al., 2006). Several algorithms have been proposed to make the hierarchical clustering techniques more robust

to noise, such as Wishart's method (Wishart, 1969), and CURE (Guha et al., 1998). Ward's minimum variance method (Ward, 1963) is also more preferable in the presence of noise. However, these algorithms have no theoretical guarantees for their robustness. Also, our empirical study demonstrates that our algorithm has better tolerance to noise.

On the theoretical side, the simple strict separation property discussed above is generalized to the ν-strict separation property (Balcan et al., 2008). The generalization requires that after a small number of outliers have been removed all points are strictly more similar to points in their own cluster than to points in other clusters. They provided an algorithm for producing a hierarchy such that the target clustering is close to some pruning of the tree, but via a much more computationally expensive (non-agglomerative) algorithm. Our algorithm is simpler and substantially faster. As discussed in Section 2.1, the good neighborhood property is much broader than the ν-strict separation property, so our algorithm is much more generally applicable compared to their algorithm specifically designed for ν-strict separation.

In a different statistical model, a generalization of Wishart's method is proposed Chaudhuri and Dasgupta (2010). The authors proved that given a sample from a density function, the method constructs a tree that is consistent with the cluster tree of the density. Although not directly targeting at robustness, the analysis shows the method successfully identifies salient clusters separated by low density regions, which suggests the method can be robust to the noise represented by the low density regions.

For general clustering beyond hierarchical clustering, there are also works proposing robust algorithms and analyzing robustness of the algorithms; see (García-Escudero et al., 2010) for a review. In particular, the trimmed k-means algorithm (García-Escudero and Gordaliza, 1999), a variant of the classical k-means algorithm, updates the centers after trimming points that are far away and thus are likely to be noise. An interesting mathematical probabilistic framework for clustering in the presence of outliers is introduced (Gallegos, 2002; Gallegos and Ritter, 2005), which used maximum likelihood approach to estimate the underlying parameters. An algorithm combining the above two approaches is later proposed (García-Escudero et al., 2008). The robustness of some classical algorithms such as k-means is also studied from the perspective of how the clusters are changed after adding some additional points (Hennig, 2008; Ackerman et al., 2013).

1.3 Structure of the Paper

The rest of the paper is organized as follows. In Section 2, we formalize our model and define the good neighborhood property. We describe our algorithm and prove it succeeds under the good neighborhood property in Section 3. We then prove that it also succeeds under a generalization of the good neighborhood property in Section 4. In Section 5, we show how to adapt our algorithm to the inductive setting with formal correctness guarantees. We provide the experimental results in Section 6, and conclude the paper in Section 7.

2. Definitions. A Formal Setup

We consider a clustering problem (S, ℓ) specified as follows. Assume we have a data set S of n objects. Each $x \in S$ has some (unknown) "ground-truth" label $\ell(x)$ in $Y = \{1, \ldots, k\}$, where we will think of k as much smaller than n. Let $C_i = \{x \in S : \ell(x) = i\}$ denote

the set of points of label i (which could be empty), and denote the target clustering as $\mathcal{C} = \{C_1, \ldots, C_k\}$. Let $C(x)$ be a shorthand of $C_{l(x)}$, and n_C denote the size of a cluster C.

Given another proposed clustering h, $h : S \to Y$, we define the error of h with respect to the target clustering to be

$$\mathbf{err}(h) = \min_{\sigma \in \mathcal{S}_k} \left[\Pr_{x \in S} [\sigma(h(x)) \neq \ell(x)] \right],$$

where $\mathcal{S}_k$ is the set of all permutations on $\{1, \ldots, k\}$. Equivalently, the error of a clustering $\mathcal{C}' = \{C'_1, \ldots, C'_k\}$ is $\min_{\sigma \in \mathcal{S}_k} \frac{1}{n} \sum_i |C_i - C'_{\sigma(i)}|$. This is popularly known as *Classification Error* (Meilă and Heckerman, 2001; Balcan et al., 2013; Voevodski et al., 2012).

We will be considering clustering algorithms whose only access to their data is via a pairwise similarity function $\mathcal{K}(x, x')$ that given two examples outputs a number in the range $[-1, 1]$. We will say that $\mathcal{K}$ is a symmetric similarity function if $\mathcal{K}(x, x') = \mathcal{K}(x', x)$ for all x, x'. In this paper we assume that the similarity function $\mathcal{K}$ is symmetric.

Our goal is to produce a hierarchical clustering that contains a pruning that is close to the target clustering. Formally, the goal of the algorithm is to produce a hierarchical clustering: that is, a tree on subsets such that the root is the set S, and the children of any node S' in the tree form a partition of S'. The requirement is that there must exist a *pruning* h of the tree (not necessarily using nodes all at the same level) that has error at most ϵ. It has been shown that this type of output is necessary in order to be able to analyze non-trivial properties of the similarity function (Balcan et al., 2008). For example, even if the similarity function satisfies the requirement that all points are more similar to all points in their own cluster than to any point in any other cluster (this is called the strict separation property) and even if we are told the number of clusters, there can still be multiple different clusterings that satisfy the property. In particular, one can show examples of similarity functions and two significantly different clusterings of the data satisfying the strict separation property. See Figure 1 for an example. However, under the strict separation property, there is a single hierarchical decomposition such that any consistent clustering is a pruning of this tree. This motivates clustering in the tree model, which is the model we consider in this work as well.

Given a similarity function satisfying the strict separation property (see Figure 1 for an example), we can efficiently construct a tree such that the ground-truth clustering is a pruning of this tree (Balcan et al., 2008). Moreover, the standard linkage single linkage, average linkage, and complete linkage algorithms would work well under this property. However, one can show that if the similarity function slightly deviates from the strict separation condition, then all these standard agglomerative algorithms will fail (we elaborate on this in Section 2.2). In this context, the main question we address in this work is: Can we develop other more robust, linkage based algorithms that will succeed under more realistic and yet natural conditions on the similarity function?

Note The strict separation property does not guarantee that all the cutoffs for different points x are the same, so single linkage would not necessarily have the right clustering if it just stopped once it has k clusters; however the target clustering will provably be a pruning of the final single linkage tree; this is why we define success based on prunings.

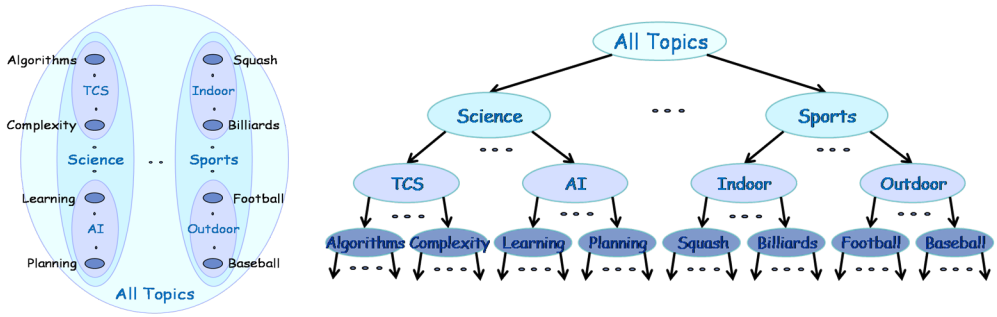


Figure 1: Consider a document clustering problem. Assume that data lies in multiple regions Algorithms, Complexity, Learning, Planning, Squash, Billiards, Football, Baseball. Suppose that the similarity $\mathcal{K}(x,y) = 0.999$ if x and y belong to the same inner region; $\mathcal{K}(x,y) = 3/4$ if $x \in$ Algorithms and $y \in$ Complexity, or if $x \in$ Learning and $y \in$ Planning, or if $x \in$ Squash and $y \in$ Billiards, or if $x \in$ Football and $y \in$ Baseball; $\mathcal{K}(x,y) = 1/2$ if x is in (Algorithms or Complexity) and y is in (Learning or Planning), or if x is in (Squash or Billiards) and y is in (Football or Baseball); define $\mathcal{K}(x,y) = 0$ otherwise. Both clusterings $\{$Algorithms $\cup$ Complexity $\cup$ Learning $\cup$ Planning, Squash $\cup$ Billiards, Football $\cup$ Baseball$\}$ and $\{$Algorithms$\cup$Complexity, Learning$\cup$Planning, Squash$\cup$Billiards$\cup$ Football $\cup$ Baseball$\}$ satisfy the strict separation property.

2.1 Properties of the Similarity Function

We describe here some natural properties of the similarity functions that we analyze in this paper. We start with a noisy version of the simple strict separation property mentioned above (Balcan et al., 2008) and then define an interesting and natural generalization of it.

Property 1 *The similarity function $\mathcal{K}$ satisfies* **ν-strict separation** *for the clustering problem (S,ℓ) if for some $S' \subseteq S$ of size $(1-\nu)n$, $\mathcal{K}$ satisfies strict separation for (S',ℓ). That is, for all $x, x', x'' \in S'$ with $x' \in C(x)$ and $x'' \notin C(x)$ we have $\mathcal{K}(x,x') > \mathcal{K}(x,x'')$.*

So, in other words we require that the strict separation is satisfied after a number of bad points have been removed. A somewhat different condition is to allow each point to have some bad immediate neighbors as long as most of its immediate neighbors are good. Formally:

Property 2 *The similarity function $\mathcal{K}$ satisfies* **α-good neighborhood** *property for the clustering problem (S,ℓ) if for all points x we have that all but αn out of their $n_{C(x)}$ nearest neighbors belong to the cluster $C(x)$.*

Note that the α-good neighborhood property is different from the ν-strict separation property. For the ν-strict separation property we can have up to νn bad points that can misbehave; in particular, these νn bad points can have similarity 1 to *all* the points in

S; however, once we remove these points the remaining points are more similar to points in their own cluster than to points in other cluster. On the other hand, for the α-good neighborhood property we require that for all points x all but αn out of their $n_{C(x)}$ nearest neighbors belong to the cluster $C(x)$. (So we cannot have a point that has similarity 1 to all the points in S.) Note however that different points might misbehave on different αn neighbors. We can also consider a property that generalizes both the ν-strict separation and the α-good neighborhood property. Specifically:

Property 3 *The similarity function $\mathcal{K}$ satisfies (α, ν)-**good neighborhood** property for the clustering problem (S, ℓ) if for some $S' \subseteq S$ of size $(1-\nu)n$, $\mathcal{K}$ satisfies α-good neighborhood for (S', ℓ). That is, for all points $x \in S'$ we have that all but αn out of their $n_{C(x)\cap S'}$ nearest neighbors in S' belong to the cluster $C(x)$.*

Clearly, the (α, ν)-good neighborhood property is a generalization of both the ν-strict separation and α-good neighborhood property. Formally,

Fact 1 *If the similarity function $\mathcal{K}$ satisfies the α-good neighborhood property for the clustering problem (S, ℓ), then $\mathcal{K}$ also satisfies the $(\alpha, 0)$-good neighborhood property for the clustering problem (S, ℓ).*

Fact 2 *If the similarity function $\mathcal{K}$ satisfies the ν-strict separation property for the clustering problem (S, ℓ), then $\mathcal{K}$ also satisfies the $(0, \nu)$-good neighborhood property for the clustering problem (S, ℓ).*

It has been shown that if $\mathcal{K}$ satisfies the ν-strict separation property with respect to the target clustering, then as long as the smallest target cluster has size $5\nu n$, one can in polynomial time construct a hierarchy such that the ground-truth is ν-close to a pruning of the hierarchy (Balcan et al., 2008). Unfortunately the algorithm presented there is computationally very expensive: it first generates a large list of $\Omega(n^2)$ candidate clusters and repeatedly runs pairwise tests in order to laminarize these clusters; its running time is a large unspecified polynomial. The new robust linkage algorithm we present in Section 3 can be used to get a simpler and much faster algorithm for clustering accurately under the ν-strict separation and the more general (α, ν)-good neighborhood property.

Generalizations Our algorithm succeeds under an even more general property called weak good neighborhood, which allows a good fraction of points to only have nice structure in their small local neighborhoods. The relations between these properties are described in Section 4.1, and the analysis under the weak good neighborhood is presented in Section 4.2.

2.2 Standard Linkage Based Algorithms Are Not Robust

As we show below, even if the data satisfies the good neighborhood property, the standard single linkage, average linkage, and complete linkage algorithms might fail. The contribution of our work is to develop a robust, linkage based algorithm that will succeed under these natural conditions. More specifically, we can show an example where the standard single linkage, average linkage, and complete linkage algorithms would perform very badly, but where our algorithm would work well. In particular, let us slightly modify the example in Figure 1, by adding a little bit of noise, to form links of high similarity between points

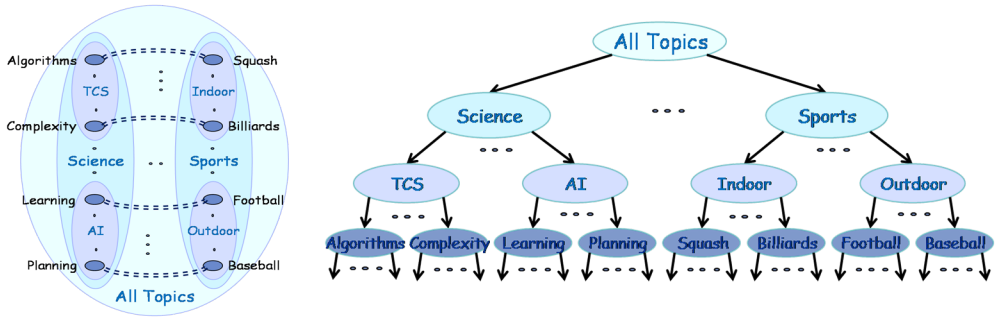


Figure 2: Same as Figure 1 except that let us match each point in Algorithms with a point in Squash, each point in Complexity with a point in Billiards, each point in Learning with a point in Football, and each point in Planning with a point in region Baseball. Define the similarities to be the same as in Figure 1 except that we let $\mathcal{K}(x, y) = 1$ if x and y are matched. Note that for $\alpha = 1/n$ the similarity function satisfies the α-good neighborhood with respect to any of the prunings of the tree above. However, single linkage, average linkage, and complete linkage would initially link the matched pairs and produce clusters with very high error with respect to any such clustering.

in different inner blobs.[2] See Figure 2 for a precise description of the similarity. In this example all the single linkage, average linkage, and complete linkage algorithms would in the first $n/2$ stages merge the matched pairs of points. From that moment on, no matter how they perform, none of the natural and desired clusterings will even be $1/2$ close to any of the prunings of the hierarchy produced. Notice however, that $\mathcal{K}$ satisfies the α-good neighborhood with respect to any of the desired clusterings (for $\alpha = 1/n$), and that our algorithm will be successful on this instance. The ν-strict separation is not satisfied in this example either, for any constant ν.

3. Robust Median Neighborhood Linkage

In this section, we propose a new algorithm, Robust Median Neighborhood Linkage, and show that it succeeds for instances satisfying the (α, ν)-good neighborhood property.

Informally, the algorithm maintains a threshold t and a list $\mathcal{C}'_t$ of subsets of points of S; these subsets are called blobs for convenience. We first initialize the threshold to a value t that is not too large and not too small ($t = 6(\alpha + \nu)n + 1$), and initialize $\mathcal{C}'_{t-1}$ to

2. Since, usually, the similarity function between pairs of objects is constructed based on heuristics, this could happen in practice; for example we could have a similarity measure that puts a lot of weight on features such as date or names, and so we could easily have a document about Learning being more similar to a document about Football than to other documents about Learning. While this example seems a little bit contrived, in Figure 7 in Section 4 we will give a naturally occurring example where the standard single linkage, average linkage, and complete linkage algorithms still fail but our algorithm succeeds because it satisfies a generalization of the good neighborhood property that we will discuss in Section 4.

Algorithm 1 Robust Median Neighborhood Linkage

Input: Similarity function $\mathcal{K}$ on a set of points S, $n = |S|$, noise parameters $\alpha > 0, \nu > 0$.

Step 1 Initialize $t = 6(\alpha + \nu)n + 1$.
Initialize $\mathcal{C}'_{t-1}$ to be a list of blobs so that each point is in its own blob.
while $|\mathcal{C}'_{t-1}| > 1$ **do**
Step 2 ▷ `Build a graph` F_t `whose vertices are points in` S `and whose edges are specified as follows.`
Let $N_t(x)$ denote the t nearest neighbors of x.
for any $x, y \in S$ that satisfy $|N_t(x) \cap N_t(y)| \geq t - 2(\alpha + \nu)n$ **do**
Connect x, y in F_t.
end for
Step 3 ▷ `Build a graph` H_t `whose vertices are blobs in` $\mathcal{C}'_{t-1}$ `and whose edges are specified as follows.`
Let $N_F(x)$ denote the neighbors of x in F_t.
for any $C_u, C_v \in \mathcal{C}'_{t-1}$ **do**
if C_u, C_v are singleton blobs, i.e., $C_u = \{x\}, C_v = \{y\}$ **then**
Connect C_u, C_v in H_t, if $|N_F(x) \cap N_F(y)| > (\alpha + \nu)n$.
else
Set $S_t(x, y) = |N_F(x) \cap N_F(y) \cap (C_u \cup C_v)|$, i.e., the number of points in $C_u \cup C_v$ that are common neighbors of x, y in F_t.
Connect C_u, C_v in H_t, if $\text{median}_{x \in C_u, y \in C_v} S_t(x, y) > \frac{|C_u| + |C_v|}{4}$.
end if
end for
Step 4 ▷ `Merge blobs in` $\mathcal{C}'_{t-1}$ `to get` $\mathcal{C}'_t$
Set $\mathcal{C}'_t = \mathcal{C}'_{t-1}$.
for any connected component V in H_t with $|\bigcup_{C \in V} C| \geq 4(\alpha + \nu)n$ **do**
Update $\mathcal{C}'_t$ by merging all blobs in V into one blob.
end for
Step 5 ▷ `Increase threshold`
$t = t + 1$.
end while

Output: Tree T with single points as leaves and internal nodes corresponding to the merges performed.

contain $|S|$ blobs, one for each point in S. For each t, the algorithm builds $\mathcal{C}'_t$ from $\mathcal{C}'_{t-1}$ by merging two or more blobs as follows. It first builds a graph F_t, whose vertices are the data points in S and whose edges are constructed by connecting any two points that share at least $t - 2(\alpha + \nu)n$ points in common out of their t nearest neighbors. Then it builds a graph H_t whose vertices correspond to blobs in $\mathcal{C}'_{t-1}$ and whose edges are specified in the following way. Two singleton blobs $C_u = \{x\}$ and $C_v = \{y\}$ are connected in H_t if the points x, y have more than $(\alpha + \nu)n$ common neighbors in F_t. For blobs C_u and C_v that are not both singleton, the algorithm performs a median test. In this test, for each

pair of points $x \in C_u, y \in C_v$, it computes the number $S_t(x,y)$ of points $z \in C_u \cup C_v$ that are the common neighbors of x and y in F_t. It then connects C_u and C_v in H_t if $\text{median}_{x \in C_u, y \in C_v} S_t(x,y)$ is larger than $1/4$ fraction of $|C_u| + |C_v|$. Once H_t is built, we analyze its connected components in order to create $\mathcal{C}'_t$. For each connected component V of H_t, if V contains sufficiently many points from S in its blobs we merge all its blobs into one blob in $\mathcal{C}'_t$. After building $\mathcal{C}'_t$, the threshold is increased and the above steps are repeated until only one blob is left. The algorithm finally outputs the tree with single points as leaves and internal nodes corresponding to the merges performed. The full details of our algorithm are described in Algorithm 1. Our main result in this section is the following:

Theorem 1 *Let $\mathcal{K}$ be a symmetric similarity function satisfying the (α, ν)-good neighborhood for the clustering problem (S, ℓ). As long as the smallest target cluster has size greater than $6(\nu + \alpha)n$, Algorithm 1 outputs a hierarchy such that a pruning of the hierarchy is ν-close to the target clustering in time $O(n^{\omega+1})$, where $O(n^{\omega})$ is the state of the art for matrix multiplication.*

In the rest of this section, we will first describe the intuition behind the algorithm in Section 3.1 and then prove Theorem 1 in Section 3.2.

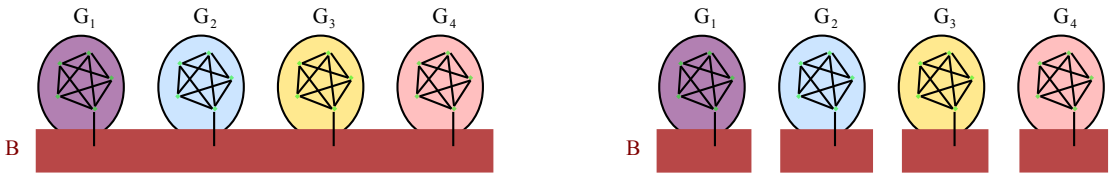
3.1 Intuition of the Algorithm under the Good Neighborhood Property

We begin with some convenient terminology and a simple fact about the good neighborhood property. In the definition of the (α, ν)-good neighborhood property (see Property 3), we call the points in S' good points and the points in $B = S \setminus S'$ bad points. Let $G_i = C_i \cap S'$ be the good set of label i. Let $G = \cup_i G_i$ denote the whole set of good points; so $G = S'$. Clearly $|G| \geq n - \nu n$. Recall that n_{C_i} is the number of points in the cluster C_i. Note that the following is a useful consequence of the (α, ν)-good neighborhood property.

Fact 3 *Suppose the similarity function $\mathcal{K}$ satisfies the (α, ν)-good neighborhood property for the clustering problem (S, ℓ). As long as t is smaller than n_{C_i}, for any good point $x \in C_i$, all but at most $(\alpha + \nu)n$ out of its t nearest neighbors lie in its good set G_i.*

Proof Let $x \in G_i$. By definition, out of its $|G_i|$ nearest neighbors in G, there are at least $|G_i| - \alpha n$ points from G_i. These points must be among its $|G_i| + \nu n$ nearest neighbors in S, since there are at most νn bad points in $S \setminus G$. This means that at most $(\alpha + \nu)n$ out of its $|G_i| + \nu n$ nearest neighbors are outside G_i. Notice $|G_i| + \nu n \geq n_{C_i}$, we have that at most $(\alpha + \nu)n$ out of its n_{C_i} nearest neighbors are outside G_i, as desired. ■

Intuition We first assume for simplicity that all the target clusters have the same size n_C and that we know n_C. In this case it is quite easy to recover the target clusters as follows. We first construct a graph F whose vertices are points in S; we connect two points in F if they share at least $n_C - 2(\nu + \alpha)n$ points in common among their n_C nearest neighbors. By Fact 3, if the target clusters are not too small (namely $n_C > 6(\nu + \alpha)n$), we are guaranteed that no two good points in different target clusters will be connected in F, and that all good points in the same target cluster will be connected in F. If there are no bad points ($\nu = 0$), then each connected component of F corresponds to a target cluster, and we could



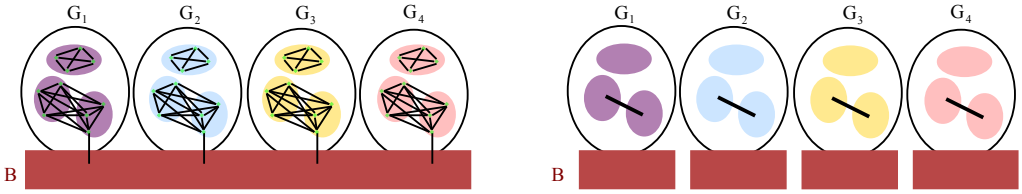
(a) F (b) H

Figure 3: Graph F and H when all target clusters are of the same size n_C, which is known. In F, no two good points in different target clusters can be connected, and all good points in the same target cluster will be connected. In H, bad points connected to good points from different target clusters are disconnected.

simply output them. Alternatively, if there are bad points ($\nu > 0$), we can still cluster well as follows. We construct a new graph H on points in S by connecting points that share more than $(\alpha + \nu)n$ neighbors in the graph F. The key point is that in F a bad point can be connected to good points from only one single target cluster. This then ensures that no good points from different target clusters are in the same connected component in H. So, if we output the largest k components of H, we will obtain a clustering with error at most νn. See Figure 3 for an illustration.

If we do not know n_C, we can still use a pretty simple procedure. Specifically, we start with a threshold $t \leq n_C$ that is not too small and not too large (say $6(\nu + \alpha)n < t \leq n_C$), and build a graph F_t on S by connecting two points if they share at least $t - 2(\nu + \alpha)n$ points in common out of their t nearest neighbors. We then build another graph H_t on S by connecting points if they share more than $(\alpha+\nu)n$ neighbors in the graph F_t. The key idea is that when $t \leq n_C$, good points from different target clusters share less than $t-2(\nu+\alpha)n$ neighbors, and thus are not connected in F_t and H_t. If the k largest connected components of H_t all have sizes greater than $(\alpha + \nu)n$ and they cover at least a $(1 - \nu)$ fraction of the whole set of points S, then these components must correspond to the target clusters and we can output them. Otherwise, we increase the critical threshold and repeat. By the time we reach n_C, all good points in the same target clusters will get connected in F_t and H_t, so we can identify the k largest components as the target clusters.

Note that as mentioned above, when $t \leq n_C$, each connected component in H_t can contain good points from only one target cluster. An alternative procedure is to reuse this information in later thresholds, so that we do not need to build the graph H_t from scratch as described in the above paragraph. Specifically, we maintain a list $\mathcal{C}'_t$ of subsets of points; these subsets are called blobs for convenience. We start with a list where each blob contains a single point. At each threshold t, we build F_t on the points in S as before, but build H_t on the blobs in $\mathcal{C}'_{t-1}$ (instead of on the points in S). When building H_t, for two singleton blobs, it is safe to connect them if their points share enough neighbors in F_t. For non-singleton



(a) F_t (b) H_t

Figure 4: Graph F_t and H_t when target clusters have the same size n_C but we do not know n_C. The figure shows the case when $t < n_C$. In F_t, no good points are connected with good points outside their target clusters; in H_t, blobs containing good points in different target clusters are disconnected.

blobs, it turns out that we can use a median test to outvote the noise.[3] In particular, for two blobs C_u and C_v that are not both singleton, we compute for all $x \in C_u$ and $y \in C_v$ the quantity $S_t(x, y)$, which is the number of points in $C_u \cup C_v$ that are the common neighbors of x and y in F_t. We then connect the two blobs in H_t if $\text{median}_{x \in C_u, y \in C_v} S_t(x, y)$ is sufficiently large. See Figure 4 for an illustration and see Step 3 in Algorithm 1 for the details.

In the general case where the sizes of the target clusters are different, similar ideas can be applied. The key point is that when $t \leq n_{C_i}$, good points from C_i share less than $t - 2(\nu + \alpha)n$ neighbors with good points outside, and thus are not connected to them in F_t. Then in H_t, we can make sure that no blobs containing good points in C_i will be connected with blobs containing good points outside C_i. When $t = n_{C_i}$, good points in C_i form a clique in F_t, then all the blobs containing good points in C_i are connected in H_t, and thus are merged. See Figure 5 for an illustration. Full details are presented in Algorithm 1 and the proof of Theorem 1 in the following subsection.

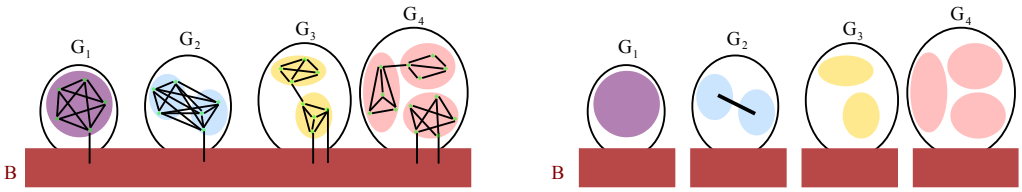
3.2 Correctness under the Good Neighborhood Property

In this subsection, we prove Theorem 1 for our algorithm. The correctness follows from Lemma 2 and the running time follows from Lemma 3. Before proving these lemmas, we begin with a useful fact which follows immediately from the design of the algorithm.

Fact 4 *In Algorithm 1, for any t, if a blob in $\mathcal{C}_t$ contains at least one good point, then at least $3/4$ fraction of the points in that blob are good points.*

Proof This is clearly true when the blob is singleton. When it is non-singleton, it must be formed in Step 4 in Algorithm 1, so it contains at least $4(\alpha + \nu)n$ points. Then the claim follows since there are at most νn bad points. ■

3. The median test is quite robust and as we show, it allows some points in these blobs to have weaker properties than the good neighborhood. See Section 4 for examples of such points and a theoretical analysis of the robustness.



(a) F_t (b) H_t

Figure 5: Graph F_t and H_t when target clusters are of different sizes. The figure shows the case when $t = n_{C_2}$. In F_t, no good points are connected with good points outside their target clusters; good points in C_2 form a clique since $t = n_{C_2}$. In H_t, blobs containing good points in different target clusters are disconnected; blobs containing good points in C_2 are all connected.

Lemma 2 *The following claims are true in Algorithm 1:*
(1) For any C_i such that $t \leq |C_i|$, any blob in $\mathcal{C}'_t$ containing good points from C_i will not contain good points outside C_i.
(2) For any C_i such that $t = |C_i|$, all good points in C_i belong to one blob in $\mathcal{C}'_t$.

Proof Before proving the claims, we first show that the graph F_t constructed in Step 2 has the following useful properties. Recall that F_t is constructed on points in S by connecting any two points that share at least $t - 2(\alpha + \nu)n$ points in common out of their t nearest neighbors. For any C_i such that $t \leq |C_i|$, we have:

(a) If x is a good point in C_i and y is a good point outside C_i, then x and y are not connected in F_t.

To see this, first note that by Fact 3, x has at most $(\alpha+\nu)n$ neighbors outside C_i out of the t nearest neighbors. For $y \in G_j$, if $n_{C_j} \geq t$, then y has at most $(\alpha+\nu)n$ neighbors in C_i; if $n_{C_j} < t$, y has at most $(\alpha+\nu)n + t - n_{C_j}$ neighbors in C_i. In both cases, y has at most $(\alpha+\nu)n + \max(0, t - n_{C_j}) < t - 5(\alpha+\nu)n$ neighbors in C_i, since $n_{C_j} > 6(\alpha+\nu)n$ and $t > 6(\alpha+\nu)n$. Then x and y have at most $t - 4(\alpha+\nu)n$ common neighbors, so they are not connected in F_t.

(b) If x is a good point in C_i, y is a good point outside C_i, and z is a bad point, then z cannot be connected to both x and y in F_t.

To prove this, we will show that if z is connected to x, then z cannot be connected to y. First, by the same argument as above, out of the t nearest neighbors, y has less than $t - 5(\alpha + \nu)n$ neighbors in C_i. Second, by Fact 3, x has at most $(\alpha + \nu)n$ neighbors outside C_i. If z has less than $t - 3(\alpha + \nu)n$ neighbors in C_i, then z and x share less than $t - 3(\alpha + \nu)n + (\alpha + \nu)n = t - 2(\alpha + \nu)n$ neighbors and will not be connected. So z must have at least $t - 3(\alpha + \nu)n$ neighbors in C_i, and thus cannot have more than $3(\alpha+\nu)n$ neighbors outside C_i. The two statements show that y and z share less than $t - 5(\alpha + \nu)n$ neighbors in C_i, and at most $3(\alpha + \nu)n$ neighbors

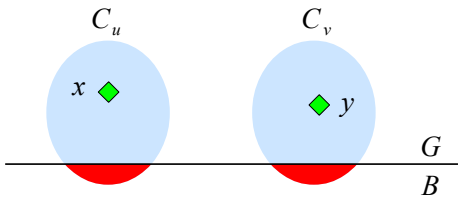


Figure 6: Illustration of the median test on C_u and C_v. At least 3/4 fraction of the points in C_u and C_v are good points, and thus more than half of the pairs (x, y) with $x \in C_u$ and $y \in C_v$ are pairs of good points.

outside C_i. So they share less than $t-2(\alpha+\nu)n+3(\alpha+\nu)n = t-2(\alpha+\nu)n$ neighbors and thus are not connected in F_t.

Now we prove Claim (1) in the lemma by induction on t. The claim is clearly true initially. Assume for induction that the claim is true for the threshold $t-1 < |C_i|$, that is, for any C_i such that $t-1 < |C_i|$, any blob in $\mathcal{C}'_{t-1}$ containing good points from C_i will not contain good points outside C_i. We now prove that the graph H_t constructed in Step 3 has the following properties, which can be used to show that the claim is still true for the threshold t.

- If $C_u \in \mathcal{C}'_{t-1}$ contains good points from C_i and $C_v \in \mathcal{C}'_{t-1}$ contains good points outside C_i, then they cannot be connected in H_t.

  If both C_u and C_v are singleton blobs, say $C_u = \{x\}, C_v = \{y\}$, then by Property (a) of F_t, the common neighbors of x and y can only be bad points, and thus C_u and C_v cannot be connected.
  If one of the two blobs (say C_u) is a singleton blob and the other is not, then C_u contains only one good point, and by Fact 4, at least 3/4 fraction of the points in C_v are good points. If both C_u and C_v are non-singleton blobs, then by Fact 4, at least 3/4 fraction of the points in C_u and C_v are good points. Therefore, in both cases, the number of pairs (x, y) with good points $x \in C_u$ and $y \in C_v$ is at least $\frac{3}{4}|C_u| \times \frac{3}{4}|C_v| > \frac{|C_u||C_v|}{2}$. That is, more than half of the pairs (x, y) with $x \in C_u$ and $y \in C_v$ are pairs of good points; see Figure 6 for an illustration. This means there exist good points $x^* \in C_u, y^* \in C_v$ such that $S_t(x^*, y^*)$ is no less than $\text{median}_{x \in C_u, y \in C_v} S_t(x, y)$. By the induction assumption, x^* is a good point in C_i and y^* is a good point outside C_i. Then by Property (a)(b) of F_t, x^* and y^* have no common neighbors in F_t, and thus $\text{median}_{x \in C_u, y \in C_v} S_t(x, y) = 0$. Therefore, C_u and C_v are not connected in H_t.

- If $C_u \in \mathcal{C}'_{t-1}$ contains good points from C_i, $C_v \in \mathcal{C}'_{t-1}$ contains good points outside C_i, and $C_w \in \mathcal{C}'_{t-1}$ contains only bad points, then C_w cannot be connected to both C_u and C_v in H_t.

  To prove this, assume for contradiction that C_w is connected to both C_u and C_v. First, note the following fact about C_w. Since any non-singleton blob must be formed

in Step 4 in the algorithm and contain at least $4(\alpha+\nu)n$ points and thus cannot contain only bad points, C_w must be a singleton blob, containing only a bad point z. Next, we show that if $C_w = \{z\}$ is connected to C_u, then z must be connected to some good point in C_i in F_t. If C_u is a singleton blob, say $C_u = \{x\}$, then by Step 4 in the algorithm, z and x share more than $(\alpha+\nu)n$ common neighbors in F_t. By Property (a)(b) of F_t, the common neighbors of x and z in F_t can only be good points in C_i or bad points. Since there are at most νn bad points, z must be connected to some good point in C_i in F_t. If C_u is not a singleton blob, then by Step 4 in the algorithm, $\text{median}_{x\in C_u} S_t(x,z) > (|C_u|+|C_w|)/4$. By Fact 4, at least 3/4 fraction of the points in C_u are good points. So there exists a good point $x^* \in C_u$ such that $S_t(x^*,z) \geq \text{median}_{x\in C_u} S_t(x,z)$, which leads to $S_t(x^*,z) > (|C_u|+|C_w|)/4 > \nu n$. By the induction assumption, x^* is a good point in C_i. Then by Property (a) of F_t, x^* is only connected to good points from C_i and bad points. Since $S_t(x^*,z) > \nu n$, z and x^* must share some common neighbors from C_i. Therefore, z is connected to some good point in C_i in F_t.
Similarly, if $C_w = \{z\}$ is connected to C_v, z must be connected to some good point outside C_i in F_t. But then z is connected to both a good point in C_i and a good point outside F_t, which contradicts Property (b) of F_t.

By the properties of H_t, no connected component contains both good points in C_i and good points outside C_i. So Claim (1) is still true for the threshold t. By induction, it is true for all thresholds.

Finally, we prove Claim (2). First, at the threshold $t = |C_i|$, all good points in C_i are connected in F_t. This is because any good point in C_i has at most $(\alpha+\nu)n$ neighbors outside C_i, so when $t = |C_i|$, any two good points in C_i share at least $t - 2(\alpha+\nu)n$ common neighbors and thus are connected in F_t.

Second, all blobs in $\mathcal{C}'_{t-1}$ containing good points in C_i are connected in H_t. There are two cases.

- If no good points in C_i have been merged, then all singleton blobs containing good points in C_i will be connected in H_t.

  This is because all good points in C_i are connected in F_t, and thus they share at least $|G_i| \geq 6(\alpha+\nu)n - \nu n$ points as common neighbors in F_t.

- If some good points in C_i have already been merged into non-singleton blobs in $\mathcal{C}'_{t-1}$, we can show that in H_t these non-singleton blobs will be connected to each other and connected to singleton blobs containing good points from C_i.

  Consider two non-singleton blobs C_u and C_v that contain good points from C_i. By Fact 4, at least 3/4 fraction of the points in C_u and C_v are good points. So there exist good points $x^* \in C_u$ and $y^* \in C_v$ such that $S_t(x^*,y^*) \leq \text{median}_{x\in C_u, y\in C_v} S_t(x,y)$. By Claim (1), x^* and y^* must be good points in C_i. Then they are connected to all good points in C_i in F_t, and thus $S_t(x^*,y^*)$ is no less than the number of good points in C_u and C_v, which is at least $3(|C_u|+|C_v|)/4$. Now we have $\text{median}_{x\in C_u, y\in C_v} S_t(x,y) \geq S_t(x^*,y^*) \geq 3(|C_u|+|C_v|)/4 > (|C_u|+|C_v|)/4$, and thus C_u, C_v are connected in H_t. Consider a non-singleton blob C_u and a singleton blob C_v that contain good points from C_i. The above argument also holds, so C_u, C_v are connected in H_t.

Therefore, in both cases, all blobs in $\mathcal{C}'_{t-1}$ containing good points in C_i are connected in H_t. Then in Step 4, all good points in C_i are merged into a blob in $\mathcal{C}'_t$. ■

Lemma 3 *Algorithm 1 has a running time of* $O(n^{\omega+1})$.

Proof The initializations in Step 1 take $O(n)$ time. To compute F_t in Step 2, for any $x \in S$, let $I_t(x,y) = 1$ if y is within the t nearest neighbors of x, and let $I_t(x,y) = 0$ otherwise. Initializing I_t takes $O(n^2)$ time. Next we compute $N_t(x,y)$, the number of common neighbors between x and y. Notice that $N_t(x,y) = \sum_{z\in S} I_t(x,z)I_t(y,z)$, so $N_t = I_t I_t^T$. Then we can compute the adjacency matrix F_t (overloading notation for the graph F_t) from N_t. These steps take $O(n^\omega)$ time.

To compute the graph H_t in Step 3, first define $NS_t = F_t(F_t)^T$. Then for two points x and y, $NS_t(x,y)$ is the number of their common neighbors in F_t. Further define a matrix FC_t as follows: if x and y are connected in F_t and are in the same blob in $\mathcal{C}'_{t-1}$, then let $FC_t(x,y) = 1$; otherwise, let $FC_t(x,y) = 0$. As a reminder, for two points x that belongs to $C_u \in \mathcal{C}'_{t-1}$ and y that belongs to $C_v \in \mathcal{C}'_{t-1}$, $S_t(x,y)$ is the number of points in $C_u \cup C_v$ they share as neighbors in common in F_t. FC_t is useful for computing S_t: since for $x \in C_u$ and $y \in C_v$,

$$\begin{aligned} S_t(x,y) &= \sum_{z\in C_v} F_t(x,z)F_t(y,z) + \sum_{z\in C_u} F_t(x,z)F_t(y,z) \\ &= \sum_{z\in S} F_t(x,z)FC_t(y,z) + \sum_{z\in S} FC_t(x,z)F_t(y,z), \end{aligned}$$

we have $S_t = F_t(FC_t)^T + FC_t(F_t)^T$. Based on NS_t and S_t, we can then build the graph H_t. All these steps take $O(n^\omega)$ time.

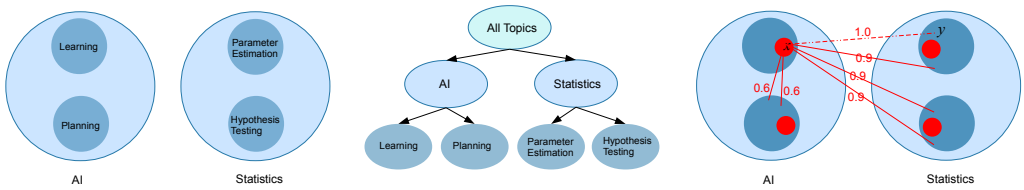
When we perform merges in Step 4 or increase the threshold in Step 5, we need to recompute the above data structures, which takes $O(n^\omega)$ time. Since there are $O(n)$ merges and $O(n)$ thresholds, Algorithm 1 takes time $O(n^{\omega+1})$ in total. ■

4. A More General Property: Weak Good Neighborhood

In this section we introduce a weaker notion of good neighborhood property and prove that our algorithm also succeeds for data satisfying this weaker property.

To motivate the property, consider a point x with the following neighborhood structure. In the neighborhood of size $n_{C(x)}$, x has a significant amount of its neighbors from other target clusters. However, in a smaller, more local neighborhood, x has most of its nearest neighbors from its target clusters $C(x)$. In practice, points close to the boundaries between different target clusters typically have such neighborhood structure; for this reason, points with such neighborhood are called boundary points.

We present an example in Figure 7. A document close to the boundary between the two fields AI and Statistics has the following neighborhood structure: out of its $n/4$ nearest neighbors, it has all its neighbors from its own field; but out of its $n/2$ nearest neighbors, it has $n/4$ neighbors outside its field. With $1/8$ fraction of such boundary



(a) (b) (c)

Figure 7: Consider a document clustering problem. Assume that there are $n/4$ documents in each of the four areas: Learning, Planning, ParameterEstimation and HypothesisTesting. The first two belong to the field AI, and the last two belong to the field Statistics. The similarities are specified as follows. (1) $\mathcal{K}(x, y) = 0.99$ if x, y belong to the same area; (2) $\mathcal{K}(x, y) = 0.8$ if x, y belong to different areas in the same field; (3) $\mathcal{K}(x, y) = 0.5$ if x, y belong to different fields. As shown in (b), there are four prunings: $\{\text{Learning}, \text{Planning}, \text{ParameterEstimation}, \text{HypothesisTesting}\}$, $\{\text{AI}, \text{ParameterEstimation}, \text{HypothesisTesting}\}$, $\{\text{Learning}, \text{Planning}, \text{Statistics}\}$, and $\{\text{AI}, \text{Statistics}\}$. All these four prunings satisfy the strict separation property, and consequently satisfy the α-good neighborhood property for $\alpha = 0$. However, this is no longer true if we take into account noise that naturally arises in practice. As shown in (c), in each area, $1/8$ fraction of the documents lie close to the boundary between the two fields. More precisely, the similarities for these boundary documents are defined as follows. (1) These documents are very similar to some document in the other field: for each boundary document x, we randomly pick one document y in the other field and set $\mathcal{K}(x, y) = \mathcal{K}(y, x) = 1.0$; (2) These documents are also closely related to the other documents in the other field: $\mathcal{K}(x, y) = \mathcal{K}(y, x) = 0.9$ when x is a boundary document and y belongs to the other field; (3) These documents are not close to those in the other area in the same field: $\mathcal{K}(x, y) = \mathcal{K}(y, x) = 0.6$ when x is a boundary document and y belongs to the other area in the same field. Then the clustering $\{\text{AI}, \text{Statistics}\}$ satisfies the (α, ν)-good neighbor property only for $\alpha \geq 1/4$ or $\nu \geq 1/8$. Similarly, $\{\text{AI}, \text{ParameterEstimation}, \text{HypothesisTesting}\}$ and $\{\text{Learning}, \text{Planning}, \text{Statistics}\}$ satisfy the property only for $\alpha \geq 1/4$ or $\nu \geq 1/16$. See the text for more details, and see Section 6.1 for simulations of this example and its variants.

points, the clustering $\{\text{AI}, \text{Statistics}\}$ satisfies the (α, ν)-good neighbor property only for $\alpha \geq 1/4$ or $\nu \geq 1/8$. This is because either we view all the boundary points as bad points in the (α, ν)-good neighborhood property which leads to $\nu \geq 1/8$, or we need $\alpha \geq 1/4$ since a boundary point has $n/4$ neighbors outside its target cluster. Similarly, $\{\text{AI}, \text{ParameterEstimation}, \text{HypothesisTesting}\}$ and $\{\text{Learning}, \text{Planning}, \text{Statistics}\}$ satisfy the property only for $\alpha \geq 1/4$ or $\nu \geq 1/16$.

For this example, either α is too large so that Theorem 1 is not applicable, or ν is too large so that the guarantee in Theorem 1 leads to constant error rate. However, it turns out that our algorithm can still successfully produce a hierarchy as in Figure 7(b) where

the desired clusterings ({Learning, Planning, ParameterEstimation, HypothesisTesting}, {AI, ParameterEstimation, HypothesisTesting}, {Learning, Planning, Statistics}, and {AI, Statistics}) are prunings of the hierarchy. As we show, the reason is that each of these prunings satisfies a generalization of the good neighborhood property which takes into account the boundary points, and for which our algorithm still succeeds. Note that the standard linkage algorithms fail on this example.[4] In the following, we first formalize this property and discuss how it relates to the properties of the similarity function described in the paper so far. We then prove that our algorithm succeeds under this property, correctly clustering all points that are not adversarially bad.

For clarity, we first relax the α-good neighborhood to the weak (α, β)-good neighborhood defined as follows.

Property 4 *A similarity function $\mathcal{K}$ satisfies* **weak (α, β)-good neighborhood** *property for the clustering problem (S, ℓ), if for each $p \in S$, there exists $A_p \subseteq C(p)$ of size greater than $6\alpha n$ such that $p \in A_p$ and*

- *any point in A_p has at most αn neighbors outside A_p out of the $|A_p|$ nearest neighbors;*
- *for any such subset $A_p \subseteq C(p)$, at least β fraction of points in A_p have all but at most αn nearest neighbors from $C(p)$ out of their $n_{C(p)}$ nearest neighbors.*

Informally, the first condition implies that every point falls into a sufficiently large subset of its target cluster, and points in the subset are close to each other in the sense that most of their nearest neighbors are in the subset. This condition is about the local neighborhood structure of the points. It shows that each point has a local neighborhood in which points closely relate to each other. Note that the local neighborhood should be large enough so that the membership of the point is clearly established: it should have size comparable to the number of connections to points outside (αn). Here we choose a minimum size of greater than $6\alpha n$ mainly because it guarantees that our algorithm can still succeed in the worst case. The second condition implies that for points in these large enough subsets, a majority of them have most of their nearest neighbors from their target cluster. This condition is about more global neighborhood structure. It shows how the subsets are closely related to those in the same target cluster in the neighborhood of size equal to the target cluster size. Note that in this more global neighborhood, we do not require all points in these subsets have most nearest neighbors from their target clusters; we allow the presence of $(1 - \beta)$ fraction of points that may have a significant number of nearest neighbors outside their target clusters.

Naturally, as we can relax the α-good neighborhood property to the (α, ν)-good neighborhood property, we can relax the weak (α, β)-good neighborhood to the weak (α, β, ν)-good neighborhood as follows. Informally, it implies that the target clustering satisfies the weak (α, β)-good neighborhood property after removing a few bad points.

4. For any fixed non-boundary point y and fixed boundary point x in the other field, the probability that y has similarity 1.0 only with x is $\frac{2}{n}(1 - \frac{2}{n})^{n/16-1} \approx \frac{2}{n}e^{-1/8}$. Since there are $n/16$ such boundary points x and $7n/8$ such non-boundary points y, when n is sufficiently large, with high probability $n/12$ non-boundary points have similarity 1.0 with one single boundary point. Then the standard linkage algorithms (in particular, single linkage, average linkage, and complete linkage) would first merge these pairs of points with similarity 1.0. From then on, no matter how they perform, any pruning of the hierarchy produced will have error higher than 1/12.

Property 5 *A similarity function $\mathcal{K}$ satisfies* **weak (α, β, ν)-good neighborhood** *property for the clustering problem (S, ℓ), if there exist a subset of points B of size at most νn, and for each $p \in S \setminus B$, there exists $A_p \subseteq C(p) \setminus B$ of size greater than $6(\alpha + \nu)n$ such that $p \in A_p$ and*

- *any point in A_p has at most αn neighbors outside A_p out of the $|A_p|$ nearest neighbors;*
- *for any such subset $A_p \subseteq C_i \setminus B$, at least β fraction of points in A_p have all but at most αn nearest neighbors from $C_i \setminus B$ out of their $|C_i \setminus B|$ nearest neighbors in $S \setminus B$.*

For convenience, we call points in B bad points. If a point in $C_i \setminus B$ has all but at most αn nearest neighbors from $C_i \setminus B$ out of its $|C_i \setminus B|$ nearest neighbors in $S \setminus B$, we call it a good point. Then the second condition in the definition can be simply stated as: any A_p has at least β fraction of good points. Note that C_i can contain points that are neither bad nor good. Such points are called boundary points, since in practice such points typically lie close to the boundaries between target clusters.

As a concrete example, consider the clustering $\{\text{AI}, \text{Statistics}\}$ in Figure 7(c). It satisfies the weak (α, β, ν)-good neighborhood property with probability at least $1 - \delta$ when the number of points $n = O(\ln \frac{1}{\delta})$. To see this, first note that for a fixed point y and a fixed boundary point x in the other field, the probability that $\mathcal{K}(y, x) = 1$ is $2/n$. Since there are $n/16$ boundary point in the other field, by Hoeffding bound, the probability that y has similarity 1 with more than $n/32$ points is bounded by $\exp\{-2 \cdot n/16 \cdot (1/2)^2\} = \exp\{-n/32\}$. By union bound, with probability at least $1 - n \exp\{-n/32\}$, no point has similarity 1 with more than $n/32$ points. Then by setting A_p as the area that p falls in, we can see that the clustering satisfies the weak (α, β, ν)-good neighborhood property for $\alpha = 1/32, \beta = 7/8$ and $\nu = 0$. Note that there may also be some adversarial bad points. Then the weak (α, β, ν)-good neighborhood property is satisfied when $\alpha = 1/32, \beta = 7/8$ and ν is the fraction of bad points. See Section 6.1 for simulations of this example and its variants.

4.1 Relating Different Versions of Good Neighborhood Properties

The relations between these properties are illustrated in Figure 8. The relations between the weak good neighborhood properties and other properties are discussed below, while the other relations in the figure follow from the facts in Section 2.1.

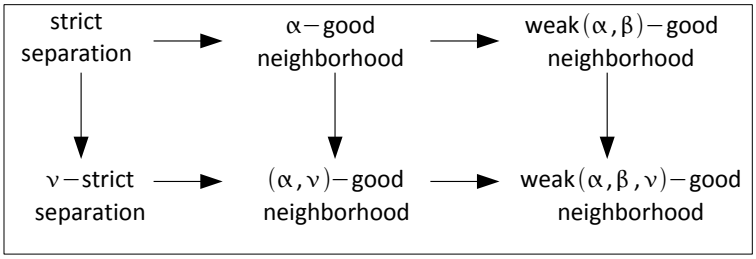


Figure 8: Relations between various properties. The arrows represent generalization.

By setting $A_p = C_i$ for $p \in C_i$ in the definition, we can see that the weak (α, β)-good neighborhood property is a generalization of the α-good neighborhood property when each target cluster has size greater than $6\alpha n$. Formally,

Fact 5 *If the similarity function $\mathcal{K}$ satisfies the α-good neighborhood property for the clustering problem (S, ℓ) and $\min_i |C_i| > 6\alpha n$, then $\mathcal{K}$ also satisfies the weak (α, β)-good neighborhood property for the clustering problem (S, ℓ) for any $0 < \beta \leq 1$.*

Proof If $\mathcal{K}$ satisfies the α-good neighborhood property and $\min_i |C_i| > 6\alpha n$, then we have: for any $p \in C_i$, there exists a subset $C_i \subseteq C_i$ of size greater than $6\alpha n$, such that out of the n_{C_i} nearest neighbors, any point $x \in C_i$ has at most αn neighbors outside C_i. So $\mathcal{K}$ satisfies both conditions of the weak (α, β)-good neighborhood property. ■

By setting $A_p = G_i$ for $p \in G_i$ in the definition, we can see that the weak (α, β, ν)-good neighborhood property generalizes the (α, ν)-good neighborhood property when each target cluster has size greater than $7(\alpha + \nu)n$. Also, by setting $\nu = 0$, we can see that the weak (α, β)-good neighborhood property is equivalent to the weak $(\alpha, \beta, 0)$-good neighborhood.

Fact 6 *If the similarity function $\mathcal{K}$ satisfies the (α, ν)-good neighborhood property for the clustering problem (S, ℓ) and $\min_i |C_i| > 7(\alpha + \nu)n$, then $\mathcal{K}$ also satisfies the weak (α, β, ν)-good neighborhood property for the clustering problem (S, ℓ) for any $0 < \beta \leq 1$.*

Proof If $\mathcal{K}$ satisfies the (α, ν)-good neighborhood property and $\min_i |C_i| > 7(\alpha + \nu)n$, then we have: for any $p \in G_i = C_i \setminus B$, there exists a subset $G_i \subseteq G_i$ of size greater than $6(\alpha + \nu)n$, such that out of the $|G_i|$ nearest neighbors in $S \setminus B$, any good point $x \in G_i$ has at most αn neighbors outside G_i. So $\mathcal{K}$ satisfies both conditions of the weak (α, β, ν)-good neighborhood property. ■

Fact 7 *If the similarity function $\mathcal{K}$ satisfies the weak (α, β)-good neighborhood property for the clustering problem (S, ℓ), then $\mathcal{K}$ also satisfies the weak $(\alpha, \beta, 0)$-good neighborhood property for the clustering problem (S, ℓ).*

Proof By setting $\nu = 0$ in the definition of the weak (α, β, ν)-good neighborhood property, we can see that it is the same as the weak (α, β)-good neighborhood property. ■

4.2 Correctness under the Weak Good Neighborhood Property

Now we prove that our algorithm also succeeds under the weak (α, β, ν)-good neighborhood property when $\beta \geq 7/8$. Formally,

Theorem 4 *Let $\mathcal{K}$ be a symmetric similarity function satisfying the weak (α, β, ν)-good neighborhood property for the clustering problem (S, ℓ) with $\beta \geq 7/8$. Then Algorithm 1 outputs a hierarchy such that a pruning of the hierarchy is ν-close to the target clustering in time $O(n^{\omega+1})$, where $O(n^{\omega})$ is the state of the art for matrix multiplication.*

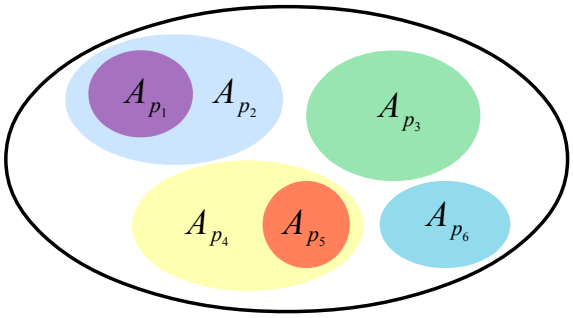


Figure 9: Illustration of a fully formed blob C_u: for any point $p \in C_u \setminus B$, $A_p \subseteq C_u$. Then we can show that sets in $\{A_p : p \in C_u \setminus B\}$ are laminar, that is, for any $p, q \in C_u \setminus B$, either $A_p \cap A_q = \emptyset$ or $A_p \subseteq A_q$ or $A_q \subseteq A_p$. For example, in the figure we have $A_{p_5} \subseteq A_{p_4}$.

Theorem 4 is a generalization of Theorem 1, and the proof follows a similar reasoning. The proof of correctness is from Lemma 8 stated and proved below and the running time follows from Lemma 3. The intuition is as follows. First, by similar arguments as for the good neighborhood property, each point p in $S \setminus B$ will only be merged with other points in A_p at $t \leq |A_p|$, and all points in A_p will belong to one blob at $t = |A_p|$ (Lemma 5), since in the local neighborhood of size $|A_p|$, the point has most of its nearest neighbor from A_p. Then, we need to show that such blobs will be correctly merged. The key point is to show that even in the presence of boundary points, the majority of points in such blobs are good points (Lemma 7). Then the median test can successfully distinguish blobs containing good points from different target clusters, and our algorithm can correctly merge blobs from the same target clusters together.

To formally prove the correctness, we begin with Lemma 5. The proof is similar to that for Lemma 2, replacing C_i with A_p.

Lemma 5 *The following claims are true in Algorithm 1:*
(1) For any point $p \in S \setminus B$ and t such that $t \leq |A_p|$, any blob in $\mathcal{C}'_t$ containing points from A_p will not contain points in $(S \setminus A_p) \setminus B$.
(2) For any point $p \in S \setminus B$ and $t = |A_p|$, all points in A_p belong to one blob in $\mathcal{C}'_t$.

Lemma 5 states that for any $p \in S \setminus B$, we will form A_p before merging them with points outside. Then we only need to make sure that these A_p formed will be correctly merged. More precisely, we need to consider the blobs that are "fully formed" in the following sense:

Definition 6 *A blob $C_u \in \mathcal{C}'_t$ in Algorithm 1 is said to be fully formed if for any point $p \in C_u \setminus B$, $A_p \subseteq C_u$.*

To show that fully formed blobs are correctly merged, the key point is to show that the majority of points in such blobs are good points, and thus the median test in the algorithm can successfully distinguish blobs containing good points from different target clusters. This key point is in fact a consequence of Lemma 5:

Lemma 7 *For any fully formed blob $C_u \in \mathcal{C}'_t$ in Algorithm 1, at least β fraction of points in $C_u \setminus B$ are good points.*

Proof It suffices to show that there exist a set of points $P \subseteq C_u \setminus B$, such that $\{A_p : p \in P\}$ is a partition of $C_u \setminus B$. Clearly $C_u \setminus B = \cup_{p \in C_u \setminus B} A_p$. So we only need to show that sets in $\{A_p : p \in C_u \setminus B\}$ are laminar, that is, for any $p, q \in C_u \setminus B$, either $A_p \cap A_q = \emptyset$ or $A_p \subseteq A_q$ or $A_q \subseteq A_p$. See Figure 9 for an illustration.

Assume for contradiction that there exist A_p and A_q such that $A_p \setminus A_q \neq \emptyset, A_q \setminus A_p \neq \emptyset$ and $A_p \cap A_q \neq \emptyset$. Without loss of generality, suppose $|A_p| \leq |A_q|$. Then by the second claim in Lemma 5, when $t = |A_p|$, all points in A_p belong to one blob in $\mathcal{C}'_t$. In other words, this blob contains $A_p \cap A_q$ and $A_p \setminus A_q$. So for $t \leq |A_q|$, the blob contains points in A_q and also points in $S \setminus B \setminus A_q$, which contradicts the first claim in Lemma 5. ■

We are now ready to prove the following lemma that implies Theorem 4.

Lemma 8 *The following claims are true in Algorithm 1:*
(1) For any C_i such that $t \leq |C_i|$, any blob in $\mathcal{C}'_t$ containing points in $C_i \setminus B$ will not contain points in $(S \setminus C_i) \setminus B$.
(2) For any C_i such that $t = |C_i|$, all points in $C_i \setminus B$ belong to one blob in $\mathcal{C}'_t$.

Proof Before proving the claims, we first show that the graph F_t constructed in Step 2 has the following useful properties by an argument similar to that in Lemma 2. Recall that F_t is constructed on points in S by connecting any two points that share at least $t - 2(\alpha + \nu)n$ points in common out of their t nearest neighbors. For any C_i such that $t \leq |C_i|$, we have:

(a) If x is a good point in C_i and y is a good point outside C_i, then x and y are not connected in F_t.

To see this, first note that by Fact 3, x has at most $(\alpha + \nu)n$ neighbors outside C_i out of the t nearest neighbors. Suppose y is a good point from C_j. If $n_{C_j} \geq t$, then y has at most $(\alpha + \nu)n$ neighbors in C_i; if $n_{C_j} < t$, y has at most $(\alpha + \nu)n + t - n_{C_j}$ neighbors in C_i. In both cases, y has at most $(\alpha+\nu)n+\max(0, t-n_{C_j}) < t-5(\alpha+\nu)n$ neighbors in C_i, since $n_{C_j} > 6(\alpha+\nu)n$ and $t > 6(\alpha+\nu)n$. Then x and y have at most $t - 4(\alpha + \nu)n$ common neighbors, so they are not connected in F_t.

(b) If x is a good point in C_i, y is a good point outside C_i, and z is a bad point, then z cannot be connected to both x and y in F_t.

To prove this, we will show that if z is connected to x, then z cannot be connected to y. First, by the same argument as above, out of the t nearest neighbors, y has less than $t - 5(\alpha + \nu)n$ neighbors in C_i. Second, by Fact 3, x has at most $(\alpha + \nu)n$ neighbors outside C_i. If z has less than $t - 3(\alpha + \nu)n$ neighbors in C_i, then z and x share less than $t - 3(\alpha + \nu)n + (\alpha + \nu)n = t - 2(\alpha + \nu)n$ neighbors and will not be connected. So z must have at least $t - 3(\alpha + \nu)n$ neighbors in C_i, and thus cannot have more than $3(\alpha+\nu)n$ neighbors outside C_i. The two statements show that y and z share less than $t - 5(\alpha + \nu)n$ neighbors in C_i, and at most $3(\alpha + \nu)n$ neighbors outside C_i. So they share less than $t-2(\alpha+\nu)n+3(\alpha+\nu)n = t-2(\alpha+\nu)n$ neighbors and thus are not connected in F_t.

Now we prove Claim (1) in the lemma by induction on t. The claim is clearly true initially. Assume for induction that the claim is true for the threshold $t-1$, that is, for any C_i such that $t-1 \le |C_i|$, any blob in $\mathcal{C}'_{t-1}$ containing points in $C_i \setminus B$ will not contain points in $(S \setminus C_i) \setminus B$. We now prove that the graph H_t constructed in Step 3 has the following properties, which can be used to show that the claim is still true for the threshold t.

- If $C_u \in \mathcal{C}'_{t-1}$ contains points from $C_i \setminus B$ and $C_v \in \mathcal{C}'_{t-1}$ contains points from $(S \setminus C_i) \setminus B$, then they cannot be connected in H_t.

  Suppose one of them (say C_u) is not fully formed, that is, there is a point $p \in C_u \setminus B$ such that $A_p \not\subseteq C_u$. Then by Lemma 5, the algorithm will not merge C_u with C_v at this threshold. More precisely, since not all points in A_p belong to C_u, we have $t-1 < |A_p|$ by Claim (2) in Lemma 5. Then by Claim (1) in Lemma 5, since C_v contains points in $(S \setminus A_p) \setminus B$, C_u and C_v will not be merged in $\mathcal{C}'_t$. So they are not connected in H_t. So we only need to consider the other case when C_u and C_v are fully formed blobs. By Lemma 7, the majority of points in the two blobs are good points. The good points from different target clusters have few common neighbors in F_t, then by the median test in our algorithm, the two blobs will not be connected in H_t. Formally, we can find two good points $x^* \in C_u, y^* \in C_v$ that satisfy the following two statements.

  - $S_t(x^*, y^*) \ge \text{median}_{x \in C_u, y \in C_v} S_t(x, y)$.
    By Lemma 7, at least $\beta \ge 7/8$ fraction of points in $C_u \setminus B$ are good points. The fraction of good points in C_u is at least

    $$\frac{\beta |C_u \setminus B|}{|C_u \setminus B| + |B|} \ge \frac{7/8 \times 6(\alpha+\nu)n}{6(\alpha+\nu)n + \nu n} \ge \frac{3}{4},$$

    since $|C_u \setminus B| \ge 6(\alpha+\nu)n$ and $|B| \le \nu n$. Similarly, at least $\frac{3}{4}$ fraction of points in C_v are good points. Then among all the pairs (x, y) such that $x \in C_u, y \in C_v$, at least $\frac{3}{4} \times \frac{3}{4} > \frac{1}{2}$ fraction are pairs of good points. So there exist good points $x^* \in C_u, y^* \in C_v$ such that $S_t(x^*, y^*) \ge \text{median}_{x \in C_u, y \in C_v} S_t(x, y)$.
  - $S_t(x^*, y^*) \le (|C_u| + |C_v|)/4$.
    The fraction of good points in $C_u \cup C_v$ is at least $\frac{3}{4}$. Since in F_t, good points in C_u are not connected to good points in C_v, we have $S_t(x^*, y^*) \le (|C_u| + |C_v|)/4$.

  Combining the two statements, we have $\text{median}_{x \in C_u, y \in C_v} S_t(x, y) \le (|C_u| + |C_v|)/4$ and thus C_u and C_v are not connected in H_t.

- If in $\mathcal{C}'_{t-1}$, C_u contains points from $C_i \setminus B$, C_v contains points from $(S \setminus C_i) \setminus B$, and C_w contains only bad points, then C_w cannot be connected to both C_u and C_v.

  By the same argument as above, we only need to consider the case when C_u and C_v are fully formed blobs. To prove the claim in this case, assume for contradiction that C_w is connected to both C_u and C_v. First, note the following fact about C_w. Since any non-singleton blob must be formed in Step 4 in the algorithm and contain at least $4(\alpha+\nu)n$ points and thus cannot contain only bad points, C_w must be a singleton blob, containing only a bad point z.
  Next, we show that if $C_w = \{z\}$ are connected to C_u in H_t, then z must be connected

to at least one good point in C_u in F_t. We have $\text{median}_{x \in C_u} S_t(x, z) > \frac{|C_u|+|C_w|}{4}$, which means z is connected to more than $\frac{|C_u|}{4}$ points in C_u in F_t. By the same argument as above, at least 3/4 fraction of points in C_u are good points, then z must be connected to at least one good point in C_u.
Similarly, if C_w is connected to C_v in H_t, then z must be connected to at least one good point in C_v in F_t. But this contradicts Property (b) of F_t, so C_w cannot be connected to both C_u and C_v in H_t.

By the properties of H_t, no connected component contains both points in $C_i \setminus B$ and points in $(S \setminus C_i) \setminus B$. So Claim (1) is still true for the threshold t. By induction, it is true for all thresholds.

Finally, we prove Claim (2). By Lemma 5, when $t = |C_i|$, for any point $p \in C_i \setminus B$, A_p belong to the same blob. So all points in $C_i \setminus B$ are in sufficiently large blobs. We will show that any two of these blobs C_u, C_v are connected in H_t, and thus will be merged into one blob. By Lemma 7, we know that more than 3/4 fraction of points in C_u (C_v respectively) are good points, and thus there exist good points $x^* \in C_u, y^* \in C_v$ such that $S_t(x^*, y^*) \leq \text{median}_{x \in C_u, y \in C_v} S_t(x, y)$. By Claim (1), all good points in C_u and C_v are from C_i, so they share at least $t - 2(\alpha + \nu)n$ neighbors when $t = |C_i|$, and thus are connected in F_t. Then $S_t(x^*, y^*)$ is at least the number of good points in $C_u \cup C_v$, which is at least $3(|C_u| + |C_v|)/4$. Then $\text{median}_{x \in C_u, y \in C_v} S_t(x, y) \geq S_t(x^*, y^*) > (|C_u| + |C_v|)/4$. Therefore, all blobs containing points from $C_i \setminus B$ are connected in H_t and thus merged into a blob. ■

5. The Inductive Setting

Many clustering applications have recently faced an explosion of data, such as in astrophysics and biology. For large data sets, it is often resource and time intensive to run an algorithm over the entire data set. It is thus increasingly important to develop algorithms that can remove the dependence on the actual size of the data and still perform reasonably well.

In this section we consider an inductive model that formalizes this problem. In this model, the given data is merely a small random subset of points from a much larger data set. The algorithm outputs a hierarchy over the sample, which also implicitly represents a hierarchy over the data set. In the following we describe the inductive version of our algorithm and prove that when the data satisfies the good neighborhood properties, the algorithm achieves small error on the entire data set, requiring only a small random sample whose size is independent of that of the entire data set.

5.1 Formal Definition

First we describe the formal definition of the inductive model. In this setting, the given data S is merely a small random subset of points from a much larger abstract instance space X. For simplicity, we assume that X is finite and that the underlying distribution is uniform over X. Let $N = |X|$ denote the size of the entire instance space, and let $n = |S|$ denote the size of the sample.

Algorithm 2 Inductive Robust Median Neighborhood Linkage

Input: similarity function $\mathcal{K}$, $n \in \mathbf{Z}^+$, parameters $\alpha > 0, \nu > 0$.

▷ `Get a hierarchy on the sample`
Sample i.i.d. examples $S = \{x_1, \ldots, x_n\}$ uniformly at random from X.
Run Algorithm 1 with parameters $(2\alpha, 2\nu)$ on S and obtain a hierarchy T.

▷ `Get the implicit hierarchy over` X
for any $x \in X$ **do**
  Let $N_S(x)$ denote the $6(\alpha + \nu)n$ nearest neighbors of x in S.
  Initialize $u = \text{root}(T)$ and $f_u(x) = 1$.
  while u is not a leaf **do**
    Let w be the child of u that contains the most points in $N_S(x)$.
    Set $u = w$ and $f_u(x) = 1$.
  end while
end for

Output: Hierarchy T and $\{f_u, u \in T\}$.

Our goal is to design an algorithm that based on the sample produces a hierarchy of small error with respect to the whole distribution. Formally, we assume that each node u in the hierarchy derived over the sample induces a cluster (a subset of X). For convenience, u is also used to denote the blob of sampled points it represents. The cluster u induces over X is implicitly represented as a function $f_u : X \to \{0, 1\}$, that is, for each $x \in X$, $f_u(x) = 1$ if x is a point in the cluster and 0 otherwise. We say that the hierarchy has error at most ϵ if it has a pruning $f_{u_1}, \ldots, f_{u_k}$ of error at most ϵ.

5.2 Inductive Robust Median Neighborhood Linkage

The inductive version of our algorithm is described in Algorithm 2. To analyze the algorithm, we first present the following lemmas showing that, when the data satisfies the good neighborhood property, a sample of sufficiently large size also satisfies the weak good neighborhood property.

Lemma 9 *Let $\mathcal{K}$ be a symmetric similarity function satisfying the (α, ν)-good neighborhood for the clustering problem (X, ℓ). Consider any fixed $x \in X \setminus B$. If the sample size satisfies $n = \Theta\left(\frac{1}{\alpha} \ln \frac{1}{\delta}\right)$, then with probability at least $1 - \delta$, x has at most $2\alpha n$ neighbors outside $(C(x) \setminus B) \cap S$ out of the $|(C(x) \setminus B) \cap S|$ nearest neighbors in $S \setminus B$.*

Proof Suppose $x \in G_i$. Let $NN(x)$ denote its $|G_i|$ nearest neighbors in X. By assumption we have that $|NN(x) \setminus G_i| \leq \alpha N$ and $|G_i \setminus NN(x)| \leq \alpha N$. Then by Chernoff bounds, with probability at least $1 - \delta$ at most $2\alpha n$ points in our sample are in $NN(x) \setminus G_i$ and at most $2\alpha n$ points in our sample are in $G_i \setminus NN(x)$.

We now argue that at most $2\alpha n$ of the $|G_i \cap S|$ nearest neighbors of x in $S \setminus B$ can be outside G_i. Let n_1 be the number of points in $(NN(x) \setminus G_i) \cap S$, n_2 be the number of

points in $(G_i \setminus NN(x)) \cap S$, and n_3 be the number of points in $(G_i \cap NN(x)) \cap S$. Then $|G_i \cap S| = n_2 + n_3$ and we know that $n_1 \leq 2\alpha n$, $n_2 \leq 2\alpha n$. We consider the following two cases.

- $n_1 \geq n_2$. Then $n_1 + n_3 \geq n_2 + n_3 = |G_i \cap S|$. This implies that the $|G_i \cap S|$ nearest neighbors of x in the sample all lie inside $NN(x)$, since by definition all points inside $NN(x)$ are closer to x than any point outside $NN(x)$. But we are given that at most $n_1 \leq 2\alpha n$ of them can be outside G_i. Thus, we get that at most $2\alpha n$ of the $|G_i \cap S|$ nearest neighbors of x are not from G_i.
- $n_1 < n_2$. This implies that the $|G_i \cap S|$ nearest neighbors of x in the sample include *all* the points in $NN(x)$ in the sample, and possibly some others too. But this implies in particular that it includes all the n_3 points in $G_i \cap NN(x)$ in the sample. So, it can include at most $|G_i \cap S| - n_3 = n_2 \leq 2\alpha n$ points not in $G_i \cap NN(x)$. Even if all those are not in G_i, the $|G_i \cap S|$ nearest neighbors of x still include at most $2\alpha n$ points not from G_i.

In both cases, at most $2\alpha n$ of the $|G_i \cap S|$ nearest neighbors of x in $S \setminus B$ can be outside G_i. ■

Lemma 10 *Let $\mathcal{K}$ be a symmetric similarity function satisfying the (α, ν)-good neighborhood for the clustering problem (X, ℓ). If the sample size satisfies $n = \Theta\left(\frac{1}{\min(\alpha,\nu)} \ln \frac{1}{\delta \min(\alpha,\nu)}\right)$, then with probability at least $1 - \delta$, $\mathcal{K}$ satisfies the $(2\alpha, 2\nu)$-good neighborhood with respect to the clustering induced over the sample S.*

Proof First, by Chernoff bounds, when $n \geq \frac{3}{\nu} \ln \frac{2}{\delta}$, we have that with probability at least $1 - \delta/2$, at most $2\nu n$ bad points fall into the sample.

Next, by Lemma 9 and union bound, when $n = \Theta\left(\frac{1}{\alpha} \ln \frac{n}{\delta}\right)$ we have that with probability at least $1 - \delta/2$, for any C_i, any $x \in G_i \cap S$, x has at most $2\alpha n$ points outside $G_i \cap S$ out of its $|G_i \cap S|$ nearest neighbors in $(X \setminus B) \cap S$. Therefore, if $n = \Theta\left(\frac{1}{\min(\alpha,\nu)} \ln \frac{n}{\delta}\right)$, then with probability at least $1 - \delta$, the similarity function satisfies the $(2\alpha, 2\nu)$-good neighborhood property with respect to the clustering induced over the sample S.

It now suffices to show n is large enough so that $n = \Theta\left(\frac{1}{\min(\alpha,\nu)} \ln \frac{n}{\delta}\right)$. To see this, let $\eta = \min(\alpha, \nu)$. Since $\ln n \leq tn - \ln t - 1$ for any $t, n > 0$, we have

$$\frac{c}{\eta} \ln n \leq \frac{c}{\eta}\left(\frac{\eta}{2c} n + \ln \frac{2c}{\eta} - 1\right) = \frac{n}{2} + \frac{c}{\eta} \ln \frac{2c}{e \cdot \eta},$$

for any constant $c > 0$. Then $n = \Theta\left(\frac{1}{\eta} \ln \frac{1}{\eta}\right)$ implies $n = \Theta\left(\frac{1}{\eta} \ln n\right)$, and $n = \Theta\left(\frac{1}{\eta} \ln \frac{1}{\delta \cdot \eta}\right)$ implies $n = \Theta\left(\frac{1}{\eta} \ln \frac{n}{\delta}\right)$. ■

Theorem 11 *Let $\mathcal{K}$ be a symmetric similarity function satisfying the (α, ν)-good neighborhood for the clustering problem (X, ℓ). As long as the smallest target cluster has size greater than $12(\nu + \alpha)N$, then Algorithm 2 with parameters $n = \Theta\left(\frac{1}{\min(\alpha,\nu)} \ln \frac{1}{\delta \cdot \min(\alpha,\nu)}\right)$ produces a hierarchy with a pruning that is $(\nu + \delta)$-close to the target clustering with probability $1 - \delta$.*

Proof Note that by Lemma 10, with probability at least $1-\delta/4$, we have that $\mathcal{K}$ satisfies the $(2\alpha, 2\nu)$-good neighborhood with respect to the clustering induced over the sample. Moreover, by Chernoff bounds, with probability at least $1-\delta/4$, each G_i has at least $6(\nu+\alpha)n$ points in the sample. Then by Theorem 1, Algorithm 1 outputs a hierarchy T on the sample S with a pruning that assigns all good points correctly. Denote this pruning as $\{u_1, \ldots, u_k\}$ such that $u_i \setminus B = (C_i \cap S) \setminus B$.

Now we want to show that $f_{u_1}, \ldots, f_{u_k}$ have error at most $\nu+\delta$ with probability at least $1-\delta/2$. For convenience, let $u(x)$ be a shorthand of $u_{\ell(x)}$. Then it is sufficient to show that with probability at least $1-\delta/2$, a $(1-\delta)$ fraction of points $x \in X \setminus B$ have $f_{u(x)}(x) = 1$.

Fix C_i and a point $x \in C_i \setminus B$. By Lemma 9, with probability at least $1-\delta^2/2$, out of the $|G_i \cap S|$ nearest neighbors of x in $S \setminus B$, at most $2\alpha n$ can be outside G_i. Recall that Algorithm 2 checks $N_S(x)$, the $6(\alpha+\nu)n$ nearest neighbors of x in S. Then out of $N_S(x)$, at most $2(\alpha+\nu)n$ points are outside $G_i \cap S$. By Lemma 2, u_i contains $G_i \cap S$, so u_i must contain at least $4(\alpha+\nu)n$ points in $N_S(x)$. Consequently, any ancestor w of u_i, including u_i, has more points in $N_S(x)$ than any other sibling of w. Then we must have $f_w(x) = 1$ for any ancestor w of u_i. In particular, $f_{u_i}(x) = 1$. So, for any point $x \in X \setminus B$, with probability at least $1-\delta^2/2$ over the draw of the random sample, $f_{u(x)}(x) = 1$.

Then by Markov inequality, with probability at least $1-\delta/2$, a $(1-\delta)$ fraction of points $x \in X \setminus B$ have $f_{u(x)}(x) = 1$. More precisely, let U_x denote the uniform distribution over $X \setminus B$, and let U_S denote the distribution of the sample S. Let $I(x, S)$ denote the event that $f_{u(x)}(x) \neq 1$. Then we have

$$\mathbf{E}_{x\sim U_x, S\sim U_S}[I(x,S)] = \mathbf{E}_{S\sim U_S}\left[\mathbf{E}_{x\sim U_x}[I(x,S)|S]\right] \leq \delta^2/2.$$

Then by Markov inequality, we have

$$\Pr_{S\sim U_S}\left[\mathbf{E}_{x\sim U_x}[I(x,S)|S] \geq \delta\right] \leq \delta/2,$$

which means that with probability at least $1-\delta/2$ over the draw of the random sample S, a $(1-\delta)$ fraction of points $x \in X \setminus B$ have $f_{u(x)}(x) = 1$. ■

Similarly, Algorithm 2 also succeeds for the weak good neighborhood property. By similar arguments as those in Lemma 9 and 10, we can prove that $\mathcal{K}$ satisfies the weak good neighborhood property over a sufficiently large sample (Lemma 12), which then leads to the final guarantee Theorem 13. For clarity, the proofs are provided in Appendix B.

Lemma 12 *Let $\mathcal{K}$ be a symmetric similarity function satisfying the weak (α, β, ν)-good neighborhood for the clustering problem (X, ℓ). Furthermore, it satisfies that for any $p \in X \setminus B$, $|A_p| > 24(\alpha+\nu)N$. If the sample size satisfies $n = \Theta\left(\frac{1}{\min(\alpha,\nu)} \ln \frac{1}{\delta \min(\alpha,\nu)}\right)$, then with probability at least $1-\delta$, $\mathcal{K}$ satisfies the $(2\alpha, \frac{15}{16}\beta, 2\nu)$-good neighborhood with respect to the clustering induced over the sample S.*

Theorem 13 *Let $\mathcal{K}$ be a symmetric similarity function satisfying the weak (α, β, ν)-good neighborhood for the clustering problem (X, ℓ) with $\beta \geq \frac{14}{15}$. Furthermore, it satisfies*

that for any $p \in X \setminus B$, $|A_p| > 24(\alpha+\nu)N$. *Then Algorithm 2 with parameters* $n = \Theta\left(\frac{1}{\min(\alpha,\nu)} \ln \frac{1}{\delta \cdot \min(\alpha,\nu)}\right)$ *produces a hierarchy with a pruning that is* $(\nu+\delta)$*-close to the target clustering with probability* $1-\delta$.

6. Experiments

In this section, we compare our algorithm (called RMNL for convenience) with popular hierarchical clustering algorithms, including standard linkage algorithms (Sneath and Sokal, 1973; King, 1967; Everitt et al., 2011), (Generalized) Wishart's Method (Wishart, 1969; Chaudhuri and Dasgupta, 2010), Ward's minimum variance method (Ward, 1963), CURE (Guha et al., 1998), and EigenCluster (Cheng et al., 2006).

To evaluate the performance of the algorithms, we use the model discussed in Section 2. Given a hierarchy output by an algorithm, we generate all possible prunings of size k, where k is the number of clusters in the target clustering.[5] Then we compute the *Classification Error* of each pruning with respect to the target clustering, and report the best error. The Classification Error of a computed clustering h with respect to the target clustering ℓ is the probability that a point chosen at random from the data is labeled incorrectly.[6] Formally,

$$\mathbf{err}(h) = \min_{\sigma \in \mathcal{S}_k} \left[\Pr_{x \in S} \left[\sigma(h(x)) \neq \ell(x)\right] \right],$$

where $\mathcal{S}_k$ is the set of all permutations on $\{1, \ldots, k\}$. For reporting results, we follow the classic methodology (Guha et al., 1998): for all algorithms accepting input parameters (including (Generalized) Wisharts' Method, CURE, and RMNL), the experiments are repeated on the same data over a range of input parameter values, and the best results are considered.

Data sets To emphasize the effect of noise on different algorithms, we perform controlled experiments on a synthetic data set AIStat. This data set contains 512 points. It is an instance of the example discussed in Section 4 and is described in Figure 7. We further consider the following real-world data sets from UCI Repository (Bache and Lichman, 2013): Wine, Iris, BCW (Breast Cancer Wisconsin), BCWD (Breast Cancer Wisconsin Diagnostic), Spambase, and Mushroom. We also consider the MNIST data set (LeCun et al., 1998) and use two subsets of the test set for our experiments: Digits0123 that contains the examples of the digits $0, 1, 2, 3$, and Digits4567 that contains the examples of the digits $4, 5, 6, 7$.

We additionally consider the 10 data sets (PFAM1 to PFAM10) (Voevodski et al., 2012), which are created by randomly choosing 8 families (of size between 1000 and 10000) from the biology database Pfam (Punta et al., 2012), version 24.0, October 2009. The similarities for the PFAM data sets are generated by biological sequence alignment software BLAST (Altschul et al., 1990). BLAST performs one versus all queries by aligning a queried sequence to sequences in the data set, and produces a score for each alignment. The score is a measure of the alignment quality and thus can be used as similarity. However, BLAST

5. Note that we generate all prunings of size k for evaluating the performance of various algorithms only. The hierarchical clustering algorithms do not need to generate these prunings when creating the hierarchies.
6. To compute this error for a computed clustering in polynomial time, we first find its best match to the target clustering using the Hungarian Method (Kuhn, 1955) for min-cost bipartite matching in time $O(n^3)$, and then calculate the error as the fraction of points misclassified in matched clusters.

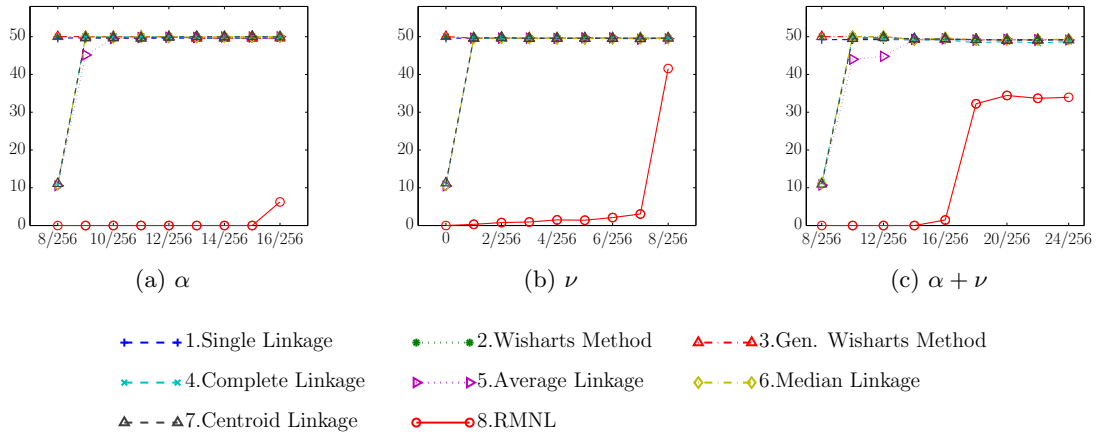


Figure 10: Classification Error on the synthetic data AIStat. The y-axis represents the % error. (a) Fix $\nu = 0$ and vary α from 1/32 to 1/16. The x-axis represents the value of α; (b) Fix $\alpha = 1/32$ and vary ν from 0 to 1/32. The x-axis represents the value of ν; (c) Vary α from 1/32 to 1/16, and vary ν from 0 to 1/32. The x-axis represents the value of $\alpha + \nu$. Note that the instance no longer satisfies the weak good neighborhood property when $\alpha + \nu \geq 1/24 \approx 11/256$.

does not consider alignments with some of the sequences, in which case we assign similarities 0 to the corresponding sequences and exclude them from the neighbors of the queried sequence.

The smaller data sets are used in the transductive setting: Wine (178 points of dimension 13), Iris (150×4), BCW (699×10), and BCWD (569×32). The larger ones are used in the inductive setting: Spambase (4601×57), Mushroom (8124×22), Digits0123 (4157×784), Digits4567 (3860×784), and PFAM1 to PFAM10 ($10000 \sim 100000$ sequences each).

6.1 Synthetic Data

Here we compare the performance of the algorithms on the synthetic data AIStat. Recall that the clustering $\{\text{AI}, \text{Statistics}\}$ satisfies the weak (α, β, ν)-good neighborhood property for $\alpha = 1/32, \beta = 7/8, \nu = 0$ with high probability (See Figure 7 in Section 4). We conduct three sets of experiments, where we vary the values of α and ν by modifying the similarities between the points.

(a) For each point x, we choose $\Delta\alpha n$ points y from the other field and set the similarities $\mathcal{K}(x, y) = \mathcal{K}(y, x) = 1$, so that the value of α is increased to $1/32 + \Delta\alpha$. By varying $\Delta\alpha$, we control $\alpha = 1/32 + i/256$ for $i = 0, \ldots, 8$ and run the clustering algorithms on the modified data.

(b) We randomly choose νn points x, and then set the similarity between x and any other point to be 1 minus the original similarity. This introduces νn bad points. We thus control $\nu = i/256$ for $i = 0, \ldots, 8$.

(c) We perform the above two modifications simultaneously, that is, we control $\alpha = 1/32 + i/256$ and $\nu = i/256$ for $i = 0, \ldots, 8$.

Note that the instance no longer satisfies the weak good neighborhood property when $\alpha+\nu \geq 1/24$. This is because the weak good neighborhood requires that each point $p \notin B$ falls into a subset A_p of size greater than $6(\alpha+\nu)n$ with desired properties (see Property 5), and the largest such subsets in AIStat have size $n/4$.

Figure 10 shows the results of these experiments, averaged over 10 runs. When $\alpha+\nu < 1/24$, the instance satisfies the weak good neighborhood property and our algorithm has error at most ν. Moreover, even if the instance does not satisfy the weak good neighborhood property when $\alpha+\nu \geq 1/24$, our algorithm still reports lower error. All the other algorithms have higher error than our algorithm and fail rapidly as $\alpha+\nu$ increases. This demonstrates that in cases modeled by the properties we propose, our algorithm will be successful while the traditional agglomerative algorithms fail.

6.2 Real-World Data

In this section, we compare the performance of our algorithm with the other algorithms on real-world data sets and show that our algorithm consistently outperforms the others.

6.2.1 Transductive Setting

Here we compare the performance of the algorithms in the transductive setting where the algorithms use all the points in the data set. Figure 11 shows that our algorithm consistently achieves lowest or close to lowest errors on all the data sets. Ward's Method is the best among the other algorithms, but still shows larger errors than our algorithms. All the other algorithms generally show worse performance, and report significantly higher errors on some of the data sets. The comparison shows the robustness of our algorithm to the noise in the real world data sets.

To further evaluate the robustness of the algorithms, in the following we show their performance when different types of noise are added to the data. Since our algorithm requires additional parameters to characterize noise, we also discuss their robustness to parameter tuning.

Robustness to Noise Here we present the performance of the algorithms when Gaussian noise or corruption noise is added and the level of noise is increased monotonically. The Gaussian noise model essentially corresponds to additive perturbations to the data entries and it is a very common type of noise studied throughout machine learning. The corruption noise models data corruption or missing values, and is also frequently studied in machine learning and coding theory (Blum et al., 2007; Feldman et al., 2008; Wigderson and Yehudayoff, 2012; Moitra and Saks, 2013). The experiments on different types of noise then evaluate the robustness of the algorithms to noise caused by different reasons in real world scenarios. Note that the instance is not in a metric space after adding noise to the similarities, so in this case, we only evaluate algorithms that can be run on non-metric instances.

We consider three types of noise: corruption noise to the attributes, corruption noise to the similarities, and Gaussian noise added to the attributes. The first type of noise is generated as follows: normalize the entries in the data matrix to $[0, 1]$; randomly pick p fraction of the entries; replace each sampled entry with a random value independently

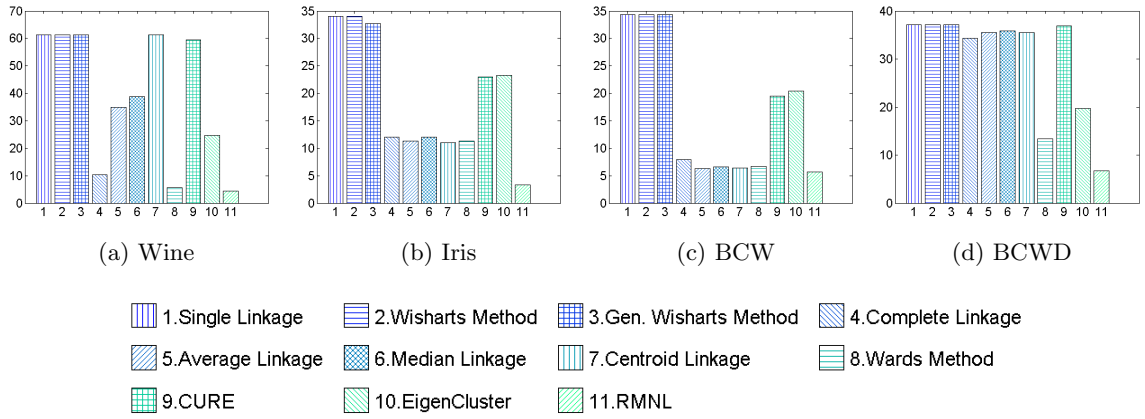


Figure 11: Classification Error in the transductive setting. The y-axis represents the % error.

generated from $N(0,1)$, where p is the parameter indicating the level of noise. The second type of noise is generated using the same approach, but is added to the similarity matrix. The third type of noise is generated as follows: normalize the entries in the data matrix to $[0,1]$; add a random value independently generated from $N(0,p^2)$ to each entry, where p is the parameter indicating the level of noise.

Figure 12 shows the results of different algorithms in the presence of noise, averaged over 30 runs. The rows correspond to different types of noise added, and the columns correspond to different data sets. The first row shows the results when corruption noise is added to the attributes. Our algorithm shows robustness to such type of noise: its error rates remain the best or close to the best up to noise level 0.2 on all data sets. EigenCluster and Ward's method also show robustness, but their error rates are generally higher than those of our algorithm. The other algorithms report high errors even when the noise level is as low as 0.04.

The second row shows the results when corruption noise is added to the similarities. We observe that the errors of our algorithm remain nearly the same up to noise level 0.2 over all the data sets, while the other algorithms report higher errors. Some algorithms (such as Complete Linkage on Wine) show comparable performance to our algorithms when there is no noise, but their errors generally increase rapidly as the noise level increases. This shows that our algorithm performs much better than the other algorithms in the presence of corruptions in the similarities.

The third row shows the results when Gaussian noise is added to the attributes. We observe that when the noise level increases, the errors of all algorithms increase. The errors of our algorithm do not increase much: they remain the best or close to the best up to the noise level 0.2 on all the data sets. Ward's method also shows robustness, since the minimum variance criterion used is insensitive to this type of noise. The other algorithms generally show higher errors than our algorithms and Ward's method.

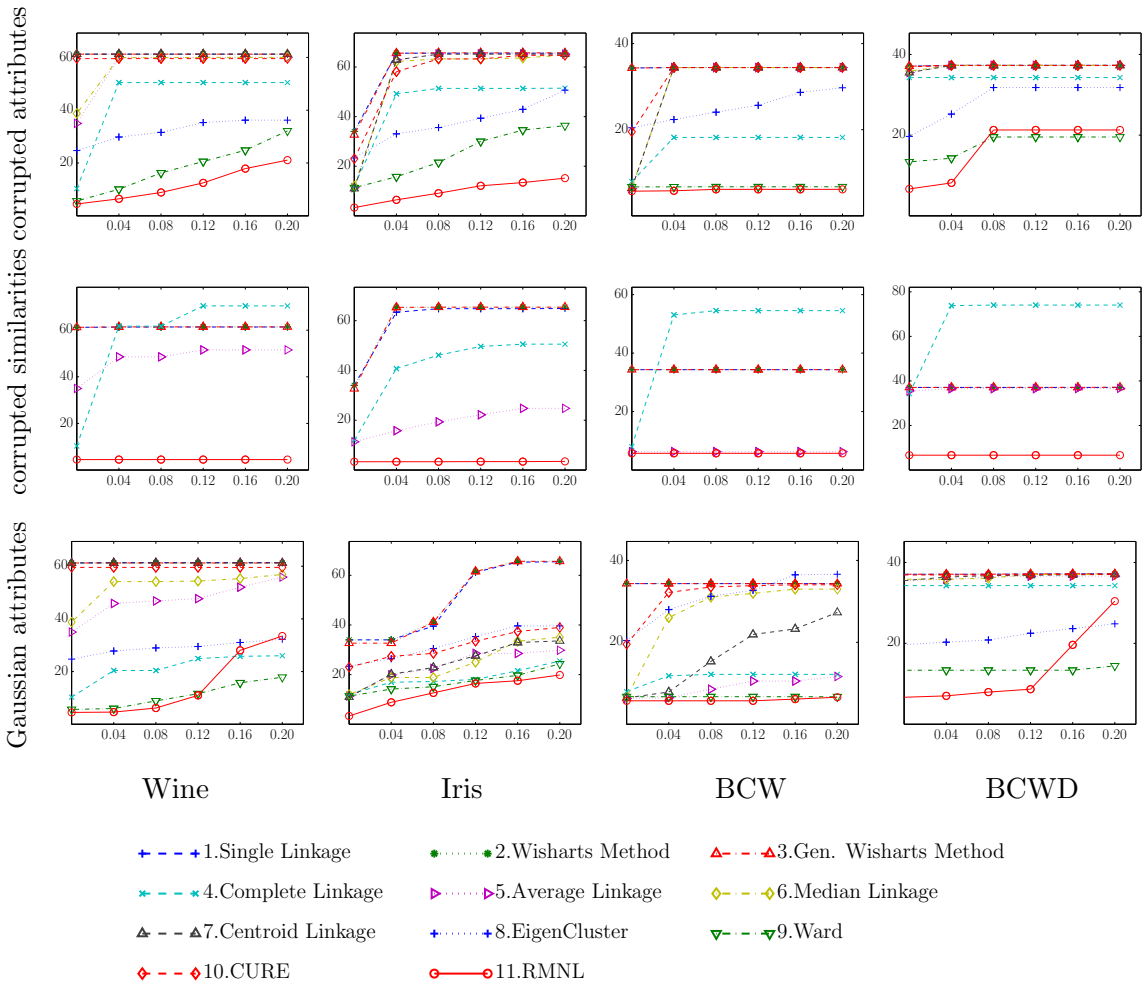
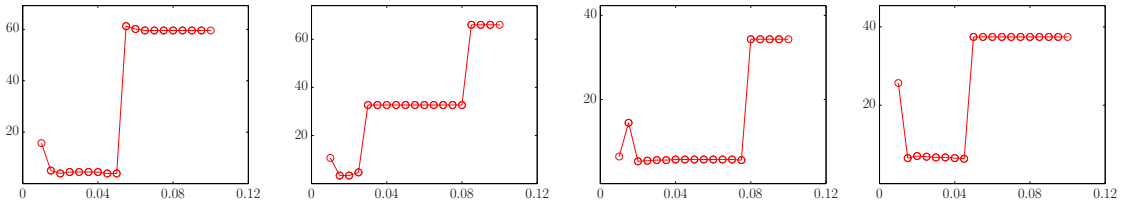


Figure 12: Classification Error in the presence of noise. Rows: corruption noise added to the attributes, corruption noise added to the similarities, Gaussian noise added to the attributes. Columns: Wine, Iris, BCW, BCWD. In each subfigure, the x-axis represents the noise level, and the y-axis represents the % error.

In conclusion, our algorithm generally outperforms the other algorithms when corruption noise is added to the data attributes or the similarities, or when Gaussian noise is added to the data attributes. Its robustness to Gaussian noise in similarities is not so significant since such noise with large variance can change the neighbor rankings of all points considerably. Still, it can tolerate such noise when the noise variance is not too large.

Robustness to Parameter Tuning Our algorithm requires extra input parameters α and ν. There may be indirect ways to set their values, for example, by estimating the size of the smallest target cluster. Still, we are not aware of any efficient algorithm to compute the approximately correct values. Since these parameters play an important role in our algorithm, it is crucial to show the robustness of the algorithm to parameter tuning. Note that the



(a) Wine (b) Iris (c) BCW (d) BCWD

Figure 13: Classification Error of RMNL using different values of parameter $(\alpha + \nu)$. The x-axis represents the value of $(\alpha + \nu)$, and the y-axis represents the % error.

two parameters are always used together as the additive term $(\alpha + \nu)$, thus essentially the algorithm takes one parameter. So for evaluation, we vary the parameter $(\alpha + \nu)$ linearly and run our algorithm over these values.

Figure 13 shows the performance of the algorithm for different parameter values. We observe that the algorithm does not require the exact value of $(\alpha + \nu)$ as it shows good performance over a continuous range of values. The range is sufficiently large for all data sets except Iris. The range for Iris is relatively small as there is little noise in it, and thus the parameter should be set to small values. In the other data sets we tried, we observed that it is easy to land in the right range with only a few runs.

6.2.2 Inductive Setting

In this subsection, we present the evaluation results in the inductive setting. In this setting, the algorithm generates a hierarchy on a small random sample of the data set, and inserts the remaining points to generate a new hierarchy over the entire data set. We repeat the sampling and evaluation for 5 times and report the average results.

We compare our inductive algorithm with the random sample algorithm (Eriksson, 2012). These algorithms sample some fraction of the similarities and use only these similarities. The percentage of sampled similarities can be tuned in these algorithms, so we compare their performance when they use the same amount of sampled similarities.

Figure 14 shows the results for eight configurations (using 5% or 10% similarities on four different data sets). Our algorithm consistently outperforms the random sampling algorithm. Figure 15 shows the results on PFAM1 to PFAM10, which approximately satisfy the good neighborhood property (Voevodski et al., 2012). On all PFAM data sets, the errors of our algorithm are low while those of the random sample algorithm are much higher. This shows the significant advantage of our algorithm when the data approximately satisfies the good neighborhood property.

7. Discussion

In this work we propose and analyze a new robust algorithm for bottom-up agglomerative clustering. We show that our algorithm can be used to cluster accurately in cases where the data satisfies a number of natural properties and where the traditional agglomerative algorithms fail. In particular, if the data satisfies the good neighborhood properties, the

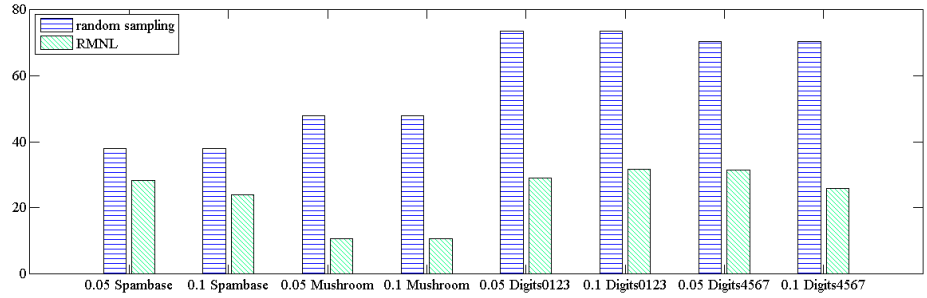


Figure 14: Classification Error of different algorithms in the inductive setting. The y-axis represents the % error. The x-axis represents data sets, where the numbers before the names of the data sets denote the fraction of similarities used by the inductive algorithms.

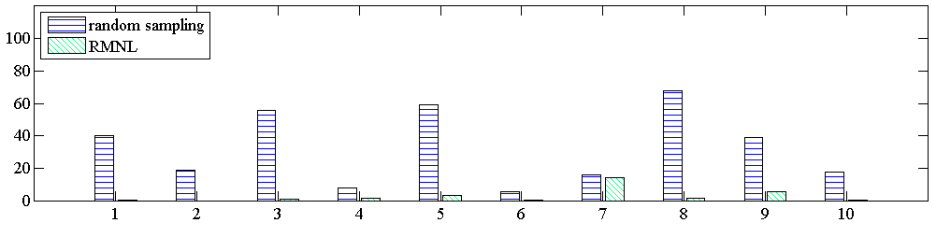


Figure 15: Classification Error on PFAM1 to PFAM10 data sets using 2.5% similarities. The y-axis in each case represents the % error, and the x-axis represents data sets.

algorithm will be successful in generating a hierarchy such that the target clustering is close to a pruning of that hierarchy.

We also show how to extend our algorithm to the inductive setting, where the given data is only a small random sample of the entire data set. Our algorithm achieves similar correctness guarantees, requiring only a small random sample whose size is independent of that of the entire data set.

We empirically show that with appropriate tuning of the noise parameters our algorithm consistently performs better than other hierarchical algorithms and are more robust to noise in the data. We also show the efficacy of the inductive version of our algorithm as a faster alternative when evaluation over the complete data is resource intensive.

Additionally, our subsequent work (Balcan and Liang, 2013) showed that the algorithm can be applied to the closely related community detection task and compares favorably with existing approaches.

It would be interesting to see if our algorithmic approach can be shown to work for other natural properties on the input similarity function. For example, it would be particularly interesting to analyze a noisy version of the max stability property (Balcan et al., 2008), which was shown to be a necessary and sufficient condition for single linkage to succeed, or of the average stability property which was shown to be a sufficient condition for average linkage to succeed. It would also be interesting to identify other natural conditions under different types of algorithms which are known to provide empirical noise robustness (e.g., the Wards method) would provably succeed. Finally, from an experimental point of view, an interesting open question is whether one can provide a wrapper for the algorithm to eliminate the need for manual tuning of the noise parameters.

Acknowledgments

We thank Avrim Blum for numerous useful discussions, and Konstantin Voevodski for providing us the PFAM data sets. We also thank the reviewers for their helpful comments and suggestions.

This work was supported in part by NSF grant CCF-0953192, AFOSR grant FA9550-09-1-0538, ONR grant N00014-09-1-0751, a Google Research Award, and a Microsoft Faculty Fellowship.

Appendix A. Implementation Details of Algorithm 1

Here we give the full details of the implementation of Algorithm 1.

First, we need some auxiliary data structures to build the graphs F_t and H_t. See Algorithm 3 for the definitions of these data structures.

Second, we specify the order of merging clusters. In the merge step in Algorithm 1, the blobs in a sufficiently large connected component of H_t can be merged in arbitrary order. In our implementation, we merge two connected C_u, C_v in H_t such that they are not both singleton blobs and they have maximum median$_{x \in C_u, y \in C_v} S_t(x, y)/(|C_u| + |C_v|)$ (so that we are most confident about merging them). Then we merge singleton clusters.

Third, for practical purposes, we can slightly modify the algorithm to speed it up on practical instances.[7] When there are less than $4(\alpha+\nu)n$ singleton blobs, we know that they cannot be merged together into one blob. So we can simply merge each singleton blob with the non-singleton blob that has the highest median similarity. This will correctly assign all but bad points under the good neighborhood properties. Similarly, when the number of singleton blobs is less than half the current threshold, we can safely merge each singleton blob with the non-singleton blob that has the highest median similarity.

Appendix B. Additional Proofs for Section 5

Here we provide the details for proving that Algorithm 2 also succeeds for the weak good neighborhood. First, by a similar argument as that in Lemma 9, we can prove Lemma 14

7. This does not change the time complexity and the correctness, but we observe that it helps speed up practical instances.

Algorithm 3 Implementation Details of Robust Median Neighborhood Linkage

Input: Similarity function $\mathcal{K}$ on a set of points S, $n = |S|$, $\alpha > 0, \nu > 0$.

Step 1 For each point, sort the other points increasingly according to the distances.
Initialize $t = 6(\alpha + \nu)n + 1$, and $\mathcal{C}'_{t-1}$ to be a list of singleton blobs.
while $|\mathcal{C}'_{t-1}| > 1$ **do**

Step 2 $\triangleright$ `Build a graph` F_t `on the points in` S `as follows:`
Set $I_t(x, y) = 1$ if y is in x's t nearest neighbors; $I_t(x, y) = 0$ otherwise.
Set $N_t = I_t(I_t)^T$.
Set $F_t(x, y) = 1$ if $N_t(x, y) \geq t - 2(\alpha + \nu)n$; $F_t(x, y) = 0$ otherwise.

Step 3 $\triangleright$ `Build a graph` H_t `on the blobs in` $\mathcal{C}'_{t-1}$ `as follows:`
Set $NS_t = F_t(F_t)^T$.
Set $FC_t(x, y) = 1$ if x, y are in the same blob in $\mathcal{C}'_{t-1}$ and $F_t(x, y) = 1$;
$FC_t(x, y) = 0$ otherwise.
Set $S_t = F_t(FC_t)^T + FC_t(F_t)^T$.
for any $C_u, C_v \in \mathcal{C}'_{t-1}$ **do**
if $C_u = \{x\}$ and $C_v = \{y\}$ **then**
$H_t(C_u, C_v) = 1$ if $NS_t(x, y) > (\alpha + \nu)n$;
$H_t(C_u, C_v) = 0$ otherwise.
else
$H_t(C_u, C_v) = 1$ if $\text{median}_{x \in C_u, y \in C_v} S_t(x, y) > \frac{|C_u| + |C_v|}{4}$;
$H_t(C_u, C_v) = 0$ otherwise.
end if
end for

Step 4 $\triangleright$ `Merge blobs`
while $\exists C_u, C_v$ with $H_t(C_u, C_v) = 1$ and $|C_u| + |C_v| > 4(\alpha + \nu)n$ **do**
Find the pair C_u, C_v with maximum $\text{median}_{x \in C_u, y \in C_v} \frac{S_t(x,y)}{|C_u| + |C_v|}$.
Merge the pair C_u, C_v, and update $\mathcal{C}'_{t-1}$. Recompute FC_t, S_t and H_t.
end while

Step 5 $\triangleright$ `Merge singletons`
while $\exists$ component V in H_t with $|\cup_{C \in V} C| \geq 4(\alpha + \nu)n$ **do**
Merge blobs in V, and update $\mathcal{C}'_{t-1}$. Recompute FC_t, S_t and H_t.
end while

Step 6 $\triangleright$ `Speed up`
if $\exists$ less than $\max\{4(\alpha + \nu)n, t/2\}$ singleton blobs in $\mathcal{C}'_{t-1}$ **then**
Merge each singleton with the non-singleton blob of highest median similarity.
Update $\mathcal{C}'_{t-1}$. Recompute FC_t, S_t and H_t.
end if
$\mathcal{C}'_t = \mathcal{C}'_{t-1}$.

Step 7 $\triangleright$ `Increase threshold`
$t = t + 1$.
end while

Output: Tree T with single points as leaves and internal nodes corresponding to the merges performed.

showing that for a fixed p in $X \setminus B$ and fixed $x \in A_p$, the first condition of the weak good neighborhood is still satisfied on a sufficiently large sample (Recall the definition of Property 5). Similarly, we can prove Lemma 15 showing that the second condition of the weak good neighborhood is also satisfied. Then, the similarity $\mathcal{K}$ satisfies the weak good neighborhood property with respect to the clustering induced over the sample (Lemma 12). Our final guarantee, Theorem 13, then follows from the lemmas.

Lemma 14 *Let $\mathcal{K}$ be a symmetric similarity function satisfying the weak (α, β, ν)-good neighborhood for the clustering problem (X, ℓ). Consider any fixed $p \in X \setminus B$ and any fixed $x \in A_p$. If the sample size satisfies $n = \Theta\left(\frac{1}{\alpha} \ln \frac{1}{\delta}\right)$, then with probability at least $1 - \delta$, x has at most $2\alpha n$ neighbors outside $A_p \cap S$ out of the $|A_p \cap S|$ nearest neighbors in $S \setminus B$.*

Lemma 15 *Let $\mathcal{K}$ be a symmetric similarity function satisfying the weak (α, β, ν)-good neighborhood for the clustering problem (X, ℓ). Consider any fixed $p \in X \setminus B$ and any fixed good point $x \in A_p$. If the sample size satisfies $n = \Theta\left(\frac{1}{\alpha} \ln \frac{1}{\delta}\right)$, then with probability at least $1 - \delta$, x has at most $2\alpha n$ neighbors outside $C(x) \cap S$ out of the $|A_p \cap S|$ nearest neighbors in $S \setminus B$.*

Lemma 12 *Let $\mathcal{K}$ be a symmetric similarity function satisfying the weak (α, β, ν)-good neighborhood for the clustering problem (X, ℓ). Furthermore, it satisfies that for any $p \in X \setminus B$, $|A_p| > 24(\alpha + \nu)N$. If the sample size satisfies $n = \Theta\left(\frac{1}{\min(\alpha,\nu)} \ln \frac{1}{\delta \min(\alpha,\nu)}\right)$, then with probability at least $1 - \delta$, $\mathcal{K}$ satisfies the $(2\alpha, \frac{15}{16}\beta, 2\nu)$-good neighborhood with respect to the clustering induced over the sample S.*

Proof Consider the first condition of the weak good neighborhood property. First, by Chernoff bounds, when $n \geq \frac{3}{\nu} \ln \frac{4}{\delta}$, we have that with probability at least $1 - \delta/4$, at most $2\nu n$ bad points fall into the sample. Next, by Lemma 14 and union bound, when $n = \Theta\left(\frac{1}{\alpha} \ln \frac{n}{\delta}\right)$ we have that with probability at least $1 - \delta/4$, for any point $p \in S \setminus B$, any point $x \in A_p \cap S$ has at most $2\alpha n$ neighbors outside $A_p \cap S$ out of the $|A_p \cap S|$ nearest neighbors in $S \setminus B$. Since $|A_p| > 24(\alpha + \nu)N$, we also have $|A_p \cap S| > 12(\alpha + \nu)n$ with probability at least $1 - \delta/4$. So the first condition of the weak good neighborhood property is satisfied.

Now consider the second condition. Fix C_i and a point $p \in (C_i \setminus B) \cap S$. When $n = \Theta\left(\frac{1}{\alpha} \ln \frac{n}{\delta}\right)$, with probability at least $1 - \delta/(8n)$, at least $\frac{15}{16}\beta$ fraction of points x in $A_p \cap S$ have all but at most αN nearest neighbors from $C_i \setminus B$ out of their $|C_i \setminus B|$ nearest neighbors in $X \setminus B$. Fix such a point $x \in A_p \cap S$. By Lemma 15, with probability at least $1 - \delta/(8n^2)$, it has all but at most $2\alpha n$ nearest neighbors from $(C_i \setminus B) \cap S$ out of their $|(C_i \setminus B) \cap S|$ nearest neighbors in $S \setminus B$. By union bound, we have that with probability at least $1 - \delta/4$, for any C_i and any $p \in C_i \setminus B$, at least $\frac{15}{16}\beta$ fraction of points in $A_p \cap S$ have all but at most $2\alpha n$ nearest neighbors from $(C_i \setminus B) \cap S$ out of their $|(C_i \setminus B) \cap S|$ nearest neighbors in $S \setminus B$. So the second condition is also satisfied.

Therefore, if $n = \Theta\left(\frac{1}{\min(\alpha,\nu)} \ln \frac{n}{\delta}\right)$, then with probability at least $1 - \delta$, the similarity function satisfies the $(2\alpha, 2\nu)$-good neighborhood property with respect to the clustering induced over the sample S. The lemma then follows from the fact that $n = \Theta\left(\frac{1}{\min(\alpha,\nu)} \ln \frac{1}{\delta \min(\alpha,\nu)}\right)$ implies $n = \Theta\left(\frac{1}{\min(\alpha,\nu)} \ln \frac{n}{\delta}\right)$. ■

Theorem 13 *Let $\mathcal{K}$ be a symmetric similarity function satisfying the weak (α, β, ν)-good neighborhood for the clustering problem (X, ℓ) with $\beta \geq \frac{14}{15}$. Furthermore, it satisfies that for any $p \in X \setminus B$, $|A_p| > 24(\alpha + \nu)N$. Then Algorithm 2 with parameters $n = \Theta\left(\frac{1}{\min(\alpha,\nu)} \ln \frac{1}{\delta \cdot \min(\alpha,\nu)}\right)$ produces a hierarchy with a pruning that is $(\nu + \delta)$-close to the target clustering with probability $1 - \delta$.*

Proof Note that by Lemma 12, with probability at least $1 - \delta/4$, we have that $\mathcal{K}$ satisfies the weak $(2\alpha, \frac{15}{16}\beta, 2\nu)$-good neighborhood with respect to the clustering induced over the sample. Then by Theorem 1, Algorithm 1 outputs a hierarchy T on the sample S with a pruning $\{u_1, \ldots, u_k\}$ such that $u_i \setminus B = (C_i \cap S) \setminus B$.

Now we want to show that $f_{u_1}, \ldots, f_{u_k}$ have error at most $\nu + \delta$ with probability at least $1 - \delta/2$. For convenience, let $u(x)$ be a shorthand of $u_{\ell(x)}$. Then it is sufficient to show that with probability at least $1 - \delta/2$, a $(1 - \delta)$ fraction of points $x \in X \setminus B$ have $f_{u(x)}(x) = 1$. Fix C_i and a point $x \in C_i \setminus B$. By Lemma 14, with probability at least $1 - \delta^2/2$, out of the $|A_x \cap S|$ nearest neighbors of x in $S \setminus B$, at most $2\alpha n$ can be outside A_x. Then out of the $6(\alpha + \nu)n$ nearest neighbors of x in S, at most $2(\alpha + \nu)n$ points are outside $A_x \cap S$. By Lemma 2, u_i contains $A_x \cap S$, so u_i must contain at least $4(\alpha + \nu)n$ points in $N_S(x)$. Consequently, any ancestor w of u_i, including u_i, has more points in $N_S(x)$ than any other sibling of w. Then we must have $f_w(x) = 1$ for any ancestor w of u_i. In particular, $f_{u_i}(x) = 1$. So, for any point $x \in X \setminus B$, with probability at least $1 - \delta^2/2$ over the draw of the random sample, $f_{u(x)}(x) = 1$. By Markov inequality, with probability at least $1 - \delta/2$, a $(1 - \delta)$ fraction of points $x \in X \setminus B$ have $f_{u(x)}(x) = 1$. ■

Appendix C. Strict Separation and Ward's Method

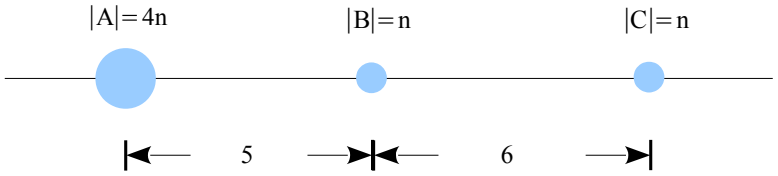


Figure 16: An example that satisfies the strict separation property but is not clustered successfully by Ward's minimum variance Method.

Here we describe an example showing that Ward's minimum variance method fails in the presence of unbalanced clusters. The clustering instance satisfies the strict separation property and thus the more general good neighborhood properties, but on this instance Ward's method leads to large classification error.

The instance is presented in Figure 16. It consists three groups of points on a line: Group A has $4n$ points, Group B has n points, and Group C has n points. The distances

between points in the same groups are 0, while the distances between points in A and points in B are 5, the distances between points in B and points in C are 6, the distances between points in A and points in C are 11.

It can be verified that the clustering $\{A \cup B, C\}$ satisfies the strict separation property. We now show that Ward's method will produce a tree that do not have this clustering as a pruning, and thus fails to cluster the instance. Recall that Ward's method starts with each point being a singleton cluster and at each step finds the pair of clusters that leads to minimum increase in total within-cluster variance after merging. Formally, it merges the two clusters U and V such that

$$(U, V) = \operatorname{argmin} \left[\mathrm{Var}(U \cup V) - \mathrm{Var}(U) - \mathrm{Var}(V) \right],$$

where

$$\mathrm{Var}(X) = \min_c \sum_{p \in X} \|p - c\|_2^2.$$

Since the distances between points in the same groups are 0, the method will first merge points in the same groups and forms three clusters A, B, and C. Now, merging A and B increases the variance by $20n$, while merging B and C increases the variance by $18n$. Therefore, B and C will be merged, and thus the best pruning in the tree produced is $\{A, B \cup C\}$. This leads to an error of $1/6 \approx 16.7\%$.

References

M. Ackerman, S. Ben-David, D. Loker, and S. Sabato. Clustering oligarchies. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2013.

S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 1990.

K. Bache and M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

M. F. Balcan and Y. Liang. Modeling and detecting community hierarchies. In *Proceedings of the International Workshop on Similarity-Based Pattern Analysis and Recognition*, 2013.

M. F. Balcan, A. Blum, and S. Vempala. A discriminative framework for clustering via similarity functions. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2008.

M. F. Balcan, A. Blum, and A. Gupta. Clustering under approximation stability. *Journal of ACM*, 2013.

A. Blum, A. Coja-Oghlan, A. Frieze, and S. Zhou. Separating populations with wide data: A spectral analysis. In *Algorithms and Computation*. 2007.

D. Bryant and V. Berry. A structured family of clustering and tree construction methods. *Advances in Applied Mathematics*, 2001.

K. Chaudhuri and S. Dasgupta. Rates of convergence for the cluster tree. *Advances in Neural Information Processing Systems*, 2010.

D. Cheng, R. Kannan, S. Vempala, and G. Wang. A divide-and-merge methodology for clustering. *ACM Transaction on Database Systems*, 2006.

S. Dasgupta and P. Long. Performance guarantees for hierarchical clustering. *Journal of Computer and System Sciences*, 2005.

R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. 2000.

B. Eriksson. Hierarchical clustering using randomly selected measurements. In *Proceedings of the IEEE Statistical Signal Processing Workshop*, 2012.

B. S. Everitt, S. Landau, M. Leese, and D. Stahl. *Hierarchical Clustering*. 2011.

J. Feldman, R. O'Donnell, and R. A. Servedio. Learning mixtures of product distributions over discrete domains. *SIAM Journal on Computing*, 2008.

M. T. Gallegos. Maximum likelihood clustering with outliers. In *Classification, Clustering, and Data Analysis*. Springer, 2002.

M. T. Gallegos and G. Ritter. A robust method for cluster analysis. *The Annals of Statistics*, 2005.

L. García-Escudero and A. Gordaliza. Robustness properties of k means and trimmed k means. *Journal of the American Statistical Association*, 94(447), 1999.

L. García-Escudero, A. Gordaliza, C. Matrán, and A. Mayo-Iscar. A general trimming approach to robust cluster analysis. *The Annals of Statistics*, 2008.

L. García-Escudero, A. Gordaliza, C. Matrán, and A. Mayo-Iscar. A review of robust clustering methods. *Advances in Data Analysis and Classification*, 4(2-3), 2010.

S. Gollapudi, R. Kumar, and D. Sivakumar. Programmable clustering. In *Symposium on Principles of Database Systems*, 2006.

J. C. Gower. A comparison of some methods of cluster analysis. *Biometrics*, 1967.

S. Guha, R. Rastogi, and K. Shim. CURE: an efficient clustering algorithm for large databases. In *ACM SIGMOD Record*, 1998.

C. Hennig. Dissolution point and isolation robustness: robustness criteria for general cluster analysis methods. *Journal of multivariate analysis*, 99(6), 2008.

A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., 1981.

A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 1999.

S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 1967.

B. King. Step-wise clustering procedures. *Journal of the American Statistical Association*, 1967.

H. W. Kuhn. The Hungarian method for the assignment algorithm. *Naval Research Logistics Quarterly*, 1955.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

M. Meilă and D. Heckerman. An experimental comparison of model-based clustering methods. *Machine Learning*, 2001.

A. Moitra and M. Saks. A polynomial time algorithm for lossy population recovery. In *Proceddings of the IEEE Annual Symposium on Foundations of Computer Science*, 2013.

G. Nagy. State of the art in pattern recognition. *Proceedings of the IEEE*, 1968.

M. Narasimhan, N. Jojic, and J. Bilmes. Q-clustering. *Advances in Neural Information Processing Systems*, 2006.

M. Punta, P. C. Coggill, R. Y. Eberhardt, J. Mistry, J. Tate, C. Boursnell, N. Pang, K. Forslund, G. Ceric, J. Clements, A. Heger, L. Holm, E. L. L. Sonnhammer, S. R. Eddy, A. Bateman, and R. D. Finn. The pfam protein families database. *Nucleic Acids Research*, 2012.

P. H. A. Sneath and R. R. Sokal. The principles and practice of numerical classification. *Numerical Taxonomy*, 1973.

K. Voevodski, M. F. Balcan, H. Röglin, S.-H. Teng, and Y. Xia. Active clustering of biological sequences. *Journal of Machine Learning Research*, 2012.

J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 1963.

A. Wigderson and A. Yehudayoff. Population recovery and partial identification. In *Proceedings of the IEEE Annual Symposium on Foundations of Computer Science*, 2012.

D. Wishart. Mode analysis: a generalization of nearest neighbour which reduces chaining effects. *Numerical Taxonomy*, 1969.

Journal of Machine Learning Research 15 (2014) 4053-4103 Submitted 4/13; Revised 6/14; Published 12/14

Parallelizing Exploration-Exploitation Tradeoffs in Gaussian Process Bandit Optimization

Thomas Desautels* TDESAUTELS@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit
University College London
Alexandra House, 17 Queen Square, London WC1N 3AR, UK

Andreas Krause KRAUSEA@ETHZ.CH
Department of Computer Science
ETH Zurich
Universitätstrasse 6, 8092 Zürich, Switzerland

Joel W. Burdick JWB@ROBOTICS.CALTECH.EDU
Department of Mechanical Engineering
California Institute of Technology
1200 E California Blvd., Pasadena, CA 91125, USA

Editor: Alexander Rakhlin

Abstract

How can we take advantage of opportunities for experimental parallelization in exploration-exploitation tradeoffs? In many experimental scenarios, it is often desirable to execute experiments simultaneously or in batches, rather than only performing one at a time. Additionally, observations may be both noisy and expensive. We introduce Gaussian Process Batch Upper Confidence Bound (GP-BUCB), an upper confidence bound-based algorithm, which models the reward function as a sample from a Gaussian process and which can select batches of experiments to run in parallel. We prove a general regret bound for GP-BUCB, as well as the surprising result that for some common kernels, the asymptotic average regret can be made independent of the batch size. The GP-BUCB algorithm is also applicable in the related case of a delay between initiation of an experiment and observation of its results, for which the same regret bounds hold. We also introduce Gaussian Process Adaptive Upper Confidence Bound (GP-AUCB), a variant of GP-BUCB which can exploit parallelism in an adaptive manner. We evaluate GP-BUCB and GP-AUCB on several simulated and real data sets. These experiments show that GP-BUCB and GP-AUCB are competitive with state-of-the-art heuristics.[1]

Keywords: Gaussian process, upper confidence bound, batch, active learning, regret bound

1. Introduction

Many problems require optimizing an unknown reward function, from which we can only obtain noisy observations. A central challenge is choosing actions that both explore (es-

∗. This research was carried out while TD was a student at the California Institute of Technology.

1. A previous version of this work appeared in the Proceedings of the 29th International Conference on Machine Learning, 2012.

timate) the function and exploit our knowledge about likely high reward regions in the function's domain. Carefully calibrating this exploration–exploitation tradeoff is especially important in cases where the experiments are costly, e.g., when each experiment takes a long time to perform and the time budget available for experiments is limited. In such settings, it may be desirable to execute several experiments in parallel. By parallelizing the experiments, substantially more information can be gathered in the same time-frame; however, future actions must be chosen without the benefit of intermediate results. One might conceptualize these problems as choosing "batches" of experiments to run simultaneously. The challenge is to assemble batches of experiments that both explore the function and exploit by focusing on regions with high estimated value.

Two key, interrelated questions arise: the *computational* question of how one should efficiently choose, out of the combinatorially large set of possible batches, those that are most effective; and the *statistical* question of how the algorithm's performance depends on the size of the batches (i.e., the degree of informational parallelism). In this paper, we address these questions by presenting GP-BUCB and GP-AUCB; these are novel, efficient algorithms for selecting batches of experiments in the Bayesian optimization setting where the reward function is modeled as a sample from a Gaussian process prior or has low norm in the associated Reproducing Kernel Hilbert Space.

In more detail, we provide the following main contributions:

- We introduce GP-BUCB, a novel algorithm for selecting actions to maximize reward in large-scale exploration-exploitation problems. GP-BUCB accommodates parallel or batch execution of the actions and the consequent observation of their reward. GP-BUCB may also be used in the setting of a bounded delay between initiation of an action and observation of its reward.
- We also introduce GP-AUCB, an algorithm which adaptively exploits parallelism to choose batches of actions, the sizes of which are limited by the conditional mutual information gained therein; this limit is such that the batch sizes are small when the algorithm selects actions for which it knows relatively little about the reward. Conversely, batch sizes may be large when the reward function is well known for the actions selected. We show that this adaptive parallelism is effective and can easily be parameterized using pre-defined limits.
- We prove sublinear bounds on the cumulative regret incurred by algorithms of a general class, including GP-BUCB and GP-AUCB, that also imply bounds on their rates of convergence.
- For some common kernels, we show that if the problem is initialized by making observations corresponding to an easily selected and provably bounded set of queries, the regret of GP-BUCB can be bounded to a constant multiplicative factor of the known regret bounds of the fully sequential GP-UCB algorithm of Srinivas et al. (2010, 2012). This implies (near-)linear speedup in the asymptotic convergence rates through parallelism.
- We demonstrate how execution of many UCB algorithms, including the GP-UCB, GP-BUCB, and GP-AUCB algorithms, can be drastically accelerated by *lazily evaluating* the posterior variance. This technique does not result in any loss in accuracy.

- We evaluate GP-BUCB and GP-AUCB on several synthetic benchmark problems, as well as two real data sets, respectively related to automated vaccine design and therapeutic spinal cord stimulation. We show that GP-BUCB and GP-AUCB are competitive with state-of-the-art heuristics for parallel Bayesian optimization. Under certain circumstances, GP-BUCB and GP-AUCB are competitive with sequential action selection under GP-UCB, despite having to cope with the disadvantage of delayed feedback.
- We consider more complex notions of execution cost in the batch and delay settings and identify areas of this cost and performance space where our algorithms make favorable tradeoffs and are therefore especially suitable for practical applications.

In the remainder of the paper, we first review the literature (Section 2) and formally describe the problem setting (Section 3). In the next section, we describe the GP-BUCB algorithm, present the main regret bound, which applies to a general class of algorithms using an upper confidence bound decision rule, and present corollaries bounding the regret of GP-BUCB and initialized GP-BUCB (Section 4). We extend this analysis to GP-AUCB, providing a regret bound for that algorithm, and discuss different possible stopping conditions for similar algorithms (Section 5). Next, we introduce the notion of lazy variance calculations (Section 6). We compare our algorithms' performance with each other and with several other algorithms across a variety of problem instances, including two real data sets (Section 7). Finally, we present our conclusions (Section 8).

2. Related Work

Our work builds on ideas from bandits, Bayesian optimization, and batch selection. In the following, we briefly review the literature in each of these areas.

2.1 Multi-armed Bandits

Exploration-exploitation tradeoffs have been classically studied in context of multi-armed bandit problems. These are sequential decision tasks where a single action is taken at each round, and a corresponding (possibly noisy) reward is observed. Early work has focused on the case of a finite number of candidate actions (arms), a total budget of actions which is at least as large as the number of arms, and payoffs that are independent across the arms (Robbins, 1952). In this setting, under some strong assumptions, optimal policies can be computed (Gittins, 1979). Optimistic allocation of actions according to upper-confidence bounds (UCB) on the payoffs has proven to be particularly effective (Auer et al., 2002). In many applications, the set of candidate actions is very large (or even infinite). In such settings, dependence between the payoffs associated with different decisions must be modeled and exploited. Various methods of introducing dependence include bandits with linear (Dani et al., 2008; Abernethy et al., 2008; Abbasi-Yadkori et al., 2011) or Lipschitz-continuous payoffs (Kleinberg et al., 2008; Bubeck et al., 2008), or bandits on trees (Kocsis and Szepesvári, 2006). In this paper we pursue a Bayesian approach to bandits, where fine-grained assumptions on the regularity of the reward function can be imposed through proper choice of the prior distribution over the payoff function. Concretely, we focus on Gaussian process priors, as considered by Srinivas et al. (2010).

2.2 Bayesian Optimization

The exploration-exploitation tradeoff has also been studied in Bayesian global optimization and response surface modeling, where Gaussian process (GP, see Rasmussen and Williams, 2006) models are often used due to their flexibility in incorporating prior assumptions about the payoff function's structure (Brochu et al., 2010). In addition to a model of the payoff function, an algorithm must have a method for selecting the next observation. Several bandit-like heuristics, such as Maximum Expected Improvement (Jones et al., 1998), Maximum Probability of Improvement (Mockus, 1989), Knowledge Gradient (Ryzhov et al., 2012), and upper-confidence-based methods (Cox and John, 1997), have been developed to balance exploration with exploitation and have been successfully applied in learning problems (e.g., Lizotte et al., 2007). In contrast, the Entropy Search algorithm of Hennig and Schuler (2012) seeks to take the action that will greedily decrease future losses, a less bandit-like and more optimization-focused heuristic. Recently, Srinivas et al. (2010, 2012) analyzed GP-UCB, an algorithm for this setting based on upper-confidence bound sampling, and proved bounds on its cumulative regret, and thus convergence rates for Bayesian global optimization. We build on this foundation and generalize it to the parallel setting.

2.3 Parallel Selection

To enable parallel selection, one must account for the delay between decisions and observations. Most existing approaches that can deal with such delay result in a multiplicative increase in the cumulative regret as the delay grows. Only recently, Dudik et al. (2011) demonstrated that it is possible to obtain regret bounds that only increase *additively* with the delay (i.e., the penalty becomes negligible for large numbers of decisions). However, the approach of Dudik et al. only applies to contextual bandit problems with finite decision sets, and thus not to settings with complex (even nonparametric) payoff functions. Similarly, contemporary work by Joulani et al. (2013) develops a meta-algorithm for converting sequential bandit algorithms to the delayed, finite decision set environment. While this algorithm has regret bounds which only increase additively with batch size, it does not generalize to the case of infinitely large decision sets and, by construction, does not take advantage of knowledge of pending observations, leading to redundant exploration, of particular concern when individual observations are expensive.

In contrast to these theoretical developments for finite bandits, there has been heuristic work on parallel Bayesian global optimization using GPs, e.g., by Ginsbourger et al. (2010). The state of the art is the *simulation matching* algorithm of Azimi et al. (2010), which uses the posterior of the payoff function at the beginning of the batch to simulate observations that the sequential algorithm would encounter if it could receive feedback during the batch, obtaining a number of Monte Carlo samples over future behaviors of the sequential algorithm. These Monte Carlo samples are then aggregated into a batch of observations which is intended to "closely match" the set of actions that would be taken by the sequential algorithm if it had been run with sequential feedback. To our knowledge, no theoretical results regarding the regret or convergence of this algorithm exist. We experimentally compare with this approach in Section 7. Azimi et al. (2012a) recently extended this construction to the batch classification setting.

Azimi et al. (2012b) also propose a very different algorithm that adaptively chooses the level of parallelism it will allow. This is done in a manner which depends on the expected prediction error between the posterior constructed with the simulated observations in the batch in progress versus the true posterior that would be available assuming the observations had actually been obtained. We also compare against this adaptive algorithm in Section 7.

Recently, Chen and Krause (2013) investigated batch-mode active learning using the notion of adaptive submodular functions. In contrast to our work, their approach focuses on active learning for estimation, which does not involve exploration–exploitation tradeoffs.

3. Problem Setting and Background

We wish to take a sequence of *actions* (or equivalently, make decisions) $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T \in D$, where D is the *decision set*, which is often (but not necessarily) a compact subset of $\mathbb{R}^d$. The subscript denotes the *round* in which that action was taken; each round is an opportunity for the algorithm to take one action. For each action $\boldsymbol{x}_t$, we observe a noisy scalar reward $y_t = f(\boldsymbol{x}_t) + \varepsilon_t$, where $f : D \to \mathbb{R}$ is an unknown function modeling the expected payoff $f(\boldsymbol{x})$ for each action $\boldsymbol{x}$. For now we assume that the noise variables ε_t are i.i.d. Gaussian with known variance σ_n^2, i.e., $\varepsilon_t \sim \mathcal{N}(0, \sigma_n^2)$, $\forall t \geq 1$. This assumption will be relaxed later in one of the cases of our main theorem. If the actions $\boldsymbol{x}_t$ are selected one at a time, each with the benefit of all observations $y_1, \ldots, y_{t-1}$ corresponding to previous actions $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}$, we shall refer to this case as the *strictly sequential* setting. In contrast, the main problem tackled in this paper is the challenging setting where action $\boldsymbol{x}_t$ must be selected using only observations $y_1, \ldots, y_{t'}$, where often $t' < t - 1$. Thus, less information is available for choosing actions as compared to the strictly sequential setting.

In selecting these actions, we wish to maximize the cumulative reward $\sum_{t=1}^{T} f(\boldsymbol{x}_t)$. Defining the *regret* of action $\boldsymbol{x}_t$ as $r_t = [f(\boldsymbol{x}^*) - f(\boldsymbol{x}_t)]$, where $\boldsymbol{x}^* \in \boldsymbol{X}^* = \operatorname{argmax}_{\boldsymbol{x} \in D} f(\boldsymbol{x})$ is an optimal action (assumed to exist, but not necessarily to be unique), we may equivalently think of maximizing the cumulative reward as minimizing the *cumulative regret*

$$R_T = \sum_{t=1}^{T} r_t.$$

By minimizing the regret, we ensure progress toward optimal actions uniformly over T. In fact, the *average regret*, R_T/T, is a natural upper bound on the suboptimality of the best action considered so far, i.e., $R_T/T \geq \min_{t \in 1,\ldots,T} [f(\boldsymbol{x}^*) - f(\boldsymbol{x}_t)]$ (where this minimum is often called the *simple regret*, Bubeck et al., 2009). Therefore bounds on the average regret imply convergence rates for global optimization. It is particularly desirable to show that R_T is sublinear, i.e., $o(T)$, such that the average regret (and thus the minimum regret) goes to zero for large T; an algorithm with this property is described as being "no-regret."

In Section 3.1, we formally define the problem setting of parallel selection. Sections 3.2 and 3.3 introduce mathematical background necessary for our analysis. Section 3.4 describes the GP-UCB algorithm and discusses why some simple attempts at generalizing it to the parallel setting are insufficient, setting the stage for GP-BUCB, the subject of Section 4.

3.1 The Problem: Parallel or Delayed Selection

In many applications, at time τ, we wish to select a *batch* of actions, e.g., $\boldsymbol{x}_\tau, ..., \boldsymbol{x}_{\tau+B-1}$, where B is the size of the batch, to be evaluated in parallel. One natural application is the design of high-throughput experiments, where several experiments are performed in parallel, but feedback is only received after the experiments have concluded. In other settings, the feedback is delayed. We can model both situations by selecting actions sequentially; however when choosing $\boldsymbol{x}_t$ in round t, we can only make use of the feedback obtained in rounds $1, \ldots, t'$, for some $t' \leq t-1$. Formally, we assume there is some mapping $\text{fb} : \mathbb{N} \to \{\mathbb{N}, 0\}$ (where $\mathbb{N}$ denotes the positive integers) such that $\text{fb}[t] \leq t-1, \ \forall t \in \mathbb{N}$, and when selecting an action at time t, we can use feedback up to and including round $\text{fb}[t]$. If $\text{fb}[t] = 0$, no observation information is available.

Here and in most of the remainder of the paper, we concentrate primarily on this perspective on parallelism, which we term the *pessimistic view*, in which we consider the problem of *coping effectively under inferior feedback*. Intuitively, given feedback such that $\text{fb}[t] \leq t-1$ and often $\text{fb}[t] < t-1$, an algorithm should be expected to underperform relative to the strictly sequential algorithm, which obtains feedback according to $\text{fb}[t] = t-1$. This view provides a natural benchmark; success is performing nearly as well as the strictly sequential algorithm, despite the disadvantageous feedback. The contrasting *optimistic view*, in which parallelism may confer an advantage over strictly sequential algorithms via the ability to take more than one action simultaneously, is equivalent to the pessimistic view via a reparameterization of time, if batches are constructed sequentially; the difference between the two is fundamentally the philosophical primacy of decision-making in the pessimistic view and the experimental process in the optimistic view. We examine our results from the optimistic perspective in Section 7.3 and in Figure 7. Unfortunately, this optimistic view of parallelism presents difficulties when comparing algorithms; there is less clearly a benchmark for comparing the regret suffered by two algorithms which have submitted the same number of batches but use different levels of parallelism, since they may at any time have different numbers of observations. We thus concentrate our analytical and experimental approach on the pessimistic view, while remaining motivated by its optimistic counterpart.

Different specifications of the feedback mapping $\text{fb}[t]$ can model a variety of realistic scenarios. As noted above, setting $B = 1$ and $\text{fb}[t] = t-1$ corresponds to the non-delayed, strictly sequential setting in which a single action is selected and the algorithm waits until the corresponding observation is returned before selecting the succeeding action. The *simple batch* setting, in which we wish to select batches of size B, can be captured by setting $\text{fb}[t]_{SB} = \lfloor (t-1)/B \rfloor B$, i.e.,

$$\text{fb}[t]_{SB} = \begin{cases} 0 & : t \in \{1, \ldots, B\} \\ B & : t \in \{B+1, \ldots, 2B\} \\ 2B & : t \in \{2B+1, \ldots, 3B\} \\ \vdots & \end{cases}.$$

Note that in the batch case, the time indexing within the batch is a matter of algorithmic construction, since the batch is built in a sequential fashion, but actions are initiated simultaneously and observations are received simultaneously. If we wish to formalize the problem of selecting actions when feedback from those actions is delayed by exactly B rounds, the

simple delay setting, we can simply define this feedback mapping as $\text{fb}[t]_{SD} = \max\{t-B, 0\}$. Note that in both the simple batch and delay cases, $B = 1$ is the strictly sequential case. In comparing these two simple cases for equal values of B, we observe that $\text{fb}[t]_{SB} \geq \text{fb}[t]_{SD}$, that is, the set of observations available in the simple batch case for selecting the tth action is always at least as large as in the simple delay case, suggesting that the delay case is in some sense "harder" than the batch case. As we will see, however, the regret bounds presented in this paper may be expressed in terms of the maximal number of pending observations (i.e., those which have been initiated, but are still incomplete), which is $B-1$ in both of these settings, resulting in unified proofs and regret bounds for the two cases.

More complex cases may also be described using a feedback mapping. For example, we may be interested in executing B experiments in parallel, where we can start a new experiment as soon as one finishes, but the length of each experiment is variable; this translates to a more complex delay setting in which the algorithm has a queue of pending observations of some finite size B and checks at each round to see whether the queue is full. If the queue is not full, the algorithm submits an action, and if it is full, it "balks," i.e., does not submit an action and continues waiting for room to open within the queue. This is a natural description of an agent which periodically monitors slow experimental processes and takes action when it discovers they have finished. Since the algorithm only selects a new action when the queue is not full, there can be at most $B-1$ pending observations at the time a new action is selected, as in the simple batch and delay cases. Again, the maximum number of pending observations is the key to bounding the regret.

Since the level of difficulty of a variety of settings may be described in terms of the maximum number of pending observations when selecting any action (which we set to be $B-1$), in our development of GP-BUCB and initialized GP-BUCB in Sections 4.4 and 4.5, we only assume that the mapping $\text{fb}[t]$ is specified as part of the problem instance and $t - \text{fb}[t] \leq B$ for a known constant B. Importantly, our algorithms do not need to know the full feedback mapping ahead of time. It suffices if $\text{fb}[t]$ is revealed to the algorithms at each time t.

3.2 Modeling f via Gaussian Processes

Regardless of when feedback is obtained, if we are to turn a finite number of observations into useful inference about the payoff function f, we will have to make assumptions about its structure. In particular, for large (possibly infinite) decision sets D there is no hope to do well, i.e., incur little regret or even simply converge to an optimal action, if we do not make any assumptions. For good performance, we must choose a regression model which is both simple enough to be learned and expressive enough to capture the relevant behaviors of f. One effective formalism is to model f as a sample from a Gaussian process[2] (GP) prior. A GP is a probability distribution across a class of—typically smooth—functions, which is parameterized by a kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$, which characterizes the smoothness of f, and a mean function $\mu(\boldsymbol{x})$. In the remainder of this section, we assume $\mu(\boldsymbol{x}) = 0$ for notational convenience, without loss of generality. We often also assume that $k(\boldsymbol{x}, \boldsymbol{x}) \leq 1$, $\forall \boldsymbol{x} \in D$, i.e., that the kernel is normalized; results obtained using this assumption can be generalized to any case where $k(\boldsymbol{x}, \boldsymbol{x})$ has a known bound. We write $f \sim \mathcal{GP}(\mu, k)$ to denote that

2. See Rasmussen and Williams (2006) for a thorough treatment.

we model f as sampled from such a GP. If noise is i.i.d. Gaussian and the distribution of f is conditional on a vector of observations $\boldsymbol{y}_{1:t-1} = [y_1, ..., y_{t-1}]^T$ corresponding to actions $\boldsymbol{X}_{t-1} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_{t-1}]^T$, one obtains a Gaussian posterior distribution $f(\boldsymbol{x})|\boldsymbol{y}_{1:t-1} \sim \mathcal{N}(\mu_{t-1}(\boldsymbol{x}), \sigma^2_{t-1}(\boldsymbol{x}))$ for each $\boldsymbol{x} \in D$, where

$$\mu_{t-1}(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{X}_{t-1})[K(\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-1}) + \sigma_n^2 I]^{-1}\boldsymbol{y}_{1:t-1} \text{ and} \tag{1}$$

$$\sigma^2_{t-1}(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) - K(\boldsymbol{x}, \boldsymbol{X}_{t-1})[K(\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-1}) + \sigma_n^2 I]^{-1}K(\boldsymbol{x}, \boldsymbol{X}_{t-1})^T. \tag{2}$$

In the above, $K(\boldsymbol{x}, \boldsymbol{X}_{t-1})$ denotes the row vector of kernel evaluations between $\boldsymbol{x}$ and the elements of $\boldsymbol{X}_{t-1}$, the set of actions taken in the past, and $K(\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-1})$ is the matrix of kernel evaluations where $[K(\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-1})]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\forall\, \boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{X}_{t-1}$, i.e., the covariance matrix of the values of f over the set so far observed. Since Equations (1) and (2) can be computed efficiently, closed-form posterior inference is computationally tractable in a GP distribution via linear algebraic operations.

3.3 Conditional Mutual Information

A number of information theoretic quantities will be essential to the analysis of the algorithms presented in this paper. In particular, we are interested in the mutual information $I(f; \boldsymbol{y}_A)$ between f and a set of observations $\boldsymbol{y}_A$, where these observations correspond to a set $A = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots\}$ and each $\boldsymbol{x}_i$ in A is also in D. For a GP, $I(f; \boldsymbol{y}_A)$ is

$$I(f; \boldsymbol{y}_A) = H(\boldsymbol{y}_A) - H(\boldsymbol{y}_A \,|\, f) = \frac{1}{2} \log \left| \mathbf{I} + \sigma_n^{-2} K(A, A) \right|,$$

where $K(A, A)$ is the covariance matrix of the values of f at the elements of the set A, $H(\boldsymbol{y}_A)$ is the differential entropy of the probability distribution over the set of observations $\boldsymbol{y}_A$, and $H(\boldsymbol{y}_A \,|\, f)$ is the corresponding value when the distribution over $\boldsymbol{y}_A$ is conditioned on f. Note that for a GP, since $\boldsymbol{y}_A$ only depends on the values of f at A, denoted $f(A)$, it follows that $H(\boldsymbol{y}_A \,|\, f) = H(\boldsymbol{y}_A \,|\, f(A))$ and so $I(f; \boldsymbol{y}_A) = I(f(A); \boldsymbol{y}_A)$.

The *conditional mutual information* with respect to f resulting from observations $\boldsymbol{y}_A$, given previous observations $\boldsymbol{y}_S$, is defined (for two finite sets $A, S \subseteq D$) as

$$I(f; \boldsymbol{y}_A \mid \boldsymbol{y}_S) = H(\boldsymbol{y}_A \mid \boldsymbol{y}_S) - H(\boldsymbol{y}_A \mid f, \boldsymbol{y}_S) = H(\boldsymbol{y}_A \mid \boldsymbol{y}_S) - H(\boldsymbol{y}_A \mid f),$$

where the second equality follows from conditional independence of the observations given f. The conditional mutual information gained from observations $\boldsymbol{y}_A$ can also be calculated as a sum of the marginal conditional mutual information gains of each observation in $\boldsymbol{y}_A$; conditioned on $\boldsymbol{y}_S$, and for $A = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T\}$, this sum is

$$I(f; \boldsymbol{y}_A \mid \boldsymbol{y}_S) = \sum_{t=1}^{T} \log\left(1 + \sigma_n^{-2}\sigma^2_{t-1}(\boldsymbol{x}_t)\right), \tag{3}$$

where the term $\sigma^2_{t-1}(\boldsymbol{x}_t)$ is the posterior variance over $f(\boldsymbol{x}_t)$, conditioned on $\boldsymbol{y}_S$ and $\{y_1, ..., y_{t-1}\} \subseteq \boldsymbol{y}_A$. It is important to note that $\sigma^2_{t-1}(\boldsymbol{x}_t)$, given by Equation (2), is independent of the values of the observations. Since the sum's value can thus be calculated without making the observations (i.e., during the course of assembling a batch), it is possible to calculate the mutual information that will be gained from any hypothetical set of

Algorithm 1 GP-UCB

Input: Decision set D, GP prior μ_0, σ_0, kernel function $k(\cdot,\cdot)$.
for $t = 1, 2, \ldots, T$ **do**
  Choose $\boldsymbol{x}_t = \text{argmax}_{\boldsymbol{x}\in D}[\mu_{t-1}(\boldsymbol{x}) + \alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x})]$
  Compute $\sigma_t(\cdot)$ via Equation (2)
  Obtain $y_t = f(\boldsymbol{x}_t) + \varepsilon_t$
  Perform Bayesian inference to obtain $\mu_t(\cdot)$ via Equation (1)
end for

observations. We will also be interested in the *maximum* information gain with respect to f obtainable from observations $\boldsymbol{y}_A$ corresponding to any set of actions A, where $|A| \leq T$,

$$\gamma_T = \max_{A\subseteq D,\, |A|\leq T} I(f; \mathbf{y}_A). \tag{4}$$

3.4 The GP-UCB Approach for Strictly Sequential Selection

Modeling f as a sample from a GP has the major benefit that the predictive uncertainty can be used to guide exploration and exploitation. This is done via a decision rule, by which the algorithm selects actions $\boldsymbol{x}_t$. While several heuristics, such as Expected Improvement (Mockus et al., 1978) and Most Probable Improvement (Mockus, 1989) have been effectively employed in practice, nothing is known about their convergence properties in the case of noisy observations. Srinivas et al. (2010), guided by the success of upper-confidence-based sampling approaches for multi-armed bandit problems (Auer, 2002), analyzed the Gaussian process Upper Confidence Bound (GP-UCB) decision rule,

$$\boldsymbol{x}_t = \operatorname*{argmax}_{\boldsymbol{x}\in D} \left[\mu_{t-1}(\boldsymbol{x}) + \alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x})\right]. \tag{5}$$

This decision rule uses α_t, a domain-specific time-varying parameter, to trade off exploitation (sampling $\boldsymbol{x}$ with high mean) and exploration (sampling $\boldsymbol{x}$ with high standard deviation). Srinivas et al. (2010) showed that, with proper choice of α_t, the cumulative regret of GP-UCB grows sublinearly for many commonly used kernel functions. This algorithm is presented in simplified pseudocode as Algorithm 1.

Implicit in the definition of the GP-UCB decision rule is the corresponding confidence interval for each $\boldsymbol{x} \in D$,

$$C_t^{\text{seq}}(\boldsymbol{x}) \equiv \left[\mu_{t-1}(\boldsymbol{x}) - \alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x}),\ \mu_{t-1}(\boldsymbol{x}) + \alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x})\right], \tag{6}$$

where this confidence interval's upper confidence bound is the value of the argument of the decision rule. For this (or any) confidence interval, we will refer to the difference between the uppermost limit and the lowermost, here $w = 2\alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x})$, as the *width*. This confidence interval is based on the posterior over f given $\boldsymbol{y}_{1:t-1}$; a new confidence interval is created for round $t+1$ after adding y_t to the set of observations. Srinivas et al. (2010) carefully select α_t such that a union bound over all $t \geq 1$ and $\boldsymbol{x} \in D$ yields a high-probability guarantee of confidence interval correctness; it is this guarantee and the direct relationship

between confidence intervals and the decision rule which enable the construction of high-probability regret bounds. Using this guarantee, Srinivas et al. (2010) then prove that the cumulative regret of the GP-UCB algorithm can be bounded as $R_T = O(\sqrt{T\alpha_T\gamma_T})$, where α_T is the confidence interval width multiplier described above. For many commonly used kernel functions, Srinivas et al. (2010) show that γ_T grows sublinearly and α_T only needs to grow polylogarithmically in T, implying that R_T is also sublinear; thus $R_T/T \to 0$ as $T \to \infty$, i.e., GP-UCB is a no-regret algorithm.

Motivated by the strong theoretical and empirical performance of GP-UCB (Srinivas et al., 2010, 2012), we explore generalizations to batch and parallel selection (i.e., $B > 1$). One naïve approach would be to update the GP-UCB score, Equation (5), only once new feedback becomes available, selecting the same action at each time step between acquisitions of new observations. In the case that the observation noise model is Gaussian, the bound of Srinivas et al. (2010) can be used together with reparameterization of time to bound the regret to no more than a factor of $\sqrt{B}$ greater than the GP-UCB algorithm. In empirical tests (Online Appendix 2), this algorithm does not explore sufficiently to perform well early on, making it of limited practical interest. To encourage more exploration, one may instead require that no action is selected twice within a batch (i.e., simply rank actions according to the GP-UCB score, and pick actions in order of decreasing score until new feedback is available). However, since f often varies smoothly, so does the GP-UCB score; under some circumstances, this algorithm would also suffer from limited exploration. Further, if the optimal set $\boldsymbol{X}^* \subseteq D$ is of size $|\boldsymbol{X}^*| < B$ and there is a finite gap between the rewards $f(\boldsymbol{x}^*)$ and $f(\boldsymbol{x})$ for all $\boldsymbol{x}^* \in \boldsymbol{X}^*, \boldsymbol{x} \notin \boldsymbol{X}^*$, the algorithm is suffers linear regret, since some $\boldsymbol{x} \notin \boldsymbol{X}^*$ must be included in every batch. This algorithm also underperforms in empirical tests (Online Appendix 2). These naïve algorithms have clear shortcomings because they do not simultaneously select diverse sets of actions and ensure appropriate convergence behavior.

In the following, we introduce the *Gaussian process Batch Upper Confidence Bound (GP-BUCB)* algorithm, which successfully balances these competing imperatives. GP-BUCB encourages diversity in exploration, uses past information in a principled fashion, and yields strong performance guarantees. We also extend it and develop the *Gaussian process Adaptive Upper Confidence Bound (GP-AUCB)* algorithm, which retains the theoretical guarantees of the GP-BUCB algorithm, but chooses batches of variable length in an adaptive, data-driven manner.

4. The GP-BUCB Algorithm and Regret Bounds

We introduce the GP-BUCB algorithm in Section 4.1. Section 4.2 states the paper's major theorem, a bound on the cumulative regret of a general class of algorithms including GP-BUCB and GP-AUCB. This main result is in terms of a quantity C, a bound on information used within a batch; this quantity is examined in detail in Section 4.3. Using these insights, Section 4.4 provides a corollary, bounding the regret of GP-BUCB specifically. Section 4.5 improves this regret bound by initializing GP-BUCB with a finite set of observations.

4.1 GP-BUCB: An Overview

A key property of GPs is that the predictive variance at time t, Equation (2), only depends on $\boldsymbol{X}_{t-1} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}\}$, i.e., *where* the observations are made, but not *which*

Algorithm 2 GP-BUCB

Input: Decision set D, GP prior μ_0, σ_0, kernel function $k(\cdot,\cdot)$, feedback mapping $\mathrm{fb}[\cdot]$.
for $t = 1, 2, \ldots, T$ **do**
  Choose $\boldsymbol{x}_t = \mathrm{argmax}_{\boldsymbol{x}\in D}[\mu_{\mathrm{fb}[t]}(\boldsymbol{x}) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x})]$
  Compute $\sigma_t(\cdot)$ via Equation (2)
  if $\mathrm{fb}[t] < \mathrm{fb}[t+1]$ **then**
    Obtain $y_{t'} = f(\boldsymbol{x}_{t'}) + \varepsilon_{t'}$ for $t' \in \{\mathrm{fb}[t]+1, \ldots, \mathrm{fb}[t+1]\}$
    Perform Bayesian inference to obtain $\mu_{\mathrm{fb}[t+1]}(\cdot)$ via Equation (1)
  end if
end for

values $\boldsymbol{y}_{1:t-1} = [y_1, \ldots, y_{t-1}]^T$ were actually observed. Thus, it is possible to compute the posterior variance that would be used by the sequential GP-UCB decision rule, Equation (5), even while certain observations are not yet available. In contrast, the predictive mean using in Equation (1) does depend on the actual observations. A natural approach towards parallel exploration is therefore to replace the GP-UCB decision rule, Equation (5), with a decision rule that sequentially chooses actions within the batch using all the information that is available so far,

$$\boldsymbol{x}_t = \operatorname*{argmax}_{\boldsymbol{x}\in D}\left[\mu_{\mathrm{fb}[t]}(\boldsymbol{x}) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x})\right]. \tag{7}$$

Here, the parameter β_t has a role analogous to the parameter α_t in the GP-UCB algorithm. The confidence intervals corresponding to this decision rule are of the form

$$C_t^{\mathrm{batch}}(\boldsymbol{x}) \equiv \left[\mu_{\mathrm{fb}[t]}(\boldsymbol{x}) - \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}),\ \mu_{\mathrm{fb}[t]}(\boldsymbol{x}) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x})\right]. \tag{8}$$

Note that this approach is equivalent to running the strictly sequential GP-UCB algorithm based on *hallucinated observations*. Concretely, we *hallucinate* observations $\boldsymbol{y}_{\mathrm{fb}[t]+1:t-1}$ for those observations that have not yet been received, simply using the most recently updated posterior mean, i.e., $\boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} = [\mu_{\mathrm{fb}[t]}(\boldsymbol{x}_{\mathrm{fb}[t]+1}), \ldots, \mu_{\mathrm{fb}[t]}(\boldsymbol{x}_{t-1})]$. As a consequence, the mean of the posterior including these hallucinated observations remains precisely $\mu_{\mathrm{fb}[t]}(\boldsymbol{x})$, but the posterior variance decreases.

The resulting GP-BUCB algorithm is shown in pseudocode as Algorithm 2. This approach naturally encourages diversity in exploration by taking into account the change in predictive variance that will eventually occur after receiving the pending observations; since the payoffs of "similar" actions are assumed to co-vary, exploring one action will automatically reduce the predictive variance of similar actions, and thus their value in terms of exploration. This decision rule appropriately deprecates those observations which will be made partially redundant by the acquisition of the pending observations, resulting in a more correct valuation of exploring any $\boldsymbol{x}$ in D.

The disadvantage of this approach appears as the algorithm progresses deeper into the batch. At each time t, the width of the confidence intervals $C_t^{\mathrm{batch}}(\boldsymbol{x})$ is proportional to $\sigma_{t-1}(\boldsymbol{x})$. As desired, shrinking the confidence intervals with respect to the start of the batch by using this standard deviation enables GP-BUCB to avoid exploratory redundancy. However, as an undesired side-effect, doing so conflates the information which is actually available, gained via the observations $\boldsymbol{y}_{1:\mathrm{fb}[t]}$, with the hallucinated information corresponding

to actions $\boldsymbol{x}_{\text{fb}[t]+1}$ through $\boldsymbol{x}_{t-1}$. Thus, the posterior reflected by $\sigma_{t-1}(\boldsymbol{x})$ is "overconfident" about the algorithm's actual state of knowledge of the function. This is problematic when using the confidence intervals to bound the regret.

To build an algorithm with rigorous guarantees on its performance while still avoiding exploratory redundancy, we must control for this overconfidence. One measure of overconfidence is the ratio $\sigma_{\text{fb}[t]}(\boldsymbol{x})/\sigma_{t-1}(\boldsymbol{x})$, which is the ratio of the width of the confidence interval derived from the set of actual observations $\boldsymbol{y}_{1:\text{fb}[t]}$ to the width of the confidence interval derived from the partially hallucinated set of observations $\boldsymbol{y}_{1:t-1}$. This ratio is related to $I(f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]})$, the hallucinated conditional mutual information with respect to $f(\boldsymbol{x})$ (as opposed to the whole of f), as follows:

Proposition 1 *The ratio of the standard deviation of the posterior over $f(\boldsymbol{x})$, conditioned on observations $\boldsymbol{y}_{1:fb[t]}$, to that conditioned on $\boldsymbol{y}_{1:fb[t]}$ and hallucinated observations $\boldsymbol{y}_{fb[t]+1:t-1}$ is*

$$\frac{\sigma_{fb[t]}(\boldsymbol{x})}{\sigma_{t-1}(\boldsymbol{x})} = \exp\Big(I(f(\boldsymbol{x}); \boldsymbol{y}_{fb[t]+1:t-1} \mid \boldsymbol{y}_{1:fb[t]})\Big).$$

Proof The proposition follows from the fact that

$$\begin{aligned} I(f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} | \boldsymbol{y}_{1:\text{fb}[t]}) &= H(f(\boldsymbol{x}) | \boldsymbol{y}_{1:\text{fb}[t]}) - H(f(\boldsymbol{x}) | \boldsymbol{y}_{1:t-1}) \\ &= 1/2 \log(2\pi e \sigma^2_{\text{fb}[t]}(\boldsymbol{x})) - 1/2 \log(2\pi e \sigma^2_{t-1}(\boldsymbol{x})) \\ &= \log(\sigma_{\text{fb}[t]}(\boldsymbol{x})/\sigma_{t-1}(\boldsymbol{x})). \end{aligned}$$

■

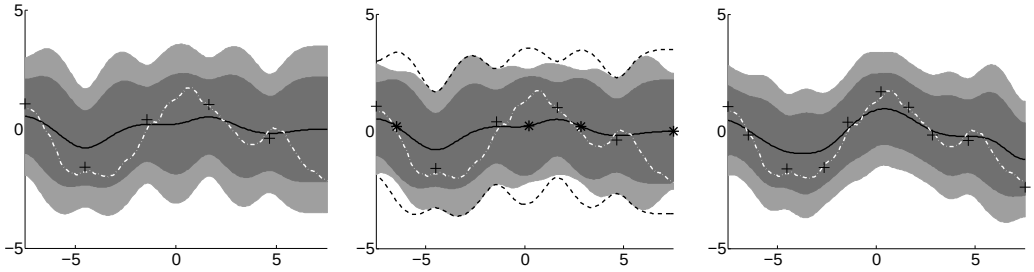
Crucially, if there exists some constant C, such that $I(f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]}) \leq C$, $\forall \boldsymbol{x} \in D, \forall t \geq 1$, the ratio $\sigma_{\text{fb}[t]}(\boldsymbol{x})/\sigma_{t-1}(\boldsymbol{x})$ can also be bounded for every $\boldsymbol{x} \in D$ as follows:

$$\frac{\sigma_{\text{fb}[t]}(\boldsymbol{x})}{\sigma_{t-1}(\boldsymbol{x})} = \exp\Big(I(f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]})\Big) \leq \exp(C). \tag{9}$$

Armed with such a bound, the algorithm can be modified to compensate for its overconfidence. Our goal is to compensate in a way that allows the algorithm to avoid redundancy, while guaranteeing accurate confidence intervals for the sake of deriving regret bounds. Our strategy is to increase the width of the confidence intervals (through proper choice of the parameter β_t), such that the confidence intervals used by GP-BUCB are conservative in their use of the hallucinated information and consequently still contain the payoff function $f(\boldsymbol{x})$ with high probability. More precisely, we will require that $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x}) \subseteq C^{\text{batch}}_t(\boldsymbol{x})$ for all t at which we select actions and all $\boldsymbol{x} \in D$; that is, the batch algorithm's confidence intervals are sufficiently large to guarantee that even for the last action selection in the batch, they contain the confidence intervals used by the GP-UCB algorithm given $\boldsymbol{y}_{1:\text{fb}[t]}$, as defined in Equation (6). Srinivas et al. (2010) provide choices of α_t such that the resulting confidence intervals have a high-probability guarantee of correctness $\forall t \geq 1, \boldsymbol{x} \in D$. Thus, if it can be shown that $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x}) \subseteq C^{\text{batch}}_t(\boldsymbol{x})$, $\forall \boldsymbol{x} \in D$, $t \in \mathbb{N}$, the batch confidence intervals inherit the high-probability guarantee of correctness.

Fortunately, the relationship between $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x})$ and $C^{\text{batch}}_t(\boldsymbol{x})$ is simple; since the partially hallucinated posterior has the same mean as that based on only $\boldsymbol{y}_{1:\text{fb}[t]}$,

$$C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x}) \subseteq C^{\text{batch}}_t(\boldsymbol{x}) \iff \beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x}) \geq \alpha^{1/2}_{\text{fb}[t]} \sigma_{\text{fb}[t]}(\boldsymbol{x}).$$



(a) *Beginning of batch 2* (b) *Last decision of batch 2* (c) *Beginning of batch 3*

Figure 1: (a): The confidence intervals $C^{\mathrm{seq}}_{\mathrm{fb}[t]+1}(\boldsymbol{x})$ (dark), computed from previous noisy observations $\boldsymbol{y}_{1:\mathrm{fb}[t]}$ (crosses), are centered around the posterior mean (solid black) and contain $f(\boldsymbol{x})$ (white dashed) w.h.p. To avoid overconfidence, GP-BUCB chooses $C^{\mathrm{batch}}_{\mathrm{fb}[t]+1}(\boldsymbol{x})$ (light gray) such that even in the worst case, the succeeding confidence intervals in the batch, $C^{\mathrm{batch}}_{\tau}(\boldsymbol{x})$, $\forall \tau : \mathrm{fb}[\tau] = \mathrm{fb}[t]$, will contain $C^{\mathrm{seq}}_{\mathrm{fb}[t]+1}(\boldsymbol{x})$. (b): Due to the observations that GP-BUCB "hallucinates" (stars), the outer posterior confidence intervals $C^{\mathrm{batch}}_{t}(\boldsymbol{x})$ shrink from their values at the start of the batch (black dashed), but still contain $C^{\mathrm{seq}}_{\mathrm{fb}[t]+1}(\boldsymbol{x})$, as desired. (c): Upon selection of the last action of the batch, the feedback for all actions is obtained, and for the subsequent action selection in round t', new confidence intervals $C^{\mathrm{seq}}_{\mathrm{fb}[t']+1}(\boldsymbol{x})$ and $C^{\mathrm{batch}}_{\mathrm{fb}[t']+1}(\boldsymbol{x})$ are computed.

If we have a suitable bound on $\sigma_{\mathrm{fb}[t]}(\boldsymbol{x})/\sigma_{t-1}(\boldsymbol{x})$ via Equation (9), all that remains is to choose β_t appropriately. If we do so by using a uniform, multiplicative increase with respect to $\alpha_{\mathrm{fb}[t]}$ for every $\boldsymbol{x} \in D$ and $t \in \mathbb{N}$, the desired redundancy avoidance property of these confidence intervals is simultaneously maintained, since the actions corresponding to pending observations (and related actions) are deprecated as if the observations had actually been obtained. Figure 1 illustrates this idea. The problem of developing a parallel algorithm with bounded delay is thus reduced to finding a value C such that $I(f(\boldsymbol{x}); \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \leq C$, $\forall \boldsymbol{x} \in D, \forall t \geq 1$, thus allowing us to select β_t to guarantee the containment of the reference sequential confidence intervals by their batch counterparts.

4.2 General Regret Bound

Our main theorem bounds the regret of GP-BUCB and related algorithms. This regret bound is formulated in terms of a bound C, which we assume to be known to the algorithm, on the maximum amount of conditional mutual information which is hallucinated with respect to $f(\boldsymbol{x})$ for any $\boldsymbol{x}$ in D. We defer discussion of methods of obtaining such a bound to Section 4.3. This bound is used to relate confidence intervals used to select actions, which incorporate this hallucinated information, to the posterior confidence intervals as of the last feedback obtained, which contain the payoff function f with high probability. This theorem holds under any of three different assumptions about f, studied by Srinivas et al. (2012) in the case of the GP-UCB algorithm, which may all be of practical interest. In particular, it

holds even if the assumption that f is sampled from a GP is replaced by the assumption that f has low norm in the associated Reproducing Kernel Hilbert Space (RKHS).[3]

Theorem 2 *Specify $\delta \in (0,1)$ and let γ_t be as defined in Equation (4). Let there exist a mapping $fb[t]$ (possibly revealed online) that dictates at which rounds new feedback becomes available. Model the payoff function f via a Gaussian process prior with bounded variance, such that for any $\boldsymbol{x}$ in the decision set D, $k(\boldsymbol{x},\boldsymbol{x}) \leq 1$. Suppose one of the following sets of assumptions holds:*

Case 1: D is a finite set and f is sampled from the assumed GP prior. The noise variables ϵ_t are i.i.d., $\epsilon_t \sim \mathcal{N}(0,\sigma_n^2)$. Choose $\alpha_t = 2\log(|D|t^2\pi^2/6\delta)$.

Case 2: $D \subseteq [0,l]^d$ is compact and convex, with $d \in \mathbb{N}$, $l > 0$, and f is sampled from the assumed zero-mean GP prior. The noise variables ϵ_t are i.i.d., $\epsilon_t \sim \mathcal{N}(0,\sigma_n^2)$. The kernel $k(\boldsymbol{x},\boldsymbol{x}')$ is such that the following bound holds with high probability on the derivatives of GP sample paths f, where a and b are constants such that $a \geq \delta/(4d)$, $b > 0$ and $bl\sqrt{\log(4da/\delta)}$ is an integer:

$$Pr\left\{\sup_{\boldsymbol{x}\in D} |\partial f/\partial x_j| > L\right\} \leq ae^{-(L/b)^2},\ j = 1,\ldots,d.$$

Choose $\alpha_t = 2\log(2t^2\pi^2/(3\delta)) + 2d\log\left(t^2dbl\sqrt{\log(4da/\delta)}\right)$.

Case 3: D is arbitrary and the squared RKHS norm of f under the kernel assumed is bounded as $||f||_k^2 \leq M$ for some constant M. The noise variables ε_t form an arbitrary *martingale difference sequence (meaning that $\mathbb{E}[\varepsilon_t \,|\, \varepsilon_1,\ldots,\varepsilon_{t-1}] = 0$ for all $t \in \mathbb{N}$), uniformly bounded by σ_n. Choose $\alpha_t = 2M + 300\gamma_t \ln^3(t/\delta)$.*

Employ the GP posterior and the GP-BUCB update rule, Equation (7), to select actions $\boldsymbol{x}_t \in D$ for all $t \geq 1$, using $\beta_t = \exp(2C)\alpha_{fb[t]+1}$ (Cases 1 & 3) or $\beta_t = \exp(2C)\alpha_t$ (Case 2), where $C > 0$ and

$$I(f(\boldsymbol{x}); \boldsymbol{y}_{fb[t]+1:t-1} \mid \boldsymbol{y}_{1:fb[t]}) \leq C, \tag{10}$$

for all $t \geq 1$ and all $\boldsymbol{x} \in D$. Under these conditions, the following statement holds with regard to the cumulative regret:

$$\Pr\left\{R_T \leq \sqrt{C_1 T \exp(2C)\alpha_T\gamma_T} + 2, \forall\, T \geq 1\right\} \geq 1-\delta,$$

where $C_1 = 8/\log(1+\sigma_n^{-2})$.

Proof The proof of this result is presented in Appendix A. ■

First, note that this guarantee holds for any amount of time the algorithm is allowed to run, since the algorithm does not use knowledge of how many actions it may yet take; thus, with high probability, R_T is less than the given expression for every T less than or equal to the number of executed actions. Second, the key quantity that controls the regret in Theorem 2 is C, the bound in Equation (10) on the maximum conditional mutual information obtainable within a batch with respect to $f(\boldsymbol{x})$ for any $\boldsymbol{x} \in D$. In particular, the cumulative regret bound of Theorem 2 is a factor $\exp(C)$ larger than the regret bound for the sequential ($B = 1$) GP-UCB algorithm. Various choices of the key parameter C are explored in the following sections.

3. See Schölkopf and Smola (2002).

4.3 Suitable Choices for C

The significance of a bound C on the information hallucinated with respect to any $f(\boldsymbol{x})$ arises through this quantity's ability to bound the degree of contamination of the GP-BUCB confidence intervals, given by Equation (8), with hallucinated information.

Two properties of the mutual information in this setting are particularly useful. These properties are monotonicity (adding an element $\boldsymbol{x}$ to the set A cannot decrease the mutual information between f and the corresponding set of observations $\boldsymbol{y}_A$) and submodularity (the increase in mutual information between f and $\boldsymbol{y}_A$ with the addition of an element $\boldsymbol{x}$ to set A cannot be greater than the corresponding increase in mutual information if $\boldsymbol{x}$ is added to A', where $A' \subseteq A$) (Krause and Guestrin, 2005). Submodularity arises because individual observations are conditionally independent, given f.

Using the time indexing notation developed in Section 3.1, the following results hold:

$$\forall \boldsymbol{x} \in D: \quad I(f(\boldsymbol{x}); \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \leq I(f; \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \tag{11}$$

$$\leq \max_{A \subseteq D, |A| \leq B-1} I(f; \boldsymbol{y}_A \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \tag{12}$$

$$\leq \max_{A \subseteq D, |A| \leq B-1} I(f; \boldsymbol{y}_A) = \gamma_{B-1}. \tag{13}$$

The first inequality follows from the monotonicity of mutual information, i.e., the information gained with respect to f as a whole must be at least as large as that gained with respect to $f(x)$. The second inequality holds because we specify the feedback mapping such that $t - \mathrm{fb}[t] \leq B$, and the third inequality holds due to the submodularity of the conditional mutual information.

Often, the terms on the right-hand side of these inequalities are easier to work with than $I(f(\boldsymbol{x}); \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]})$. The remainder of the paper is characterized by which inequality we employ in constructing an algorithm and choosing a suitable C to use with Equation (9) and Theorem 2; Sections 4.4 and 4.5 approach the problem via Inequalities (13) and (12), while Section 5.1 exploits Inequality (11) and Section 5.2 examines the consequences of directly bounding the local hallucinated information.

4.4 Corollary Regret Bound: GP-BUCB

The GP-BUCB algorithm requires that $t - \mathrm{fb}[t] \leq B,\ \forall t \geq 1$, and uses a value C such that, for any $t \in \mathbb{N}$,

$$\max_{A \subseteq D, |A| \leq B-1} I(f; \boldsymbol{y}_A \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \leq C, \tag{14}$$

thus bounding $I(f(\boldsymbol{x}); \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]})$ for all $\boldsymbol{x} \in D$ and $t \in \mathbb{N}$ via Inequality (12). Otherwise stated, in GP-BUCB, the local information gain with respect to any $f(\boldsymbol{x}), \boldsymbol{x} \in D, t \in \mathbb{N}$ is bounded by fixing the feedback times and then bounding the maximum conditional mutual information with respect to the entire function f which can be acquired by *any* algorithm which chooses any set of $B-1$ or fewer observations. This approach is sensible because such a bound C holds for any batches constructed with any algorithm. Following an approach which is less agnostic with regard to algorithm choice makes it quite difficult to disentangle the role of C in setting the exploration-exploitation tradeoff parameter β_t from its role as a bound on how much information is hallucinated by the algorithm; since a larger β_t encourages exploration under the GP-BUCB decision rule, Equation (7), a larger value of C

Algorithm 3 Uncertainty Sampling

Input: Decision set D, GP prior μ_0, σ_0, kernel function $k(\cdot,\cdot)$.
for $t = 1, 2, \ldots, T$ **do**
  Choose $\boldsymbol{x}_t = \text{argmax}_{\boldsymbol{x} \in D} \sigma_{t-1}(\boldsymbol{x})$
  Compute $\sigma_t(\cdot)$ via Equation (2)
end for

(and thus β_t) typically produces batches that explore more and thus use more hallucinated information.

It remains to choose a C which satisfies Inequality (14). We do so via Inequality (13). As noted in Section 4.3, mutual information is submodular with respect to the set of observed actions, and thus the maximum conditional mutual information which can be gained by making any set of observations is maximized when the set of observations currently available, to which these new observations will be added, is empty. Letting the maximum mutual information between f and any observation set of size $B-1$ be denoted γ_{B-1} and choosing $C = \gamma_{B-1}$ provides a bound on the possible local conditional mutual information gain for any $t \in \mathbb{N}$ and $\boldsymbol{x} \in D$, as in Inequality (13).

In practice, γ_{B-1} is often difficult to calculate; in general, this requires optimizing over the combinatorially large set of sets of actions of size $B-1$. However, Krause and Guestrin (2005) demonstrate that, due to the submodularity of the mutual information with respect to f in this setting, there is an easily obtained upper bound on γ_{B-1}. Specifically, they use uncertainty sampling, a greedy procedure, shown here as Algorithm 3, and show that $e/(e-1)\, I(f; \boldsymbol{y}^{US}_{B-1}) \geq \gamma_{B-1}$, where $I(f; \boldsymbol{y}^{US}_{B-1})$ is the information gained by observing the set of observations $\boldsymbol{y}^{US}_{B-1}$ corresponding to the actions $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{B-1}\}$ selected using uncertainty sampling. This insight enables efficient computation of upper bounds on γ_{B-1}.

Choosing $C = \gamma_{B-1}$ yields the following Corollary, a special case of Theorem 2:

Corollary 3 *Assume the GP-BUCB algorithm is employed with a constant B such that $t - fb[t] \leq B$ for all $t \geq 1$. Let $\delta \in (0,1)$, and let the requirements of one of the numbered cases of Theorem 2 be met. Choose $\beta_t = \exp(2C)\alpha_{fb[t]+1}$ (Cases 1 & 3) or $\beta_t = \exp(2C)\alpha_t$ (Case 2) and select actions $\boldsymbol{x}_t$ for all $t \geq 1$. Then*

$$\Pr\left\{R_T \leq \sqrt{C_1 T \exp(2\gamma_{B-1})\alpha_T \gamma_T} + 2, \quad \forall T \geq 1\right\} \geq 1 - \delta,$$

where $C_1 = 8/\log(1+\sigma_n^{-2})$ and γ_{B-1} and γ_τ are as defined in Equation (4).

Unfortunately, the choice $C = \gamma_{B-1}$ is not especially satisfying from the perspective of asymptotic scaling. The maximum information gain γ_{B-1} usually grows at least as $\Omega(\log B)$, implying that $\exp(C)$ grows at least linearly in B, yielding a regret bound which is also at least linear in B. Fortunately, the analysis of Section 4.5 shows that the GP-BUCB algorithm can be modified such that a constant choice of C *independent* of B suffices.

4.5 Better Bounds Through Initialization

To obtain regret bounds *independent* of batch size B, the monotonicity properties of conditional mutual information can again be exploited. This can be done by structuring GP-BUCB as a two-stage procedure. First, an *initialization set* D^{init} of size $|D^{\text{init}}| = T^{\text{init}}$ is

selected nonadaptively (i.e., without any feedback); following the selection of this entire set, feedback $\boldsymbol{y}_{D^{\text{init}}}$ for all actions in $D^{\text{init}} = \{\boldsymbol{x}_1^{\text{init}}, \ldots, \boldsymbol{x}_{T^{\text{init}}}^{\text{init}}\}$ is obtained. In the second stage, GP-BUCB is applied to the posterior Gaussian process distribution, conditioned on $\boldsymbol{y}_{D^{\text{init}}}$.

Notice that if we define

$$\gamma_T^{\text{init}} = \max_{A \subseteq D, |A| \leq T} I(f; \boldsymbol{y}_A \mid \boldsymbol{y}_{D^{\text{init}}}),$$

then, under the assumptions of Theorem 2, using $C = \gamma_{B-1}^{\text{init}}$, the regret of the two-stage algorithm is bounded by $R_T = O\left(T^{\text{init}} + (T\gamma_T \alpha_T \exp(2C))^{1/2}\right)$. In the following, we show that it is indeed possible to construct an initialization set D^{init} such that the size T^{init} is dominated by $(T\gamma_T\alpha_T \exp(2C))^{1/2}$, and—crucially—that $C = \gamma_{B-1}^{\text{init}}$ can be bounded *independently* of the batch size B.

The initialization set D^{init} which enables us to make this argument is constructed by running the uncertainty sampling algorithm (Algorithm 3) for T^{init} rounds and setting D^{init} to the selected actions. Note that uncertainty sampling can be viewed as a special case of the GP-BUCB algorithm with a constant prior mean of 0 and the requirement that for all $1 \leq t \leq T^{\text{init}}$, $\text{fb}[t] = 0$, i.e., no feedback is taken into account for the first T^{init} iterations.

Under this procedure, we have the following key result about the maximum residual information gain γ^{init}:

Lemma 4 *Suppose uncertainty sampling is used to generate an initialization set D^{init} of size T^{init}. Then*

$$\gamma_{B-1}^{init} \leq \frac{B-1}{T^{init}} \gamma_{T^{init}}. \tag{15}$$

Proof The proof of this lemma is presented in Appendix B. ■

Whenever γ_T is sublinear in T, the bound on $\gamma_{B-1}^{\text{init}}$ given by Inequality (15) converges to zero for sufficiently large T^{init}; thus for *any* constant $C > 0$, we can choose T^{init} as a function of B such that $\gamma_{B-1}^{\text{init}} < C$. Using this choice of C in Theorem 2 bounds the post-initialization regret. In order to derive bounds on T^{init}, we in turn need a bound on γ_T which is analytical and sublinear. Fortunately, Srinivas et al. (2010) prove suitable bounds on how the information gain γ_T grows for some of the most commonly used kernels. We summarize our analysis below in Theorem 5. For sake of notation, define R_T^{seq} to be the regret bound of Corollary 3 with $B = 1$ (i.e., that of Srinivas et al., 2010, associated with the sequential GP-UCB algorithm).

Theorem 5 *Suppose the assumptions of one of the cases of Theorem 2 are satisfied. Further, suppose the kernel and T^{init} are as listed in Table 1, and $B \geq 2$. Fix $\delta > 0$. Let R_T be the cumulative regret at round T of the two-stage initialized GP-BUCB algorithm, which ignores feedback for the first T^{init} rounds. Then there exists a constant C' independent of B such that*

$$Pr\left\{R_T \leq C' R_T^{seq} + 2||f||_\infty T^{init}, \forall T \geq 1\right\} \geq 1 - \delta, \tag{16}$$

where C' takes the value shown in Table 1.

Proof The proof of this result and the values in Table 1 are presented in Appendix B. ■

In Table 1, $\lceil \cdot \rceil$ denotes the first integer greater than or equal to the argument. Note that

Kernel Type	**Size T^{init} of Initialization Set D^{init}**	**Regret Multiplier C'**
Linear: $\gamma_t \leq \eta d \log(t+1)$	$\left\lceil \max \left[\log(B), \frac{\log\eta + \log d + 2\log(B)}{2\log(B)-1} e\eta d(B-1)\log(B) \right] \right\rceil$	$\exp(2/e)$
Matérn: $\gamma_t \leq \nu t^{\epsilon}$	$\lceil (\nu(B-1))^{1/(1-\epsilon)} \rceil$	e
RBF: $\gamma_t \leq \eta(\log(t+1))^{d+1}$	$\left\lceil \max \left[(\log(B))^{d+1}, \left(\frac{e}{d+1}\frac{\log\eta+(d+2)\log(B)}{2\log(B)-1}\right)^{d+1} \eta(B-1)(\log(B))^{d+1} \right] \right\rceil$	$\exp((\frac{2d+2}{e})^{d+1})$

Table 1: Initialization set sizes for Theorem 5.

the particular values of C' used in Table 1 are not the only ones possible; they are chosen simply because they yield relatively clean algebraic forms for T^{init}. The most important component of this result is the scaling of the regret R_T with T and B. As compared to Theorem 2, which bounds R_T via the product $\exp(2C)T\alpha_T\gamma_T$, where C is a function of B, Theorem 5 replaces the root of this product with a sum of two terms, one in each of B and T; the term $C'R_T^{\text{seq}}$ in Inequality (16) is the cost paid for running the algorithm post-initialization (dependent on T, but not B), whereas the second term is the cost of performing the initialization (dependent on B, but not T). Notice that whenever $B = O(\text{polylog}(T))$, $T^{\text{init}} = O(\text{polylog}(T))$, and further, note $R_T^{\text{seq}} = \Omega(\sqrt{T})$. Thus, as long as the batch size does not grow too quickly, the term $O(T^{\text{init}})$ is dominated by $C'R_T^{\text{seq}}$ and the regret bounds of GP-BUCB are only a constant factor, *independent of* B, worse than those of GP-UCB.

In practice, D^{init} should not be constructed by running uncertainty sampling for T^{init} rounds, but rather by running until $\gamma_{B-1}^{\text{init}} \leq C$ for the pre-specified C; one online check can be constructed using Lemma 4. This procedure cannot take more than T^{init} rounds for the kernels discussed and may take considerably fewer. Further, this procedure is applicable to any kernel with sublinear γ_T, generalizing this initialization technique to kernels other than those we have examined.

5. Adaptive Parallelism: GP-AUCB

While the analysis of the GP-BUCB algorithm in Sections 4.4 and 4.5 used feedback mappings $\text{fb}[t]$ specified by the problem instance, it may be useful to let the algorithm control when to request feedback, and to allow this feedback period to vary in some range not easily described by any constant B. For example, allowing the algorithm to control parallelism is desirable in situations where the cost of executing the algorithm's requested actions depends on both the number of batches and the number of individual actions or experiments in those batches. Consider a chemical experiment, in which the cost may depend on the time to complete the batch of reactions and the cost of the reagents needed for each individual experiment. In such a case, confronting an initial state of relative ignorance about the reward function, it may be desirable to avoid using a wasteful level of parallelism. Motivated by this, we develop an alternative to our requirement in GP-BUCB that $t - \text{fb}[t] \leq B$; we will instead specify a $C > 0$ and choose the feedback mapping $\text{fb}[t]$ in concert with the sequence of actions selected by the algorithm such that $I(f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]}) \leq C, \forall \boldsymbol{x} \in D, \forall t \geq 1$. This requirement on $\text{fb}[t]$ in terms of C may appear stringent, but in fact it can be easily

satisfied by on-line, data-driven construction. The GP-AUCB algorithm adaptively controls feedback through precisely such a mechanism.

Section 5.1 introduces GP-AUCB and states a corollary regret bound for this algorithm. A few comments on local versus global stopping criteria for adaptivity of algorithms follow in Section 5.2.

5.1 GP-AUCB Algorithm

The key innovation of the GP-AUCB algorithm is in choosing $\text{fb}[t]$ online, using a limit on the amount of information hallucinated within the batch. Such adaptive batch length control is possible because we can measure online the amount of information hallucinated with respect to f using Equation (3), even in the absence of the observations themselves. This quantity can be used in a stopping condition; when it exceeds a pre-defined constant C, the algorithm terminates the batch and waits for the environment to return observations for the pending actions. The feedback mapping fb is then updated to include these new observations and the selection of a new batch begins. The resulting algorithm, GP-AUCB, is shown in Algorithm 4.

GP-AUCB is also applicable in the delay setting. In Section 3.1, a view of the delay setting was presented in which an algorithm maintains a queue of pending observations, where this queue is of size B and the algorithm submits a query in any round during which the queue is not full. This is natural for GP-BUCB, particularly if the delay on any observation is known to be bounded by B', i.e., $t - \text{fb}[t] \leq B'$; in such a case, choosing $B = B'$ gives an algorithm which submits an action every round. However, if B' is unknown, the queue size B would have to be chosen in some other way, such that potentially $B < B'$. In this case, the algorithm might have B pending observations at the beginning of a round, a full queue, and so decline to submit an action in that round, i.e., balk. Analogously, GP-AUCB in the delay setting implements a queue which is bounded by the conditional mutual information of the corresponding observations and f, given the current posterior. At each round, GP-AUCB checks if this quantity is more than a pre-defined value C, and only submits a query if it is not. Consequently, if $C < \gamma_{B'-1}$, the algorithm may balk on some rounds.

By terminating batches (or balking) such that no action is selected when the conditional information of the pending observations with respect to f is more than C, the GP-AUCB algorithm ensures that

$$I(f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]}) \leq I(f; \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]}) \leq C, \ \forall \boldsymbol{x} \in D, \ \forall t \geq 1,$$

where t indexes all actions selected by the algorithm, the first inequality follows from the monotonicity of conditional mutual information, and the second inequality follows from the stopping condition. This result implies that Inequality (10) is satisfied, a key requirement of Theorem 2. In contrast, GP-BUCB satisfies the requirement that the second inequality hold by selecting a value for C greater than the conditional information which could be gained in *any* batch of a fixed size, as in Inequality (14), potentially resulting a choice of C larger than necessary for a given B. Since GP-AUCB considers the batches which are actually constructed, it can be expected to enable a higher level of parallelism for the same C, or a comparable level of parallelism for a smaller C.

It is also important to contrast the behavior of GP-AUCB with a scheduled, monotonically increasing level of parallelism. Under the stopping condition, the batch length is

Algorithm 4 GP-AUCB

Input: Decision set D, GP prior μ_0, σ_0, kernel $k(\cdot,\cdot)$, information gain threshold C.
Set $\text{fb}[t'] = 0,\ \forall t' \geq 1$, $G = 0$.
for $t = 1, 2, \ldots, T$ **do**
  if $G > C$ **then**
    Obtain $y_{t'} = f(\boldsymbol{x}_{t'}) + \varepsilon_{t'}$ for $t' \in \{\text{fb}[t-1], \ldots, t-1\}$
    Perform Bayesian inference to obtain $\mu_{t-1}(\cdot)$ via Equation (1)
    Set $G = 0$
    Set $\text{fb}[t'] = t-1,\ \forall t' \geq t$
  end if
  Choose $\boldsymbol{x}_t = \text{argmax}_{\boldsymbol{x} \in D}[\mu_{\text{fb}[t]}(\boldsymbol{x}) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x})]$
  Set $G = G + \frac{1}{2}\log\left(1 + \sigma_n^{-2}\sigma_{t-1}^2(\boldsymbol{x}_t)\right)$
  Compute $\sigma_t(\cdot)$ via Equation (2)
end for

chosen in response to the algorithm's need to explore or exploit as dictated by the decision rule, Equation (7). This does tend to cause an increase in parallelism; the batch length may possibly become quite large as the shape of f is better and better understood and the variance of $f(\boldsymbol{x}_t)$ tends to decrease. However, if exploratory actions are chosen, the high information gain of these actions contributes to a relatively early arrival at the information gain threshold C and thus relatively short batch length, even late in the algorithm's run.

Since all actions are selected when $I(f; \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]}) \leq C$ for all $\boldsymbol{x} \in D$, this approach meets the conditions of Theorem 2, yielding the following corollary:

Corollary 6 *Let the GP-AUCB algorithm be employed with a specified constant $\delta \in (0,1)$ and a specified constant $C > 0$, for which the resulting feedback mapping fb : $\mathbb{N} \to \mathbb{N}$ guarantees $I(f; \boldsymbol{y}_{fb[t]+1:t-1} \mid \boldsymbol{y}_{1:fb[t]}) \leq C, \forall t \geq 1$. If the conditions of one case of Theorem 2 are met, and $\beta_t = \exp(2C)\alpha_{fb[t]+1}$ (Case 1 & 3) or $\beta_t = \exp(2C)\alpha_t$ (Case 2), then*

$$\Pr\left\{R_T \leq \sqrt{C_1 T \exp(2C)\alpha_T \gamma_T} + 2,\ \forall T \geq 1\right\} \geq 1 - \delta$$

where $C_1 = 8/\log(1 + \sigma_n^{-2})$.

Importantly, the specification of C directly specifies the regret bound under Corollary 6. Describing a problem in terms of C is thus natural in the case that we wish to parallelize an experimental process and our specification is what factor additional regret is acceptable, as compared to the sequential GP-UCB algorithm. The batch sizes or balking which result can then be regarded as those which follow from this specification.

Despite the advantages of this approach, C is abstract and less natural for an experimentalist to specify than a maximum batch size or delay length. However, some intuition with regard to C may be obtained. First, C can be selected to deliver batches with a specified minimum size B_{min}. To ensure this occurs, C can be set such that $C > \gamma_{(B_{min}-1)}$, i.e., no set of queries of size less than B_{min} could possibly gain enough information to end the batch. A satisfactory C can be found by either obtaining $\gamma_{(B_{min}-1)}$ directly (tractable for small B_{min}) or via a constant factor bound (Krause and Guestrin, 2005) using the amount

of information which could be gained during uncertainty sampling (Algorithm 3). Note that it is also possible to combine the results of Section 4.5 with Corollary 6 to produce a two-stage adaptive algorithm which can deliver high starting parallelism, very high parallelism as the run proceeds, and a low information gain bound C, yielding a favorable asymptotic regret bound. This may be done by initializing thoroughly enough that, for a pre-specified C and B_{min}, $\gamma^{\text{init}}_{B_{min}-1} < C$, such that the stopping condition cannot take effect until the batch size is at least B_{min}, and then running the GP-AUCB algorithm. This procedure ensures that all batches are of size at least B_{min} and no action is selected using more than C hallucinated information. Alternatively, for uninitialized GP-AUCB, note that C could be quite small, e.g., γ_1; a very small choice for C should produce GP-UCB-like, fully-sequential behavior while the algorithm knows very little, but as the algorithm begins repeatedly selecting actions within a small, well-characterized set, it will permit a greater level of parallelism.

In Section 3.1, the pessimistic and optimistic views of parallelism discussed therein could respectively be viewed as emphasizing one or the other of action selection or feedback receipt as the most important clock by which the system's progress could be judged. However, in the simple batch and delay settings, these perspectives were fixed to one another by the constant B, governing the maximum level of parallelism. Allowing adaptive or stochastic delay and balking breaks this fixed linkage and can be thought of as creating a third clock timing the *opportunities* for the algorithm to select a single action. If the delay is fixed in terms of the number of such opportunities between action and observation, rather than the number of actions between these events, this gives a more natural notion of waiting for observations and allows a better comparison of the tradeoffs inherent in such policies. In our experiments, this opportunity-for-action perspective is explicitly used for all adaptive algorithms shown in Figures 3 and 6, which apply the adaptive algorithms to the delay setting. We also take our previous, pessimistic or action-centered perspective in Figure 5 when looking at adaptive batch size selection, allowing examination of how much regret adaptive batch size selection incurs as compared to fully sequential of fixed parallel algorithms.

5.2 Locally Stopped Adaptive Algorithms

Recently, Azimi et al. (2012b) proposed the Hybrid Batch Bayesian Optimization algorithm (HBBO). HBBO implements a check on the faithfulness of a hallucinated posterior, similar to our approach. This check is expressed not in terms of information gain, but rather expected prediction error versus the true posterior if all information had been acquired. Their stopping condition is also only locally checked at the selected $\boldsymbol{x}_t$, rather than all $\boldsymbol{x}$ in D. Azimi et al. (2012b) employ this stopping condition along with a constraint that the size of the batch assembled can never exceed a pre-specified B_{max}. They show that, in practice, much of the time the algorithm is "safe" under the local faithfulness condition and the level of parallelism is actually controlled by B_{max}. In this section, we consider how our results similarly extend to local stopping conditions.

Theorem 2's requirement on the hallucinated conditional information gain is stated in terms of Equation (10), a bound on hallucinated information with respect to $f(\boldsymbol{x})$ for all $\boldsymbol{x} \in D$. Through Equation (9), this bound ensures that the confidence intervals used to select actions are still sufficiently faithful to those based on the true posterior, i.e., that

Algorithm 5 GP-AUCB Local

Input: Decision set D, GP prior μ_0, σ_0, kernel $k(\cdot,\cdot)$, information gain threshold C, maximum batch size B_{max}.
Set $\text{fb}[t'] = 0,\ \forall t' \geq 1$.
for $t = 1, 2, \ldots, T$ **do**
  if $t - \text{fb}[t] > B_{max}$ **or** $\exists\, \boldsymbol{x} \in D : \sigma_{\text{fb}[t]}(\boldsymbol{x})/\sigma_{t-1}(\boldsymbol{x}) > \exp(C)$ **then**
    Obtain $y_{t'} = f(\boldsymbol{x}_{t'}) + \varepsilon_{t'}$ for $t' \in \{\text{fb}[t-1], \ldots, t-1\}$
    Perform Bayesian inference to obtain $\mu_{t-1}(\cdot)$ via Equation (1)
    Set $\text{fb}[t'] = t-1,\ \forall t' \geq t$
  end if
  Choose $\boldsymbol{x}_t = \text{argmax}_{\boldsymbol{x} \in D}[\mu_{\text{fb}[t]}(\boldsymbol{x}) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x})]$
  Compute $\sigma_t(\cdot)$ via Equation (2)
end for

$\sigma_{t-1}(\boldsymbol{x})$ does not become too small with respect to $\sigma_{\text{fb}[t]}(\boldsymbol{x})$. In the previous analysis, we ensured that this bound held for all $\boldsymbol{x} \in D$ by bounding $I(f; \boldsymbol{y}_{\text{fb}[t]+1:t-1} | \boldsymbol{y}_{1:\text{fb}[t]})$, an upper bound on each of the local information gains. However, in order to select actions, $\sigma_{t-1}(\boldsymbol{x})$ is calculated on-line; if D is of finite size, it is thus possible (if expensive) to compute the ratio $\sigma_{\text{fb}[t]}(\boldsymbol{x})/\sigma_{t-1}(\boldsymbol{x})$ for every $\boldsymbol{x}$ in D and every time step. Similar to GP-AUCB, it is possible to create an algorithm which uses Equation (7) to select actions and which terminates batches adaptively whenever there is any $\boldsymbol{x} \in D$ where this ratio is greater than $\exp(C)$ for a specified $C > 0$. Such an algorithm retains the regret bounds of Theorem 2. With the additional constraint that the assembled batch size not exceed a specified B_{max}, we denote this algorithm GP-AUCB Local and present it as Algorithm 5. We also test this algorithm in some of the experiments and figures in Section 7, along with HBBO.

A number of statements may be made regarding GP-AUCB Local. First, in the case of a flat prior, e.g., $f \sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}'))$, Equation (7) reduces to $x_t = \text{argmax}_{\boldsymbol{x} \in D}\, \sigma_{t-1}(\boldsymbol{x})$ until feedback is obtained at the end of the first batch, i.e., uncertainty sampling (Algorithm 3). GP-AUCB Local's first batch may thus contain a very large number of actions, broadly initializing the decision set. Such a procedure resembles the typical initialization of bandit algorithms and may be attractive in some settings, particularly those in which parallelism is essentially unlimited and the central concern is the number of batches. Second, in practice, nearly all of batches of GP-AUCB Local are stopped via the maximum batch size constraint because the largest local information gain may be small, even for a large batch. This means that this algorithm is effectively implementing GP-BUCB in the simple parallel case, where $B = B_{max}$, albeit with a tighter regret bound, since the specified C only needs to exceed the *local* information gain, rather than the maximum global information gain.

6. Lazy Variance Calculations

In this section, we introduce the notion of lazy variance calculations, which may be used to greatly accelerate the computation of many UCB-based algorithms, including GP-UCB, GP-BUCB, and GP-AUCB, without any loss of performance.

While the probabilistic inference carried out by GP-UCB, GP-BUCB, and GP-AUCB may be implemented in closed form, without the need for expensive approximate inference, the computational cost of the algorithms may still be high, particularly as the number of observations increases. In applications where a finite decision set is considered at every time t, the major computational bottleneck is calculating the posterior mean $\mu_{\mathrm{fb}[t]}(\boldsymbol{x})$ and variance $\sigma^2_{t-1}(\boldsymbol{x})$ for the candidate actions, as required to calculate the decision rule and choose an action $\boldsymbol{x}_t$. The mean is updated only whenever feedback is obtained, and—upon computation of the Cholesky factorization of $\mathbf{K}(X_{\mathrm{fb}[t]}, X_{\mathrm{fb}[t]}) + \sigma_n^2 I$—the calculation of the posterior mean $\mu_{\mathrm{fb}[t]}(\boldsymbol{x})$ takes $O(t)$ additions and multiplications. On the other hand, $\sigma^2_{t-1}(\boldsymbol{x})$ must be recomputed for every $\boldsymbol{x} \in D$ after every round, and requires solving backsubstitution, which requires $O(t^2)$ computations. For large decision sets D, the variance computation thus dominates the computational cost of GP-BUCB.

Fortunately, for any fixed decision $\boldsymbol{x}$, $\sigma^2_t(\boldsymbol{x})$ is non-increasing in t. This fact can be exploited to dramatically improve the running time of GP-BUCB. The key idea is that instead of recomputing $\sigma_{t-1}(\boldsymbol{x})$ for all candidate actions $\boldsymbol{x}$ in every round t, we can maintain an upper bound $\widehat{\sigma}_{t-1}(\boldsymbol{x})$, initialized to $\widehat{\sigma}_0(\boldsymbol{x}) = \infty$. In every round, we lazily apply the GP-BUCB rule with this upper bound to identify

$$\boldsymbol{x}_t = \operatorname*{argmax}_{\boldsymbol{x} \in D} \left[\mu_{\mathrm{fb}[t]}(\boldsymbol{x}) + \beta_t^{1/2} \widehat{\sigma}_{t-1}(\boldsymbol{x}) \right]. \tag{17}$$

We then recompute $\widehat{\sigma}_{t-1}(\boldsymbol{x}_t) \leftarrow \sigma_{t-1}(\boldsymbol{x}_t)$. If $\boldsymbol{x}_t$ still lies in the argmax of Equation (17), we have identified the next action to take, and set $\widehat{\sigma}_t(\boldsymbol{x}) = \widehat{\sigma}_{t-1}(\boldsymbol{x})$ for all $\boldsymbol{x} \in D$. Minoux (1978) proposed a similar technique, concerning calculating the greedy action for submodular maximization, which the above technique generalizes to the bandit setting. A similar idea was also employed by Krause et al. (2008) in the Gaussian process setting for experimental design. The lazy variance calculation method leads to dramatically improved empirical computational speed, discussed in Section 7.4. Note also that the quantities needed for a rank-1 update of the Cholesky decomposition of the observation covariance $\mathbf{K}(X_t, X_t) + \sigma_n^2 I$ are obtained at no additional cost; in order to select $\boldsymbol{x}_t$, we calculate the posterior standard deviation $\sigma_{t-1}(\boldsymbol{x}_t)$, which requires precisely these values.

Locally stopped algorithms (Section 5.2) may have stopping conditions which require $\sigma_{t-1}(\boldsymbol{x})$ for every $\boldsymbol{x} \in D$, which would seem to indicate that the lazy approach is not applicable. However, they may also benefit from lazy variance calculations. Since the global conditional information gain bounds the local information gain for all $\boldsymbol{x} \in D$, as in Inequality (11), we obtain the implication

$$I(f; \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \leq C \implies \nexists\, \boldsymbol{x} \in D : I(f(\boldsymbol{x}); \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) > C$$

that is, that until the stopping condition for GP-AUCB is met, the stopping condition for GP-AUCB Local is also not met, and thus no local calculations need be made. In implementing GP-AUCB Local, we may run what is effectively lazy GP-AUCB until the global stopping condition is met, at which time we transition to GP-AUCB Local. For a fixed maximum batch size B_{max}, it is often the case that local variance calculations become only very rarely necessary after the first few batches.

We have so far in this section concentrated on the case where D is of finite size. It is in general challenging to optimize the decision rule (a possibly multimodal function) over D if

D is a continuous set, as in Case 2 of Theorem 2. Many heuristics are reasonable, but any heuristic which re-uses candidate actions from round to round (e.g., one which considers repeating past actions $\boldsymbol{x}_{t'}, \forall t' < t$, or employs an expanding, finite discretization of D) could also be accelerated by this method.

7. Experiments

We compare GP-BUCB with several alternatives: (1) The strictly sequential GP-UCB algorithm ($B = 1$), which *immediately* receives feedback from each action without batching or delay, thus providing the baseline comparison from the pessimistic perspective (see Section 3.1); (2) Two versions of a state-of-the-art algorithm for Batch Bayesian optimization proposed by Azimi et al. (2010), which can use either a UCB or Maximum Expected Improvement (MEI) decision rule, herein SM-UCB and SM-MEI respectively. Note that the algorithm of Azimi et al. (2010) is not applicable to the delay setting and so does not appear in our delay experiments. Similarly, we compare GP-AUCB against two other adaptive algorithms: (1) HBBO, proposed by Azimi et al. (2012b), which checks an expected prediction error stopping condition, makes decisions using either an MEI or a UCB decision rule, and is applicable only to the batch setting; and (2) GP-AUCB Local, a local information gain-checking adaptive algorithm described in Section 5.2. We also present some experimental comparisons across these two sets of algorithms.

In Section 7.1, we describe the computational experiments in more detail. We perform each of these experiments for several data sets. These data sets and the corresponding experimental results are presented in Section 7.2. We highlight the optimistic perspective on parallelism and the tradeoffs inherent in adaptive parallelism in Section 7.3. Finally, we present the results of the computational time comparisons in Section 7.4.

7.1 Experimental Comparisons

We perform a number of different experiments using this set of algorithms: (1) A simple experiment in the batch case, in which the non-adaptive batch length algorithms are compared against one another, using a single batch length of $B = 5$ (Figure 2); (2) A corresponding experiment in the delay case using a delay of $B = 5$ rounds between action and the corresponding observation, comparing GP-UCB, GP-BUCB, GP-AUCB, and GP-AUCB Local against one another, where the two adaptive algorithms may balk (Figure 3); (3) An experiment examining how changes in the batch length over the range $B = 5, 10$, and 20 affect performance of the non-adaptive algorithms (Figure 4), and a similar experiment where the adaptive algorithms may terminate batches freely, with the restriction that batches must contain at least one and at most 5, 10, or 20 actions (Figure 5); (4) A corresponding experiment in the delay setting, examining how fixed delay length values of $5, 10$, and 20 rounds affect algorithm performance, and in which the adaptive algorithms may balk (Figure 6); (5) An experiment which examines how parallelism and different parameterizations of execution cost may be traded off (Figure 7); and (6) an experiment comparing execution time for various algorithms in the batch case, comparing basic and lazy versions (see Section 6) of the algorithms presented (Figure 8). In the interest of space, some plots are reserved to Online Appendix 2. We also present the results of the experiments in tabular form in Online Appendix 3. The algorithms do not receive an initialization set of observations in

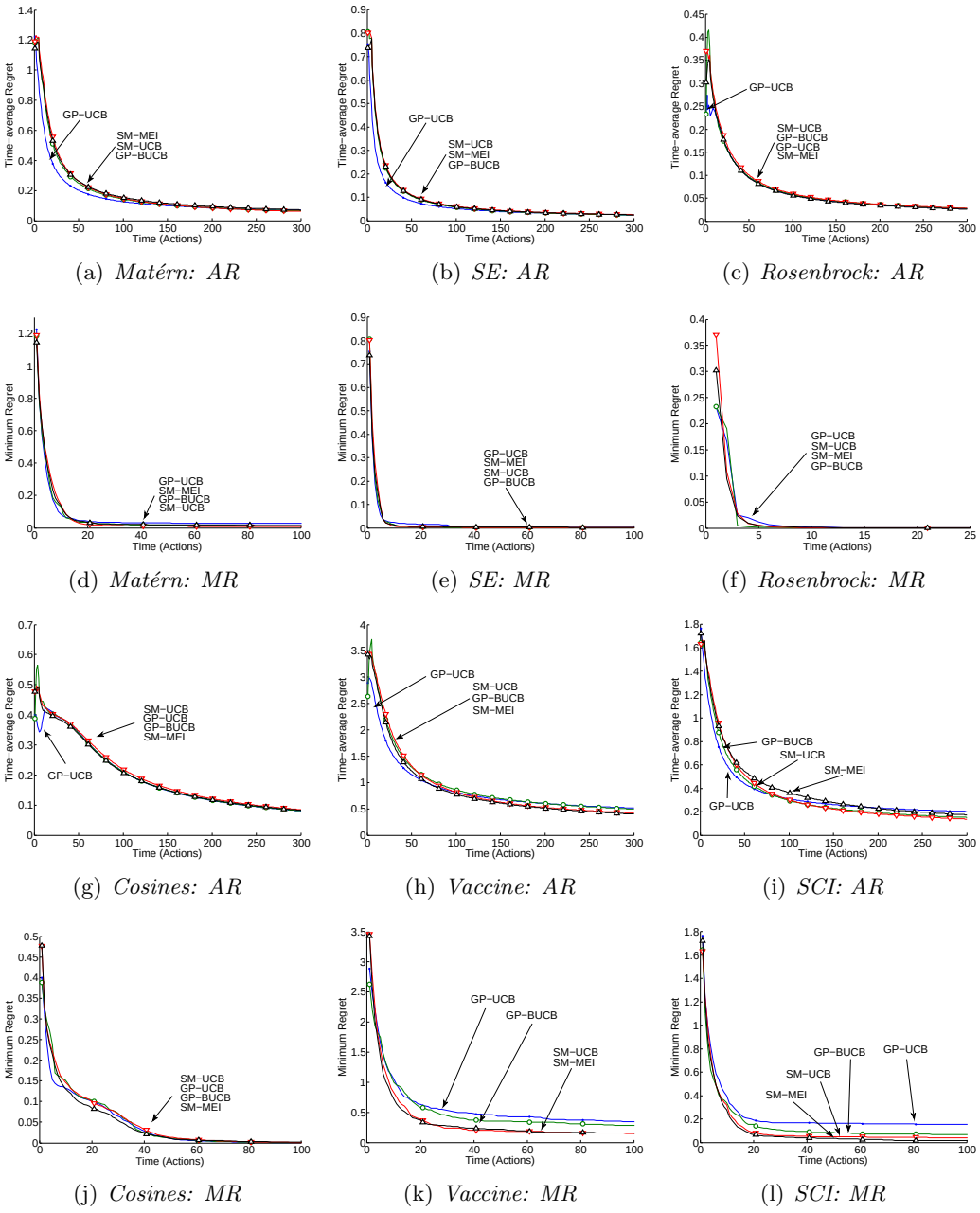
any of the experiments. All experiments were performed in MATLAB using custom code, which we make publicly available.[4]

Comparisons of reward and regret among the algorithms discussed above are presented in terms of their cumulative regret, as well as their simple regret (the function's maximum value minus the best reward obtained). Execution time comparisons are performed using wall-clock time elapsed since the beginning of the experiment, recorded at ends of algorithmic time steps. All experiments were repeated for 200 trials, with pseudo-independent tie-breaking and observation noise for each trial. Additionally, in those experimental cases where the reward function was a draw from a GP (the SE and Matérn problems), each trial used a pseudo-independent draw from the same GP.

In the theoretical analysis in Section 4, the crucial elements in proving the regret bounds of GP-BUCB and GP-AUCB are C, the bound on the information which can be hallucinated within a batch and β_t, the exploration-exploitation tradeoff parameter, which is set with reference to C to ensure confidence interval containment of the reward function. For practical purposes, it is often necessary to define β_t and the corresponding parameter of GP-UCB, α_t, in a fashion which makes the algorithm considerably more aggressive than the regret bound requires. This aggressiveness is particularly important in cases where each observation is very expensive. Setting α_t or β_t in this fashion removes the high-probability guarantees in the regret bound, but often produces excellent empirical performance. On the other hand, leaving the values for α_t and β_t as would be indicated by the theory results in heavily exploratory behavior and very little exploitation. In this paper, in all algorithms which use the UCB or BUCB decision rules, the value of α_t has been set such that it has a small premultiplier (0.05 or 0.1, see Table 2), yielding substantially smaller values for α_t. Further, despite the rigors of analysis explored above in Section 4, we choose to set $\beta_t = \alpha_{\mathrm{fb}[t]+1}$ for the batch and delay algorithms, without reference to the value of C or the batch length B. Taking either of these measures removes the guarantees of correctness as carefully crafted in Section 4. However, as verified by the experiments comparing batch sizes, this is often not a substantial detriment to performance, even for large batch sizes; the batch algorithms generally remain quite competitive with the sequential GP-UCB algorithm. This approach is additionally supported by interactions between local information gain and batch size constraints seen in practice with GP-AUCB Local. One experimental advantage of this approach is that (with some limitations necessitated by the adaptive algorithms) the various algorithms using a UCB decision rule are using the same exploration-exploitation tradeoff parameter at the same iteration, including GP-UCB, GP-BUCB, GP-AUCB, and even SM-UCB and HBBO when using the UCB decision rule. This choice enables us to remove a confounding factor in comparing how well the algorithms overcome the disadvantages inherent in the batch and delay settings.

In the adaptive algorithms (GP-AUCB and GP-AUCB Local), C still establishes the stopping condition, even though it is not used in setting β_t. For GP-AUCB, we specify a minimum batch size or acceptable number of queued observations B_{min} and use uncertainty sampling to calculate a constant-ratio upper bound on $\gamma_{B_{min}}$, as discussed in Section 4.4. Since the ratio $e/(e-1)$ in this bound is > 1, we also use a linear upper bound $\gamma_1 B_{min}$ and set C to the smaller of the two bounds. This choice ensures that the algorithm will always be

4. See `www.its.caltech.edu/~tadesaut/`.



(a) *Matérn: AR* (b) *SE: AR* (c) *Rosenbrock: AR*

(d) *Matérn: MR* (e) *SE: MR* (f) *Rosenbrock: MR*

(g) *Cosines: AR* (h) *Vaccine: AR* (i) *SCI: AR*

(j) *Cosines: MR* (k) *Vaccine: MR* (l) *SCI: MR*

Figure 2: Time-average (AR) and minimum (MR) regret, simple batch setting, batch size of 5. GP-UCB is shown in blue, GP-BUCB in green with circular markers, SM-MEI in black, with triangles, and SM-UCB red, with inverted triangles. When more than one algorithm name is associated with a single arrow, the vertical order of the labels indicates the local vertical order of the regret curves.

Problem Setting	Kernel Function	Hyperparameters	Noise Variance σ_n^2	Premultiplier (on α_t, β_t)
Matérn	covMaterniso	$l = 0.1$, $\sigma^2 = 0.5$	0.0250	0.1
SE	covSEiso	$l = 0.2$, $\sigma^2 = 0.5$	0.0250	0.1
Rosenbrock	RBF	$l^2 = 0.1$, $\sigma^2 = 1$	0.01	0.1
Cosines	RBF	$l^2 = 0.03$, $\sigma^2 = 1$	0.01	0.1
Vaccine	covLINone	$t_2 = 0.8974$	1.1534	0.05
SCI	covSEard	$l = [0.988, 1.5337, 1.0051, 1.5868]$, $\sigma^2 = 1.0384$	0.0463	0.1

Table 2: Experimental kernel functions and parameters.

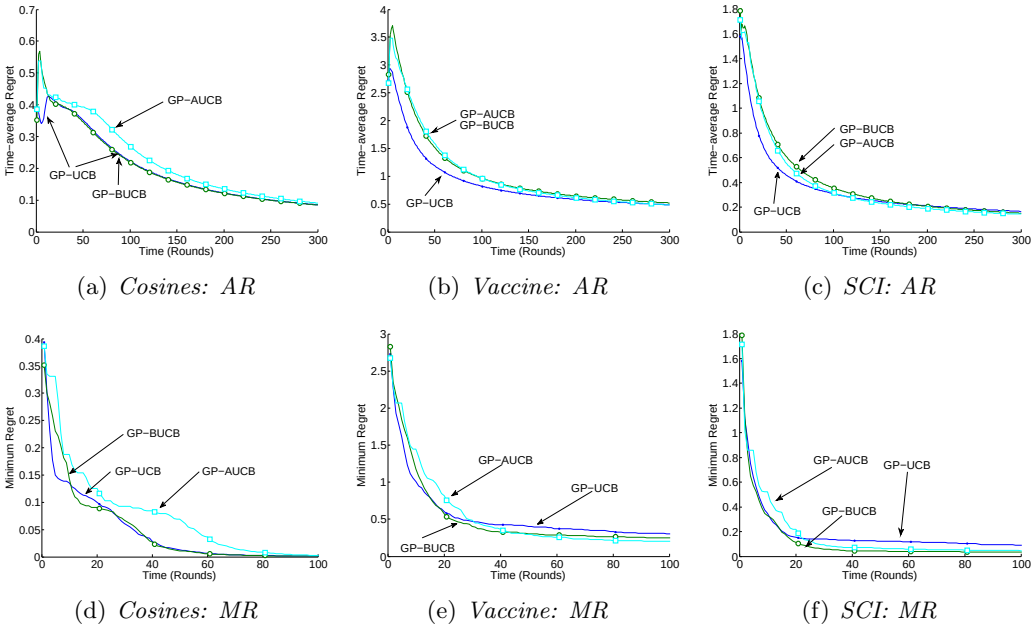
able to select at least B_{min} actions before receiving feedback. In GP-AUCB Local, it is more difficult to choose C appropriately, but we set $C = \max_{\boldsymbol{x} \in D} 1/2 \log(1 + B_{min}\sigma_n^{-2}\sigma_0^2(\boldsymbol{x}))$, where $\sigma_0^2(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x})$ is the prior variance at $\boldsymbol{x}$. This is the maximum information about any $f(\boldsymbol{x})$ which would result from noisily observing $f(\boldsymbol{x})$ B_{min} times. Since for both GP-AUCB and GP-AUCB Local we used B_{min} rather than $B_{min} - 1$ to set C, we implement the stopping condition using a strict inequality for the threshold, requiring that the information gain be $< C$ rather than $\leq C$. In experimental setting (3), we set $B_{min} = 1$, in line with HBBO, and in experimental settings (2), (4), and (5), we use $B_{min} = 2$.

7.2 Data Sets

We empirically evaluate GP-BUCB and GP-AUCB on several synthetic benchmark problems as well as two real applications. For each of the experimental data sets used in this paper, the kernel functions and experimental constants are listed in Table 2. Where applicable, the covariance function from the GPML toolbox (Ver. 3.1, Rasmussen and Nickisch, 2010) used is also listed by name. For all experiments, $\delta = 0.1$ (see Theorem 2) for UCB-based algorithms and tolerance $\epsilon = 0.02$ for HBBO. Each of the experiments discussed above is performed for each of the data sets described below and their results are presented, organized by experimental comparison (e.g., delay, adaptive batch size, etc.), in the accompanying figures.

7.2.1 Synthetic Benchmark Problems

We first test GP-BUCB and GP-AUCB in conditions where the true prior is known. A set of 100 example functions was drawn from a zero-mean GP with Matérn kernel over the interval $[0, 1]$. The kernel, its parameters, and the noise variance are known to each algorithm and D is the discretization of $[0, 1]$ into 1000 evenly spaced points. These experiments are also repeated with a Squared-Exponential kernel. Broadly speaking, these two problems are quite easy; the functions are fairly smooth, and for all algorithms considered, the optimum was found nearly every time, even for long batch sizes or delay lengths. For long batch lengths, substantial regret is incurred during the first batch, since no feedback is available; this is visible in Figures 11(a) and 11(b), in Online Appendix 2. For the batch lengths studied, the first batch of feedback provides a good localization of the optimum because the first few observations are highly informative; for this reason, subsequent values of the minimum regret are typically very small. For the same reason, average regret is largely driven by the length of the first batch. In the delay length experiments, the relative ease of the problems also means that the adaptive algorithms were able to use only relatively few actions and still obtain effective initialization.



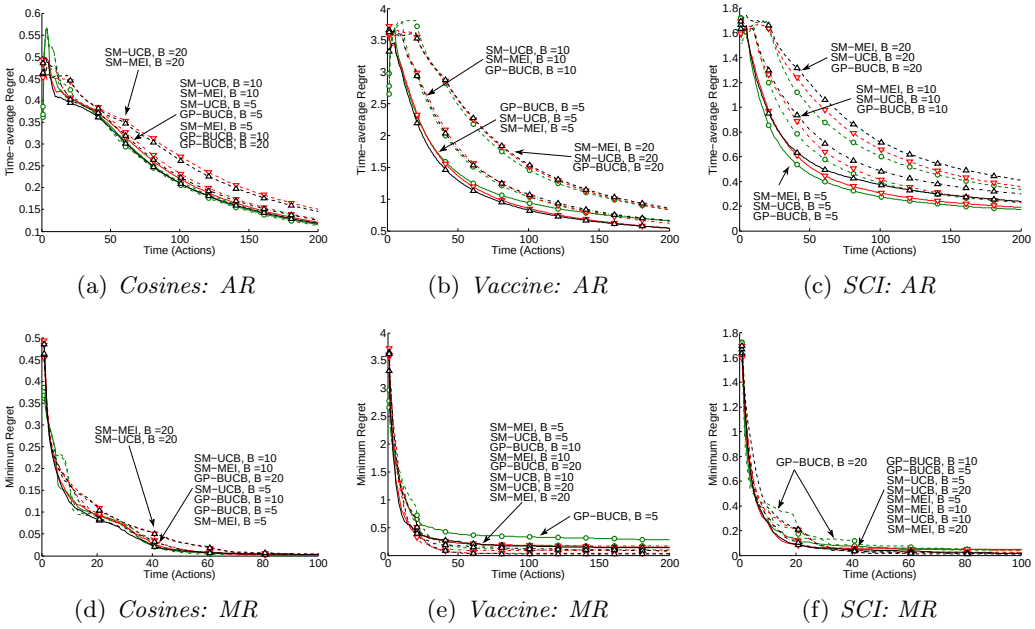
(a) *Cosines: AR* (b) *Vaccine: AR* (c) *SCI: AR*

(d) *Cosines: MR* (e) *Vaccine: MR* (f) *SCI: MR*

Figure 3: Time-average (AR) and minimum (MR) regret plots, delay setting, with a delay length of 5 rounds between action and observation. GP-AUCB is shown in cyan with square markers.

The Rosenbrock and Cosines test functions used by Azimi et al. (2010) are also considered, using the same Squared-Exponential kernel as employed in their experiments, though with somewhat different length scales. For both functions, D is a 31x31 grid of evenly-spaced points on $[0,1]^2$; D is thus similar in size to its counterpart in the Matérn and Squared-Exponential experiments. The values of the Rosenbrock test function at these points are heavily skewed toward the upper end of the reward range, such that the minimum regret is often nearly zero before the first feedback is obtained. In our experiments on the Rosenbrock function, similar performance was obtained across algorithms at each batch size in terms of both average and minimum regret. One result of interest is visible in Figure 6(c), which concerns delay length changes; it is possible to see that GP-AUCB balked too often in this setting, leading to substantial losses in performance relative to GP-AUCB Local and GP-BUCB. The Cosines test function also shows broadly similar results across specific problem instances, with only a small spread in regret among the algorithms tested. Because the Cosines function is multi-modal, the average regret seems to show two-phase convergence behavior, in which individual runs may be approaching local optima and subsequently finding the global optimum. The overly frequent balking by GP-AUCB present in the Rosenbrock test function is also present for longer delays in the Cosines function, as can be seen in 6(g).

In both delay experiments, this behavior may be explained by how the kernel chosen interacts with the stopping condition, which requires that the information gain with respect to the reward function f as a whole be less than a chosen constant C. With a flat prior,



(a) *Cosines: AR* (b) *Vaccine: AR* (c) *SCI: AR*

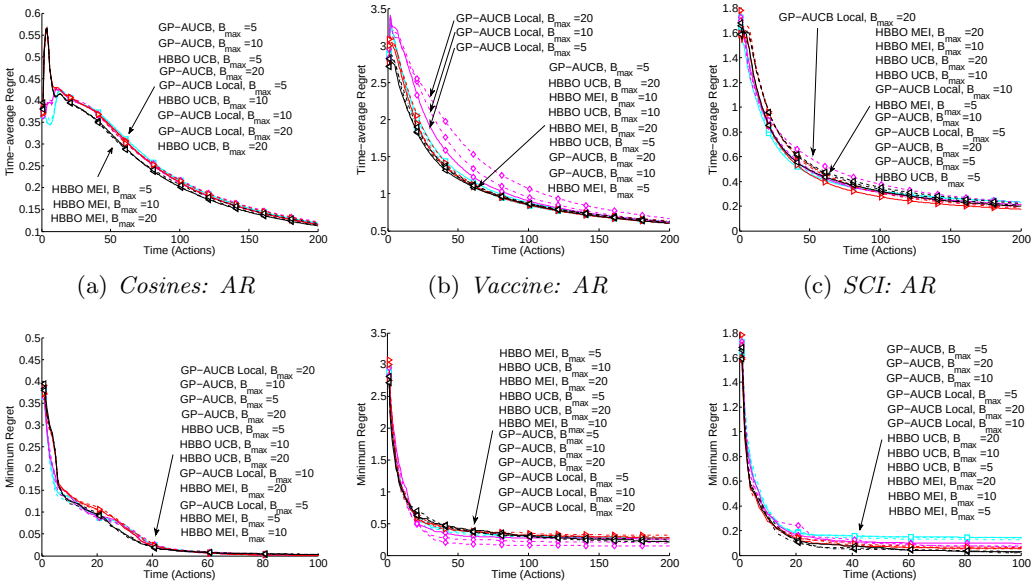
(d) *Cosines: MR* (e) *Vaccine: MR* (f) *SCI: MR*

Figure 4: Time-average (AR) and minimum (MR) regret plots, non-adaptive batch algorithms, batch sizes 5 (solid), 10 (dash-dot), and 20 (dashed).

GP-BUCB, GP-AUCB and GP-AUCB Local all initially behave like uncertainty sampling (see Sections 4.5 and 5.2). Since uncertainty sampling gains a great deal of information globally, GP-AUCB thus tends to balk; on the other hand, since uncertainty sampling scatters queries widely, the information gained with respect to any individual reward $f(\boldsymbol{x})$ may be comparatively small, and so GP-AUCB Local balks less or not at all. If the informativeness of the observations selected is overestimated, perhaps by poor specification of the long-range covariance properties of the assumed kernel function, this greater degree of balking by GP-AUCB may result in overall losses in performance.

7.2.2 Automated Vaccine Design

We also test GP-BUCB and GP-AUCB on a database of Widmer et al. (2010), as considered for experimental design by Krause and Ong (2011). This database describes the binding affinity of various peptides with a Major Histocompatibility Complex (MHC) Class I molecule, of importance when designing vaccines to exploit peptide binding properties. Algorithmic parallelization in such broad chemical screens is particularly attractive because automated, parallel equipment for carrying out these experiments is available. Each of the peptides which bound with the MHC molecule is described by a set of chemical features in $\mathbb{R}^{45}$, where each dimension corresponds to a chemical feature of the peptide. The binding affinity of each peptide, which is treated as the reward or payoff, is described as an offset IC_{50} value. The experiments use an isotropic linear kernel fitted on a different MHC molecule from the same data set. Since the data describe a phenomenon which has a mea-



(a) *Cosines: AR* (b) *Vaccine: AR* (c) *SCI: AR*

(d) *Cosines: MR* (e) *Vaccine: MR* (f) *SCI: MR*

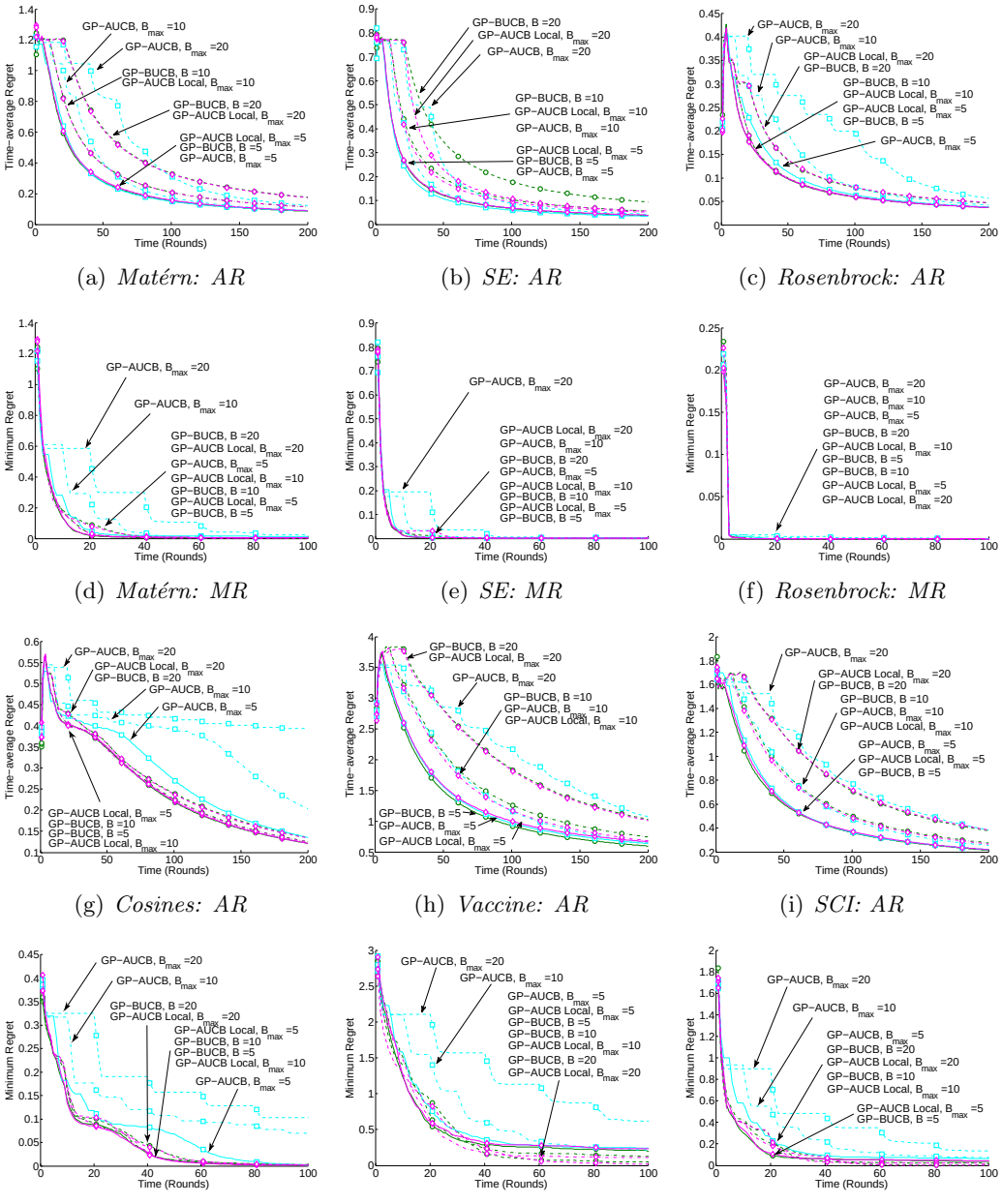
Figure 5: Time-average (AR) and minimum (MR) regret plots, adaptive batch algorithms, maximum batch sizes 5, 10, and 20. HBBO is shown in black with left pointing triangle makers when using an MEI decision rule and in red with right pointing triangle makers when using a UCB decision rule, while GP-AUCB Local is shown in pink with diamond markers. For the adaptive algorithms, minimum batch size B_{min} was set to 1, as in HBBO. The algorithms tended to run fully sequentially at the beginning, but quite rapidly switched to maximal parallelism.

surable limit, many members of the data set are optimal; out of 3089 elements of D, 124, or about 4%, are in the maximizing set. In the simple batch experiments, Figures 2(h) and 2(k), GP-BUCB performs competitively with SM-MEI and SM-UCB, both in terms of average and minimum regret, and converges to the performance of GP-UCB. In the simple delay setting, Figures 3(b) and 3(e), both GP-BUCB and GP-AUCB produce superior minimum regret curves to that of GP-UCB, while performing comparably in terms of long-run average regret; this indicates that the more thorough initialization of GP-AUCB and GP-BUCB versus GP-UCB may enable them to avoid early commitment to the wrong local optimum, thus finding a member of the maximizing set more consistently. This is consistent with the results of the non-adaptive batch size comparison experiment, Figures 4(b) and 4(e), which shows that as the batch size B grows, the algorithm must pay more "up front" due to its more enduring ignorance, but also tends to avoid missing the optimal set entirely. This same sort of tradeoff of average regret against minimum regret is clearly visible for the GP-AUCB Local variants in the experiments sweeping maximal batch size for adaptive algorithms, Figures 5(b) and 5(e).

7.2.3 Spinal Cord Injury (SCI) Therapy

Lastly, we compare the algorithms on a data set of leg muscle activity triggered by therapeutic spinal electrostimulation in spinal cord injured rats. From the 3-by-9 grid of electrodes on the array, a pair of electrodes is chosen to activate, with the first element of the pair used as the cathode and the second used as the anode. Electrode configurations are represented in $\mathbb{R}^4$ by the cathode and anode locations on the array. These active array electrodes create an electric field which may influence both incoming sensory information in dorsal root processes and the function of interneurons within the spinal cord, but the precise mechanism of action is poorly understood. Since the goal of this therapy is to improve the motor control functions of the lower spinal cord, the designated experimental objective is to choose the stimulus electrodes which maximize the resulting activity in lower limb muscles, as measured by electromyography (EMG). Batch or delay algorithms are particularly suited to this experimental setting because the time to process the EMG information needed to assess experimental stimuli may be quite long as compared to the time required to actually test a stimulus, and because idle time during the experimental session should be avoided to the degree possible. We use data with a stimulus amplitude of 5 V and seek to maximize the peak-to-peak amplitude of the recorded EMG waveforms from the right medial gastrocnemius muscle in a time window corresponding to a single interneuronal delay. This objective function attempts to measure the degree to which the selected stimulus activates interneurons controlling reflex activity in the spinal gray matter. This response signal is non-negative and for physical reasons does not generally rise above 3 mV. A Squared-Exponential ARD kernel was fitted using experimental data from 12 days post-injury. Algorithm testing is done using an reward function composed of data from 116 electrode pairs tested on the 14th day post-injury.

Like the Vaccine data set, the SCI data set displays a number of behaviors which indicate that the problem instance is difficult; in particular, the same tendency that algorithms which initialize more thoroughly eventually do better in both minimum and average regret was observed. This tendency is visible in the simple batch setting (Figures 2(i) and 2(l)), where GP-UCB is not clearly superior to either GP-BUCB or GP-AUCB. This is surprising because the pessimistic perspective on parallelism suggests that being required to work in batches, rather than one query at a time, might be expected to give the algorithm less information at any given round, and should thus be a disadvantage. This under-exploration in GP-UCB may be a result of the exploration-exploitation tradeoff parameter α_t being chosen to promote greater aggressiveness across all algorithms. Interestingly, this data set also displays both a small gap between the best and second-best values of the reward function (approximately 0.9% of the range) and a large gap between the best and third-best (approximately 7% of the range). When examining how many out of the individual experimental runs simulated selected $\boldsymbol{x}^* = \text{argmax}_{\boldsymbol{x} \in D} f(\boldsymbol{x})$ on the 200th query in the simple batch case, only 20% of GP-UCB runs choose $\boldsymbol{x}^*$; the numbers are considerably better for GP-BUCB, SM-UCB, and SM-MEI, at 35%, 30.5%, and 36%, but are still not particularly good. If the first suboptimal action is also included, these numbers improve substantially, to 63.5% for GP-UCB and 84%, 91%, and 96.5% for GP-BUCB, SM-UCB, and SM-MEI. These results indicate that the second-most optimal $\boldsymbol{x}$ is actually easier to find than the most optimal, to a fairly substantial degree. It is also important to place these results in the context of the



(a) *Matérn: AR* (b) *SE: AR* (c) *Rosenbrock: AR*

(d) *Matérn: MR* (e) *SE: MR* (f) *Rosenbrock: MR*

(g) *Cosines: AR* (h) *Vaccine: AR* (i) *SCI: AR*

(j) *Cosines: MR* (k) *Vaccine: MR* (l) *SCI: MR*

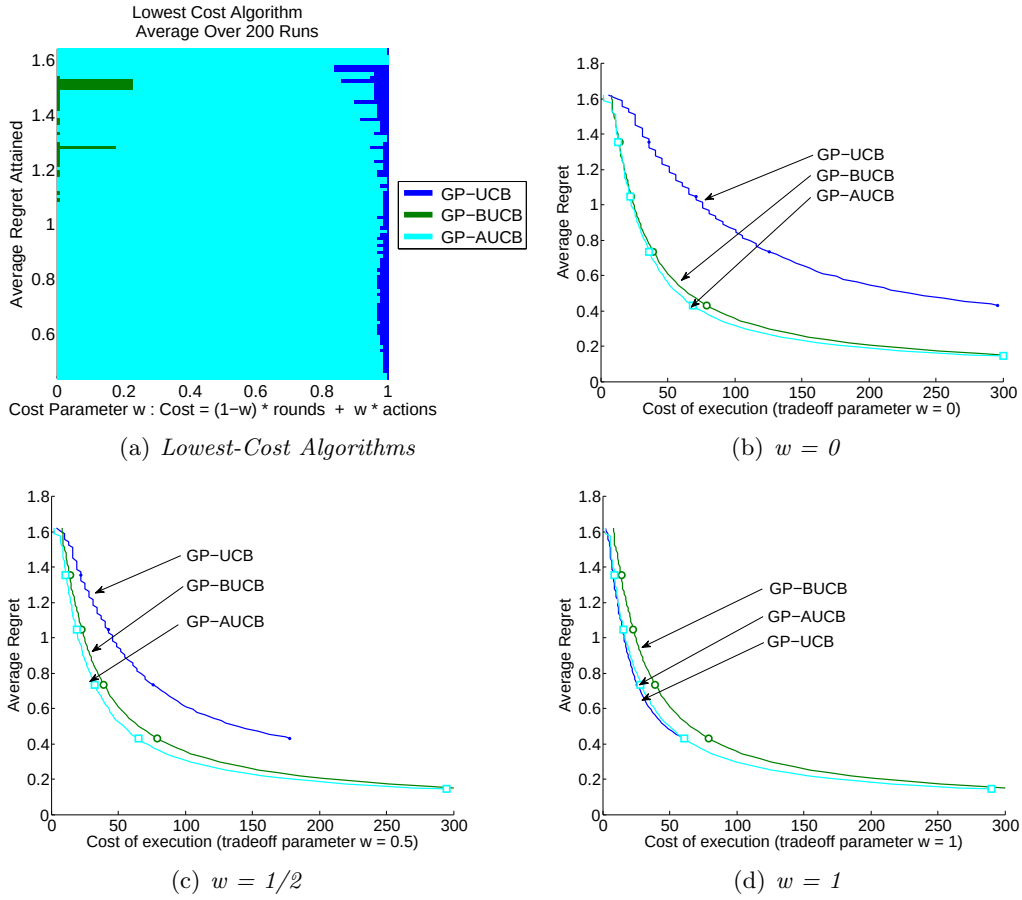
Figure 6: Time-average (AR) and minimum (MR) regret plots, delay setting, with delay lengths of 5, 10, and 20 rounds between action and observation. This experiment examines the degree to which these algorithms are able to cope with long delays between action and observation. Note that the adaptive algorithms, GP-AUCB and GP-AUCB Local, may balk at some rounds. The time-average regret is calculated with respect to the number of actions actually executed as of that round; this means that the number of queries submitted as of any particular round is hidden with respect to the plots shown, and may vary across runs of the same algorithm.

experimental setting; even assuming that the measured response values are reflective of a difference in spinal excitability between these two highest-performing stimuli, it may be that this very small difference in excitability would not yield any difference in therapeutic outcome. Since all of GP-BUCB, SM-UCB, and SM-MEI more consistently found one of the two best actions in the decision set than GP-UCB, all of them show strong performance in comparison to GP-UCB.

7.3 Parallelism: Costs and Tradeoffs

We have presented several algorithms, but an important question is which should be chosen to control any particular experimental process. Our motivation in pursuing parallel algorithms is the setting in which there is a cost—not accumulated in the regret—associated with the experimental process, such that each round or opportunity to submit a query is expensive, but the additional marginal cost of taking an action at that round is not very large. It is interesting to consider more precisely what we mean by "expensive" or "not very large," and also what effect varying these costs with respect to one another might have on which algorithm or level of parallelism is appropriate. In particular, one would expect a low level of parallelism to be beneficial if per-action costs are much higher than per-opportunity (i.e., when speed is less important than economy), while a high level of parallelism would be beneficial if the opposite is true, with intermediate levels of parallelism being superior in the middle. This intuition can be tested by measuring the costs and regret incurred by several algorithms solving the same problem. It is necessary to have a measure by which the performance of different algorithms can be compared, given a particular parameterization of costs. Here, we use an experiment in the delay setting, where the algorithm chooses to either take an action or balk at each round, and employ the average total cost up to the round in which a given average regret is first obtained.

Given N sample runs, a successful algorithm should have a (nearly) monotonically decreasing average regret curve, defined as $\bar{r}(T) = 1/N \sum_{n=1}^{N} R_{T,n}/T$, where $R_{T,n}$ is the cumulative regret of run n after T rounds; these regret curves are the same ones presented in previous experiments. After averaging over many runs, this curve can be inverted to find the first round $\tau(\bar{r})$ in which the sample average regret is at or below a particular $\bar{r}$. The average cost of running the algorithm until round $\tau(\bar{r})$ can then be computed. The cost of run n is the sum of two contributions, the first for running $\tau(\bar{r})$ rounds of the algorithm and the second for the actual execution of $a_n(\tau(\bar{r}))$ actions, where the number of actions executed varies depending on the data acquired. Parameterizing the relative costs of each round and each action using w, the average cost $C(\bar{r}, w) = (1 - w)\tau(\bar{r}) + w \cdot \bar{a}(\tau(\bar{r}))$ corresponding to a particular average regret value $\bar{r}$ can be obtained, where $\bar{a}(\tau(\bar{r})) = 1/N \sum_{n=1}^{N} a_n(\tau(\bar{r}))$. Note that $w \in [0, 1]$ translates to any constant, non-negative ratio of the cost of a single action to that of a single round. This procedure is not equivalent to fixing a value of $\bar{r}$, running each sample run of the algorithm until $R_{t,n}/t \leq \bar{r}$ and averaging over the costs incurred in so doing; in particular, if an algorithm has a non-zero probability of failing to ever obtain $\bar{r}$, individual sample runs may not terminate, making sensible comparison impossible. The calculation of $C(\bar{r}, w)$ as proposed here is robust to this case, giving an estimate of the expected cost to run the algorithm until a round in which the expected cumulative average regret is $\leq \bar{r}$.



(a) *Lowest-Cost Algorithms* (b) $w = 0$

(c) $w = 1/2$ (d) $w = 1$

Figure 7: Parameterized cost comparison on the SCI data set, simple delay case, $B = 5$. The same experiment is also presented in Figure 3(c), but in that figure, we take the pessimistic perspective and compare GP-BUCB and GP-AUCB with GP-UCB, where GP-UCB receives feedback every round. Here, we take the optimistic perspective, which treats parallelism as a potential advantage, and impose the same delay on all algorithms. Figure 7(a): the space of cost-tradeoff parameter w and attained average regrets $\bar{r}$ is colored according to which algorithm has the lowest mean cost at the round in which the mean, time-average regret is first $\leq \bar{r}$. Figures 7(b), (c), and (d) show $\bar{r}$ as a function of C and correspond to vertical slices through Figure 7(a) at the left, center, and right. Since GP-AUCB and GP-UCB pass on some rounds, the terminal cost of GP-AUCB and GP-UCB is possibly < 300.

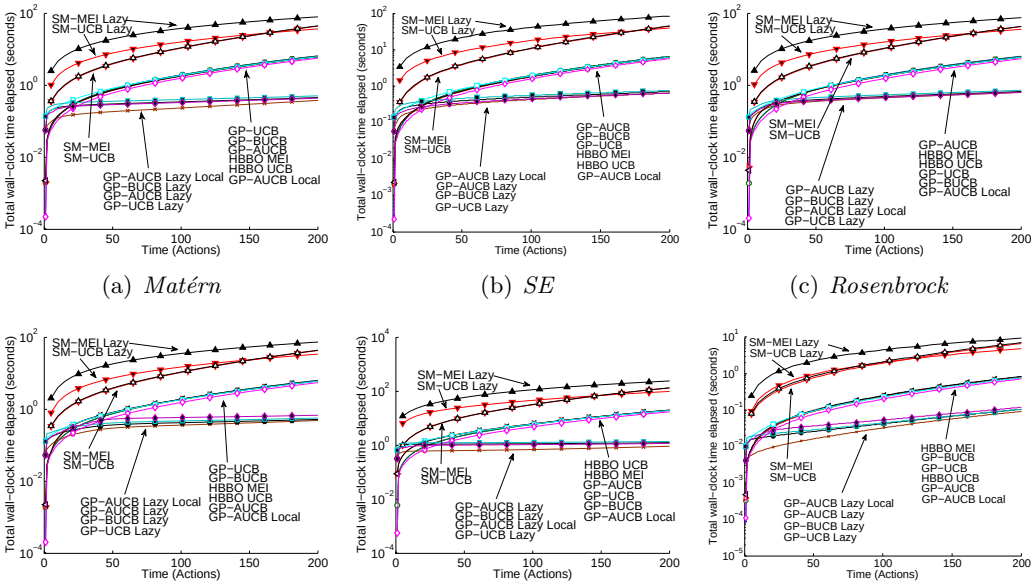
Among a set of algorithms, and given a test problem, one can find which among them has the lowest value of $C(\bar{r}, w)$ at any particular point in the $\bar{r}, w$ space. Similarly, for any fixed value of w, it is possible to once more invert the function and plot $\bar{r}_w(C)$; this plot resembles conventional average regret plots, and corresponds to intersections of each algorithm's $C(\bar{r}, w)$ surface with the plane at a fixed w.

We compare GP-BUCB, GP-AUCB, and GP-UCB in the SCI therapy setting, with a simple delay ($B = 5$). In this setting, GP-BUCB selects an action every round (filling its queue of pending experiments to 5, and then keeping it full) and GP-AUCB may balk, but will tend to fill its queue fully by the end of the experiment. Note that here, we employ GP-UCB under the same feedback mapping as the other algorithms, rather than its use as a benchmark in all of our previous experiments; it thus only submits an action when its queue of pending observations is empty, i.e., every fifth round. The results of this experiment are shown in Figure 7. In this scenario, GP-AUCB costs the least through most of the parameter space, due to its tendency to pass in early rounds, when the potential for exploitation is lowest. In line with the intuition described at the beginning of this section, the advantage changes to the fully sequential algorithm when w is large (i.e., parallelism is expensive), and to GP-BUCB when w is small. Many real-world situations lie somewhere between these extremes, suggesting that GP-AUCB may be useful in a variety of scenarios.

7.4 Computational Performance

We also examined the degree to which lazy variance calculations, as described in Section 6, reduce the computational overhead of each of the algorithms discussed. These results are presented in Figure 8. Note that for algorithms which appear as both lazy and non-lazy versions, the only functional difference between the two is the procedure by which the action is selected, not the action selection itself; all computational gains are without sacrificing accuracy and without requiring any algorithmic approximations. All computational time experiments were performed on a desktop computer (quad-core Intel i7, 2.8 GHz, 8 GB RAM, Ubuntu 10.04) running a single MATLAB R2012a process.

For all data sets, the algorithms lie in three broad classes: Class 1, comprised of the lazy GP-UCB family of algorithms; Class 2, the non-lazy versions of the GP-UCB family of algorithms, as well as the HBBO UCB and MEI variants; and Class 3, consisting of the SM-MEI and SM-UCB algorithms, in both lazy and non-lazy versions. Class 1 algorithms run to completion about one order of magnitude faster than those in Class 2, which in turn are about one order of magnitude faster than those in Class 3. The various versions of the simulation matching algorithm of Azimi et al. (2010) require multiple samples from the posterior over f to aggregate together into a batch, the composition of which is intended to match or cover the performance of the corresponding sequential UCB or MEI algorithm. The time difference between Class 2 and Class 3, approximately one order of magnitude, reflects the choice to run 10 such samples. Within Class 3, our implementation of the lazy version of SM-MEI is slower than the non-lazy version, largely due to the increased overhead of sorting the decision rule and computing single values of the variance; a more efficient implementation of either or both of these elements could perhaps improve on this tradeoff. The lazy algorithms also tend to expend a large amount of computational time early, computing upper bounds on later uncertainties, but tend to make up for this early investment later; this is even visible with regard to the lazy version of SM-UCB, which is initially slower than the non-lazy version, but scales better and, in all six data sets examined, ends up costing substantially less computational time by the 200th query.



(a) *Matérn* (b) *SE* (c) *Rosenbrock*

(d) *Cosines* (e) *Vaccine* (f) *SCI*

Figure 8: Elapsed computational time in batch experiments, B = 5. Lazy versions of algorithms (except GP-UCB) are shown will filled markers. Note the logarithmic vertical scaling in all plots. Note also the substantial separation between the three groups of algorithms, discussed in Section 7.4.

8. Conclusions

We develop the GP-BUCB and GP-AUCB algorithms for parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. We present a unified theoretical analysis, hinging on a natural notion of conditional mutual information accumulated while making selections without observing feedback. Our analysis allows us to bound the regret of GP-BUCB and GP-AUCB, as well as similar GP-UCB-type algorithms. In particular, Theorem 2 provides high-probability bounds on the cumulative regret of algorithms in this class, applicable to both the batch and delay setting. These bounds also imply bounds on the convergence rate of such algorithms. Further, we prove Theorem 5, which establishes a regret bound for a variant GP-BUCB using uncertainty sampling as initialization. Crucially, this bound scales *independently* of the batch size or delay length B, if B is constant or polylogarithmic in T. Finally, we introduce lazy variance calculations, which dramatically accelerate computational performance of GP-based active learning decision rules.

Across the experimental settings examined, GP-BUCB and GP-AUCB performed comparably with state-of-the-art parallel and adaptive parallel Bayesian optimization algorithms, which are not equipped with theoretical bounds on regret. GP-BUCB and GP-AUCB also perform comparably to the sequential GP-UCB algorithm, indicating that GP-BUCB and GP-AUCB successfully overcome the disadvantages of only receiving delayed or batched feedback. As the family of algorithms we describe offers a spectrum of parallelism, we also

examine a parameterization of cost to achieve a given level of regret. In this comparison, GP-AUCB appears to offer substantial advantages over the fully parallel or fully sequential approaches. We believe that our results provide an important step towards solving complex, large-scale exploration-exploitation tradeoffs.

Acknowledgments

The authors thank Daniel Golovin for helpful discussions, the Edgerton laboratory at UCLA for the SCI data set, and the anonymous reviewers for their careful reading and many helpful suggestions. This work was partially supported by NIH project R01 NS062009, SNSF grant 200021_137971, NSF IIS-0953413, DARPA MSEE FA8650-11-1-7156, ERC StG 307036, a Microsoft Research Faculty Fellowship (AK) and a ThinkSwiss Research Scholarship (TD).

Appendix A. Proof of Theorem 2

In order to prove Theorem 2, this appendix first establishes a series of supporting lemmas. For clarity of development, we present the proof of the first case in detail, followed by the required lemmas and modifications required to prove the second and third cases. Since our three cases are those treated by Srinivas et al. (2012) for the GP-UCB algorithm, our proofs use Proposition 1 to generalize their theoretical analysis to the batch and delay cases. In the following, $\mu_{t-1}(\boldsymbol{x})$ and $\sigma_{t-1}(\boldsymbol{x})$ are found via Equations (1) and (2), which assume i.i.d. Gaussian noise of variance σ_n^2, even in Case 3, where the actual noise is non-Gaussian.

A.1 Case 1: Finite D

In all three cases, the first component of the proof is the establishment of confidence intervals which contain the payoff function f with high-probability. In Case 1, this is done by using a result established by Srinivas et al. (2012), presented here as Lemma 7.

Lemma 7 *(Lemma 5.1 of Srinivas et al., 2012) Specify $\delta \in (0,1)$ and set $\alpha_t = 2\log(|D|\pi_t/\delta)$, where $\sum_{t=1}^{\infty} \pi_t^{-1} = 1, \pi_t > 0$. Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \in D$ be an arbitrary sequence of actions. Then,*

$$P(|f(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})| \leq \alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x}),\, \forall \boldsymbol{x} \in D, \forall t \geq 1) \geq 1 - \delta.$$

Proof For $a \sim \mathcal{N}(0,1)$, $P(a > c) \leq 1/2\exp(-c^2/2)$. Conditioned on actions $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}\}$ and corresponding observations $\{y_1, \ldots, y_{t-1}\}$, $f(\boldsymbol{x}) \sim \mathcal{N}(\mu_{t-1}(\boldsymbol{x}), \sigma_{t-1}^2(\boldsymbol{x}))$; for any $\alpha_t > 0$,

$$P\left(\frac{f(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})}{\sigma_{t-1}(\boldsymbol{x})} > \alpha_t^{1/2}\right) = P\left(\frac{f(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})}{\sigma_{t-1}(\boldsymbol{x})} < -\alpha_t^{1/2}\right) \leq \frac{1}{2}\exp(-\alpha_t/2).$$

Note that these two events are the two ways the confidence interval on $f(\boldsymbol{x})$ could fail to hold, i.e., that $f(\boldsymbol{x}) \notin [\mu_{t-1}(\boldsymbol{x}) - \alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x}),\, \mu_{t-1}(\boldsymbol{x}) + \alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x})]$. Union bounding these confidence interval failure probabilities over D, $P(\exists \boldsymbol{x} \in D : f(\boldsymbol{x}) \notin [\mu_{t-1}(\boldsymbol{x}) - \alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x}),\, \mu_{t-1}(\boldsymbol{x}) + \alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x})]) \leq |D|\exp(-\alpha_t/2)$. Let $\delta/\pi_t = |D|\exp(-\alpha_t/2)$, implicitly defining α_t as specified. Union bounding in time and taking the complement yields

$$P(|f(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})| \leq \alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x}), \forall \boldsymbol{x} \in D, \forall t \in \{1, \ldots, T\}) \geq 1 - \delta\sum_{t=1}^{T} \pi_t^{-1}.$$

If $\pi_t > 0$ is chosen such that $\sum_{t=1}^{\infty} \pi_t^{-1} \leq 1$, the result follows. ■

This series convergence condition on π_t corresponds to a requirement that α_t grow sufficiently fast as to make confidence interval failures vanishingly unlikely as $t \to \infty$. Lemma 7 also implies that for $\mathcal{S}$, a subset of the positive integers, $P(|f(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})| \leq \alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x}), \forall \boldsymbol{x} \in D, \forall t \in \mathcal{S}) \geq 1 - \delta$, since $\pi_t > 0 \implies \sum_{t \in \mathcal{S}} \pi_t^{-1} \leq 1$.

Next, we must establish a link between confidence intervals which use a fully updated posterior and for which we have high probability guarantees of correctness (e.g., those in in Lemma 7), and the confidence intervals used in Equation (7), which use a hallucinated posterior. Lemma 8 shows that a bound on the local information hallucinated during the batch implies such a link between batch and sequential confidence intervals.

Lemma 8 *Suppose that at round t, there exists $C > 0$ such that*

$$I(f(\boldsymbol{x}); \boldsymbol{y}_{fb[t]+1:t-1} \mid \boldsymbol{y}_{1:fb[t]}) \leq C, \ \forall \boldsymbol{x} \in D. \tag{18}$$

Choose

$$\beta_t = \exp(2C)\alpha_{fb[t]+1}, \tag{19}$$

where Equation (6) relates sequential confidence intervals $C^{seq}_{fb[t]+1}(\boldsymbol{x})$ with the parameter $\alpha_{fb[t]+1}$ and Equation (8) relates batch confidence intervals $C^{batch}_t(\boldsymbol{x})$ with the parameter β_t. If $f(\boldsymbol{x}) \in C^{seq}_{fb[t]+1}(\boldsymbol{x})$, for all $\boldsymbol{x} \in D$, then $f(\boldsymbol{x}) \in C^{batch}_{t'}(\boldsymbol{x})$ for all $\boldsymbol{x} \in D$ and all t' such that $fb[t] + 1 \leq t' \leq t$.

Proof Noting that the confidence intervals $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x})$ and $C^{\text{batch}}_t(\boldsymbol{x})$ are both centered on $\mu_{\text{fb}[t]}(\boldsymbol{x})$,

$$C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x}) \subseteq C^{\text{batch}}_t(\boldsymbol{x}), \ \forall \boldsymbol{x} \in D \iff \alpha^{1/2}_{\text{fb}[t]+1}\sigma_{\text{fb}[t]}(\boldsymbol{x}) \leq \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}), \ \forall \boldsymbol{x} \in D.$$

By the definition of the conditional mutual information with respect to $f(\boldsymbol{x})$, and by employing Equation (18), Equation (9) follows. Choosing β_t as in Equation (19), it follows that

$$\alpha^{1/2}_{\text{fb}[t]+1}\sigma_{\text{fb}[t]}(\boldsymbol{x}) = \beta_t^{1/2} \exp(-C) \cdot \sigma_{\text{fb}[t]}(\boldsymbol{x}) \leq \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}),$$

where the inequality is based on Equation (9), thus implying $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x}) \subseteq C^{\text{batch}}_t(\boldsymbol{x}) \forall \boldsymbol{x} \in D$. In turn, if $f(\boldsymbol{x}) \in C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x})$, then $f(\boldsymbol{x}) \in C^{\text{batch}}_t(\boldsymbol{x})$. Further, since Equation (19) relates β_t to $\alpha_{\text{fb}[t]+1}$, then $\beta_{t'} = \beta_t$ for all $t' \in \{\text{fb}[t] + 1, \ldots, t\}$. Since $\sigma_{t'}(\boldsymbol{x})$ is non-increasing, $C^{\text{batch}}_{t'}(\boldsymbol{x}) \supseteq C^{\text{batch}}_t(\boldsymbol{x})$ for all such t', completing the proof. ■

With a bound C on the conditional mutual information gain with respect to $f(\boldsymbol{x})$ for any $\boldsymbol{x} \in D$, as in Equation (18), Lemma 8 links the confidence intervals and GP-BUCB decision rule at time t with the GP posterior after observation fb$[t]$. Lemma 9 extends this link to all $t \geq 1$ and all $\boldsymbol{x} \in D$, given a high-probability guarantee of confidence interval correctness at the beginning of all batches.

Lemma 9 *Let there exist a constant $C > 0$, a sequence of actions $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}\}$, and a feedback mapping $fb[t]$ such that for all $\boldsymbol{x} \in D$*

$$I(f(\boldsymbol{x}); \boldsymbol{y}_{fb[t]+1:t-1} \mid \boldsymbol{y}_{1:fb[t]}) \leq C, \ \forall t \geq 1.$$

Let $\beta_t = \exp(2C)\alpha_{fb[t]+1}$, $\forall t \geq 1$; then

$$P(f(\boldsymbol{x}) \in C^{seq}_{fb[t]+1}(\boldsymbol{x}), \forall \boldsymbol{x} \in D, \forall t \geq 1) \geq 1 - \delta$$
$$\implies P(f(\boldsymbol{x}) \in C^{batch}_{t}(\boldsymbol{x}), \forall \boldsymbol{x} \in D, \forall t \geq 1) \geq 1 - \delta.$$

Proof For every $t \geq 1$, there exists a $\tau \geq 0$ such that $\tau = \text{fb}[t]$; let $\mathcal{S} = \{\tau_1, \tau_2, \ldots\}$ be the set of all such images under fb, such that $\text{fb}[t] \in \mathcal{S}$ for all $t \geq 1$. If β_t is chosen as specified, then for any t and $\tau = \text{fb}[t]$, if $f(\boldsymbol{x}) \in C^{\text{seq}}_{\tau+1}(\boldsymbol{x})$, Lemma 8 implies that $f(x) \in C^{\text{batch}}_t(\boldsymbol{x})$. If $f(\boldsymbol{x}) \in C^{\text{seq}}_{\tau+1}(\boldsymbol{x})$ for all $\boldsymbol{x} \in D$ and $\tau \in \mathcal{S}$, then $f(\boldsymbol{x}) \in C^{\text{batch}}_t(\boldsymbol{x})$ for all $\boldsymbol{x} \in D$ and all $t \geq 1$ because every t has an image in $\mathcal{S}$. Thus $f(\boldsymbol{x}) \in C^{\text{seq}}_{\tau+1}(\boldsymbol{x}), \forall \boldsymbol{x} \in D, \forall \tau \in \mathcal{S} \implies f(\boldsymbol{x}) \in C^{\text{batch}}_t(\boldsymbol{x}), \forall \boldsymbol{x} \in D, \forall t \geq 1$. The lemma follows because if the probability of the sufficient condition is at least $1 - \delta$, then the probability of the implied condition must also be at least $1 - \delta$. ■

The high-probability confidence intervals are next related to the instantaneous regret and thence to the cumulative regret. We first state several supporting lemmas.

Lemma 10 *(From Lemma 5.2 of Srinivas et al., 2012) If $f(\boldsymbol{x}) \in C^{batch}_t(\boldsymbol{x})$ for all $\boldsymbol{x} \in D$ and all $t \geq 1$, when actions are selected via Equation (7), $r_t \leq 2\beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t)$, $\forall t \geq 1$.*

Proof By Equation (7), $\boldsymbol{x}_t$ is chosen at each time t such that $\mu_{\text{fb}[t]}(\boldsymbol{x}) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}) \leq \mu_{\text{fb}[t]}(\boldsymbol{x}_t) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t), \forall \boldsymbol{x} \in D$, including for any optimal choice $\boldsymbol{x} = \boldsymbol{x}^*$. Since the instantaneous regret is defined as $r_t = f(\boldsymbol{x}^*) - f(\boldsymbol{x}_t)$ and by assumption both $f(\boldsymbol{x}^*)$ and $f(\boldsymbol{x}_t)$ are contained within their respective confidence intervals,

$$\begin{aligned} r_t &\leq [\mu_{\text{fb}[t]}(\boldsymbol{x}^*) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}^*)] - [\mu_{\text{fb}[t]}(\boldsymbol{x}_t) - \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t)] \\ &\leq [\mu_{\text{fb}[t]}(\boldsymbol{x}_t) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t)] - [\mu_{\text{fb}[t]}(\boldsymbol{x}_t) - \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t)] \\ &\leq 2\beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t). \end{aligned}$$

■

Lemma 11 *(Lemma 5.3 of Srinivas et al., 2012) The mutual information gain with respect to f for the actions selected, $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$, can be expressed in terms of the predictive variances as*

$$I(f; \boldsymbol{y}_{1:T}) = \frac{1}{2}\sum_{t=1}^{T} \log(1 + \sigma_n^{-2}\sigma^2_{t-1}(\boldsymbol{x}_t)).$$

This statement is a result of the additivity of the conditional mutual information gain of observations of a Gaussian.

Lemma 12 *(Extension of Lemma 5.4 of Srinivas et al., 2012) Let $k(\boldsymbol{x}, \boldsymbol{x}) \leq 1, \forall \boldsymbol{x} \in D$. If $f(\boldsymbol{x}) \in C_t^{batch}(\boldsymbol{x})$, $\forall \boldsymbol{x} \in D$, $\forall t \geq 1$, and given that actions $\boldsymbol{x}_t$, $\forall t \in \{1, \ldots, T\}$ are selected using Equation (7), it holds that*

$$R_T \leq \sqrt{TC_1\beta_T\gamma_T},$$

where $C_1 = 8/\log(1 + \sigma_n^{-2})$, γ_T is defined in Equation (4), and β_t is defined in Equation (19).

Proof Given $f(\boldsymbol{x}) \in C_t^{\text{batch}}(\boldsymbol{x})$, $\forall \boldsymbol{x} \in D$, $\forall t \geq 1$, Lemma 10 bounds the instantaneous regret r_t as $r_t \leq 2\beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t)$, $\forall t \geq 1$. The square of the right-hand quantity may be manipulated algebraically to show that $4\beta_t\sigma_{t-1}^2(\boldsymbol{x}_t) \leq C_1\beta_t[1/2\log(1 + \sigma_n^{-2}\sigma_{t-1}^2(\boldsymbol{x}_t))]$. This manipulation exploits the facts that $\sigma_{t-1}^2(\boldsymbol{x}) \leq k(\boldsymbol{x}, \boldsymbol{x}) \leq 1, \forall \boldsymbol{x} \in D$ and that $x/\log(1 + x)$ is non-decreasing for $x \in [0, \infty)$. Summing in time and noting that β_t is non-decreasing, $\sum_{t=1}^T 4\beta_t\sigma_{t-1}^2(\boldsymbol{x}_t) \leq C_1\beta_T\sum_{t=1}^T 1/2\log(1 + \sigma_n^{-2}\sigma_{t-1}^2(\boldsymbol{x}_t)) = C_1\beta_T I(f; \boldsymbol{y}_{1:T})$ by Lemma 11. Thus, by Equation (4), $\sum_{t=1}^T r_t^2 \leq C_1\beta_T\gamma_T$. The claim then follows as a consequence of the Cauchy-Schwarz inequality, since $R_T^2 \leq T\sum_{t=1}^T r_t^2$. ■

Proof [Proof of Theorem 2, Case 1] Taken together, Lemmas 8 through 12, a bound C satisfying Equation (18), and a high-probability guarantee that some set of sequential confidence intervals always contain the values of f allow us to construct a batch algorithm with high-probability regret bounds. Lemma 7 gives us precisely such a guarantee in the case that D is of finite size. Employing Lemma 7 and Lemma 12, and noting that the result holds for all $T \geq 1$, Case 1 of Theorem 2 follows as an immediate corollary. ■

A.2 Case 2: $D \subset \mathbb{R}^d$

Case 2 of Theorem 2 deals with decision sets which are continuous regions of $\mathbb{R}^d$. As a note, we assume that it is possible to select $\boldsymbol{x}_t \in D$ according to Equation (7), i.e., as the maximizer of the decision rule over D. This assumption is non-trivial in practice; this is a non-convex optimization problem in general, though of a function which is perhaps not too ill-behaved, e.g., it is differentiable under the assumptions of Case 2. Nevertheless, we make this assumption and proceed with our analysis.

In the proof of Lemma 7, $P(\exists \boldsymbol{x} \in D : |f(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})| > \beta_t^1/2\sigma_{t-1}(\boldsymbol{x}))$ is bounded via a union bound over D as at most $|D|\exp(-\alpha_t/2)$. Unfortunately, since this bound scales directly with the number of elements in D, this is not useful when D is continuous. We instead use a very similar analysis to establish high-probability confidence intervals on a *subset* D_t of D; using a high-probability bound on the derivatives of the sample paths drawn from the GP, we then proceed to upper-bound $f(\boldsymbol{x})$ for $\boldsymbol{x} \in D \setminus D_t$. Next, we establish a high-probability guarantee for the containment of the reward corresponding to the actions actually taken within their respective confidence bounds at any time, and combine these results to bound the regret r_t suffered in round t. With some slight modifications and careful choices of the scaling of D_t and β_t, the remainder of the analysis from Case 1 can be employed to establish the required bound on R_T for all $T \geq 1$.

Lemma 13 *(From Lemma 5.6 of Srinivas et al., 2012) Specify a discrete set $D_t \subset D$ for every $t \geq 1$, where $D \subseteq [0,l]^d$ and $|D_t|$ is finite. Also specify $\delta \in (0,1)$ and let $\beta_t = 2\exp(2C)\log(|D_t|\pi_t/\delta)$, where $\pi_t > 0, \forall t \geq 1$, $\sum_{t=1}^{\infty} \pi_t^{-1} = 1$, and $I(f(\boldsymbol{x}); \boldsymbol{y}_{fb[t]+1:t-1}|\boldsymbol{y}_{1:t}) \leq C, \forall \boldsymbol{x} \in D, \forall t \geq 1$. Then,*

$$P(|f(\boldsymbol{x}) - \mu_{fb[t]}(\boldsymbol{x})| \leq \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}), \forall \boldsymbol{x} \in D_t, \forall t \geq 1) \geq 1 - \delta.$$

Proof The proof of this lemma is very similar to that of Lemma 7. First, conditioned on actions $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{\text{fb}[t]}\}$ and observations $\{y_1, y_2, \ldots, y_{\text{fb}[t]}\}$, $f(\boldsymbol{x}) \sim \mathcal{N}(\mu_{\text{fb}[t]}(\boldsymbol{x}), \sigma^2_{\text{fb}[t]}(\boldsymbol{x}))$; thus, $(f(\boldsymbol{x}) - \mu_{\text{fb}[t]}(\boldsymbol{x}))/\sigma_{\text{fb}[t]}(\boldsymbol{x}) \sim \mathcal{N}(0,1)$. If $a \sim \mathcal{N}(0,1)$, $c \geq 0$, then $P(a > c) \leq 1/2\exp(-c^2/2)$. Using this inequality and a union bound over all $\boldsymbol{x} \in D_t$, we obtain the following result for general $\beta_t > 0$:

$$\begin{aligned} P\left(\frac{|f(\boldsymbol{x}) - \mu_{\text{fb}[t]}(\boldsymbol{x})|}{\sigma_{\text{fb}[t]}(\boldsymbol{x})} \leq \beta_t^{1/2}\frac{\sigma_{t-1}(\boldsymbol{x})}{\sigma_{\text{fb}[t]}}, \forall \boldsymbol{x} \in D_t\right) &\geq 1 - |D_t|\exp\left(-\frac{\beta_t}{2}\frac{\sigma^2_{t-1}(\boldsymbol{x})}{\sigma^2_{\text{fb}[t]}}\right) \\ &\geq 1 - |D_t|\exp(-\beta_t\exp(-2C)/2). \end{aligned}$$

Analogous to the proof of Lemma 7, let $\delta/\pi_t = |D_t|\exp(-\beta_t\exp(-2C)/2)$, implicitly defining β_t, and union bound in time, yielding the desired result. ■

Note that if at each time $t \geq 1$ we specify a particular $\mathbf{z}_t \in D$ and choose $D_t = \mathbf{z}_t$, Lemma 13 implies that $P(|f(\mathbf{z}_t) - \mu_{\text{fb}[t]}(\mathbf{z}_t)| \leq \tilde{\beta}_t^{1/2}\sigma_{t-1}(\mathbf{z}_t), \forall t \geq 1) \geq 1 - \delta$, where $\tilde{\beta}_t \geq 2\exp(2C)\log(\pi_t/\delta)$. This fact will be employed for $\mathbf{z}_t = \boldsymbol{x}_t$ below.

Next, we upper bound the value of $f(\boldsymbol{x}^*)$, where $\boldsymbol{x}^*$ is possibly within $D \setminus D_t$. Let $[\boldsymbol{x}]_t = \text{argmin}_{\boldsymbol{x}' \in D_t} ||\boldsymbol{x} - \boldsymbol{x}'||_1$, i.e., the closest point in D_t to $\boldsymbol{x}$, in the sense of 1-norm. As a technical point, $[\boldsymbol{x}]_t$ may not be uniquely determined; any 1-norm minimizing element of D_t is sufficient for the following discussion.

Lemma 14 *(From Lemma 5.7 of Srinivas et al., 2012) Specify $\delta \in (0,1)$ and let τ_t be a time-varying parameter. Let $D_t \subset D$ be chosen such that $||\boldsymbol{x} - [\boldsymbol{x}]_t||_1 \leq ld/\tau_t, \forall \boldsymbol{x} \in D, \forall t \geq 1$. Let the statement*

$$P(\sup_{\boldsymbol{x} \in D} |\partial f(\boldsymbol{x})/\partial \boldsymbol{x}_i| < L, \forall i \in \{1, \ldots, d\}) \geq 1 - da\exp(-L^2/b^2)$$

hold for any $L > 0$ for some corresponding $a \geq \delta/(2d)$, $b > 0$, where $\boldsymbol{x}_i$ denotes the ith dimension of $\boldsymbol{x}$. Choose $L = b\sqrt{\log(2da/\delta)}$, $\tau_t = dt^2bl\sqrt{\log(2da/\delta)}$, and $\beta_t \geq 2\exp(2C) \cdot \log(2|D_t|\pi_t/\delta)$, where $\pi_t > 0, \forall t \geq 1$, $\sum_{t=1}^{\infty} \pi_t^{-1} = 1$, and $I(f(\boldsymbol{x}); \boldsymbol{y}_{fb[t]+1:t-1}|\boldsymbol{y}_{1:t}) \leq C, \forall \boldsymbol{x} \in D, \forall t \geq 1$. Then,

$$P\left(|f(\boldsymbol{x}^*) - \mu_{fb[t]}([\boldsymbol{x}^*]_t)| < \beta_t^{1/2}\sigma_{t-1}([\boldsymbol{x}^*]_t) + \frac{1}{t^2}, \forall t \geq 1\right) \geq 1 - \delta$$

Proof For the specified choice of L, we obtain $P(|\partial f(\boldsymbol{x})/\partial \boldsymbol{x}_i| < b\sqrt{\log(2da/\delta)}, \forall \boldsymbol{x} \in D, \forall i \in \{1, \ldots, d\}) \geq 1 - \delta/2$. Thus, with probability $\geq 1 - \delta/2$,

$$\begin{aligned} |f(\boldsymbol{x}^*) - f([\boldsymbol{x}^*]_t)| &\leq b\sqrt{\log(2da/\delta)} \cdot ||\boldsymbol{x}^* - [\boldsymbol{x}^*]_t||_1 \\ &\leq b\sqrt{\log(2da/\delta)} \cdot ld/\tau_t \\ &\leq \frac{1}{t^2}. \end{aligned}$$

By Lemma 13, for β_t as chosen above, $|f([\boldsymbol{x}^*]_t) - \mu_{\text{fb}[t]}([\boldsymbol{x}^*]_t)| \leq \beta_t^{1/2}\sigma_{t-1}([\boldsymbol{x}^*]_t), \forall t \geq 1$ with probability $\geq 1 - \delta/2$. The result follows by a union bound on the possible failures these two events. ■

Note that Lemma 14 states that if we know the size of a suitable discretization D_t of D, we may choose β_t such that we may establish a high probability upper bound on f over all of D. Note also that a larger β_t is acceptable and that D_t itself is not required to prove the result. Our next result proves the existence and size of a sufficient discretization of D; we will then choose β_t according to this provably existent discretization and entirely avoid its explicit construction.

In the following lemma, $\lceil \tau_t \rceil$ denotes the smallest integer which is at least τ_t.

Lemma 15 *There exists a discretization D_t of $D \subseteq [0, l]^d$, D compact and convex, where $|D_t| \leq \lceil \tau_t \rceil^d$ and $||\boldsymbol{x} - [\boldsymbol{x}]_t||_1 \leq \frac{ld}{\tau_t}, \forall \boldsymbol{x} \in D$.*

Proof It is sufficient to construct an example discretization. One way to do so is to first generate an ϵ-cover of $[0, l]^d \supseteq D$, where $\epsilon = ld/2\tau_t$; this can be done by placing a ball centered at each location in $\{\frac{l}{2\lceil \tau_t \rceil}, \frac{3l}{2\lceil \tau_t \rceil}, \ldots, \frac{(2\lceil \tau_t \rceil - 1)l}{2\lceil \tau_t \rceil}\}^d$, such that each point in $[0, l]^d$ (and thus D) is at most $ld/2\lceil \tau_t \rceil \leq ld/2\tau_t$ from the nearest point in this set (i.e., is within at least one of the closed balls) and every ball center lies within $[0, l]^d$. Denote this set of ball centers $\mathcal{A}$ and note that $|\mathcal{A}| = \lceil \tau_t \rceil^d$. For each $\boldsymbol{x} \in \mathcal{A}$, denote the corresponding 1-norm ball of radius $ld/2\tau_t$ as $B^1_{ld/2\tau_t}(\boldsymbol{x})$. We now use $\mathcal{A}$ to construct D_t, such D_t is an ϵ-cover for D, for $\epsilon = ld/\tau_t$. Begin with D_t empty and iterate over $\boldsymbol{x} \in \mathcal{A}$. If $\boldsymbol{x} \in D$, add it to D_t. If $\boldsymbol{x} \notin D$, but $B^1_{ld/2\tau_t}(\boldsymbol{x}) \cap D \neq \emptyset$, add any point in this intersection to D_t. If $\boldsymbol{x} \notin D$ and $B^1_{ld/2\tau_t}(\boldsymbol{x}) \cap D = \emptyset$, do not add $\boldsymbol{x}$ to D_t. By construction, $\boldsymbol{x} \in D_t \implies \boldsymbol{x} \in D$. Since the triangle inequality implies $B^1_{ld/2\tau_t}(\boldsymbol{x}) \subset B^1_{ld/\tau_t}(\boldsymbol{x}'), \forall \boldsymbol{x}' \in B^1_{ld/2\tau_t}(\boldsymbol{x})$, we have the result that $\bigcup_{\boldsymbol{x}' \in D_t} B^1_{ld/\tau_t}(\boldsymbol{x}') \supseteq \bigcup_{\boldsymbol{x}: \boldsymbol{x} \in \mathcal{A}, B^1_{ld/2\tau_t}(\boldsymbol{x}) \cap D \neq \emptyset} B^1_{ld/2\tau_t}(\boldsymbol{x})$, where this second union is by definition a cover for D. D_t is thus an ϵ-cover for D for $\epsilon = ld/\tau_t$ and therefore a satisfactory discretization of size no more than $\lceil \tau_t \rceil^d$. ■

Lemma 14 also rests on a bound on the derivatives of $f(\boldsymbol{x})$ with respect to $\boldsymbol{x}_i, \forall i = 1, \ldots, d$. Such a bound can be created if the kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ defining the distribution over f is sufficiently many times differentiable with respect to $\boldsymbol{x}$ and $\boldsymbol{x}'$.

Lemma 16 *(From Srinivas et al., 2012, Appendix I) If $f \sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}'))$ and derivatives of $k(\boldsymbol{x}, \boldsymbol{x}')$ exist up to fourth order with respect to $\boldsymbol{x}, \boldsymbol{x}' \in D$, then f is almost surely continuously differentiable and there exist positive constants a, b, and L, such that*

$$P(|f(\boldsymbol{x}) - f(\boldsymbol{x}')| \leq L||\boldsymbol{x} - \boldsymbol{x}'||_1, \forall \boldsymbol{x}, \boldsymbol{x}' \in D) \geq 1 - \delta,$$

where $L = b\sqrt{\log(da/\delta)}$.

Proof If all fourth order partial derivatives of $k(\boldsymbol{x}, \boldsymbol{x}')$ exist, the derivatives of f are themselves Gaussian processes with a kernel function corresponding to the twice differentiated k and there exist positive constants $a, \{b_1, \ldots, b_d\}$ such that $P(\sup_{\boldsymbol{x} \in D} |\partial f / \partial \boldsymbol{x}_j| > L_1) \leq$

$a\exp(-b_jL_1^2), \forall j \in \{1,\ldots,d\}$, for any $L_1 > 0$ (Theorem 5 of Ghosal and Roy, 2006). Let $L_1 = \sqrt{\log(da/\delta)}/(\min_j \sqrt{b_j})$. Note that a can be chosen arbitrarily large and $a \geq \delta/d$ (required so the argument of the logarithm is ≥ 1) implies $(da/\delta)^{(-b_j/(\min_j b_j))} \leq \delta/da$; union bounding in j, we thus obtain that $P(\sup_{\boldsymbol{x}\in D} |\partial f/\partial \boldsymbol{x}_j| > L_1, \forall j \in \{1,\ldots,d\}) \leq \delta$. Reparameterizing, choose $b \geq \max_j b_j^{-1/2} > 0$, and define $L = b\sqrt{\log(da/\delta)}$. Using this bound on the supremum of the derivatives of f, and a piecewise construction of the path from $\boldsymbol{x}$ to $\boldsymbol{x}'$ according to the unit vectors, the result follows. ■

In the proof of Lemma 15, we bound the size of the virtual decision set D_t as $\lceil \tau_t \rceil^d$. We can instead use τ_t^d if τ_t is an integer. Luckily, we can always make a and b bigger, e.g., such that $bl\sqrt{\log(da/\delta)}$ is an integer. If this quantity is an integer, so is $\tau_t = dt^2 bl\sqrt{\log(da/\delta)}$.

Next, we bound the regret r_t incurred at round t.

Lemma 17 *(From Lemma 5.8 of Srinivas et al., 2012) Specify $\delta \in (0,1)$. Let the statement*

$$P(\sup_{\boldsymbol{x}\in D} |\partial f(\boldsymbol{x})/\partial \boldsymbol{x}_i| < L, \forall i \in \{1,\ldots,d\}) \geq 1 - da\exp(-L^2/b^2)$$

hold for any $L > 0$ and positive constants a, b. Choose a, b such that $a \geq \delta/(4d)$ and $bl\sqrt{\log(4da/\delta)}$ is an integer. Let actions be selected using Equation (7), where

$$\beta_t = 2\exp(2C)[\log(4\pi_t/\delta) + d\log(dt^2 bl\sqrt{\log(4da/\delta)})],$$

$\pi_t > 0, \forall t \geq 1$, $\sum_{t=1}^{\infty} \pi_t^{-1} = 1$, and $I(f(\boldsymbol{x}); \boldsymbol{y}_{fb[t]+1:t-1} | \boldsymbol{y}_{1:t}) \leq C, \forall \boldsymbol{x} \in D, \forall t \geq 1$. Then,

$$P(r_t \leq \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}) + t^{-2}, \forall t \geq 1) \geq 1 - \delta.$$

Proof By Lemma 15, for any round t, we may construct a discretization D_t of size no more than $\lceil \tau_t \rceil^d$, such that $||\boldsymbol{x} - [\boldsymbol{x}]_t||_1 \leq ld/\tau_t$, where $\tau_t = dt^2 bl\sqrt{\log(4da/\delta)}$ and τ_t is an integer. Choosing β_t as specified and implicitly constructing such a D_t, by Lemma 14, it follows that $P(|f(\boldsymbol{x}^*) - \mu([\boldsymbol{x}^*]_{\text{fb}[t]})| < \beta_t^{1/2}\sigma_{t-1}([\boldsymbol{x}^*]_t) + t^{-2}) \geq 1 - \delta/2$. Applying Lemma 13 to every $\boldsymbol{x}_t$, $t \geq 1$, for any $\tilde{\beta}_t \geq 2\exp(2C)\log(2\pi_t/\delta)$, $P(|f(\boldsymbol{x}_t) - \mu_{\text{fb}[t]}(\boldsymbol{x}_t)| \leq \tilde{\beta}_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t) \forall t \geq 1) \geq 1 - \delta/2$. As specified, $\beta_t \geq 2\exp(2C)\log(2\pi_t/\delta)$, and so $f(\boldsymbol{x}_t)$ is bounded with probability $\geq 1 - \delta/2$. By Equation (7), $\mu_{\text{fb}[t]}(\boldsymbol{x}_t) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t) \geq \mu_{\text{fb}[t]}(\boldsymbol{x}) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}), \forall \boldsymbol{x} \in D, \forall t \geq 1$, and so,

$$\begin{aligned}
r_t &= f(\boldsymbol{x}^*) - f(\boldsymbol{x}_t) \\
&\leq [\mu_{\text{fb}[t]}([\boldsymbol{x}^*]_t) + \beta_t^{1/2}\sigma_{t-1}([\boldsymbol{x}^*]_t) + t^{-2}] - [\mu_{\text{fb}[t]}(\boldsymbol{x}_t) - \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t)] \\
&\leq [\mu_{\text{fb}[t]}(\boldsymbol{x}_t) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t) + t^{-2}] - [\mu_{\text{fb}[t]}(\boldsymbol{x}_t) - \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t)] \\
&= 2\beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t) + t^{-2},
\end{aligned}$$

with probability $\geq 1 - \delta$. ■

Proof [Proof of Theorem 2, Case 2] By Lemma 17, for $\beta_t = 2\exp(2C)[\log(4\pi_t/\delta) + d\log(dt^2 bl\sqrt{\log(4da/\delta)})]$, it follows that $r_t \leq 2\beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t) + t^{-2}, \forall t \geq 1$, with probability

$\geq 1-\delta$. Consequently,

$$R_T = \sum_{t=1}^{T} r_t \leq \sum_{t=1}^{T} 2\beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}_t) + \sum_{t=1}^{T} t^{-2}$$
$$\leq \sqrt{TC_1\beta_T\gamma_T} + \pi^2/6,$$

for all $T \geq 1$, with probability $\geq 1-\delta$, where $C_1 = 8/\log(1+\sigma_n^{-2})$ and the second inequality follows via the argument advanced in the proof of Lemma 12 in Case 1; this argument uses the information gain and Lemma 11 to bound the sum of the squares of the terms, and then the Cauchy-Schwarz inequality to bound the original sum. Theorem 2, Case 2 follows. ■

A natural extension is to the case where the GP prior mean is non-zero, but is known and Lipschitz-continuous. This is straight-forward, since bounds on $\sup_{\boldsymbol{x}\in D} |\partial f(\boldsymbol{x})/\partial \boldsymbol{x}_i|$ or the generalization error in the prior mean may be naturally obtained.

A.3 Case 3: Finite RKHS Norm of f

Case 3 involves a reward function f with a finite RKHS norm with respect to the algorithm's GP prior. Fortunately, Srinivas et al. (2012) again have a result which creates confidence intervals in this situation.

Lemma 18 *(Theorem 6 from Srinivas et al., 2012) Specify $\delta \in (0,1)$. Let $k(\boldsymbol{x},\boldsymbol{x}')$ be an assumed Gaussian process kernel and let $||f||_k$ be the RKHS norm of f with respect to k. Let k be such that $k(\boldsymbol{x},\boldsymbol{x}) \leq 1$ for all $\boldsymbol{x} \in D$. Assume noise variables ϵ_t are from a martingale difference sequence, such that they are uniformly bounded by σ_n. Define*

$$\alpha_t = 2M + 300\gamma_t \log^3(t/\delta),$$

where $M \geq ||f||_k^2$. Then

$$P(|f(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})| \leq \alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x}), \forall \boldsymbol{x} \in D, \forall t \geq 1) \geq 1-\delta.$$

In brief, the reproducing property of kernels implies that the corresponding inner product of $g(\boldsymbol{x})$ with $k(\boldsymbol{x},\boldsymbol{x}')$, denoted $\langle g(\boldsymbol{x}), k(\boldsymbol{x},\boldsymbol{x}')\rangle_k$, is $\langle g(\boldsymbol{x}), k(\boldsymbol{x},\boldsymbol{x}')\rangle_k = g(\boldsymbol{x}')$ for any kernel k, and so, using such an inner product and the Cauchy-Schwarz inequality,

$$|\mu_t(\boldsymbol{x}) - f(\boldsymbol{x})| = \sqrt{\langle \mu_t(\boldsymbol{x}') - f(\boldsymbol{x}'), k_t(\boldsymbol{x},\boldsymbol{x}')\rangle_{k_t}^2}$$
$$\leq \sqrt{\langle k_t(\boldsymbol{x},\boldsymbol{x}'), k_t(\boldsymbol{x},\boldsymbol{x}')\rangle_{k_t}\langle \mu_t(\boldsymbol{x}) - f(\boldsymbol{x}), \mu_t(\boldsymbol{x}) - f(\boldsymbol{x})\rangle_{k_t}}$$
$$= \sqrt{k_t(\boldsymbol{x},\boldsymbol{x})}||\mu_t - f||_{k_t} = \sigma_t(\boldsymbol{x})||\mu_t - f||_{k_t},$$

where k_t is the posterior kernel and μ_t is the posterior mean at time t. This implies that an upper bound on $||\mu_t - f||_{k_t}$ gives an upper bound on the regret which can be incurred at the selection of action $t+1$ (via the argument of Lemma 10), suggesting that our UCB width multiplier $\alpha_t^{1/2}$ should have a form related to the growth of $||\mu_t - f||_{k_t}$. Since $||g||_{k_t}^2 = ||g||_k^2 + \sigma_n^{-2}\sum_{\tau=1}^{t} g(\boldsymbol{x}_\tau)^2$ for a function g, we may substitute $g = \mu_t - f$ and

then factor the squared norm. We thus obtain a term in $||f||_k^2$ and a second term in the observations and noise terms y_t and ϵ_t. We assume we have a bound $M \geq ||f||_k^2$, and so our problem reduces to choosing an α_t sufficiently large to surpass the growth of the second term. Through an inductive argument on the probability of confidence interval failure up to the tth action and using the α_t chosen, Srinivas et al. (2012) show that these confidence intervals hold with probability at least $1 - \delta$. Importantly, this argument does *not* use the decision-making process of the algorithm, instead only relying on the algorithm's internal GP posterior over f and the characteristics of the martingale noise process. We refer the interested reader to Appendix II of Srinivas et al. (2012) for the details.[5]

In terms of proving Case 3 of Theorem 2, Lemma 18 is just the sort of statement we need; it establishes a precisely analogous result to Lemma 7, providing a set of confidence intervals $C_\tau^{\text{seq}}(\boldsymbol{x})$ such that for $\tau = \text{fb}[t] + 1$, we can construct β_t and thus $C_t^{\text{batch}}(\boldsymbol{x})$ such that $C_t^{\text{batch}}(\boldsymbol{x}) \supseteq C_{\text{fb}[t]+1}^{\text{seq}}(\boldsymbol{x}), \forall \boldsymbol{x} \in D, \forall t \geq 1$.

Proof [Proof of Theorem 2, Case 3] By Lemma 18, for the specified value of δ, and choosing $\alpha_t = 2M + 300\gamma_t \log^3(t/\delta)$, $P(f(\boldsymbol{x}) \in C_\tau^{\text{seq}}(\boldsymbol{x}), \forall \boldsymbol{x} \in D, \forall \tau \geq 1) \geq 1 - \delta$. Let $\beta_t = \exp(2C)\alpha_{\text{fb}[t]+1}$, where C satisfies Equation (18); by Lemmas 8 and 9, $P(f(\boldsymbol{x}) \in C_t^{\text{batch}}(\boldsymbol{x}), \forall \boldsymbol{x} \in D, \forall t \geq 1) \geq 1 - \delta$. Then, by Lemma 10, the instantaneous regret is bounded as $r_t \leq \beta_t^{1/2}\sigma_{t-1}^2(\boldsymbol{x})$ for each $t \geq 1$. Application of Lemmas 11 and 12 yields the result. ■

Appendix B. Initialization Set Size Bounds

Thorough initialization of GP-BUCB can drive down the constant C, which bounds the information which can be hallucinated in the course of post-initialization batches and also governs the asymptotic scaling of the regret bound with batch size B. First, we connect the information which can be gained in post-initialization batches with the amount of information being gained in the initialization, through Lemma 4, the formal statement of which is in Section 4.5, and the proof of which is presented here.

Proof [Proof of Lemma 4] Since the initialization procedure is greedy, the marginal information gain $1/2 \log(1 + \sigma_n^{-2}\sigma_{t-1}^2(\boldsymbol{x}))$ is a monotonic function of $\sigma_{t-1}(\boldsymbol{x})$, and information gain is submodular (See Section 3.3), the information gain from $\boldsymbol{y}_{T^{\text{init}}}$, which corresponds to $\boldsymbol{x}_{T^{\text{init}}}^{\text{init}}$, the last element of the initialization set, must be the smallest marginal information gain in the initialization process, and thus no greater than the mean information gain, i.e.,

$$I\left(f; \boldsymbol{y}_{T^{\text{init}}} \mid \boldsymbol{y}_{D_{T^{\text{init}}-1}^{\text{init}}}\right) \leq I\left(f; \boldsymbol{y}_{D_{T^{\text{init}}}^{\text{init}}}\right) / T^{\text{init}}.$$

5. In the proof of Lemma 7.2 of Srinivas et al. (2012), the (GP-UCB-type) *algorithm's internal* GP posterior model of f is exploited to examine $||\mu_t - f||_{k_t}$ in terms of the model's individual conditional distributions for $y_\tau, \tau = 1, \ldots, t$. This argument relies on the GP model and its assumption of i.i.d. Gaussian noise, but does *not* change or violate the problem assumption that the actual observation noise ϵ_t is from an arbitrary, uniformly bounded martingale difference sequence.

Further, again because information gain is submodular and the initialization set was constructed greedily, no subsequent action can yield greater marginal information gain. Thus,

$$\gamma^{\text{init}}_{B-1} \leq (B-1) \cdot I\left(f; \boldsymbol{y}_{T^{\text{init}}} \mid \boldsymbol{y}_{D^{\text{init}}_{T^{\text{init}}-1}}\right).$$

Combining these two inequalities with the definition of $\gamma_{T^{\text{init}}}$ yields the result. ■

Next, we examine how Lemma 4 can be used to bound the regret of the two-stage algorithm. In the two stage algorithm, we may consider two sets of confidence intervals, which do not coincide during the construction of the initialization set, and do coincide afterward; specifically, let $\tilde{\text{fb}}[t]$ be a virtual feedback mapping which receives feedback at every point the actual feedback mapping $\text{fb}[t]$ does, i.e., at time T^{init} and at times thereafter such that Equation (10) is satisfied for all $t \geq T^{\text{init}}+1$ for $C = 1/2\log(C')$, and in addition, *receives feedback during the construction of the initialization set* such that Equation (10) is also satisfied during this time. While the virtual feedback mapping $\tilde{\text{fb}}[t]$ is of course not used by the algorithm to construct confidence intervals or make decisions, it will prove useful for our proof.

Proof [Theorem 5] Assume that there can be constructed an initialization set D^{init} of size T^{init}, subsequent to which the information gain of any batch selected by the GP-BUCB decision rule with respect to $f(\boldsymbol{x})$ for any $\boldsymbol{x} \in D$ is no more than C. Then, for the values of β_t given in the statement of Theorem 2 and using $C = 1/2\log(C')$, where C' is dictated by the choice of kernel,

$$P(|f(\boldsymbol{x}) - \mu_{\tilde{\text{fb}}[t]}(\boldsymbol{x})| \leq \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}), \forall \boldsymbol{x} \in D, \forall t \geq 1) \geq 1-\delta$$

in Cases 1 & 3, and the corresponding statement with $|f(\boldsymbol{x}) - \mu_{\tilde{\text{fb}}[t]}(\boldsymbol{x})| \leq \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}) + t^{-2}$ in Case 2. This result follows from the following sets of lemmas: 7, 8, and 9 (Case 1); 13, 14, 15, and 16 (Case 2); and 18, 8, and 9 (Case 3). Since the actual feedback mapping $\text{fb}[t]$ and $\tilde{\text{fb}}[t]$ coincide for $t \geq T^{\text{init}}+1$, the virtual feedback mapping's probability of confidence interval containment (correctness to known error in Case 2) at all times $t \geq 1$ is a lower bound on the probability that the true, post-initialization confidence intervals constructed using $\text{fb}[t]$ are correct (correct to known error in Case 2); i.e.,

$$\begin{aligned} P(|f(\boldsymbol{x}) - \mu_{\text{fb}[t]}(\boldsymbol{x})| &\leq \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}), \forall \boldsymbol{x} \in D, \forall t \geq T^{\text{init}}+1) \\ &\geq P(|f(\boldsymbol{x}) - \mu_{\tilde{\text{fb}}[t]}(\boldsymbol{x})| \leq \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}), \forall \boldsymbol{x} \in D, \forall t \geq 1) \\ &\geq 1-\delta \end{aligned}$$

in Cases 1 & 3 and similarly,

$$P(|f(\boldsymbol{x}) - \mu_{\text{fb}[t]}(\boldsymbol{x})| \leq \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x}) + t^{-2}, \forall \boldsymbol{x} \in D, \forall t \geq T^{\text{init}}+1) \geq 1-\delta$$

in Case 2. By Lemma 11, $I(\boldsymbol{y}_{T^{\text{init}}+1:T}; \{f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_T)\}) = \frac{1}{2}\sum_{t=T^{\text{init}}+1}^{T} \log(1+\sigma_n^{-2}\sigma_{t-1}^2(\boldsymbol{x}_t))$. Define $R_{T^{\text{init}}+1:T} = \sum_{t=T^{\text{init}}+1}^{T} r_t$. By the same Cauchy-Schwarz argument used in each proof in Appendix A, with probability $\geq 1-\delta$,

$$R_{T^{\text{init}}+1:T} \leq \sqrt{(T-T^{\text{init}})C_1\beta_T\gamma^{\text{init}}_{(T-T^{\text{init}})}} \leq \sqrt{TC_1\beta_T\gamma_T},$$

for all $T \geq 1$, in Cases 1 & 3. The second inequality in the above argument is wasteful, but simplifies the statement of the theorem without affecting asymptotic scaling. In Case 2, we must amend the final bound by adding a term $\pi^2/6$ to the right-hand side as an upper bound on $\sum_{t=T^{\text{init}}+1}^{T} t^{-2}$, itself bounding the generalization error from the virtual discretization. Noting that $R_T = R_{T^{\text{init}}} + R_{T^{\text{init}}+1:T}$, $R_{T^{\text{init}}} = \sum_{t=1}^{T^{\text{init}}} f(x^*) - f(x_t) \leq 2T^{\text{init}}||f||_\infty$, $C' \geq 1$, and $\beta_t = (C')^2 \alpha_{\text{fb}[t]+1}$ (α_t in Case 2), the result follows. ■

B.1 Initialization Set Size: Linear Kernel

For the linear kernel, there exists a logarithmic bound on the maximum information gain of a set of queries, precisely, $\exists\, \eta \geq 0 : \gamma_t \leq \eta d \log(t+1)$ (Srinivas et al., 2010). We attempt to initialize GP-BUCB with a set D^{init} of size T^{init}, where, motivated by this bound and the form of Inequality (15), we assume T^{init} is of the form

$$T^{\text{init}} = k\eta d(B-1)\log B. \tag{20}$$

We must show that there exists a k of finite size for which an initialization set of size T^{init}, as in Equation (20), implies that any subsequent set $\mathcal{S}$, $|\mathcal{S}| = B-1$, produces a conditional information gain with respect to f of no more than C. This requires showing that the inequality $\frac{B-1}{T^{\text{init}}}\gamma_{T^{\text{init}}} \leq C$ holds for this choice of k and thus T^{init}. Since we consider non-trivial batches, i.e., $B-1 \geq 1$, if k is large enough that $k\eta d(B-1) \geq 1$,

$$\log(\log(B) + 1/(k\eta d(B-1))) \leq \log(\log(B)+1) \leq \log B.$$

Using Equation (20) and the bound for $\gamma_{T^{\text{init}}}$, and following algebraic rearrangement, this inequality implies that if $k\eta d(B-1) \geq 1$,

$$\frac{B-1}{T^{\text{init}}}\gamma_{T^{\text{init}}} \leq C \Longleftarrow \frac{\log k}{k\log B} + \frac{\log\eta + \log d}{k\log B} + \frac{2}{k} \leq C.$$

By noting that the maximum of $\frac{\log k}{k}$ over $k \in (0,\infty)$ is $1/e$ and choosing for convenience $C = 2/e$, we obtain for $k \geq 1/(\eta d(B-1))$:

$$\frac{B-1}{T^{\text{init}}}\gamma_{T^{\text{init}}} \leq \frac{2}{e} \Longleftarrow \frac{1}{e\log B} + \frac{1}{k}\left(\frac{\log\eta + \log d + 2\log B}{\log B}\right) \leq \frac{2}{e}.$$

Choosing k to satisfy both constraints simultaneously,

$$\frac{B-1}{T^{\text{init}}}\gamma_{T^{\text{init}}} \leq \frac{2}{e} \Longleftarrow k \geq \max\left[\frac{1}{\eta d(B-1)}, \frac{e(\log\eta + \log d + 2\log B)}{2\log(B) - 1}\right].$$

Thus, for a linear kernel and such a k, an initialization set D^{init} of size T^{init}, where $T^{\text{init}} \geq k\eta d(B-1)\log(B)$, ensures that the hallucinated conditional information in any future batch of size B is $\leq \frac{2}{e}$.

B.2 Initialization Set Size: Matérn Kernel

For the Matérn kernel, $\gamma_t \leq \nu t^{\epsilon}$, $\epsilon \in (0,1)$ for some $\nu > 0$ (Srinivas et al., 2010). Hence:

$$\frac{(B-1)}{T^{\text{init}}}\gamma_{T^{\text{init}}} \leq C \Longleftarrow \frac{\nu(B-1)(T^{\text{init}})^{\epsilon}}{T^{\text{init}}} = \nu(B-1)(T^{\text{init}})^{\epsilon-1} \leq C$$
$$\iff T^{\text{init}} \geq \left(\frac{\nu(B-1)}{C}\right)^{1/(1-\epsilon)}.$$

Thus, for a Matérn kernel, an initialization set D^{init} of size $T^{\text{init}} \geq \left(\frac{\nu(B-1)}{C}\right)^{1/(1-\epsilon)}$ implies that the conditional information gain of any future batch is $\leq C$. Choosing $C = 1$, we obtain the results presented in the corresponding row of Table 1.

B.3 Initialization Set Size: Squared-Exponential (RBF) Kernel

For the RBF kernel, the information gain is bounded by an expression similar to that of the linear kernel, $\gamma_t \leq \eta(\log(t+1))^{d+1}$ (Srinivas et al., 2010). Again, motivated by Inequality (15), one reasonable choice for an initialization set size is $T^{\text{init}} = k\eta(B-1)(\log B)^{d+1}$. We again attempt to show that there exists a finite k such that the conditional information gain of any post-initialization batch is $\leq C$. By a similar parallel argument to that for the linear kernel (Appendix B.1), and assuming that $B \geq 2$ and $k\eta(B-1) \geq 1$, it follows that

$$\frac{B-1}{T^{\text{init}}}\gamma_{T^{\text{init}}} \leq C \Longleftarrow \frac{\log k + \log\eta + \log(B-1)}{k^{1/(d+1)}(\log B)} \frac{\log\left[(\log B)^{d+1}+1\right]}{k^{1/(d+1)}(\log B)} \leq C^{1/(d+1)}$$
$$\Longleftarrow \frac{\log k}{k^{1/(d+1)}(\log B)} + \frac{\log\eta}{k^{1/(d+1)}(\log B)} + \frac{(d+2)}{k^{1/(d+1)}} \leq C^{1/(d+1)},$$

where the last implication follows because for $a \geq 0, b \geq 1, (a^b+1) \leq (a+1)^b$.

By noting that the maximum of $k^{-1/(d+1)}\log k$ over $k \in (0,\infty)$ is $(d+1)/e$ and choosing $C = (2(d+1)/e)^{d+1}$, we obtain for $k \geq 1/(\eta(B-1))$:

$$\frac{B-1}{T^{\text{init}}}\gamma_{T^{\text{init}}} \leq \left(\frac{2d+2}{e}\right)^{d+1} \Longleftarrow \frac{d+1}{e\log B} + \frac{1}{k^{1/(d+1)}}\left(\frac{\log\eta + (d+2)\log B}{\log B}\right) \leq \frac{2d+2}{e},$$

or equivalently, incorporating the constraint $k \geq 1/(\eta(B-1))$ explicitly,

$$\frac{B-1}{T^{\text{init}}}\gamma_{T^{\text{init}}} \leq \left(\frac{2d+2}{e}\right)^{d+1} \Longleftarrow k \geq \max\left[\frac{1}{\eta(B-1)}, \left(\frac{e(\log\eta + (d+2)\log B)}{(d+1)(2\log(B)-1)}\right)^{d+1}\right].$$

Thus, for a Squared-Exponential kernel and such a k, an initialization set D^{init} of size T^{init}, where $T^{\text{init}} \geq k\eta(B-1)(\log(B))^{d+1}$, ensures that the hallucinated conditional information in any future batch of size B is no more than $\left(\frac{2d+2}{e}\right)^{d+1}$.

References

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2312–2320, 2011.

Jacob Abernethy, Elad Hazan, and Alexander Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT)*, pages 263–274, 2008.

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research (JMLR)*, 3:397–422, 2002.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

Javad Azimi, Alan Fern, and Xiaoli Fern. Batch Bayesian optimization via simulation matching. In *Advances in Neural Information Processing Systems (NIPS)*, pages 109–117, 2010.

Javad Azimi, Alan Fern, Xiaoli Fern, Glencora Borradale, and Brent Heeringa. Batch active learning via coordinated matching. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1199–1206, 2012a.

Javad Azimi, Ali Jalali, and Xiaoli Fern. Hybrid batch Bayesian optimization. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1215–1222, 2012b.

Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *Computing Research Repository (CoRR)*, abs/1012.2599, 2010.

Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. Online optimization in X-armed bandits. In *Advances in Neural Information Processing Systems (NIPS)*, pages 201–208, 2008.

Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandit problems. In *Proceedings of the 20th International Conference on Algorithmic Learning Theory (ALT)*, pages 23–37, 2009.

Yuxin Chen and Andreas Krause. Near-optimal batch mode active learning and adaptive submodular optimization. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 160–168, 2013.

Dennis D. Cox and Susan John. SDO: A statistical method for global optimization. *Multidisciplinary Design Optimization: State of the Art*, pages 315–329, 1997.

Varsha Dani, Thomas P. Hayes, and Sham M. Kakade. Stochastic linear optimization under bandit feedback. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT)*, pages 355–366, 2008.

Miroslav Dudik, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. Efficient optimal learning for contextual bandits. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 169–178, 2011.

Subhashis Ghosal and Anindya Roy. Posterior consistency of Gaussian process prior for nonparametric binary regression. *The Annals of Statistics*, 34(5):2413–2429, Oct. 2006.

David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. Kriging is well-suited to parallelize optimization. In Yoel Tenne and Chi-Keong Goh, editors, *Computational Intelligence in Expensive Optimization Problems*, volume 2 of *Adaptation, Learning, and Optimization*, pages 131–162. Springer Berlin Heidelberg, 2010.

John Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society, B*, 41(2):148–177, 1979.

Philipp Hennig and Christian J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research (JMLR)*, 13:1809–1837, June 2012.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

Pooria Joulani, András György, and Csaba Szepesvári. Online learning under delayed feedback. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 1453–1461, 2013.

Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *ACM Symposium on Theory of Computing (STOC)*, pages 681–690. Association for Computing Machinery, Inc., May 2008.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *Machine Learning: ECML*, pages 282–293, 2006.

Andreas Krause and Carlos Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 324–331, 2005.

Andreas Krause and Cheng Soon Ong. Contextual Gaussian process bandit optimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2447–2455, 2011.

Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research (JMLR)*, 9:235–284, February 2008.

Daniel Lizotte, Tao Wang, Michael Bowling, and Dale Schuurmans. Automatic gait optimization with Gaussian process regression. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 944–949, 2007.

Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In J. Stoer, editor, *Optimization Techniques*, volume 7 of *Lecture Notes in Control and Information Sciences*, pages 234–243. Springer, 1978.

Jonas Mockus. *Bayesian Approach to Global Optimization: Theory and Applications*. Kluwer Academic Publishers, 1989.

Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. *The application of Bayesian methods for seeking the extremum*, volume 2, pages 117–129. Elsevier, 1978.

Carl Edward Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (GPML) toolbox. *Journal of Machine Learning Research (JMLR)*, 11:3011–3015, 2010.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

Ilya O. Ryzhov, Warren B. Powell, and Peter I. Frazier. The knowledge gradient algorithm for a general class of online learning problems. *Operations Research*, 60(1):180–195, 2012.

Bernhard Schölkopf and Alex J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2002.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 1015–1022, 2010.

Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *Information Theory, IEEE Transactions on*, 58(5):3250–3265, 2012.

Christian Widmer, Nora C. Toussaint, Yasemin Altun, and Gunnar Rätsch. Inferring latent task structure for multitask learning by multiple kernel learning. *BMC Bioinformatics*, 11(Suppl 8):S5, 2010.

Journal of Machine Learning Research 15 (2014) 4105-4143 Submitted 12/12; Revised 5/14; Published 12/14

Active Imitation Learning: Formal and Practical Reductions to I.I.D. Learning

Kshitij Judah JUDAHK@EECS.OREGONSTATE.EDU
Alan P. Fern AFERN@EECS.OREGONSTATE.EDU
Thomas G. Dietterich TGD@EECS.OREGONSTATE.EDU
Prasad Tadepalli TADEPALL@EECS.OREGONSTATE.EDU
School of Electrical Engineering and Computer Science
Oregon State University
1148 Kelley Engineering Center
Corvallis, OR 97331-5501, USA

Editor: Joelle Pineau

Abstract

In standard passive imitation learning, the goal is to learn a policy that performs as well as a target policy by passively observing full execution trajectories of it. Unfortunately, generating such trajectories can require substantial expert effort and be impractical in some cases. In this paper, we consider *active imitation learning* with the goal of reducing this effort by querying the expert about the desired action at individual states, which are selected based on answers to past queries and the learner's interactions with an environment simulator. We introduce a new approach based on reducing active imitation learning to active i.i.d. learning, which can leverage progress in the i.i.d. setting. Our first contribution is to analyze reductions for both non-stationary and stationary policies, showing for the first time that the label complexity (number of queries) of active imitation learning can be less than that of passive learning. Our second contribution is to introduce a practical algorithm inspired by the reductions, which is shown to be highly effective in five test domains compared to a number of alternatives.

Keywords: imitation learning, active learning, active imitation learning, reductions

1. Introduction

Traditionally, passive imitation learning involves learning a policy that performs nearly as well as an expert's policy based on a set of trajectories of that policy. However, generating such trajectories is often tedious or even impractical for an expert (e.g., real-time low-level control of multiple game agents). In order to address this issue, we consider active imitation learning where full trajectories are not required, but rather the learner asks queries about specific states, which the expert labels with the correct actions. The goal is to learn a policy that is nearly as good as the expert's policy using as few queries as possible.

The active learning problem for i.i.d. supervised learning[1] has received considerable attention both in theory and in practice (Settles, 2012), which motivates leveraging that

1. i.i.d. stands for independent and identically distributed. The i.i.d. supervised learning setting is where the input data during both training and testing are drawn independently from the same data distribution.

work for active imitation learning. However, the direct application of i.i.d. approaches to active imitation learning can be problematic. This is because active i.i.d. learning algorithms assume access to either a target distribution over unlabeled input data (in our case states) or a large sample drawn from it. The goal then is to select the most informative query to ask, usually based on some combination of label (in our case actions) uncertainty and unlabeled data density. Unfortunately, in active imitation learning, the learner does not have direct access to the target state distribution, which is the state distribution induced by the unknown expert policy.

In principle, one could approach active imitation learning by assuming a uniform or an arbitrary distribution over the state space and then apply an existing active i.i.d. learner. However, such an approach can perform very poorly. This is because if the assumed distribution is considerably different from that of the expert, then the learner is prone to ask queries in states rarely or even never visited by the expert. For example, consider a bicycle balancing problem. Clearly, asking queries in states where the bicycle has entered an unavoidable fall is not very useful, because no action can prevent a crash. However, an active i.i.d. learning technique will tend to query in such uninformative states, leading to poor performance, as shown in our experiments. Furthermore, in the case of a human expert, a large number of such queries poses serious usability issues, since labeling such states is clearly wasted effort from the expert's perspective.

In this paper, we consider the problem of reducing active imitation learning to active i.i.d. learning both in theory and practice. Our first contribution is to analyze the Probably Approximately Correct[2] (PAC) label complexity (number of expert queries) of a reduction for learning non-stationary policies, which requires only minor modification to existing results for passive learning. Our second contribution is to introduce a reduction for learning stationary policies resulting in a new algorithm, *Reduction-based Active Imitation Learning (RAIL)*, and an analysis of the label complexity. The resulting complexities for active imitation learning are expressed in terms of the label complexity for the i.i.d. case and show that there can be significant query savings compared to existing results for passive imitation learning. Our third contribution is to describe a new practical algorithm, RAIL-DA (for data aggregation), inspired by the RAIL algorithm, which makes a series of calls to an active i.i.d. learning algorithm. We evaluate RAIL-DA in five test domains and show that it is highly effective when used with an i.i.d. algorithm that takes the unlabeled data density into account.

The rest of the paper is organized as follows. We begin by reviewing the relevant related work in Section 2. In Section 3, we present the necessary background material and describe the active imitation learning problem setup. In Section 4, we present the proposed reductions for the cases of non-stationary and stationary policies. In Section 5, we present the RAIL-DA algorithm. In Section 6, experimental results are presented. In Section 7, we summarize and present some directions for future research.

2. Related Work

Active learning has been studied extensively in the i.i.d. supervised learning setting (Settles, 2012) but to a much lesser degree for sequential decision making, which is the focus of

2. See Section 3.3.

active imitation learning. Several studies have considered active learning for *reinforcement learning (RL)* (Clouse, 1996; Mihalkova and Mooney, 2006; Gil et al., 2009; Doshi et al., 2008), where learning is based on both autonomous exploration and queries to an expert. In our imitation learning framework, in contrast, we do not assume a reward signal and learn only from expert queries. Other work (Shon et al., 2007) studies active imitation learning in a multiagent setting, where the expert is itself a reward seeking agent which acts to maximize its own reward, and hence is not necessarily helpful for learning. In the current setting, we only consider helpful experts.

One approach to imitation learning is *inverse RL (IRL)* (Ng and Russell, 2000), where a reward function is learned based on a set of target policy trajectories. The learned reward function and transition dynamics are then given to a planner to obtain a policy. There has been limited work on active IRL. This includes Active Sampling (Lopes et al., 2009), a Bayesian approach where the posterior over reward functions is used to select the state with maximum uncertainty over the actions. Another Bayesian approach (Cohn et al., 2010, 2011) models uncertainty about the entire MDP model and uses a decision-theoretic criterion, Expected Myopic Gain (EMG), to select various types of queries to pose to the expert, e.g., queries about transition dynamics, the reward function, or optimal action at a particular state. For autonomous navigation tasks, Silver proposed two active IRL techniques that request demonstrations from the expert on examples that are either novel (novelty reduction) or uncertain (uncertainty reduction) to the learner (Silver et al., 2012). Specifically, in novelty reduction, a start and goal location is selected such that the path that may most likely be demonstrated by the expert results in the learner seeing novel portions of the navigation terrain. This helps in learning behavior on previously unseen regions of the navigation terrain. In uncertainty reduction, a start and goal location is selected such that there is high uncertainty about the best path from start to goal.

While promising, the scalability of these approaches is hindered by the assumptions made by IRL, and these approaches have only been demonstrated on small problems. In particular, they require that the exact domain dynamics are provided or can be learned, which is often not realistic, for example, in the Wargus domain considered in this paper. Furthermore, even when a model is available, prior IRL approaches require an efficient planner for the MDP model. For the large domains considered in this paper, standard planning techniques for flat MDPs, which scale polynomially in the number of states, are not practical. While there has been substantial work on MDP planning for large domains via the use of factored representations (e.g., Boutilier et al. 1999) or simulators (e.g., Kocsis and Szepesvri 2006), robustness and scalability are still problematic in general.

To facilitate scalability, rather than following an IRL framework we consider a *direct imitation* framework where we attempt to directly learn a policy instead of the reward function and/or transition dynamics. Unlike inverse RL, this framework does not require an exact dynamic model nor an efficient planner. Rather, our approach requires only a simulator of the environment dynamics that is able to generate trajectories given a policy. The simulator may or may not be exact, and the performance of our approach will depend on how precise the simulator is. Even though a precise simulator is not always available for a real world domain, for many domains such a simulator is often available (e.g., flight simulators, computer network simulators etc.), even when a compact description of the transition dynamics and/or a planner are not.

Active learning work in the direct imitation framework includes Confidence Based Autonomy (CBA) (Chernova and Veloso, 2009), and the related dogged learning framework (Grollman and Jenkins, 2007), where a policy is learned in an online manner as it is executed. When the learner is uncertain about what to do at the current state, the policy is paused and the expert is queried about what action to take, resulting in a policy update. The execution resumes from the current state with the learner taking the action suggested by the expert. One can roughly view CBA as a reduction of imitation learning to stream-based active learning where the learner receives unlabeled inputs (states) one at a time and must decide whether or not to request the label (action) of the current input. CBA makes this decision by estimating its uncertainty about the action to take at a given state and requesting an action label for states with uncertainty above a threshold. One difficulty in applying this approach is setting the uncertainty threshold for querying the expert. While an automated threshold selection approach is suggested by Chernova and Veloso (Chernova and Veloso, 2009), our experiments show that it is not always effective (See Section 6). In particular, we observed that the proposed threshold selection mechanism is quite sensitive to the initial training data supplied to the learner.

Recently, Ross et al. proposed novel algorithms for imitation learning that are able to actively query the expert on states encountered during the execution of the policy being trained (Ross and Bagnell, 2010; Ross et al., 2011). The motivation behind these algorithms is to eliminate the discrepancy between the training (expert's) and test (learner's) state distributions that arises in the traditional passive imitation learning approach whenever the learned policy is unable to exactly mimic the expert's policy. This discrepancy often leads to the poor performance of the traditional approach. Note that such an issue is not present in the i.i.d. learning setting, where mistakes made by the learner do not influence the distribution of future test samples. They show that under certain assumptions their algorithms achieve better theoretical performance guarantees than traditional passive imitation learning.

However, because the primary goal of these algorithms is not to minimize the labeling effort of the expert, these algorithms query the expert quite aggressively, which makes them impractical for human experts or computationally expensive with automated experts. To see this, consider the first iteration of the DAGGER algorithm proposed by Ross et al. (Ross et al., 2011). In the first iteration, DAGGER trains a policy on a set of expert-generated trajectories, as in passive imitation learning. Thus, in practice the query complexity of the first iteration of DAGGER will be similar to that of passive imitation learning. In subsequent iterations, additional queries are asked by querying the expert on states along trajectories produced by learned policies in prior iterations. In contrast, our work focuses on active querying for the purpose of minimizing the expert's labeling effort. In particular, we show that an active approach can achieve an improved query complexity over passive in theory (under certain assumptions) and in practice. Like our work, their approach also requires a dynamics simulator to help select queries.

Our goal in this paper is to study the problem of active imitation learning and show that it can achieve better label complexity than passive imitation learning. To this end, we mention some prior work on the theoretical analysis of the label complexity of passive imitation learning. Khardon formalized a model for passive imitation learning of deterministic stationary policies in the realizable setting and gave a PAC-style label complexity result

(Khardon, 1999). He showed that for any policy class for which there exists a consistent learner, the class is efficiently learnable in the sense that only a polynomial number of expert trajectories are required by the learner to produce a policy as good as the expert's policy. However, the result holds for only deterministic policies in the realizable setting and the generalizations to stochastic policies and the agnostic setting were left as future work.

More recently, Syed and Schapire performed theoretical analysis of passive imitation learning in a more general setting where the expert policy is allowed to be stochastic and the learning can be agnostic (Syed and Schapire, 2010). In their analysis, they take a reduction based approach, where the problem of passive imitation learning is reduced to classification, and they relate the performance of the learned policy to the accuracy of the classifier. Standard PAC analysis can then be used to show that only a polynomial number of expert trajectories are required to achieve the desired level of performance. A similar analysis was done by Ross and Bagnell (Ross and Bagnell, 2010). To our knowledge, no prior work has addressed the relative sample complexity of active versus passive imitation learning, which is one of the primary contributions of this paper. Some of the material in this paper appeared in an earlier version of the paper (Judah et al., 2012).

3. Problem Setup and Background

In this section, we present the necessary background material and formally set up the active imitation learning problem.

3.1 Markov Decision Processes

We consider imitation learning in the framework of Markov decision processes (MDPs). An MDP is a tuple $\langle S, A, P, R, I \rangle$, where S is the set of states, A is the finite set of actions, $P(s, a, s')$ is the transition function denoting the probability of transitioning to state s' upon taking action a in state s, $R(s, a) \in [0, 1]$ is the reward function giving the immediate reward in state s upon taking action a, and I is the initial state distribution. A stationary policy $\pi : S \mapsto A$ is a deterministic mapping from states to actions such that $\pi(s)$ indicates the action to take in state s when executing π. A non-stationary policy is a tuple $\pi = (\pi_1, \ldots, \pi_T)$ of T stationary policies such that $\pi(s, t) = \pi_t(s)$ indicates the action to take in state s at time t when executing π, where T is the time horizon. The expert's policy, which we assume is deterministic, is denoted by π^*.

A key concept used in this paper is the notion of a state distribution of a policy at a particular time step. We use $d_\pi^t : S \mapsto [0, 1]$ to denote the state distribution induced at time step t by starting in $s_1 \sim I$ and then executing π. Note that $d_\pi^1 = I$ for all policies. We use $d_\pi = \frac{1}{T} \sum_{t=1}^{T} d_\pi^t$ to denote the state distribution induced by policy π over T time steps. To sample an (s, a) pair from d_π^t, we start in $s_1 \sim I$, execute π to generate a trajectory $\mathcal{T} = (s_1, a_1, \ldots, s_T, a_T, s_{T+1})$ and set $(s, a) = (s_t, a_t)$. Similarly, to sample from d_π, we first sample a random time step $t \in \{1, \ldots, T\}$, and then sample an (s, a) pair from d_π^t. Note that in order to sample from d_{π^*} (or $d_{\pi^*}^t$), we need to execute π^*. Throughout the paper, we assume that the only way π^* can be executed is by querying the expert for an action in the current state and executing the given action, which puts significant burden on the expert.

The T-horizon value of a policy $V(\pi)$ is the expected total reward of trajectories that start in $s_1 \sim I$ at time $t = 1$ and then execute π for T steps. This can be expressed as

$$V(\pi) = T \cdot \mathbb{E}_{s \sim d_\pi}[R(s, \pi(s))].$$

The regret of a policy π with respect to an expert policy π^* is equal to $V(\pi^*) - V(\pi)$.

3.2 Problem Setup

Passive Imitation Learning. In imitation learning, the goal is to learn a policy π with a small regret with respect to the expert. In this work, we consider the direct imitation learning setting, where the learner directly selects a policy π from a hypothesis class Π (e.g., linear action classifiers). In the *passive imitation learning* setup, the protocol is to provide the learner with a training set of full execution trajectories of π^* and the state-action pairs (or a sample of them) are passed to a passive i.i.d. supervised learning algorithm L_p. The hypothesis $\pi \in \Pi$ that is returned by L_p is used as the learned policy.

Active Imitation Learning. To help avoid the cost of generating full trajectories, the *active imitation learning* setup allows the learner to pose *action queries*. Each action query involves presenting a state s to the expert and then obtaining the desired action $\pi^*(s)$ from the expert. In addition to having access to the expert for answering queries, we assume that the learner has access to a simulator of the MDP. The input to the simulator is a policy π and a horizon T. The simulator output is a state trajectory that results from executing π for T steps starting in the initial state. The learner is allowed to interact with this simulator as part of its query selection process. The simulator is not assumed to provide a reward signal, which means that the learner cannot find π by pure reinforcement learning. The only way for the learner to gain information about the target policy is through queries to the expert at selected states.

Given access to the expert and the simulator of the MDP, the goal in active imitation learning is to learn a policy $\pi \in \Pi$ that has a small regret by posing as few queries to the expert as possible. Note that it is straightforward for the active learner to generate full expert trajectories by querying the expert at each state of the simulator it encounters. Thus, an important baseline active learning approach is to generate an appropriate number N of expert trajectories for consumption by a passive learner. The number of queries for this baseline is $N \cdot T$. A fundamental question that we seek to address is whether an active learner can achieve the same performance with significantly fewer queries both in theory and in practice.

3.3 Background on I.I.D. Learning

Since our analysis in the next two sections is based on reducing to active i.i.d. learning and comparing to passive i.i.d. learning, we briefly review the *Probably Approximately Correct (PAC)* (Valiant, 1984) learning formulation for the i.i.d. setting. Here we consider the realizable PAC setting, which will be the focus of our initial analysis. Section 4.3 extends to the non-realizable or agnostic setting.

Passive Learning. In passive i.i.d. supervised learning, N i.i.d. data samples are drawn

from an unknown distribution $D_{\mathcal{X}}$ over an input space $\mathcal{X}$ and are labeled according to an unknown target classifier $f : \mathcal{X} \mapsto \mathcal{Y}$, where $\mathcal{Y}$ denotes the label space. In the realizable PAC setting it is assumed that f is an element of a known class of classifiers H and, given a set of N examples, a learner outputs a hypothesis $h \in H$. Let $e_f(h, D_{\mathcal{X}}) = \mathbb{E}_{x \sim D_{\mathcal{X}}}[h(x) \neq f(x)]$ denote the generalization error of the returned classifier h. Standard PAC learning theory provides a bound on the number of labeled examples that is sufficient to guarantee that for any distribution $D_{\mathcal{X}}$, with probability at least $1 - \delta$, the returned classifier h will satisfy $e_f(h, D_{\mathcal{X}}) \leq \epsilon$. We will denote this bound by $N_p(\epsilon, \delta)$, which corresponds to the label/query complexity of i.i.d. passive supervised learning for a class H. We will also denote a passive learner that achieves this label complexity as $L_p(\epsilon, \delta)$.

Active Learning. In active i.i.d. learning, the learner is given access to two resources rather than just a set of training data: 1) A "cheap" resource (Sample) that can draw an unlabeled sample from $D_{\mathcal{X}}$ and provide it to the learner when requested, 2) An "expensive" resource (Label) that can label a given unlabeled sample according to target concept f when requested. Given access to these two resources, an active learning algorithm is required to learn a hypothesis $h \in H$ while posing as few queries to Label as possible. It can, however, pose a much larger number of queries to Sample (though still polynomial), as it is cheap.

Unlike passive i.i.d. learning, formal label/query complexity results for active i.i.d. learning depend not only on the hypothesis class being considered, but also on joint properties of the target hypothesis and data distribution (e.g., as measured by the disagreement coefficient proposed by Hanneke, 2009). We use $N_a(\epsilon, \delta, D_{\mathcal{X}})$ to denote the label complexity (i.e., calls to Label) that is sufficient for an active learner to return an h that for distribution $D_{\mathcal{X}}$ with probability at least $1 - \delta$ satisfies $e_f(h, D_{\mathcal{X}}) \leq \epsilon$. Note that here we did not explicitly parameterize N_a by the target hypothesis f since, in the context of our work, f will correspond to the expert policy and can be considered as fixed. We will denote an active learner that achieves this label complexity as $L_a(\epsilon, \delta, D)$, where the final argument D indicates that the Sample function used by L_a samples from distribution D.

It has been shown that for certain problem classes, N_a can be exponentially smaller than N_p (Hanneke, 2009; Dasgupta, 2011). For example, in the realizable learning setting (i.e., the target concept is in the hypothesis space), for any active learning problem with finite VC-dimension and finite disagreement coefficient, the sample complexity is exponentially smaller for active learning compared to passive learning with respect to $\frac{1}{\epsilon}$. That is, ignoring the dependence on δ, $N_p = O(\frac{1}{\epsilon})$ whereas $N_a = O(\log(\frac{1}{\epsilon}))$. A concrete problem for which this is the case is when the data are uniformly distributed on a unit sphere in a d dimensional input space $\mathbb{R}^d$, and the hypothesis space H consists of homogeneous linear separators. As an example active learning algorithm that achieves this performance, the algorithm of Cohn et al. (Cohn et al., 1994) simply samples a sequence of unlabeled examples and queries for the label of example x only when there are at least two hypotheses that disagree on the label of x, but agree on all previously labeled examples.

While results such as the above gives some theoretical justification for the use of active learning over passive learning in the i.i.d. setting, the results and understanding are not nearly as broad as for passive learning. Further, there are known limitations to the advantages of active versus passive learning. For example, lower bounds have been shown (Dasgupta, 2006; Beygelzimer et al., 2009a) implying that no active learning algorithm can

asymptotically improve over passive learning across all problems with finite VC-dimension. However, despite the limited theoretical understanding, there is much empirical evidence that in practice active learning algorithms can often dramatically reduce the required amount of labeled data compared to passive learning. Further, there are active learning algorithms that in the worst case are guaranteed to achieve performance similar to passive learning in the worst case, while also showing exponential improvement in the best case (Beygelzimer et al., 2009a).[3]

4. Reductions for Active Imitation Learning

One approach to solving novel machine learning problems is via reduction to well-studied core problems. A key advantage of this reduction approach is that theoretical and empirical advances on the core problems can be translated to the more complex problem. For example, i.i.d. multi-class and cost-sensitive classification have been reduced to i.i.d. binary classification (Zadrozny et al., 2003; Beygelzimer et al., 2009b). In particular, these reductions allow guarantees regarding binary classification to translate to the target problems. Further, the reduction-based algorithms have shown equal or better empirical performance compared to specialized algorithms. More closely related to our work, in the context of sequential decision making, both imitation learning and structured prediction have been reduced to i.i.d. classification (Daumé et al., 2009; Syed and Schapire, 2010; Ross and Bagnell, 2010).

In this section, we consider a reduction approach to active imitation learning. In particular, we reduce to active i.i.d. learning, which is a core problem that has been the focus of much theoretical and empirical work. The key result is to relate the label complexity of active imitation learning to the label complexity of active i.i.d. learning. In doing so, we can assess when improved label complexity (either empirical or theoretical) of active i.i.d. learning over passive i.i.d. learning can translate to improved label complexity of active imitation learning over passive imitation learning. In what follows, we first present a reduction for the case of deterministic non-stationary policies. Next, we give a reduction for the more difficult case of deterministic stationary policies.

4.1 Non-Stationary Policies

Syed and Schapire analyze the traditional reduction from passive imitation learning to passive i.i.d. learning for non-stationary policies (Syed and Schapire, 2010). The algorithm receives N expert trajectories as input, noting that the state-action pairs at time t across trajectories can be viewed as i.i.d. draws from distribution $d^t_{\pi^*}$. The algorithm, then returns the non-stationary policy $\hat{\pi} = (\hat{\pi}_1, \ldots, \hat{\pi}_T)$, where $\hat{\pi}_t$ is the policy returned by running the learner L_p on examples from time t.

3. A simple example of such an algorithm is the previously mentioned algorithm of Cohn et al. (Cohn et al., 1994) for the realizable learning setting. In this case, active learning can be stopped after drawing a number of unlabeled instances equal to the passive query complexity of the hypothesis class. This is because, for each instance, the algorithm either asks a query to get the label or the algorithm knows the label in cases when there is no disagreement. Thus, in the worst case, the algorithm will query each drawn example and ask for the same number of labels as a passive algorithm. But when the disagreement coefficient is finite, exponentially fewer queries will be made.

Let $\epsilon_t = e_{\pi_t^*}(\hat{\pi}_t, d^t_{\pi^*})$ be the generalization error of $\hat{\pi}_t$ at time t. Syed and Schapire (Syed and Schapire, 2010, Lemma 3)[4] show that if at each time step $\epsilon_t \leq \epsilon'$, then $V(\hat{\pi}) \geq V(\pi^*) - \epsilon' T^2$. Hence, if we are interested in learning a $\hat{\pi}$ whose regret is no more than ϵ with high probability, then we must simultaneously guarantee that with high probability $\epsilon_t \leq \frac{\epsilon}{T^2}$ at all time steps. This can be achieved by calling the passive learner L_p at each time step with $N_p(\frac{\epsilon}{T^2}, \frac{\delta}{T})$ examples. Thus, the overall passive label complexity of this algorithm (i.e., the number of actions provided by the expert) is $T \cdot N_p(\frac{\epsilon}{T^2}, \frac{\delta}{T})$. To our knowledge, this is the best known label complexity for passive imitation learning of non-stationary policies.

Our goal now is to provide a reduction from active imitation learning to active i.i.d. learning that can achieve an improved label complexity. A naive way to do this would simply replace calls to L_p in the above approach with calls to an active learner L_a. Note, however, that in order to do this the active learner at time step t requires the ability to sample from the unlabeled distribution $d^t_{\pi^*}$. Generating each such unlabeled sample requires executing the expert policy for t steps from the initial state, which in turn requires t label queries to the expert. Thus, the label complexity of this naive approach will be at least linearly related to the number of unlabeled examples required by the active i.i.d. learning algorithm. Typically, this number is similar to the passive label complexity rather than the potentially much smaller active label complexity. Thus, the naive reduction does not yield an advantage over passive imitation learning.

It turns out that for a slightly more sophisticated reduction to passive i.i.d. learning, introduced by Ross and Bagnell (Ross and Bagnell, 2010), it is possible to simply replace L_p with L_a and maintain the potential benefit of active learning. Ross and Bagnell introduced the forward training algorithm for non-stationary policies, which trains a non-stationary policy in a series of T iterations. In particular, iteration t trains policy $\hat{\pi}_t$ by calling a passive learner L_p on a labeled data set drawn from the state distribution induced at time t by the non-stationary policy $\hat{\pi}^{t-1} = (\hat{\pi}_1, \ldots, \hat{\pi}_{t-1})$, where $\hat{\pi}_1$ is learned on states drawn from the initial distribution I. The motivation for this approach is to train the policy at time step t based on the same state-distribution that it will encounter when being run after learning. By doing this, they show that the algorithm has a worst case regret of ϵT^2 and under certain assumptions can achieve a regret as low as $O(\epsilon T)$.

Importantly, the state-distribution used to train $\hat{\pi}_t$ given by $d^t_{\hat{\pi}^{t-1}}$ is easy for the learner to sample from without making queries to the expert. In particular, to generate a sample the learner can simply simulate $\hat{\pi}^{t-1}$, which is available from previous iterations, from a random initial state and return the state at time t. Thus, we can simply replace the call to L_p at iteration t with a call to L_a with unlabeled state distribution $d^t_{\hat{\pi}^{t-1}}$ as input. More formally, the *active forward training algorithm* is presented in Algorithm 1.

Ross and Bagnell (Ross and Bagnell, 2010, Theorem 3.1) give the worst case bound on the regret of the forward training algorithm which assumes the generalization error at each iteration is bounded by ϵ. Since we also maintain that assumption when replacing L_p with L_a (the active variant) we immediately inherit that bound.

4. The main result of Syed and Schapire (Syed and Schapire, 2010) holds for stochastic expert policies and requires a more complicated analysis that results in a looser bound. Lemma 3 is strong enough for deterministic expert policies, which is the assumption made in our work.

Algorithm 1 Active Forward Training

Input: active i.i.d. learning algorithm L_a, ϵ, δ

Output: non-stationary policy $\hat{\pi} = (\hat{\pi}_1, \ldots, \hat{\pi}_T)$

1: Initialize $\hat{\pi}_1 = L_a(\epsilon, \frac{\delta}{T}, I)$ ▷ queries by L_a answered by expert; unlabeled data is generated from initial state distribution I.
2: **for** $t = 2$ to T **do**
3: $\quad \hat{\pi}^{t-1} = (\hat{\pi}_1, \ldots, \hat{\pi}_{t-1})$
4: $\quad \hat{\pi}_t = L_a(\epsilon, \frac{\delta}{T}, d^t_{\hat{\pi}^{t-1}})$ ▷ queries by L_a answered by expert; unlabeled data is generated using simulator and $\hat{\pi}^{t-1}$ as described in the main text.
5: **end for**
6: **return** $\hat{\pi} = (\hat{\pi}_1, ..., \hat{\pi}_T)$

Proposition 1 *Given a PAC active i.i.d. learning algorithm L_a, if active forward training is run by giving L_a parameters ϵ and $\frac{\delta}{T}$ at each step, then with probability at least $1 - \delta$ it will return a non-stationary policy $\hat{\pi}$ such that $V(\hat{\pi}) \geq V(\pi^*) - \epsilon T^2$.*

Note that L_a is run with $\frac{\delta}{T}$ as the reliability parameter to ensure that all T iterations succeed with probability at least $1 - \delta$.

We can apply Proposition 1 to obtain the overall label complexity of active forward training required to achieve a regret of less than ϵ with probability at least $1 - \delta$. In particular, we must run the active learner at each of the T iterations with parameters $\frac{\epsilon}{T^2}$ and $\frac{\delta}{T}$, giving an overall label complexity of $\sum_{t=1}^{T} N_a(\frac{\epsilon}{T^2}, \frac{\delta}{T}, d^t_{\hat{\pi}^{t-1}})$, where $d^1_{\hat{\pi}^0} = I$ and the $\hat{\pi}^{t-1}$ are random variables in this expression. Recall, from above, that the best known label complexity of passive imitation learning is $T \cdot N_p(\frac{\epsilon}{T^2}, \frac{\delta}{T})$.

Comparing these quantities we see that if we use an active learning algorithm whose sample complexity is no worse than that of passive, i.e., $N_a(\frac{\epsilon}{T^2}, \frac{\delta}{T}, d^t_{\hat{\pi}^{t-1}})$ is no worse than $N_p(\frac{\epsilon}{T^2}, \frac{\delta}{T})$ for any t, then the expected sample complexity of active imitation learning will be no worse than the passive case. As mentioned in the previous section, such i.i.d. active learning algorithms can be realized. Further, if in addition, for some iterations the expected value of $N_a(\frac{\epsilon}{T^2}, \frac{\delta}{T}, d^t_{\hat{\pi}^{t-1}})$ for some values of t is better than the passive complexity, then there will be an overall expected improvement over passive imitation learning. While this additional condition cannot be verified in general, we know that such cases can exist, including cases of exponential improvement. Further, empirical experience in the i.i.d. setting also suggests that in practice N_a can often be expected to be substantially smaller than N_p and rarely worse. The above result suggests that those empirical gains will be able to transfer to the imitation learning setting.

4.2 Stationary Policies

A drawback of active forward training is that it is impractical for large T and the resulting policy cannot be run indefinitely. We now consider the case of learning stationary policies; first we review the existing results for passive imitation learning.

In the traditional approach, a stationary policy $\hat{\pi}$ is trained on the expert state distribution d_{π^*} using a passive learning algorithm L_p and returning a stationary policy $\hat{\pi}$. Ross and Bagnell (Ross and Bagnell, 2010, Theorem 2.1) show that if the generalization error of $\hat{\pi}$

Algorithm 2 RAIL

Input: active i.i.d. learning algorithm L_a, ϵ, δ
Output: stationary policy $\hat{\pi}$

1: Initialize $\hat{\pi}^0$ to arbitrary policy or based on prior knowledge
2: **for** $t = 1$ to T **do**
3: $\quad \hat{\pi}^t = L_a(\epsilon, \frac{\delta}{T}, d_{\hat{\pi}^{t-1}})$ ▷ queries by L_a answered by expert; unlabeled data is generated using simulator as described in Section 3
4: **end for**
5: **return** $\hat{\pi}^T$

with respect to the i.i.d. distribution d_{π^*} is bounded by ϵ' then $V(\hat{\pi}) \geq V(\pi^*) - \epsilon' T^2$. Since generating i.i.d. samples from d_{π^*} can require up to T queries (see Section 3) the passive label complexity of this approach for guaranteeing a regret less than ϵ with probability at least $1-\delta$ is $T \cdot N_p(\frac{\epsilon}{T^2}, \delta)$. Again, to our knowledge, this is the best known label complexity for passive imitation learning. Further, Ross and Bagnell (Ross and Bagnell, 2010) show that there are imitation learning problems where this bound is tight, showing that in the worst case, the traditional approach cannot be shown to do better.

The above approach cannot be converted into an active imitation learner by simply replacing the call to L_p with L_a, since again we cannot sample from the unlabeled distribution d_{π^*} without querying the expert. To address this issue, we introduce a new algorithm called *RAIL (Reduction-based Active Imitation Learning)* which makes a sequence of T calls to an active i.i.d. learner, noting that it is likely to find a useful stationary policy well before all T calls are issued. RAIL is an idealized algorithm intended for analysis, which achieves the theoretical goals but has a number of inefficiencies from a practical perspective. Later in Section 5, we describe the practical instantiation that is used in our experiments.

RAIL is similar in spirit to active forward training, though its analysis is quite different and more involved. Like forward-training, RAIL iterates for T iterations, but on each iteration, RAIL learns a new stationary policy $\hat{\pi}^t$ that can be applied across all time steps $t = 1 \ldots T$. Note that T denotes the length of the horizon as well as the total number of iterations that RAIL runs for. Similarly t denotes a single time step as well as a single iteration of RAIL. Iteration $t+1$ of RAIL learns a new policy $\hat{\pi}^{t+1}$ that achieves a low error rate at predicting the expert's actions with respect to the state distribution of the previous policy $d_{\hat{\pi}^t}$. More formally, Algorithm 2 gives pseudocode for RAIL. The initial policy $\hat{\pi}^0$ is arbitrary and could be based on prior knowledge and the algorithm returns the final policy $\hat{\pi}^T$, which is learned using the active learning applied to unlabeled state distribution $d_{\hat{\pi}^{T-1}}$.

Similar to active forward training, RAIL makes a sequence of T calls to an active learner. Unlike forward training, however, the unlabeled data distributions used at each iteration contains states from all time points within the horizon, rather than being restricted to states arising at a particular time point. Because of this difference, the active learner is able to ask queries across a range of time points and we might expect policies learned in earlier iterations to achieve non-trivial performance throughout the entire horizon. In contrast, at iteration t the policy produced by forward training is only well defined up to time t.

The complication faced by RAIL, however, compared to forward training, is that the distribution used to train $\hat{\pi}^{t+1}$ differs from the state distribution of the expert policy d_{π^*}. This is particularly true in early iterations of RAIL, since $\hat{\pi}^0$ is initialized arbitrarily. Intuitively, however, we might expect that as the iterations proceed, the unlabeled distributions used for training d_{π^t} will become similar to d_{π^*}. To see this, consider the first iteration. While $d_{\hat{\pi}^0}$ need not be at all similar to d_{π^*} overall, we know that they will agree on the initial state distribution. That is, we have that $d^1_{\hat{\pi}^0} = d^1_{\pi^*} = I$. Because of this, the policy $\hat{\pi}_1$ learned on $d_{\hat{\pi}^0}$ can be expected to agree with the expert on the first step. This implies that the states encountered after the first action of the expert and learned policy will tend to be similar. That is $d^2_{\hat{\pi}^1}$ will be similar to $d^2_{\pi^*}$. In this same fashion we might expect $d^{t+1}_{\hat{\pi}^t}$ to be similar to $d^{t+1}_{\pi^*}$ after iteration t. We now show that this intuition can be formalized in order to bound the disparity between $d_{\hat{\pi}^T}$ and d_{π^*}, which will allow us to bound the regret of the learned policy. We first state the main result, which we prove below.

Theorem 2 *Given a PAC active i.i.d. learning algorithm L_a, if RAIL is run with parameters ϵ and $\frac{\delta}{T}$ passed to L_a at each iteration, then with probability at least $1-\delta$ it will return a stationary policy $\hat{\pi}$ such that $V(\hat{\pi}) \geq V(\pi^*) - \epsilon T^3$.*

Recall that the corresponding regret for active forward training of non-stationary policies was ϵT^2. From this we see that the impact of moving from non-stationary to stationary policies in the worst case is a factor of T in the regret bound. Similarly the bound is a factor of T worse than the comparable result above for passive imitation learning, which suffered a worst-case regret of ϵT^2. From this we see that the total label complexity for RAIL required to guarantee a regret of ϵ with probability $1-\delta$ is $\sum_{t=1}^{T} N_a(\frac{\epsilon}{T^3}, \frac{\delta}{T}, d_{\hat{\pi}^{t-1}})$ compared to the above label complexity of passive learning $T \cdot N_p(\frac{\epsilon}{T^2}, \delta)$.

We first compare these quantities in the worst case. If, in each iteration, the active i.i.d. label complexity is the same as the passive complexity, then active imitation learning via RAIL can ask more queries than passive. That is, the active complexity would scale as $T \cdot N_p(\frac{\epsilon}{T^3}, \frac{\delta}{T})$ versus $T \cdot N_p(\frac{\epsilon}{T^2}, \delta)$, which is dominated by the factor of $\frac{1}{T}$ difference in the accuracy parameters. In the realizable setting with finite VC-dimension, RAIL's complexity could be a factor of T higher than passive in this worst-case scenario.

However, if across the iterations the expected active i.i.d. label complexity $N_a(\frac{\epsilon}{T^3}, \frac{\delta}{T}, d_{\hat{\pi}^{t-1}})$ is substantially better than $N_p(\frac{\epsilon}{T^3}, \frac{\delta}{T})$, then RAIL will leverage those savings. For example, in the realizable setting with finite VC-dimension, if all distributions $d_{\hat{\pi}^{t-1}}$ result in a finite disagreement coefficient, then we can get exponential savings. In particular, ignoring the dependence on δ (which is only logarithmic), we get an active label complexity of $O(T \log \frac{T^3}{\epsilon})$ versus the corresponding passive complexity of $O(\frac{T^3}{\epsilon})$.

The above analysis points to an interesting open problem. Is there an active imitation learning algorithm that can guarantee to never perform worse than passive, while at the same time showing exponential improvement in the best case?

For the proof of Theorem 2, we introduce the quantity $P^t_\pi(M)$, which is the probability that a policy π is consistent with a length t trajectory generated by the expert policy π^* in MDP M. It will also be useful to index the state distribution of π by the MDP M, denoted by $d_\pi(M)$. The main idea is to show that at iteration t, $P^t_{\hat{\pi}^t}(M)$ is not too small, meaning that the policy at iteration t mostly agrees with the expert for the first t actions. We first

state two lemmas, that are useful for the final proof. First, we bound the regret of a policy in terms of $P_\pi^T(M)$.

Lemma 3 *For any policy π, if $P_\pi^T(M) \geq 1-\epsilon$, then $V(\pi) \geq V(\pi^*) - \epsilon T$.*

Proof Let Γ^* and Γ be all state-action sequences of length T that are consistent with π^* and π respectively. If $R(\mathcal{T})$ is the total reward for a sequence $\mathcal{T}$ then we get the following

$$\begin{aligned}
V(\pi) &= \sum_{\mathcal{T}\in\Gamma} \Pr(\mathcal{T} \mid M, \pi) R(\mathcal{T}) \\
&\geq \sum_{\mathcal{T}\in\Gamma\cap\Gamma^*} \Pr(\mathcal{T} \mid M, \pi) R(\mathcal{T}) \\
&= \sum_{\mathcal{T}\in\Gamma^*} \Pr(\mathcal{T} \mid M, \pi^*) R(\mathcal{T}) - \sum_{\mathcal{T}\in\Gamma^*-\Gamma} \Pr(\mathcal{T} \mid M, \pi^*) R(\mathcal{T}) \\
&= V(\pi^*) - \sum_{\mathcal{T}\in\Gamma^*-\Gamma} \Pr(\mathcal{T} \mid M, \pi^*) R(\mathcal{T}) \\
&\geq V(\pi^*) - T \cdot \sum_{\mathcal{T}\in\Gamma^*-\Gamma} \Pr(\mathcal{T} \mid M, \pi^*) \\
&\geq V(\pi^*) - \epsilon T.
\end{aligned}$$

The last two inequalities follow since the reward for a sequence must be no more than T, and due to our assumption about $P_\pi^T(M)$. ■

Next, we show how the value of $P_\pi^t(M)$ changes across one iteration of RAIL. We show that if we learn a policy $\hat{\pi}$ on state distribution $d_\pi(M)$ of policy π whose error rate $e_{\pi^*}(\hat{\pi}, d_\pi(M))$ (see Section 3.3) w.r.t. to the expert's policy π^* is no more than ϵ, then $P_{\hat{\pi}}^{t+1}(M)$ is at least as large as $P_\pi^t(M) - T\epsilon$. When π and $\hat{\pi}$ correspond to policies learned at iteration t and $(t+1)$ respectively, then Lemma 4 describes change in the value of $P_\pi^t(M)$ across one iteration.

Lemma 4 *For any policies π and $\hat{\pi}$ and $1 \leq t < T$, if $e_{\pi^*}(\hat{\pi}, d_\pi(M)) \leq \epsilon$, then $P_{\hat{\pi}}^{t+1}(M) \geq P_\pi^t(M) - T\epsilon$.*

Proof We define $\hat{\Gamma}$ to be all sequences of state-action pairs of length $t+1$ that are consistent with $\hat{\pi}$. Also define Γ to be all length $t+1$ state-action sequences that are consistent with π on the first t state-action pairs (so need not be consistent on the final pair). We also define M' to be an MDP that is identical to M, except that the transition distribution of any state-action pair (s, a) is equal to the transition distribution of action $\pi(s)$ in state s. That is, all actions taken in a state s behave like the action selected by π in s.

We start by arguing that if $e_{\pi^*}(\hat{\pi}, d_\pi(M)) \leq \epsilon$ then $P_{\hat{\pi}}^{t+1}(M') \geq 1 - T\epsilon$, which relates our error assumption to the MDP M'. To see this, note that for MDP M', all policies, including π^*, have state distribution given by d_π. Thus by the union bound $1 - P_{\hat{\pi}}^{t+1}(M') \leq \sum_{i=1}^{t+1} \epsilon_i$, where ϵ_i is the error of $\hat{\pi}$ at predicting π^* on distribution d_π^i. This sum is bounded by $T\epsilon$

since $e_{\pi^*}(\hat{\pi}, d_\pi(M)) = \frac{1}{T}\sum_{i=1}^{T}\epsilon_i$. Using this fact we can now derive the following

$$\begin{aligned}
P_{\hat{\pi}}^{t+1}(M) &= \sum_{\mathcal{T}\in\hat{\Gamma}} \Pr(\mathcal{T} \mid M, \pi^*) \\
&\geq \sum_{\mathcal{T}\in\Gamma\cap\hat{\Gamma}} \Pr(\mathcal{T} \mid M, \pi^*) \\
&= \sum_{\mathcal{T}\in\Gamma} \Pr(\mathcal{T} \mid M, \pi^*) - \sum_{\mathcal{T}\in\Gamma-\hat{\Gamma}} \Pr(\mathcal{T} \mid M, \pi^*) \\
&= P_\pi^t(M) - \sum_{\mathcal{T}\in\Gamma-\hat{\Gamma}} \Pr(\mathcal{T} \mid M, \pi^*) \\
&= P_\pi^t(M) - \sum_{\mathcal{T}\in\Gamma-\hat{\Gamma}} \Pr(\mathcal{T} \mid M', \pi^*) \\
&\geq P_\pi^t(M) - \sum_{\mathcal{T}\notin\hat{\Gamma}} \Pr(\mathcal{T} \mid M', \pi^*) \\
&\geq P_\pi^t(M) - (1 - P_{\hat{\pi}}^{t+1}(M')) \\
&\geq P_\pi^t(M) - T\epsilon.
\end{aligned}$$

The equality of the fourth line follows because Γ contains all sequences whose first t actions are consistent with π with all possible combinations of the remaining action and state transition. Thus, summing over all such sequences yields the probability that π^* agrees with the first t steps. The equality of the fifth line follows because $\Pr(\mathcal{T} \mid M, \pi^*) = \Pr(\mathcal{T} \mid M', \pi^*)$ for any $\mathcal{T}$ that is in Γ and for which π^* is consistent (has non-zero probability under π^*). The final line follows from the above observation that $P_{\hat{\pi}}^{t+1}(M') \geq 1 - T\epsilon$. ■

We can now complete the proof of the main theorem.

Proof [Proof of Theorem 2] Using failure parameter $\frac{\delta}{T}$ in the call to L_a in each iteration of RAIL ensures that with at least probability $(1-\delta)$ that for all $1 \leq t \leq T$, we will have $e_{\pi^*}(\hat{\pi}^t, d_{\hat{\pi}^{t-1}}(M)) \leq \epsilon$, where $d_{\hat{\pi}^0}(M)$ denotes the state distribution of the initial policy $\hat{\pi}^0$. This can be easily shown using the union bound. Next, we show using induction that for $1 \leq t \leq T$, we have $P_{\hat{\pi}^t}^t \geq 1 - tT\epsilon$. As a base case for iteration $t = 1$, we have $P_{\hat{\pi}^1}^1 \geq 1 - T\epsilon$, since the the error rate of $\hat{\pi}^1$ relative to the initial state distribution at time step $t = 1$ is at most $T\epsilon$ (this is the worst case when all errors are committed at time step 1). Assume that the inequality holds for $t = k$, i.e., $P_{\hat{\pi}^k}^k \geq 1 - kT\epsilon$. Consider $\hat{\pi}^{k+1}$ trained on $d_{\hat{\pi}^k}(M)$. By the union bound argument above, we know that $e_{\pi^*}(\hat{\pi}^{k+1}, d_{\hat{\pi}^k}(M)) \leq \epsilon$. Hence, $\hat{\pi}^{k+1}$ and $\hat{\pi}^k$ satisfy the precondition of Lemma 4. Therefore we have

$$\begin{aligned}
P_{\hat{\pi}^{k+1}}^{k+1} &\geq P_{\hat{\pi}^k}^k - T\epsilon && \text{(by Lemma 4)} \\
&\geq 1 - kT\epsilon - T\epsilon && \text{(by inductive argument)} \\
&\geq 1 - (k+1)T\epsilon.
\end{aligned}$$

Hence, for $1 \leq t \leq T$, we have $P_{\hat{\pi}^t}^t \geq 1 - tT\epsilon$. In particular, when $t = T$, we have $P_{\hat{\pi}^T}^T \geq 1 - T^2\epsilon$. Combining this with Lemma 3 completes the proof. ■

4.3 Agnostic Case

Above we considered the realizable setting, where the expert's policy was assumed to be in a known hypothesis class H. In the agnostic case, we do not make such an assumption. The learner still outputs a hypothesis from a class H, but the unknown policy is not necessarily in H. The agnostic i.i.d. PAC learning setting is defined similarly to the realizable setting, except that rather than achieving a specified error bound of ϵ with high probability, a learner must guarantee an error bound of $\inf_{\pi \in H} e_f(\pi, D_{\mathcal{X}}) + \epsilon$ with high probability (where f is the target), where $D_{\mathcal{X}}$ is the unknown data distribution. That is, the learner is able to achieve close to the best possible accuracy given class H. In the agnostic case, it has been shown that exponential improvement in label complexity with respect to $\frac{1}{\epsilon}$ is achievable when $\inf_{\pi \in H} e_f(\pi, D_{\mathcal{X}})$ is relatively small compared to ϵ (Dasgupta, 2011). Further, there are many empirical results for practical active learning algorithms that demonstrate improved label complexity compared to passive learning.

It is straightforward to extend our above results for non-stationary and stationary policies to the agnostic case by using agnostic PAC learners for L_p and L_a. Here we outline the extension for RAIL. Note that the RAIL algorithm will call L_a using a sequence of unlabeled data distributions, where each distribution is of the form d_π for some $\pi \in H$ and each of which may yield a different minimum error given H. For this purpose, we define $\epsilon^* = \sup_{\pi \in H} \inf_{\hat{\pi} \in H} e_{\pi^*}(\hat{\pi}, d_\pi)$ to be the minimum generalization error achievable in the worst case considering all possible state distributions d_π that RAIL might possibly encounter. With minimal changes to the proof of Theorem 1, we can get an identical result, except that the regret is $(\epsilon^* + \epsilon)T^3$ rather than just ϵT^3. A similar change in regret holds for passive imitation learning. This shows that in the agnostic setting we can get significant improvements in label complexity via active imitation learning when there are significant savings in the i.i.d. case.

5. RAIL-DA: A Practical Variant of RAIL

Despite the theoretical guarantees, there are at least two potential drawbacks of the RAIL algorithm from a practical perspective. First, RAIL does not share labeled data across iterations which is potentially wasteful in practice, though important for our analysis. In practice, we might expect that aggregating labeled data across iterations would be beneficial due to the larger amount of data. This is somewhat confirmed by the empirical success of the DAGGER algorithm (Ross et al., 2011) and motivates evaluating the use of data aggregation within RAIL. Incorporating data aggregation into RAIL, however, complicates the theoretical analysis, which we leave for future work. The second practical inefficiency of RAIL is that the unlabeled state distributions used at early iterations may be quite different from d_{π^*}. In particular, the state distribution of policy $\hat{\pi}^t$, which is used to train the policy at iteration $t+1$, is only guaranteed to be close to the expert's state distribution for the first t time steps and can (in the worst case) differ arbitrarily at later times. Thus, early iterations may focus substantial query effort on parts of the state space that are not the most relevant for learning π^*.

Algorithm 3 RAIL$^+$

Input: L_0, n, *AccumData*, N, K
▷ L_0 : initial set of labeled data
▷ n : no. of queries per iteration
▷ *AccumData* : accumulate data across iterations or not
▷ N : committee size for DWQBC
▷ K : # trajectories used to generate unlabeled data
Output: stationary policy $\hat{\pi}$

1: Initialize $L = L_0$
2: **while** query budget remaining **do**
3: $U =$ `SampleUnlabeledData`(K,L) ▷ generates pool of unlabeled data
4: **if** !*AccumData* **then**
5: Initialize $L = L_0$
6: **end if**
7: **for** $i = 1$ to n **do** ▷ select n queries from pool U
8: $s =$ `DWQBC`(L,U,N) ▷ density-weighted QBC is used as active i.i.d. learner
9: $L = L \cup \{(s, \texttt{Label}(\text{s}))\}$ ▷ obtain label from expert
10: **end for**
11: **end while**
12: **return** $\hat{\pi} =$ `SupervisedLearn`(L)

We now describe a parameterized practical instantiation of RAIL used in our experiments, which is intended to address the above issues and also specify certain other implementation details. Algorithm 3 gives pseudocode for this algorithm, which we call RAIL$^+$ to distinguish it from the idealized version of RAIL in our analysis. In Section 6, we will compare different instances of RAIL$^+$, including parameterizations corresponding to pure RAIL and RAIL-DA (for data aggregation), which is the primary algorithm in our empirical study. We note that our description assumes the use of a pool-based active learner, which is a common active learning setting, meaning that the learner requires as input a pool of unlabeled examples U that represents the unlabeled target distribution.

5.1 Data Aggregation and Incremental Querying

The first major difference compared to RAIL is that RAIL$^+$ can aggregate data across iterations when the Boolean parameter *AccumData* is set to true. In this case, during each iteration the newly labeled data is added to the set of labeled data from previous iterations (lines 7-10). Otherwise, the labeled data from previous iterations is discarded after generating unlabeled data U (lines 4-6).

The second major difference is that RAIL$^+$ may ask fewer queries per iteration than RAIL as specified by the parameter n. RAIL corresponds to a version of RAIL$^+$ that does not use data aggregation and has $n = N_a$. Because RAIL$^+$ can aggregates data, it opens up the possibility of asking only a small number of queries per iteration ($n \ll N_a$) and hence behaving more incrementally. This is reflected in the main loop of Algorithm 3

Algorithm 4 Density-Weighted Query-By-Committee Algorithm

1: **procedure** `DWQBC`(L,U,N)
2: C = `SampleCommittee`(N,L) ▷ committee represents posterior over policies
3: $\hat{d}_L$ =`EstimateDensity`(U) ▷ estimate density of states in U (see text)
4: $s^* = \text{argmax}\{VE(s, C) * \hat{d}_L(s) : s \in U\}$ ▷ selection heuristic (see text)
5: **return** s^*
6: **end procedure**

(lines 2-11). Each iteration starts with the current set of labeled examples L, which have been accumulated across all previous iterations by RAIL$^+$. This set of examples is used to generate a pool U of unlabeled examples/states (see details below) intended to represent the unlabeled target distribution. A pool-based active learner, DWQBC (explained later), is then called n times to select n queries from this pool. Each query is labeled by the expert and added to the growing set of labeled training data L. After the n queries have been issued and L is updated, the next iteration begins.

This incremental version of RAIL allows for rapid updating of the unlabeled state distributions used for learning (represented via U) and prevents RAIL from using its query budget on earlier less accurate distributions. In our experiments, we find that using $n = 1$ is most effective compared to larger values, which facilitates the most rapid update of the distribution. In this case, each query selected by the active i.i.d. learner is based on the most up-to-date state distribution, which takes all prior data into account. We refer to this best performing variant of RAIL$^+$ with $n = 1$ and data aggregation as RAIL-DA.

An interesting variation to RAIL$^+$ is when queries take a non-trivial amount of real-time to answer and there are k experts available that can answer queries in parallel. In this case, it can be beneficial to ask k simultaneous queries per iteration in order to reduce the amount of real-time required to learn a policy. The problem of selecting k such queries is known as *batch active learning*, and a variety of approaches are available for the i.i.d. setting (Brinker, 2003; Xu et al., 2007; Hoi et al., 2006a,b; Guo and Schuurmans, 2008; Azimi et al., 2012). An advantage of our reduction-based approach to active imitation learning is that we can directly plug in the i.i.d. batch active learner in our framework without requiring any other changes to be made.

5.2 Density Weighted QBC

Since it is important that the active i.i.d. learner be sensitive to the unlabeled data distribution, we choose a density-weighted learning algorithm. In particular, we use density-weighted query-by-committee (McCallum and Nigam, 1998) in our implementation. Given a set of labeled data L and unlabeled data U, this approach first uses bootstrap aggregation on L in order to generate a policy committee for query selection (Algorithm 4, line 2). In our experiments we use a committee of size 5. The approach then computes a density estimator over the unlabeled data U (line 3). In our implementation we use a simple distance based binning approach to density estimation, though more complex approaches could be used. The selected query is the state that maximizes the product of state density and committee disagreement (line 4). As a measure of disagreement we use the entropy of the vote distribution (Dagan and Engelson, 1995) (denoted as VE), which is a common choice.

Algorithm 5 Procedure SampleUnlabeledData

```
1: procedure SampleUnlabeledData(K,L)
2:     C = SampleCommittee(K,L)      ▷ committee represents posterior over policies
3:     U = {}      ▷ initialize multi-set of unlabeled data
4:     for π ∈ C do
5:         S = SimulateTrajectory(π)      ▷ states generated on trajectory of π
6:         U = U ∪ S
7:     end for
8:     return U
9: end procedure
```

Intuitively, the selection heuristic of DWQBC attempts to trade off the uncertainty about what to do at a state (measured by VE) with the likelihood that the state is relevant to learning the target policy (measured by the density).

5.3 Bayesian Learner

The final choice we make while implementing RAIL$^+$ (and hence RAIL-DA) is again motivated by the goal of arriving at an accurate unlabeled data distribution as quickly as possible. Recall that at iteration $t+1$, RAIL learns using an unlabeled data distribution $d_{\hat{\pi}^t}$, where $\hat{\pi}^t$ is a point estimate of the policy based on the labeled data from iteration t. In order to help improve the accuracy of this unlabeled distribution (with respect to d_{π^*}), instead of using a point estimate, we adopt a Bayesian approach in RAIL$^+$. In particular, at iteration t let L be the set of state-action pairs collected from the previous iteration (or from all previous iterations if data is accumulated). We use this to define a posterior $P(\hat{\pi}|L)$ over policies in our policy class H. This distribution, in turn, defines a posterior unlabeled state distribution $d_L = \mathbb{E}_{\hat{\pi}\sim P(\hat{\pi}|\mathcal{L})}[d_{\hat{\pi}}(s)]$ that RAIL$^+$ effectively uses in place of $d_{\hat{\pi}^t}$ as used in RAIL. Note that we can sample states from d_L by first sampling a policy $\hat{\pi}$ and then sampling a state from $d_{\hat{\pi}}$, all of which can be done without interaction with the expert.

Our implementation of this idea uses bootstrap aggregation (Breiman, 1996) in order to approximate d_L by an unlabeled data pool U via a call to the procedure SampleUnlabeledData in Algorithm 3 (line 3). Our implementation assumes a class of linear parametric policies with a zero-mean Gaussian prior over the parameters. The procedure first uses bootstrap aggregation to approximate sampling a set of policies from the posterior (Algorithm 5, line 2) forming a "committee" C of K policies. We view C as an empirical distribution representing the posterior over policies. In particular, each policy is the result of first generating a bootstrap sample of the current labeled data and then calling a supervised learner on the sampled data. Each member of the committee is then simulated to form a state trajectory, and the states on those trajectories are aggregated to produce the unlabeled data pool U (Algorithm 5, lines 4-7). Our implementation uses $K = 5$.

From a theoretical perspective, the use of a Bayesian classifier does not impact the validity of RAIL's performance guarantee provided that the Bayesian approach provides PAC guarantees. In fact, early theoretical work in active learning (Freund et al., 1997) used exactly this type of assumption in their analysis of the query-by-committee algorithm.

Algorithm 6 Procedure SampleCommittee

```
1: procedure SampleCommittee(K,L)
2:     C = {}        ▷ initialize the committee
3:     for i = 1 to K do
4:         L' = BootstrapSample(L)        ▷ create a bootstrap sample of L
5:         π' = SupervisedLearn(L')       ▷ learn a classifier(policy) using L'
6:         C = C ∪ π'
7:     end for
8:     return C
9: end procedure
```

In practice, d_L is a significantly more useful estimate of d_{π^*} than the point estimate with respect to learning a policy. This is because it places more weight on states that are more frequently visited by policies drawn from the posterior rather than just a single policy. As an extreme example of the advantage of using d_L in practice, consider active imitation learning in an MDP with deterministic dynamics and a single start state. At iteration t, RAIL will use the state distribution of the point estimate $\hat{\pi}^{t-1}$, which for our assumed MDP will be uniform over the deterministic state sequence generated by $\hat{\pi}^{t-1}$. Thus, active learning will place equal emphasis on learning among that set of states. This is despite the fact that, in early iterations, we should expect that states appearing later in the trajectory are less likely to be relevant to learning π^*. This is because inaccuracies in $\hat{\pi}^{t-1}$ lead to error propagation as the trajectory unfolds. In contrast, d_L will weigh states according to the trajectories produced by all policies, weighted by the posterior. The practical effect is a non-uniform distribution over states in those trajectories, roughly weighted by how many policies visit the states. Thus, states at the tail end of trajectories in early iterations will generally carry very little weight, since they are only visited by one or a small number of policies.

6. Experiments

We conduct our empirical evaluation on five domains: 1) Cart-pole, 2) Bicycle, 3) Wargus, 4) Driving, and 5) NETtalk. Below we first describe the details of these domains. We then present various experiments that we performed in these domains. In the first experiment, we study the impact of data aggregation and query size on the performance of RAIL$^+$ from Algorithm 3. For this we compare several parameter settings, varying from versions that are close to pure RAIL to RAIL-DA ($n = 1$ with data aggregation). We show that versions closer to RAIL-DA that aggregate data and ask queries incrementally are more effective in practice. In particular, we show that RAIL-DA (which aggregates data and asks only one query per iteration) is the most effective parameterization. Next, we evaluate the impact of another implementation choice discussed in Section 5, the choice of base active learner in RAIL$^+$. We show that a density-weighted base active learner leads to better performance in practice than using other base active learners that ignore the data distribution. Once we have shown that RAIL-DA with a density-weighted base active learner is the best, we

compare it with a number of baseline approaches to active imitation learning in our last set of experiments. Finally, we provide some overall observations.

For all the learners in the experiments that are presented in this section, we employed the SimpleLogistic classifier from Weka (Hall et al., 2009) to learn policies over the set of features that were provided for each domain.

6.1 Domain Details

In this subsection, we give the details of all the domains used in our experiments.

6.1.1 Cart-Pole

Cart-pole is a well-known RL benchmark. In this domain, there is a cart on which rests a vertical pole. The objective is to keep the attached pole balanced by applying left or right force to the cart. An episode ends when either the pole falls or the cart goes out of bounds. There are two actions, left and right, and four state variables $(x, \dot{x}, \theta, \dot{\theta})$ describing the position and velocity of the cart and the angle and angular velocity of the pole. We made slight modifications to the usual setting where we allow the pole to fall down and become horizontal and the cart to go out of bounds (we used [-2.4, 2.4] as the in bounds region). We let each episode run for a fixed length of 5000 time steps. This opens up the possibility of generating several "undesirable" states where either the pole has fallen or the cart is out of bounds that are rarely or never generated by the expert's state distribution.

For all the experiments in cart-pole, the learner's policy is represented via a linear logistic regression classifier using features of state-action pairs where features correspond to state variables. The expert policy was a hand-coded policy that can balance the pole indefinitely. For each learner, we ran experiments from 150 random initial states close to the equilibrium start state $((x, \dot{x}, \theta, \dot{\theta}) = (0.0, 0.0, 0.0, 0.0))$. For each start state a policy is learned and a learning curve is generated measuring the performance as function of number of queries posed to the expert. To measure performance, we use a reward function (unknown to the learner) that gives $+1$ reward for each time step where the pole is kept balanced and the cart is within bounds and -1 otherwise. The final learning curve is the average of the individual curves.

6.1.2 Bicycle Balancing

This domain is a variant of the RL benchmark of bicycle balancing and riding (Randløv and Alstrøm, 1998). The goal is to balance a bicycle moving at a constant speed for 1000 time steps. If the bicycle falls, it remains fallen for the rest of the episode. Similar to the cart-pole domain, in bicycle balancing there is a huge possibility of spending significant amount of time in "undesirable" states where the bicycle has fallen down. The state space is described using nine variables $(\omega, \dot{\omega}, \theta, \dot{\theta}, \psi, x_f, y_f, x_b, y_b)$, where ω and $\dot{\omega}$ are the vertical angle and angular velocity of the bicycle, θ and $\dot{\theta}$ are the angle and angular velocity of the handlebar, ψ is the angle of the bicycle to the goal, x_f and y_f are x and y coordinates of the front tire and x_b and y_b are x and y coordinates of the rear tire of the bicycle. There are five possible actions $A = \{(\tau = 0, v = -0.02), (\tau = 0, v = 0), (\tau = 0, v = 0.02), (\tau = 2, v = 0), (\tau = -2, v = 0)\}$, where the first component is the torque applied to the handlebar and the second is the displacement of the rider.

As in the cart-pole domain, for all experiments in bicycle balancing, the learner's policy is represented as a linear logistic regression classifier over features of state-action pairs. A feature vector for a state-action pair is defined as follows: Given a state s, a vector consisting of following 20 basis functions is computed:

$$(1,\ \omega,\ \dot{\omega},\ \omega^2,\ \dot{\omega}^2,\ \omega\dot{\omega},\ \theta,\ \dot{\theta},\ \theta^2,\ \dot{\theta}^2,\ \theta\dot{\theta},\ \omega\theta,\ \omega\theta^2,\ \omega^2\theta,\ \psi,\ \psi^2,\ \psi\theta,\ \bar{\psi},\ \bar{\psi}^2,\ \bar{\psi}\theta)^T,$$

where $\bar{\psi} = \pi - \psi$ if $\psi > 0$ and $\bar{\psi} = -\pi - \psi$ if $\psi < 0$. This vector of basis functions is repeated for each of the 5 actions giving a feature vector of length 100. The expert policy was hand-coded and can balance the bicycle for up to 26K time steps. We used a similar evaluation procedure as for cart-pole where we generated 150 random start states and for each start state, a policy was learned using each of the learning algorithms and a learning curve was generated measuring total reward as function of number of queries posed to the expert. We give a $+1$ reward for each time step where the bicycle is kept balanced and a -1 reward otherwise.

6.1.3 Wargus

We consider controlling a group of 5 friendly close-range military units against a group of 5 enemy units in the real-time strategy game Wargus, similar to the setup used by Judah et al. (Judah et al., 2010). The objective is to win the battle while minimizing the loss in total health of friendly units. The set of actions available to each friendly unit is to attack any one of the remaining units present in the battle (including other friendly units, which is always a bad choice). In our setup, we allow the learner to control one of the units throughout the battle, whereas the other friendly units are controlled by a fixed "reasonably good" policy. This situation would arise when training the group via coordinate ascent on the performance of individual units. The expert policy corresponds to the same policy used by the other units. Note that poor behavior from even a single unit generally results in a huge loss.

The learner's policy is represented using 27 state-action features that capture different information about the current battle situation such as the distance between the friendly agent and the target of attack, whether the target is already under attack by other friendly units, health of the target relative to friendly unit, whether the target is actually a friendly unit, etc. Providing full demonstrations in real time in such tactical battles is very difficult for human players and quite time consuming if demonstrations are done in slow motion, which motivates state-based active learning for this domain. For experiments, we designed 21 battle maps differing in the initial unit positions, using 5 for training and 16 for testing. We report results in the form of learning curves showing the performance metric as a function of number of queries posed to the expert. We use the difference in the total health of friendly and enemy units at the end of the battle as the performance metric (which is positive for a win). Due to the slow pace of the experiments running on the Wargus infrastructure, we average results across at most 20 trials.

6.1.4 Driving Domain

The driving domain is a traffic navigation problem often used as a test domain in the imitation learning literature (Abbeel and Ng, 2004). This domain was also the main test

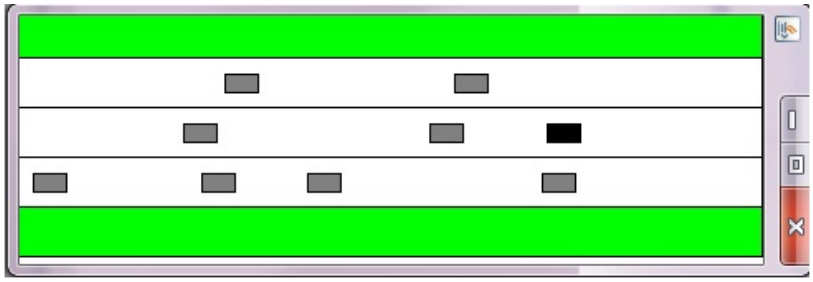


Figure 1: Screenshot of the driving simulator.

bed used to evaluate the confidence based autonomy (CBA) learner in prior work (Chernova and Veloso, 2009). Here we evaluate RAIL-DA on a particular implementation of the driving domain used by Cohn et al. (Cohn et al., 2011). In this domain, the goal is to successfully navigate a car through traffic on a busy five lane highway. The highway consists of three traffic lanes and two shoulder lanes (see Figure 1). The learner controls the black car, which moves at a constant speed. The other cars move at a randomly chosen continuous-valued constant speed, and they don't change lanes. At each discrete time step, the learner controls the car by taking one of the three actions: 1) *Left*, which moves the car to the adjacent left lane, 2) *Right*, which moves the car to the adjacent right lane, and 3) *Stay*, which keeps the car in the current lane. The agent is allowed to drive on the shoulder lane but cannot move off the shoulder lanes.

The learner's policy is represented as a linear logistic regression classifier that maps a given state to one of the three actions. The state space is represented using 68 features. The first 5 features are binary features that specify the learner's current lane. The next 3 binary features specify whether the learner is colliding with, tailgating or trailing another car. The remaining 60 features, consisting of three parts, one for the learner's current lane and two for the two adjacent lanes, which are binary features that specify whether the learner's car will collide with or pass another car in $2X$ time steps, where X ranges from 0 to 19. This captures the agent's view of the traffic while taking car velocities into account.

To conduct experiments in the driving domain, we carefully designed a reward function that induces good driving behavior and used Sarsa(λ) (Sutton and Barto, 1998) with linear function approximation to learn a policy to serve as the expert policy. The expert policy uses the same set of 68 features as used by the agent for value function approximation. For each learner, we ran 100 different learning trials where during each trial the learner is allowed to pose a maximum of 500 (1000 in Experiment 1) queries to the expert and learn from it. For each trial, a learning curve is generated that measures the performance as a function of the number of queries posed to the expert. To measure performance, after each query is posed and the policy is updated, the updated policy is allowed to navigate the car for 1000 time steps in a test episode, and the total reward is recorded. The final performance measure is the average total reward per episode measured over 500 test episodes. The final learning curve is the average over all 100 trials.

6.1.5 Structured Prediction

We evaluate RAIL-DA on two structured prediction tasks, stress prediction and phoneme prediction, both based on the NETtalk data set (Dietterich et al., 2008). In stress prediction, given a word, the goal is to assign one of the 5 stress labels to each letter of the word in left-to-right order so that the word is pronounced correctly. The output labels are '2' (strong stress), '1' (medium stress), '0' (light stress), '<' (unstressed consonant, center of syllable to the left), and '>' (unstressed consonant, center of syllable to the right). In phoneme prediction, the task is to assign one of the 51 phoneme labels to each letter of the word. It is straightforward to view structured prediction as imitation learning (see for example Ross et al., 2011,Daumé et al., 2009) where at each time step (letter location), the learner has to execute the correct action (i.e., predict correct label) given the current state. The state consists of features describing the input (the current letter and its immediate neighbors) and the previous L predictions made by the learner (the prediction context). In our experiments, we use $L = 1, 2$.

The NETtalk data set consists of 2000 words divided into 1000 training words and 1000 test words. Each method is allowed to select a state located on any of the words in the training data and pose it as a query. The expert reveals the correct label at that location. We use character accuracy as a measure of performance. The details of how in each learning trial RAIL-DA and other baselines select a query from the set of training words will be described when we present our results in the following subsections. We report final performance in the form of learning curves averaged across 50 learning trials.

6.2 Experiment 1: Evaluation of the Effects of Data Aggregation and Query Size

We compare different versions of RAIL$^+$, by varying the parameters *AccumData* and n in Algorithm 3, in order to observe the impact of data aggregation and query size. We use the notation *RAIL$^+$-n-DA* for versions that aggregate data and ask n queries per iteration and use *RAIL$^+$-n* to denote variants that ask n queries per iteration without data aggregation. Note that RAIL$^+$-1-DA is the same as RAIL-DA and that RAIL$^+$-n with a large value of n corresponds to the original version of RAIL from our analysis. Note that all these variants use the DWQBC active i.i.d. learner as described in Section 5.

For this experiment, we focus on three domains: 1) Cart-Pole, 2) Bicycle Balancing, and 3) Driving Domain. In each domain, we evaluated RAIL$^+$-n-DA and RAIL$^+$-n for values of n starting at $n = 1$ in increments until some maximum value. The maximum value of n in each domain was selected to be a value where i.i.d. active learning from the true expert state distribution reliably converged to a near perfect policy. For each RAIL$^+$ variant and domain, we show the averaged learning curve as described earlier.

Figure 2 shows the results of the experiment. We first discuss results in the Bicycle domain due to more discernible trends in this domain. Figure 2(b) shows results of the experiment in the Bicycle domain. First, observe the impact of data aggregation by comparing each pair RAIL$^+$-n and RAIL$^+$-n-DA. Clearly, RAIL$^+$-n-DA is significantly better than RAIL$^+$-n for each value of n, indicating that it is generally better to aggregate data across iterations in this domain. This is likely explained by the fact that the use of aggregation allows for the algorithms to learn from more data at later iterations. This is

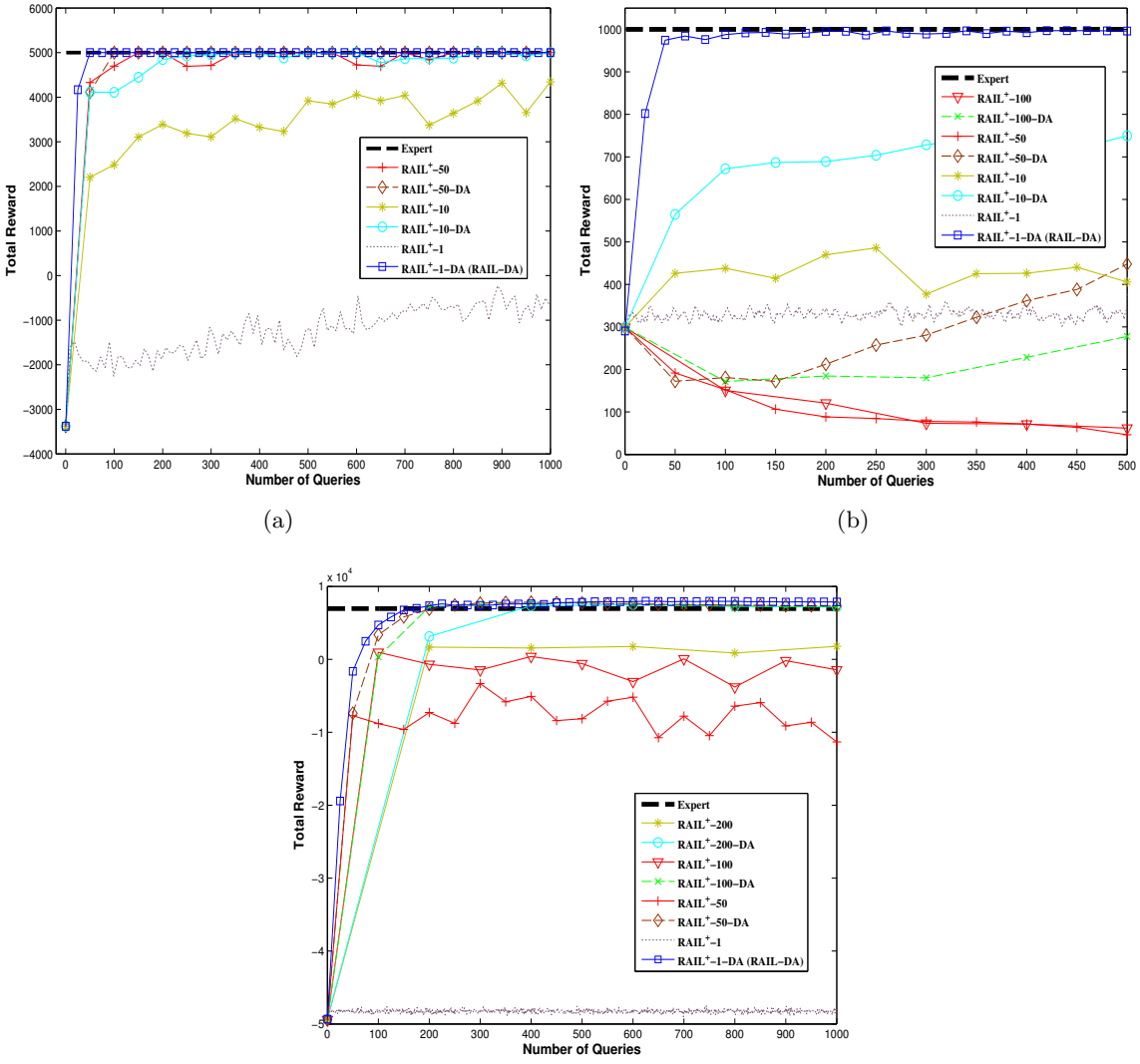


Figure 2: Evaluation of the effects of data aggregation and query size in RAIL$^+$: (a) Cart-pole (b) Bicycle balancing (c) Driving domain.

particularly important for small values of n, such as $n = 1$ and $n = 10$, where without aggregation each iteration is learning from only a small amount of data (the small amount of initial data + n).

Now consider the impact of varying n with the use of aggregation in Bicycle. The clear trend is that when aggregation is being used the performance improves significantly as n decreases from $n = 100$ to $n = 1$. That is, the more incremental variants of RAIL$^+$ dominate, with $n = 1$ (i.e. the RAIL-DA algorithm) dominating all others by a large margin. The main reason for this particularly striking trend is that in Bicycle the distributions being learned from in early iterations are quite distant from that of the expert. In particular, the early iterations involve learning from state distributions of policies that result in early crashes, and hence many useless states from the point of view of learning. This means that when using $n = 100$, the first 100 queries are selected from such a distribution and not much is learned. In fact, we see a downward trend, indicating that learning from those distributions hurts performance. On the other hand, as n decreases fewer queries are spent on those early inaccurate distributions, and the data from a small number of queries per iteration accumulates, resulting in policies that have better state distributions to learn from. Indeed, for $n = 1$, only one query is asked for each distribution, and we see very rapid improvement until reaching expert performance after just over 50 queries.

Figure 2(a) shows results for Cart-Pole. The target concept in this domain is simpler, and hence the learning curves improve more quickly compared to Bicycle. However, we see the same general trends as for Bicycle. Aggregation is beneficial and we get the best performance for small values of n when using aggregation. Again we see that RAIL$^+$-1-DA (i.e., RAIL-DA) is the top performer. These same trends are observed in Figure 2(c) for Driving.

Overall, these experiments provide strong evidence that in practice aggregation is an important enhancement to RAIL$^+$ over RAIL, and that more incremental variants are preferable. Thus, for the remainder of the paper we will use the RAIL-DA algorithm which uses aggregation and $n = 1$.

6.3 Experiment 2: Evaluation of RAIL-DA with Different Base Active Learners

In Section 5, we mentioned that it is important that the active i.i.d. learner used with RAIL be sensitive to the unlabeled data distribution. To test this hypothesis, we conducted experiments to study the effects of using active i.i.d. learners that take data distribution into consideration against active learners that ignore the data distribution altogether in RAIL-DA. We compare the performance of three different versions of RAIL-DA: 1) *RAIL-DA*, the RAIL-DA algorithm from the previous experiment that uses density-weighted QBC as the base active learner, 2) *RAIL-DA-QBC*, RAIL-DA but with density-weighted QBC replaced with the standard QBC (without density weighting), and 3) *RAIL-DA-RAND*, which uses random selection of unlabeled data points.

Figure 3 shows the performance of the three versions of RAIL-DA on our first four test domains. We see that, in Cart-Pole, Bicycle and Wargus, RAIL-DA performs better than both RAIL-DA-QBC and RAIL-DA-RAND. This shows that it is critical for the active i.i.d. learner to exploit the state density information that is estimated by RAIL-DA at

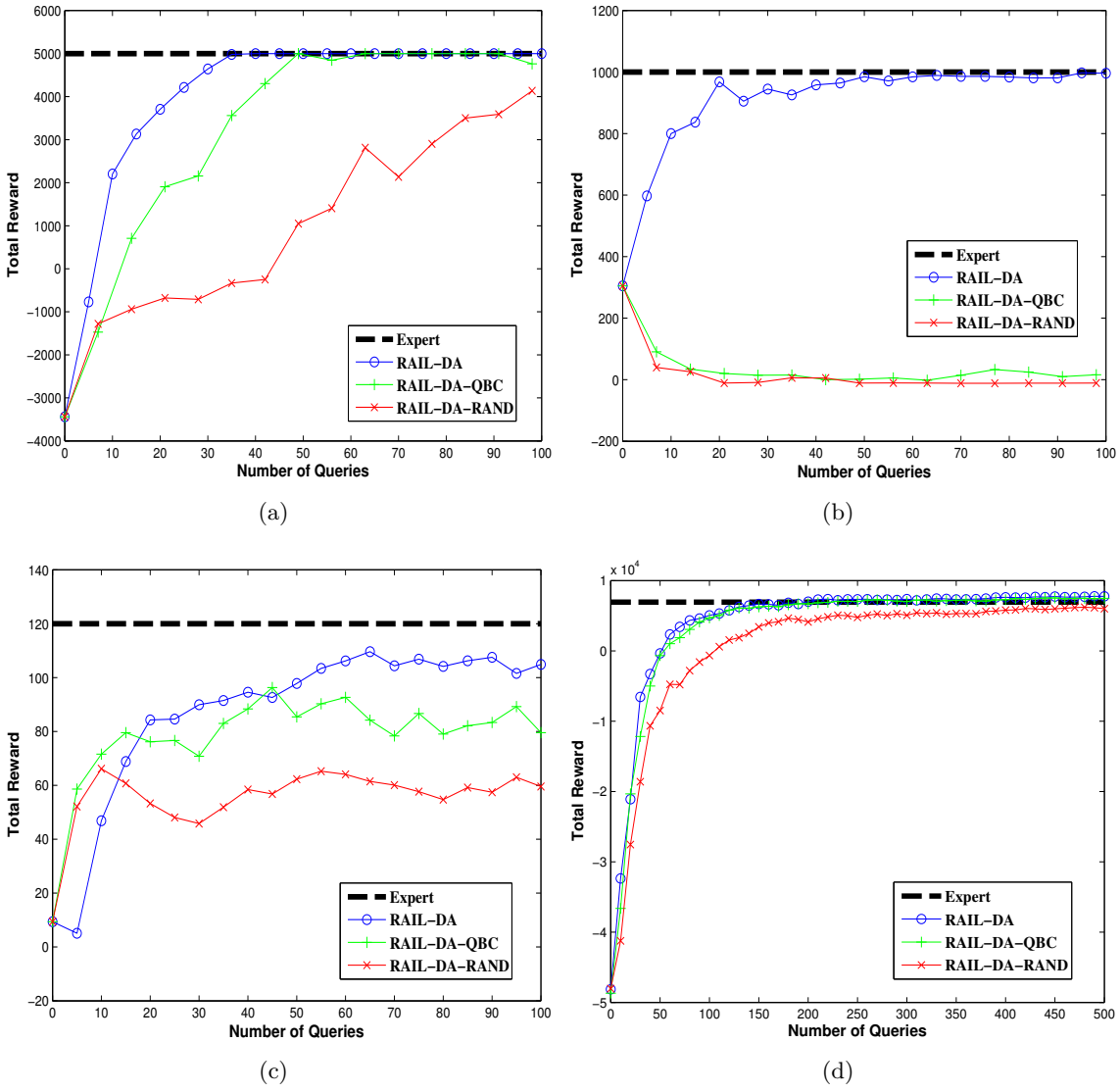


Figure 3: Performance of RAIL-DA with different base active learners on (a) Cart-pole (b) Bicycle balancing (c) Wargus (d) Driving Domain.

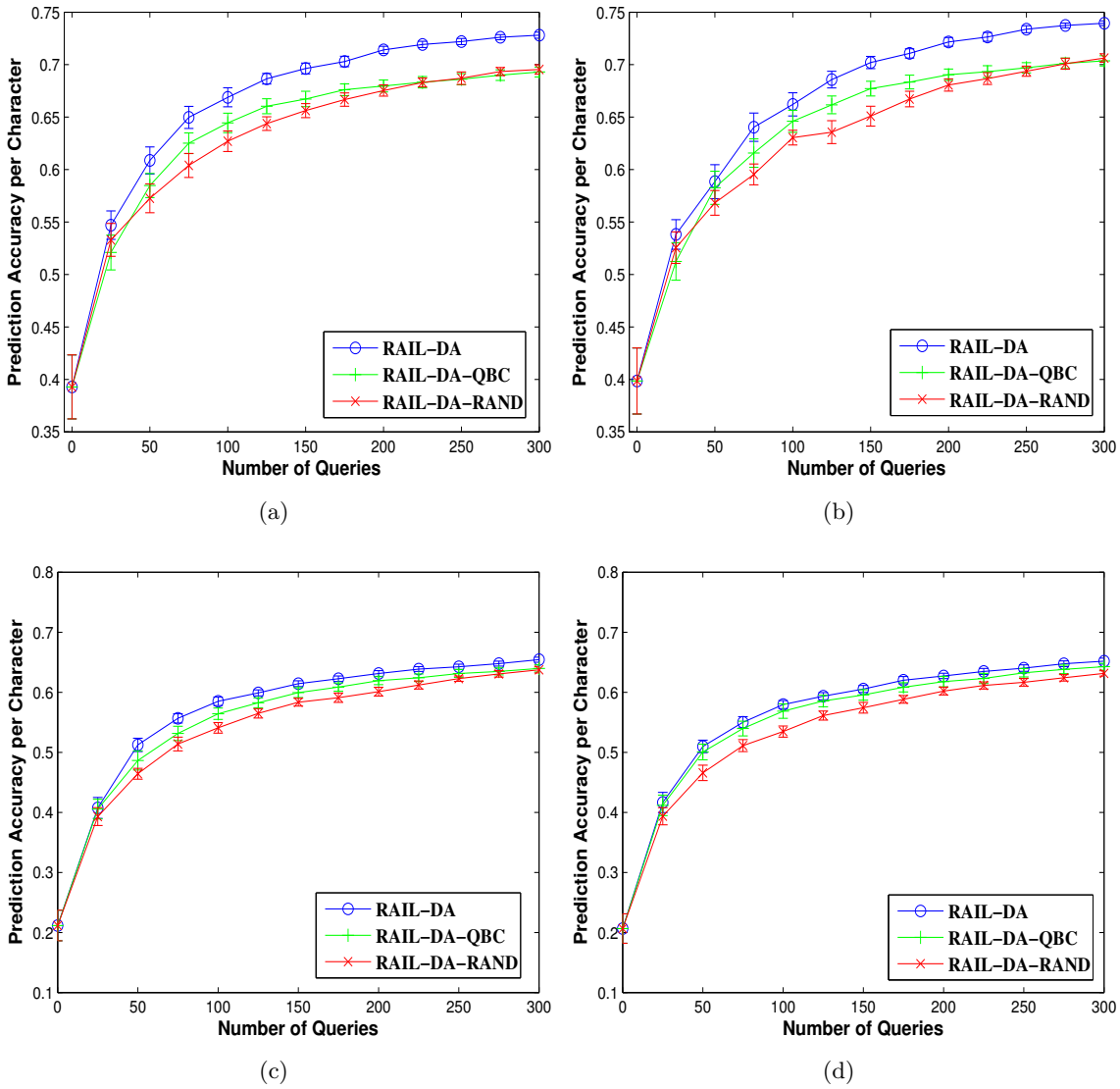


Figure 4: Performance of RAIL-DA with different base active learners on NETtalk: (a) Stress prediction, $L = 1$ (b) Stress prediction, $L = 2$ (c) Phoneme prediction, $L = 1$ (d) Phoneme prediction, $L = 2$. Character accuracy of the expert is 1.

each iteration. Recall that in these domains, especially during early iterations, RAIL-DA is quite likely to generate several uninformative states that are far from the expert's state distribution (states with fallen pole or bike, or states where the friendly team is heavily outnumbered by the enemy team in Wargus). Taking state density information into account helps to avoid querying in such states compared to a learner that ignores density information. In the driving domain, we see that although RAIL-DA performs better than RAIL-DA-RAND, its performance is comparable to that of RAIL-DA-QBC. This is because in the driving domain, even states that are not close to the expert's state distribution provide informative training information. This allows RAIL-DA-QBC to pose useful queries and generalize well from them.

Figure 4 shows the performance of the three versions of RAIL-DA on the NETtalk data set along with 95% confidence intervals. RAIL-DA and RAIL-DA-QBC can select the best query across the entire training set. RAIL-DA-RAND selects a random query from the set of unlabeled states generated on a random training sequence. For stress prediction, we see that RAIL-DA performs better than both RAIL-DA-QBC and RAIL-DA-RAND. For phoneme prediction, RAIL-DA performs better than RAIL-DA-RAND, but its performance is comparable to RAIL-DA-QBC. Overall, we see that RAIL-DA performs best, so it will be compared against the other baselines in the next section.

6.4 Experiment 3: Comparison of RAIL-DA with Baselines

We compare RAIL-DA against the following baselines:

1. *Passive.* This baseline simulates the traditional approach by starting at the initial state and querying the expert about what to do at each visited state.

2. *unif-QBC.* This baseline views all the MDP states as i.i.d. according to the uniform distribution and applies the standard query-by-committee (QBC) (Seung et al., 1992) active learning approach. Intuitively, this approach will select the state with highest action uncertainty according to the current data set and ignores the state distribution.

3. *unif-RAND.* This baseline selects states to query uniformly at random.

4. *Confidence based autonomy (CBA)* (Chernova and Veloso, 2009). This approach requires the use of policies that provide some form of confidence estimate (e.g., probabilities over actions). Given the current set of labeled data, the approach executes trajectories of the current policy until it reaches a state where the policy confidence falls below a threshold. It then queries the expert for the correct action and updates the policy accordingly. It then executes the correct action and continues until the next low confidence state is reach. It is possible for CBA to stop asking queries once the confidence exceeds the threshold in all states visited by the current policy. We use the same automated threshold adjustment strategy proposed by Chernova and Veloso (Chernova and Veloso, 2009). We also experimented with other threshold adjustment mechanisms as well as fixed thresholds, but were unable to find an improvement that performed better across our domains.

Figure 5 shows the results of this experiment along with 95% confidence intervals on our first four test domains. Figure 5(a) shows the performance of RAIL-DA on cart-pole.

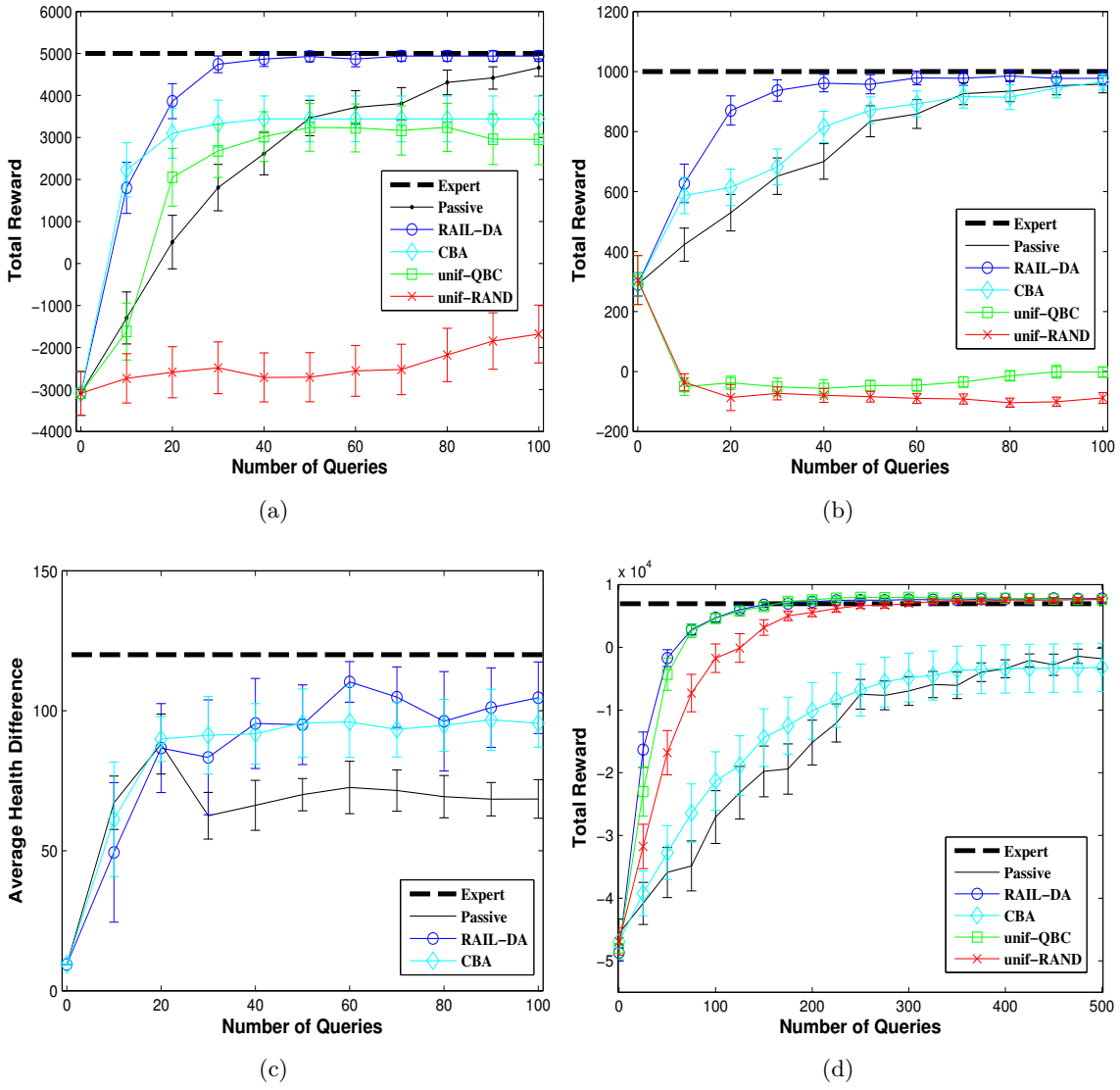
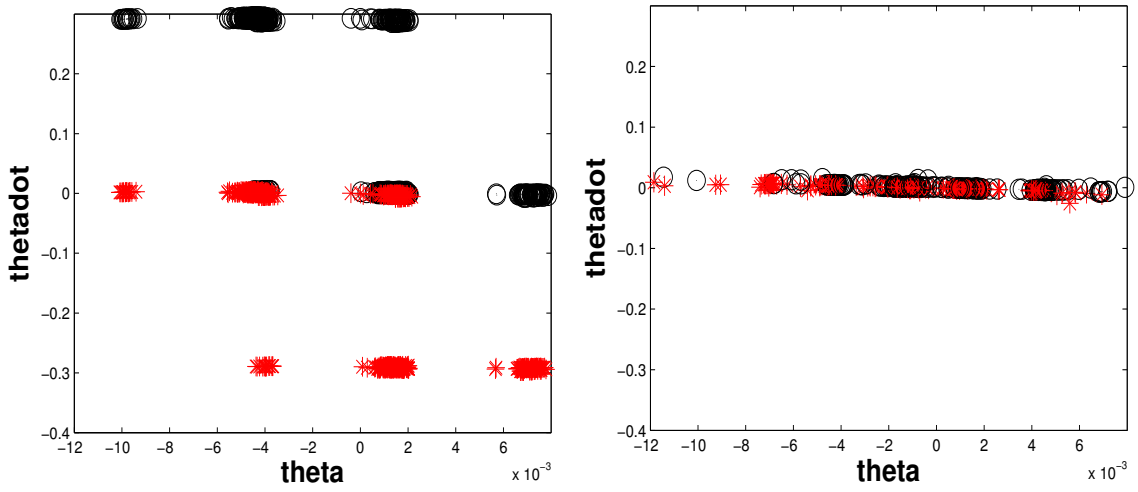


Figure 5: Active imitation learning results: (a) Cart-pole (b) Bicycle balancing (c) Wargus (d) Driving domain.



(a) (b)

Figure 6: Queried states in Cart-pole for learners: (a) Passive, (b) RAIL-DA. The black circles represent states where the right action is suggested by the expert policy. The red stars represent states where the left action is suggested. The decision boundary separating these two sets of states is a line very close to $\dot{\theta} = 0$. We see that unlike Passive, RAIL-DA focuses its queries around the decision boundary. This figure is best viewed in color.

We observe that RAIL-DA learns quickly and achieves optimal performance with only 30-35 queries. Passive, on the other hand, takes 100 queries to get close to the optimal performance. The reason for this difference is clear when one visualizes the states queried by RAIL-DA versus Passive. Figure 6(a) shows the states queried by Passive. The black circles represent states where the right action is suggested by the expert. The red stars represent states where the left action is optimal according to the expert. The decision boundary of the expert policy is the line separating these two sets of states (very close to $\dot{\theta} = 0$). We notice that Passive asks many uninformative queries that are not close to the decision boundary. Figure 6(b) shows the queries posed by RAIL-DA. We see that the queries posed by RAIL-DA tend to be close to the decision boundary of the expert policy.

A naive reduction to active learning can be dangerous, as demonstrated by the poor performance of unif-QBC. Further, RAIL-DA performs much better than random query selection as demonstrated by the performance of unif-RAND. By ignoring the real data distribution altogether and incorrectly assuming it to be uniform, these naive methods end up asking many queries that are not relevant to learning the expert policy (e.g., states where the pole is in an irrecoverable fall or the cart is out of bounds). CBA, like RAIL-DA, learns quickly but settles at a suboptimal level of performance. This is because it becomes confident prematurely and stops asking queries. This shows that CBA's automatic threshold adjustment mechanism did not work well in this domain. We did experiment with several modifications to the threshold adjustment strategy, but we were unable to find one that was robust across all our domains. Thus we report results for the original strategy.

Figure 5(b) compares each approach in the bicycle domain. The results are similar to those of cart-pole with RAIL-DA being the top performer. Unif-RAND and Unif-QBC show notably poor performance in this domain. This is because bicycle balancing is a harder learning problem than cart-pole with many more uninformative states (an unrecoverable fall or fallen state). We found that almost all queries posed by Unif-RAND and Unif-QBC were in these states. CBA does only slightly better than Passive, though unlike cart-pole, it achieves near-optimal asymptotic performance.

The results in the Wargus domain are shown in Figure 5(c). Passive learns along the expert's trajectory in each map on all 5 training maps considered sequentially according to a random ordering. For RAIL-DA, in each iteration a training map is selected randomly and a query is posed in the chosen map. For CBA, a map is selected randomly and CBA is allowed to play an episode in it, pausing and querying as and when needed. If the episode ends, another map is chosen randomly and CBA continues to learn in it. After each query, the learned policy is tested on the 16 test maps. We use the difference in the total health of friendly and enemy units at the end of the battle as the performance metric (which is positive for a win). We did not run experiments for unif-QBC and unif-RAND, because it is difficult to define the space of feasible states over which to sample uniformly.

We see that although Passive learns quickly for the first 20 queries, it fails to improve further. This shows that the states located in this initial prefix of the expert's trajectory are very useful, but thereafter Passive gets stuck on the uninformative part of the trajectory until its query budget is over. On the other hand, RAIL-DA and CBA continue to improve beyond Passive, with the performance of CBA being comparable to RAIL-DA in this domain, which indicates that both these active learners are able to locate and query more informative states.

The results for the driving domain are shown in Figure 5(d). We see qualitatively similar performance trends as in the previous three domains with RAIL-DA still being the best performing learner and outperforming most of the competing baselines. The exception however is that unif-QBC and unif-RAND perform quite well in this domain, both being able to outperform Passive and even CBA. Furthermore, performance of unif-QBC is quite comparable to that of RAIL-DA. The good performance of unif-QBC and unif-RAND in the driving domain is due to the fact that obtaining action labels on most states in the driving domain can serve as useful training data for learning the expert policy. This is unlike the previous three domains, where labels obtained for many states were relatively useless for learning the target policy (e.g., states with a fallen pole or bike).

Our final set of results are in the structured prediction domain. Passive learns along the expert's trajectory on each training sequence considered in the order it appears in the training set. Therefore, Passive always learns on the correct context, i.e., previous L characters correctly labeled. RAIL-DA and unif-QBC can select the best query across the entire training set. CBA, like Passive, considers training sequences in the order they appear in the training set and learns on each sequence by pausing and querying as and when needed. To minimize the effects of the ordering of the training sequences on the performance of Passive and CBA, for all learners, we ran 50 different trials where in each trial we randomize the order of the training sequences. We report final performance as the learning curves averaged across all 50 trials. The learning curves along with 95% confidence intervals are shown in figure 7.

Figures 7(a) and (b) presents the stress prediction results. The results are qualitatively similar to cart-pole, except that Unif-QBC and unif-RAND do quite well in this domain but not as well as RAIL-DA. We see that CBA performs quite poorly because we found that in several trials it prematurely stops asking queries,[5] which reveals its sensitivity to its threshold adjustment mechanism. Similar trends are seen in the phoneme prediction results shown in Figures 7(c) and (d). CBA does well on this task but not as well as RAIL-DA.

6.5 Overall Observations

We can draw a number of conclusions from the experiments. First, the implementation choices discussed in Section 5 are necessary to make RAIL more practical. In particular, RAIL-DA, which aggregates data and asks only one query per iteration, proved to be the most robust among all the variants of RAIL. Second, the choice of the active i.i.d. learner used for RAIL is important. In particular, performance can be poor when the active learning algorithm does not take density information into account. Using density weighted query-by-committee was effective in all of our domains. Third, RAIL-DA proved to be the most robust and effective active imitation learning algorithm among several baselines in all of our domains. It outperformed all other baselines in our experiments. Fourth, we found that CBA is quite sensitive to the threshold adjustment mechanism, and we were unable to find an alternative mechanism that works across our domains. In the original CBA paper by Chernova and Veloso (Chernova and Veloso, 2009), CBA was tested only on

5. To plot the learning curve for CBA, for each trial, we took the character accuracy after the last query and extrapolated it to 300 queries to obtain a curve. The final curve is the average of the extrapolated curves.

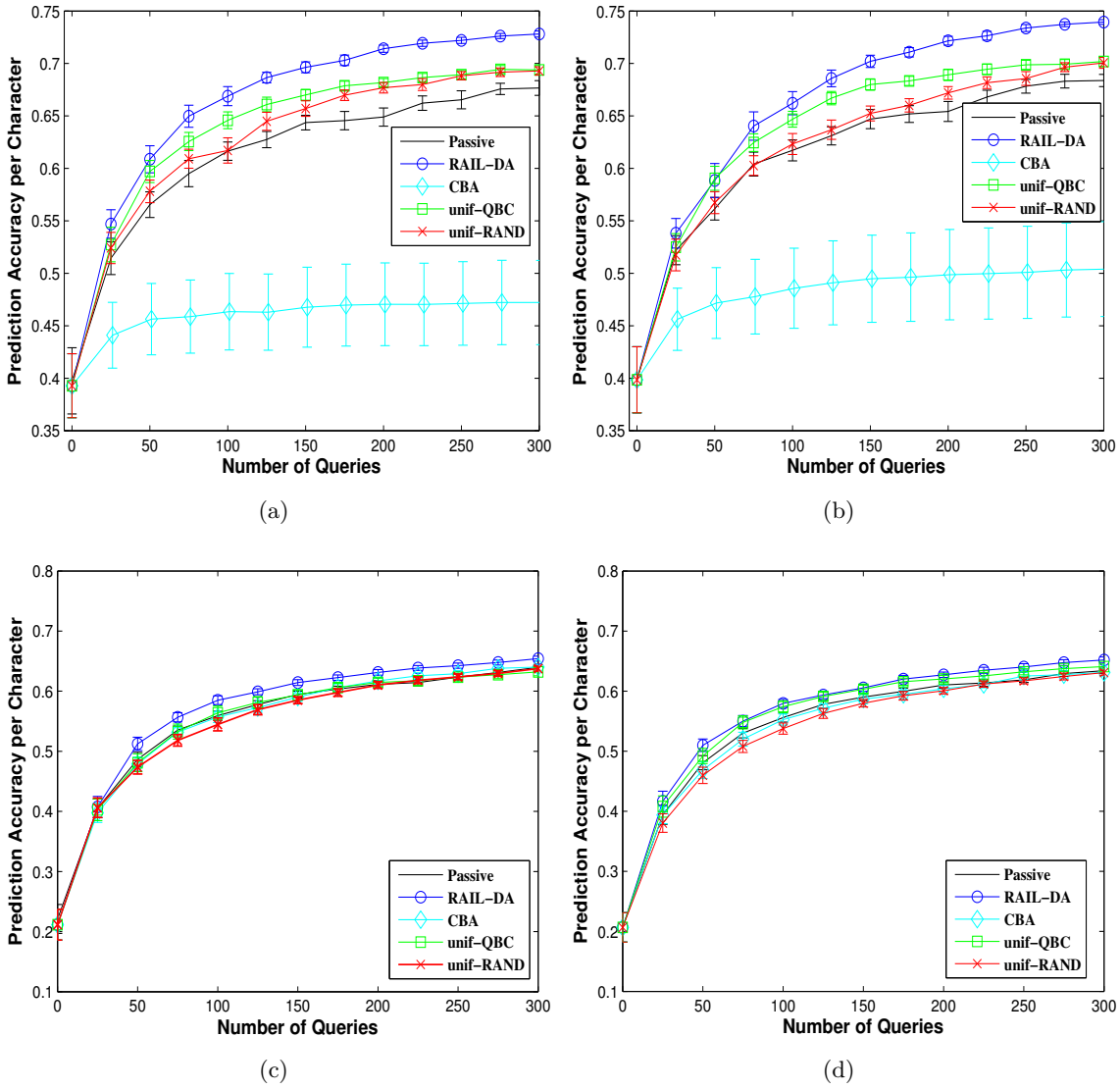


Figure 7: Active imitation learning results on NETtalk: (a) Stress prediction, $L = 1$ (b) Stress prediction, $L = 2$ (c) Phoneme prediction, $L = 1$ (d) Phoneme prediction, $L = 2$. Character accuracy of the expert is 1.

	Passive Imitation Learning	Active Imitation Learning
Non-stationary	$T \cdot N_p(\frac{\epsilon}{T^2}, \frac{\delta}{T})$	$\sum_{t=1}^{T} N_a(\frac{\epsilon}{T^2}, \frac{\delta}{T}, d^t_{\hat{\pi}^{t-1}})$
Stationary	$T \cdot N_p(\frac{\epsilon}{T^2}, \delta)$	$\sum_{t=1}^{T} N_a(\frac{\epsilon}{T^3}, \frac{\delta}{T}, d_{\hat{\pi}^{t-1}})$

Table 1: A comparison of the best known label complexities for PAC learning deterministic stationary and non-stationary policies in the passive and active settings. The label complexities are for learning policies with regret no more than ϵ with probability at least $1 - \delta$. $N_p(\epsilon', \delta')$ and $N_a(\epsilon', \delta', D)$ are label complexities of passive and active i.i.d. learning respectively with accuracy and reliability parameters ϵ' and δ' and data distribution D. When N_a is exponentially smaller than N_p on some or all of the distributions $d_{\hat{\pi}^{t-1}}, t = 1 \ldots T$, then these savings in label complexity in the i.i.d. setting translate to imitation learning resulting in improved label complexity of active imitation learning compared to passive.

the driving domain, and in their experiment CBA was initialized with a large amount of training data. We found that initializing CBA with a larger training set also resulted in improved performance of CBA in all of our domains. However, this conflicts with our goal of minimizing the amount of training data required from the expert, and hence we initialized all learners with the minimum training data required by the SimpleLogistic classifier. This turned out to be insufficient for CBA to function properly. Fifth, we showed that a more naive application of active i.i.d. learning in the imitation setting is not always effective.

7. Summary and Future Work

We considered reductions from active imitation learning to active i.i.d. learning, which allow for advances in the i.i.d. setting to translate to imitation learning. First, we analyzed the label complexity of reductions for both non-stationary and stationary policies, showing the number of queries required for active imitation learning in terms of the active sample complexity in the i.i.d. setting. These results for the realizable learning setting are summarized in Table 1. In the non-stationary case, the results show that active IL will not be worse than passive IL provided that the i.i.d. active learning algorithm is guaranteed to be no worse than passive. Further we can expect significant improvement in query complexity when the active i.i.d. algorithm is significantly better than the passive i.i.d. learner.

For the case of stationary policies, our current reduction RAIL only guarantees improvement or equivalence to passive IL when there is significant reduction in sample complexity of i.i.d. active learning over passive learning. While this is often the case in practice, it leaves an open theoretical problem. If we use an active i.i.d. learner that is guaranteed to do no worse than passive, then can we find a reduction such that active IL also has that guarantee?

Our second contribution was to introduced RAIL-DA, a practical variant of the reduction for stationary policies. RAIL-DA employees data aggregation and incremental learning in order to address several practical inefficiencies noted for the RAIL algorithm studied in the analysis. Our experiments showed that RAIL-DA significantly improved over RAIL and

other variants of RAIL$^+$. Further, we showed that in five domains RAIL-DA significantly outperformed a number of natural alternatives and the CBA algorithm from prior work.

The work presented in this paper is a first theoretical effort towards analyzing an active imitation learning approach and showing that it enjoys better label complexity than the traditional passive approach. In addition to the above open problem, an interesting line of followup work is to analyze RAIL-DA or other variants of RAIL that use data aggregation. Further, it is of interest to consider the online active learning setting, where the learner is embedded in a real environment, rather than having access to a simulator that can be reset. Such an algorithm might resemble the CBA algorithm, which would continually execute the current policy and only query the expert when it was uncertain. This is similar to the traditional QBC active learning algorithm in the i.i.d. setting. The only difference is that in the imitation learning setting the unlabeled data stream does not come from a fixed i.i.d. distribution, but rather from the policy being executed. It seems plausible that the theoretical results for QBC could be extended to the imitation learning setting.

We are also interested in extending the allowed query responses. For example, it is natural to allow the expert to declare a query as "bad" and refuse to provide an action label. This is useful in situations where the queries are posed at states that the expert would rarely or never encounter, and hence may have not natural preference about what to do. In our bicycle balancing example, these correspond to states where the bicycle is in an unavoidable fall, and no action can prevent the crash. In such cases, the expert is likely to be uncertain or agnostic about the action, since the choice does not arise for the expert or is unimportant. We would like to study how to incorporate the "bad query" response into the query selection process and update policy parameters based on such responses as those responses effectively indicate states to be avoided. A first step in this direction has already been taken, where we used the "bad query" responses within a Bayesian active learning framework to select queries (Judah et al., 2011). However, the responses were not incorporated into the learning of policy parameters (or updating the posterior) in that work.

Finally, we would like to apply active imitation learning to other types of imitation learning applications, e.g., the development of policy learning agents that learn by imitation of computationally expensive automated experts. For example, given a domain model or a simulator, automated experts based on various types of search can make near optimal decisions, if provided enough time. However, generating full trajectories of near-optimal behavior can be extremely time consuming, and require many trajectories if the learned policy representation is complex. Active imitation learning could be a viable approach to speed up the learning of more reactive policies, based on data from such computationally expensive experts.

It is important to consider usability issues that arise when interacting with human experts. In particular, it is likely that the cost model of queries studied in this paper (cost of 1 per query) is overly simplistic. For example, it may be less work for an expert to answer sets of queries about related states/scenarios, rather than arbitrary sets of queries. Understanding how to acquire and use such cost models for active learning is an interesting future direction. Finally, real experts will often not correspond to deterministic policies. Rather, they may appear to be non-deterministic or stochastic policies. Studying both passive and active imitation learning in such a setting is an interesting an important direction.

Acknowledgments

The authors acknowledge support of the ONR ATL program N00014-11-1-0106. 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 ONR. We would also like to thank Robert Cohn and Aaron Wilson for making driving and bicycle simulators available.

References

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 1–8, 2004.

J. Azimi, A. Fern, X. Fern, G. Borradaile, and B. Heeringa. Batch active learning via coordinated matching. In *Proceedings of the Twenty-Ninth International Conference on Machine Learning*, pages 1199–1206, 2012.

A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. In *Proceedings of the Twenty-Sixth International Conference on Machine Learning*, pages 49–56, 2009a.

A. Beygelzimer, J. Langford, and P. Ravikumar. Error-correcting tournaments. In *Proceedings of the Twentieth International Conference on Algorithmic Learning Theory*, pages 247–262, 2009b.

C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2):49–107, 1999.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

K. Brinker. Incorporating diversity in active learning with support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 59–66, 2003.

S. Chernova and M. Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34:1–25, 2009.

J. A. Clouse. An introspection approach to querying a trainer. Technical report, University of Massachusetts, Amherst, MA, USA, 1996.

D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

R. Cohn, M. Maxim, E. Durfee, and S. Singh. Selecting operator queries using expected myopic gain. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 40–47, 2010.

R. Cohn, E. Durfee, and S. Singh. Comparing action-query strategies in semi-autonomous agents. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 1102–1107, 2011.

I. Dagan and S. P. Engelson. Committee-based sampling for training probabilistic classifiers. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 150–157, 1995.

S. Dasgupta. Coarse sample complexity bounds for active learning. In *Advances in Neural Information Processing Systems 18*, pages 235–242. MIT Press, 2006.

S. Dasgupta. Two faces of active learning. *Theoretical Computer Science*, 412(19):1767–1781, 2011.

H. Daumé, J. Langford, and D. Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

T. Dietterich, G. Hao, and A. Ashenfelter. Gradient tree boosting for training conditional random fields. *Journal of Machine Learning Research*, 9:2113–2139, 2008.

F. Doshi, J. Pineau, and N. Roy. Reinforcement learning with limited reinforcement: using bayes risk for active learning in POMDPs. In *Proceedings of the Twenty-Fifth International Conference on Machine Learning*, pages 256–263, 2008.

Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.

A. Gil, H. Stern, and Y. Edan. A Cognitive Robot Collaborative Reinforcement Learning Algorithm. *International Journal of Information and Mathematical Sciences*, 5:273–280, 2009.

D. H. Grollman and O. C. Jenkins. Dogged learning for robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2483–2488, 2007.

Y. Guo and D. Schuurmans. Discriminative batch mode active learning. In *Advances in Neural Information Processing Systems 20*, pages 593–600. Curran Associates, Inc., 2008.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

S. Hanneke. *Theoretical Foundations of Active Learning*. PhD thesis, CMU Machine Learning Department, 2009.

S. C. H. Hoi, R. Jin, and M. R. Lyu. Large-scale text categorization by batch mode active learning. In *Proceedings of the Fifteenth International Conference on World Wide Web*, pages 633–642, 2006a.

S. C. H. Hoi, R. Jin, J. Zhu, and M. R. Lyu. Batch mode active learning and its application to medical image classification. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, pages 417–424, 2006b.

K. Judah, S. Roy, A. Fern, and T. Dietterich. Reinforcement learning via practice and critique advice. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 481–486, 2010.

K. Judah, A. Fern, and T. Dietterich. Active imitation learning via state queries. In *Proceedings of the ICML Workshop on Combining Learning Strategies to Reduce Label Cost*, 2011.

K. Judah, A. Fern, and T. Dietterich. Active imitation learning via reduction to i.i.d. active learning. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artifial Intelligence*, pages 428–437, 2012.

R. Khardon. Learning to take actions. *Machine Learning*, 35(1):57–90, 1999.

L. Kocsis and C. Szepesvri. Bandit based monte-carlo planning. In *Proceedings of the Seventeenth European Conference on Machine Learning*, pages 282–293, 2006.

M. Lopes, F. Melo, and L. Montesano. Active learning for reward estimation in inverse reinforcement learning. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 31–46, 2009.

A. McCallum and K. Nigam. Employing em and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 359–367, 1998.

L. Mihalkova and R. Mooney. Using active relocation to aid reinforcement learning. In *Prodeedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference*, pages 580–585, 2006.

A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 663–670, 2000.

J. Randløv and P. Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 463–471, 1998.

S. Ross and J. A. Bagnell. Efficient reductions for imitation learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 661–668, 2010.

S. Ross, G. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artifical Intelligence and Statistics*, pages 627–635, 2011.

B. Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 287–294, 1992.

A. P. Shon, D. Verma, and R. P. N. Rao. Active imitation learning. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 756–762, 2007.

D. Silver, J. A. Bagnell, and A. Stentz. Active learning from demonstration for robust autonomous navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 200–207, 2012.

R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachusetts, 1998.

Umar Syed and Robert E. Schapire. A reduction from apprenticeship learning to classification. In *Advances in Neural Information Processing Systems 23*, pages 2253–2261. Curran Associates, Inc., 2010.

L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Z. Xu, R. Akella, and Y. Zhang. Incorporating diversity and density in active learning for relevance feedback. In *Proceedings of the Twenty-Ninth European Conference on IR Research*, pages 246–257, 2007.

B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Proceeings of the IEEE International Conference on Data Mining*, pages 435–442, 2003.