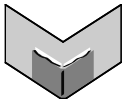
The Journal of Machine Learning Research
Volume 15
Print-Archive Edition

Pages 1371–2772



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) 1371-1429 Submitted 5/12; Revised 1/14; Published 4/14

Towards Ultrahigh Dimensional Feature Selection for Big Data

Mingkui Tan TANMINGKUI@GMAIL.COM
School of Computer Engineering
Nanyang Technological University
Blk N4, #B1a-02, Nanyang Avenue 639798, Singapore

Ivor W. Tsang IVOR.TSANG@GMAIL.COM
Center for Quantum Computation & Intelligent Systems
University of Technology Sydney, Sydney P.O. Box 123,
Broadway, NSW 2007 Sydney, Australia

Li Wang LIWANGUCSD@GMAIL.COM
Department of Mathematics, University of California
San Diego 9500 Gilman Drive La Jolla, CA 92093, USA

Editor: Sathiya Keerthi

Abstract

In this paper, we present a new adaptive feature scaling scheme for ultrahigh-dimensional feature selection on *Big Data*, and then reformulate it as a convex semi-infinite programming (SIP) problem. To address the SIP, we propose an efficient *feature generating paradigm*. Different from traditional gradient-based approaches that conduct optimization on all input features, the proposed paradigm iteratively activates a group of features, and solves a sequence of multiple kernel learning (MKL) subproblems. To further speed up the training, we propose to solve the MKL subproblems in their primal forms through a modified accelerated proximal gradient approach. Due to such optimization scheme, some efficient cache techniques are also developed. The feature generating paradigm is guaranteed to converge globally under mild conditions, and can achieve lower feature selection bias. Moreover, the proposed method can tackle two challenging tasks in feature selection: 1) group-based feature selection with complex structures, and 2) nonlinear feature selection with explicit feature mappings. Comprehensive experiments on a wide range of synthetic and real-world data sets of tens of million data points with $O(10^{14})$ features demonstrate the competitive performance of the proposed method over state-of-the-art feature selection methods in terms of generalization performance and training efficiency.

Keywords: big data, ultrahigh dimensionality, feature selection, nonlinear feature selection, multiple kernel learning, feature generation

1. Introduction

With the rapid development of the Internet, *big data* of large volume and ultrahigh dimensionality have emerged in various machine learning applications, such as text mining and information retrieval (Deng et al., 2011; Li et al., 2011, 2012). For instance, Weinberger et al. (2009) have studied a collaborative email-spam filtering task with 16 trillion (10^{13}) unique features. The ultrahigh dimensionality not only incurs unbearable memory

requirements and high computational cost in training, but also deteriorates the generalization ability because of the "curse of dimensionality" issue (Duda et al., 2000.; Guyon and Elisseeff, 2003; Zhang and Lee, 2006; Dasgupta et al., 2007; Blum et al., 2007). Fortunately, for many data sets with ultrahigh dimensions, most of the features are irrelevant to the output. Accordingly, dropping the irrelevant features and selecting the most relevant features can vastly improve the generalization performance (Ng, 1998). Moreover, in many applications such as bioinformatics (Guyon and Elisseeff, 2003), a small number of features (genes) are required to interpret the results for further biological analysis. Finally, for ultrahigh-dimensional problems, a sparse classifier is important for faster predictions.

Ultrahigh dimensional data also widely appear in many nonlinear machine learning tasks. For example, to tackle the intrinsic nonlinearity of data, researchers proposed to achieve fast training and prediction through linear techniques using explicit feature mappings (Chang et al., 2010; Maji and Berg, 2009). However, most of the explicit feature mappings will dramatically expand the data dimensionality. For instance, the commonly used 2-degree polynomial kernel feature mapping has a dimensionality of $O(m^2)$, where m denotes the number of input features (Chang et al., 2010). Even with a medium m, the dimensionality of the induced feature space is very huge. Other typical feature mappings include the spectrum-based feature mapping for string kernel (Sonnenburg et al., 2007; Sculley et al., 2006), histogram intersection kernel feature expansion (Wu, 2012), and so on.

Numerous feature selection methods have been proposed for classification tasks in the last decades (Guyon et al., 2002; Chapelle and Keerthi, 2008). In general, existing feature selection methods can be classified into two categories, namely filter methods and wrapper methods (Kohavi and John, 1997; Ng, 1998; Guyon et al., 2002). Filter methods, such as the signal-to-noise ratio method (Golub et al., 1999) and the spectral feature filtering (Zhao and Liu, 2007), own the advantages of low computational cost, but they are incapable of finding an optimal feature subset w.r.t. a predictive model of interest. On the contrary, by incorporating the inductive learning rules, wrapper methods can select more relevant features (Xu et al., 2009a; Guyon and Elisseeff, 2003). However, in general, the wrapper methods are more computationally expensive than the filter methods. Accordingly, how to scale the wrapper methods to big data is an urgent and challenging issue, and is also a major focus of this paper.

One of the most famous wrapper methods is the support vector machine (SVM) based recursive feature elimination (SVM-RFE), which has shown promising performance in the Microarray data analysis, such as gene selection task (Guyon et al., 2002). Specifically, SVM-RFE applys a recursive feature elimination scheme, and obtains nested subsets of features based on the weights of SVM classifiers. Unfortunately, the nested feature selection strategy is "monotonic" and suboptimal in identifying the most informative feature subset (Xu et al., 2009a; Tan et al., 2010). To address this drawback, non-monotonic feature selection methods have gained great attention (Xu et al., 2009a; Chan et al., 2007). Basically, the non-monotonic feature selection requires the convexity of the objective in order to easily find a global solution. To this end, Chan et al. (2007) proposed two convex relaxations to an ℓ_0-norm sparse SVM, namely QSSVM and SDP-SSVM. The resultant models are convex, and can be solved by the convex quadratically constrained quadratic programming (QCQP) and the semi-definite programming (SDP), respectively. These two methods belong to the non-monotonic feature selection methods. However, they are very expen-

sive especially for high dimensional problems. Xu *et al.* proposed another non-monotonic feature selection method, namely NMMKL (Xu et al., 2009a). Unfortunately, NMMKL is computationally infeasible for high dimensional problems since it involves a QCQP problem with many quadratic constraints.

Focusing on the logistic loss, recently, some researchers proposed to select features using greedy strategies (Tewari et al., 2011; Lozano et al., 2011), which iteratively include one feature into a feature subset. For example, Lozano et al. (2011) proposed a group orthogonal matching pursuit. Tewari et al. (2011) further introduced a general greedy scheme to solve more general sparsity constrained problems. Although promising performance has been observed, the greedy methods have several drawbacks. For example, since only one feature is involved in each iteration, these greedy methods are very expensive when there are a large number of features to be selected. More critically, due to the absence of appropriate regularizer in the objective function, the over-fitting problem may happen, which may deteriorate the generalization performance (Lozano et al., 2011; Tewari et al., 2011).

Given a set of labeled patterns $\{\mathbf{x}_i, y_i\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^m$ is an instance with m features, and $y_i \in \{\pm 1\}$ is the output label. To avoid the over-fitting problem or induce sparsity, people usually introduce some regularizers to the loss function. For instance, to select features that contribute the most to the margin, we can learn a sparse decision function $d(\mathbf{x}) = \mathbf{w}'\mathbf{x}$ by solving:

$$\min_{\mathbf{w}} \quad \|\mathbf{w}\|_0 + C\sum_{i=1}^{n} l(-y_i \mathbf{w}'\mathbf{x}_i),$$

where $l(\cdot)$ is a convex loss function, $\mathbf{w} \in \mathbb{R}^m$ is the weight vector, $\|\mathbf{w}\|_0$ denotes the ℓ_0-norm that counts the number of non-zeros in $\mathbf{w}$, and $C > 0$ is a regularization parameter. Unfortunately, this problem is NP-hard due to the ℓ_0-norm regularizer. Therefore, many researchers resort to learning a sparse decision rule through an ℓ_1-convex relaxation instead (Bradley and Mangasarian, 1998; Zhu et al., 2003; Fung and Mangasarian, 2004):

$$\min_{\mathbf{w}} \quad \|\mathbf{w}\|_1 + C\sum_{i=1}^{n} l(-y_i \mathbf{w}'\mathbf{x}_i), \tag{1}$$

where $\|\mathbf{w}\|_1 = \sum_{j=1}^{m} |w_j|$ is the ℓ_1-norm on $\mathbf{w}$. The ℓ_1-regularized problem can be efficiently solved, and many optimization methods have been proposed to solve this problem, including Newton methods (Fung and Mangasarian, 2004), proximal gradient methods (Yuan et al., 2011), coordinate descent methods (Yuan et al., 2010, 2011), and so on. Interested readers can find more details of these methods in (Yuan et al., 2010, 2011) and references therein. Beside these methods, recently, to address the big data challenge, great attention has been paid on online learning methods and stochastic gradient descent (SGD) methods for dealing with big data challenges (Xiao, 2009; Duchi and Singer, 2009; Langford et al., 2009; Shalev-Shwartz and Zhang, 2013).

However, there are several deficiencies regarding these ℓ_1-norm regularized model and existing ℓ_1-norm methods. Firstly, since the ℓ_1-norm regularization shrinks the regressors, the feature selection bias inevitably exists in the ℓ_1-norm methods (Zhang and Huang, 2008; Zhang, 2010b; Lin et al., 2010; Zhang, 2010a). To demonstrate this, let $L(\mathbf{w}) =$

$\sum_{i=1}^{n} l(-y_i \mathbf{w}'\mathbf{x}_i)$ be the empirical loss on the training data, then $\mathbf{w}^*$ is an optimal solution to (1) if and only if it satisfies the following optimality conditions (Yuan et al., 2010):

$$\begin{cases} \nabla_j L(\mathbf{w}^*) = -1/C & \text{if } w_j^* > 0, \\ \nabla_j L(\mathbf{w}^*) = 1/C & \text{if } w_j^* < 0, \\ -1/C \leq \nabla_j L(\mathbf{w}^*) \leq 1/C & \text{if } w_j^* = 0. \end{cases} \tag{2}$$

According to the above conditions, one can achieve different levels of sparsity by changing the regularization parameter C. On one hand, using a small C, minimizing $\|\mathbf{w}\|_1$ in (1) would favor selecting only a few features. The sparser the solution is, the larger the predictive risk (or empirical loss) will be (Lin et al., 2010). In other words, the solution bias will happen (Figueiredo et al., 2007). In an extreme case, where C is chosen to be tiny or even close to zero, none of the features will be selected according to the condition (2), which will lead to a very poor prediction model. On the other hand, using a large C, one can learn an more fitted prediction model to to reduce the empirical loss. However, according to (2), more features will be included. In summary, the sparsity and the unbiased solutions cannot be achieved simultaneously via solving (1) by changing the tradeoff parameter C. A possible solution is to do de-biasing with the selected features using re-training. For example, we can use a large C to train an unbiased model with the selected features (Figueiredo et al., 2007; Zhang, 2010b). However, such de-biasing methods are not efficient.

Secondly, when tackling big data of ultrahigh dimensions, the ℓ_1-regularization would be inefficient or infeasible for most of the existing methods. For example, when the dimensionality is around 10^{12}, one needs about 1 TB memory to store the weight vector $\mathbf{w}$, which is intractable for existing ℓ_1-methods, including online learning methods and SGD methods (Langford et al., 2009; Shalev-Shwartz and Zhang, 2013). Thirdly, due to the scale variation of $\mathbf{w}$, it is also non-trivial to control the number of features to be selected meanwhile to regulate the decision function.

In the conference version of this paper, an ℓ_0-norm sparse SVM model is introduced (Tan et al., 2010). Its nice optimization scheme has brought significant benefits to several applications, such as image retrieval (Rastegari et al., 2011), multi-label prediction (Gu et al., 2011a), feature selection for multivariate performance measures (Mao and Tsang, 2013), feature selection for logistic regression (Tan et al., 2013), and graph-based feature selection (Gu et al., 2011b). However, several issues remain to be solved. First of all, the tightness of the convex relation remains unclear. Secondly, the adopted optimization strategy is incapable of dealing with very large-scale problems with many training instances. Thirdly, the presented feature selection strategy was limited to linear features, but in many applications, one indeed needs to tackle nonlinear features that are with complex structures.

Regarding the above issues, in this paper, we propose an adaptive feature scaling (AFS) for feature selection by introducing a continuous feature scaling vector $\mathbf{d} \in [0, 1]^m$. To enforce the sparsity, we impose an explicit ℓ_1-constraint $||\mathbf{d}||_1 \leq B$, where the scalar B represents the least number of features to be selected. The solution to the resultant optimization problem is non-trivial due to the additional constraint. Fortunately, by transforming it as a convex semi-infinite programming (SIP) problem, an efficient optimization scheme can be developed. In summary, this paper makes the following extensions and improvements.

- A feature generating machine (FGM) is proposed to efficiently address the ultrahigh-dimensional feature selection task through solving the proposed SIP problem. Instead

of performing the optimization on all input features, FGM iteratively infers the most informative features, and then solves a reduced multiple kernel learning (MKL) subproblem, where each base kernel is defined on a set of features.[1]

- The proposed optimization scheme mimics the re-training strategy to reduce the feature selection bias with little effort. Specifically, the feature selection bias can be effectively alleviated by separately controlling the complexity and sparsity of the decision function, which is one of the major advantages of the proposed scheme.

- To speed up the training on big data, we propose to solve the primal form of the MKL subproblem by a modified accelerated proximal gradient method. As a result, the memory requirement and computational cost can be significantly reduced. The convergence rate of the modified APG is also provided. Moreover, several cache techniques are proposed to further enhance the efficiency.

- The feature generating paradigm is also extended to group feature selection with complex group structures and nonlinear feature selection using explicit feature mappings.

The remainder of this paper is organized as follows. In Section 2, we start by presenting the adaptive feature scalings (AFS) for linear feature selection and group feature selection, and then present the convex SIP reformulations of the resultant optimization problems. After that, to solve the SIP problems, in Section 3, we propose the feature generating machine (FGM) which includes two core steps, namely the worst-case analysis step and the subproblem optimization step. In Section 4, we illustrate the worst-case analysis for a number of learning tasks, including the group feature selection with complex group structures and the nonlinear feature selection with explicit feature mappings. We introduce the subproblem optimization in Section 5 and extend FGM for multiple kernel learning w.r.t. many additive kernels in Section 6. Related studies are presented in Section 7. We conduct empirical studies in Section 8, and conclude this work in Section 9.

2. Feature Selection Through Adaptive Feature Scaling

Throughout the paper, we denote the transpose of vector/matrix by the superscript $'$, a vector with all entries equal to one as $\mathbf{1} \in \mathbb{R}^n$, and the zero vector as $\mathbf{0} \in \mathbb{R}^n$. In addition, we denote a data set by $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]' = [\mathbf{f}^1, ..., \mathbf{f}^m]$, where $\mathbf{x}_i \in \mathbb{R}^m$ represents the ith instance and $\mathbf{f}^j \in \mathbb{R}^n$ denotes the jth feature vector. We use $|\mathcal{G}|$ to denote cardinality of an index set $\mathcal{G}$ and $|v|$ to denote the absolute value of a real number v. For simplicity, we denote $\mathbf{v} \succeq \boldsymbol{\alpha}$ if $v_i \geq \alpha_i, \forall i$ and $\mathbf{v} \preceq \boldsymbol{\alpha}$ if $v_i \leq \alpha_i, \forall i$. We also denote $\|\mathbf{v}\|_p$ as the ℓ_p-norm of a vector and $\|\mathbf{v}\|$ as the ℓ_2-norm of a vector. Given a vector $\mathbf{v} = [\mathbf{v}_1', ..., \mathbf{v}_p']'$, where $\mathbf{v}_i$ denotes a sub-vector of $\mathbf{v}$, we denote $\|\mathbf{v}\|_{2,1} = \sum_{i=1}^p \|\mathbf{v}_i\|$ as the mixed ℓ_1/ℓ_2 norm (Bach et al., 2011) and $\|\mathbf{v}\|_{2,1}^2 = (\sum_{i=1}^p \|\mathbf{v}_i\|)^2$. Accordingly, we call $\|\mathbf{v}\|_{2,1}^2$ as an $\ell_{2,1}^2$ regularizer. Following Rakotomamonjy et al. (2008), we define $\frac{x_i}{0} = 0$ if $x_i = 0$ and ∞ otherwise. Finally, $\mathbf{A} \odot \mathbf{B}$ represents the element-wise product between two matrices $\mathbf{A}$ and $\mathbf{B}$.

1. The C++ and MATLAB source codes of the proposed methods are publicly available at `http://www.tanmingkui.com/fgm.html`.

2.1 A New AFS Scheme for Feature Selection

In the standard support vector machines (SVM), one learns a linear decision function $d(\mathbf{x}) = \mathbf{w}'\mathbf{x} - b$ by solving the following ℓ_2-norm regularized problem:

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} l(-y_i(\mathbf{w}'\mathbf{x}_i - b)), \tag{3}$$

where $\mathbf{w} = [w_1, \ldots, w_m]' \in \mathbb{R}^m$ denotes the weight of the decision hyperplane, b denotes the shift from the origin, $C > 0$ represents the regularization parameter and $l(\cdot)$ denotes a convex loss function. In this paper, we concentrate on two kinds of loss functions, namely the squared hinge loss

$$l(-y_i(\mathbf{w}'\mathbf{x}_i - b)) = \frac{1}{2}\max(1 - y_i(\mathbf{w}'\mathbf{x}_i - b), 0)^2$$

and the logistic loss

$$l(-y_i(\mathbf{w}'\mathbf{x}_i - b)) = \log(1 + \exp(-y_i(\mathbf{w}'\mathbf{x}_i - b))).$$

For simplicity, herein we concentrate the squared hinge loss only.

In (3), the ℓ_2-regularizer $||\mathbf{w}||^2$ is used to avoid the over-fitting problem (Hsieh et al., 2008), which, however, cannot induce sparse solutions. To address this issue, we introduce a feature scaling vector $\mathbf{d} \in [0, 1]^m$ such that we can scale the importance of features. Specifically, given an instance $\mathbf{x}_i$, we impose $\sqrt{\mathbf{d}} = [\sqrt{d_1}, \ldots, \sqrt{d_m}]'$ on its features (Vishwanathan et al., 2010), resulting in a re-scaled instance

$$\widehat{\mathbf{x}}_i = (\mathbf{x}_i \odot \sqrt{\mathbf{d}}). \tag{4}$$

In this scaling scheme, the jth feature is selected if and only if $d_j > 0$.

Note that, in many real-world applications, one may intend to select a desired number of features with acceptable generalization performance. For example, in the Microarray data analysis, due to expensive bio-diagnosis and limited resources, biologists prefer to select less than 100 genes from hundreds of thousands of genes (Guyon et al., 2002; Golub et al., 1999). To incorporate such prior knowledge, we explicitly impose an ℓ_1-norm constraint on $\mathbf{d}$ to induce the sparsity:

$$\sum_{j=1}^{m} d_j = ||\mathbf{d}||_1 \leq B,\ d_j \in [0, 1],\ j = 1, \cdots, m,$$

where the integer B represents the least number of features to be selected. Similar feature scaling scheme has been used by many works (e.g., Weston et al., 2000; Chapelle et al., 2002; Grandvalet and Canu, 2002; Rakotomamonjy, 2003; Varma and Babu, 2009; Vishwanathan et al., 2010). However, different from the proposed scaling scheme, in these scaling schemes, $\mathbf{d}$ is not bounded in $[0,\ 1]^m$.

Let $\mathcal{D} = \left\{\mathbf{d} \in \mathbb{R}^m \middle| \sum_{j=1}^{m} d_j \leq B,\ d_j \in [0, 1],\ j = 1, \cdots, m\right\}$ be the domain of $\mathbf{d}$, the proposed AFS can be formulated as the following problem:

$$\begin{aligned} \min_{\mathbf{d}\in\mathcal{D}} \min_{\mathbf{w},\boldsymbol{\xi},b} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 \\ \text{s.t.} \quad & y_i\left(\mathbf{w}'(\mathbf{x}_i \odot \sqrt{\mathbf{d}}) - b\right) \geq 1 - \xi_i, \quad i = 1, \cdots, n, \end{aligned} \tag{5}$$

where C is a regularization parameter that trades off between the model complexity and the fitness of the decision function, and $b/\|\mathbf{w}\|$ determines the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$. This problem is non-convex w.r.t. $\mathbf{w}$ and $\mathbf{d}$ simultaneously, and the compact domain $\mathcal{D}$ contains infinite number of elements. However, for a fixed $\mathbf{d}$, the inner minimization problem w.r.t. $\mathbf{w}$ and $\boldsymbol{\xi}$ is a standard SVM problem:

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi},b} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 \\ \text{s.t.} \quad & y_i\left(\mathbf{w}'(\mathbf{x}_i \odot \sqrt{\mathbf{d}}) - b\right) \geq 1 - \xi_i, \quad i = 1, \cdots, n, \end{aligned} \tag{6}$$

which can be solved in its dual form. By introducing the Lagrangian multiplier $\alpha_i \geq 0$ to each constraint $y_i\left(\mathbf{w}'(\mathbf{x}_i \odot \sqrt{\mathbf{d}}) - b\right) \geq 1 - \xi_i$, the Lagrangian function is:

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 - \sum_{i=1}^{n}\alpha_i\left(y_i\left(\mathbf{w}'(\mathbf{x}_i \odot \sqrt{\mathbf{d}}) - b\right) - 1 + \xi_i\right). \tag{7}$$

By setting the derivatives of $\mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, b, \boldsymbol{\alpha})$ w.r.t. $\mathbf{w}$, $\boldsymbol{\xi}$ and b to $\mathbf{0}$, respectively, we get

$$\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}}), \boldsymbol{\alpha} = C\boldsymbol{\xi}, \text{ and } \sum_{i=1}^{n}\alpha_i y_i = 0. \tag{8}$$

Substitute these results into (7), and we arrive at the dual form of problem (6) as:

$$\max_{\boldsymbol{\alpha}\in\mathcal{A}} \quad -\frac{1}{2}\left\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\right\|^2 - \frac{1}{2C}\boldsymbol{\alpha}'\boldsymbol{\alpha} + \boldsymbol{\alpha}'\mathbf{1},$$

where $\mathcal{A} = \{\boldsymbol{\alpha} | \sum_{i=1}^{n}\alpha_i y_i = 0, \boldsymbol{\alpha} \succeq \mathbf{0}\}$ is the domain of $\boldsymbol{\alpha}$. For convenience, let $\mathbf{c}(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i \in \mathbb{R}^m$, we have $\left\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\right\|^2 = \sum_{j=1}^{m} d_j[c_j(\boldsymbol{\alpha})]^2$, where the jth coordinate of $\mathbf{c}(\boldsymbol{\alpha})$, namely $c_j(\boldsymbol{\alpha})$, is a function of $\boldsymbol{\alpha}$. For simplicity, let

$$f(\boldsymbol{\alpha}, \mathbf{d}) = \frac{1}{2}\sum_{j=1}^{m} d_j[c_j(\boldsymbol{\alpha})]^2 + \frac{1}{2C}\boldsymbol{\alpha}'\boldsymbol{\alpha} - \boldsymbol{\alpha}'\mathbf{1}.$$

Apparently, $f(\boldsymbol{\alpha}, \mathbf{d})$ is linear in $\mathbf{d}$ and concave in $\boldsymbol{\alpha}$, and both $\mathcal{A}$ and $\mathcal{D}$ are compact domains. Problem (5) can be equivalently reformulated as the following problem:

$$\min_{\mathbf{d}\in\mathcal{D}}\max_{\boldsymbol{\alpha}\in\mathcal{A}} \quad -f(\boldsymbol{\alpha}, \mathbf{d}), \tag{9}$$

However, this problem is still difficult to be addressed. Recall that both $\mathcal{A}$ and $\mathcal{D}$ are convex compact sets, according to the minimax saddle-point theorem (Sion, 1958), we immediately have the following relation.

Theorem 1 *According to the minimax saddle-point theorem (Sion, 1958), the following equality holds by interchanging the order of* $\min_{\mathbf{d}\in\mathcal{D}}$ *and* $\max_{\boldsymbol{\alpha}\in\mathcal{A}}$ *in (9),*

$$\min_{\mathbf{d}\in\mathcal{D}}\max_{\boldsymbol{\alpha}\in\mathcal{A}} \quad -f(\boldsymbol{\alpha}, \mathbf{d}) = \max_{\boldsymbol{\alpha}\in\mathcal{A}}\min_{\mathbf{d}\in\mathcal{D}} \quad -f(\boldsymbol{\alpha}, \mathbf{d}).$$

Based on the above equivalence, rather than solving the original problem in (9), hereafter we address the following minimax problem instead:

$$\min_{\boldsymbol{\alpha}\in\mathcal{A}} \max_{\mathbf{d}\in\mathcal{D}} \quad f(\boldsymbol{\alpha}, \mathbf{d}). \tag{10}$$

2.2 AFS for Group Feature Selection

The above AFS scheme for linear feature selection can be extended for group feature selections, where the features are organized into groups defined by $\mathcal{G} = \{\mathcal{G}_1, ..., \mathcal{G}_p\}$, where $\cup_{j=1}^{p} \mathcal{G}_j = \{1, ..., m\}$, $p = |\mathcal{G}|$ denotes the number of groups, and $\mathcal{G}_j \subset \{1, ..., m\}, j = 1, ..., p$ denotes the index set of feature supports belonging to the jth group. In the group feature selection, a feature in one group is selected if and only if this group is selected (Yuan and Lin, 2006; Meier et al., 2008). Let $\mathbf{w}_{\mathcal{G}_j} \in \mathbb{R}^{|\mathcal{G}_j|}$ and $\mathbf{x}_{\mathcal{G}_j} \in \mathbb{R}^{|\mathcal{G}_j|}$ be the components of $\mathbf{w}$ and $\mathbf{x}$ related to $\mathcal{G}_j$, respectively. The group feature selection can be achieved by solving the following non-smooth group lasso problem (Yuan and Lin, 2006; Meier et al., 2008):

$$\min_{\mathbf{w}} \quad \lambda \sum_{j=1}^{p} ||\mathbf{w}_{\mathcal{G}_j}||_2 + \sum_{i=1}^{n} l(-y_i \sum_{j=1}^{p} \mathbf{w}'_{\mathcal{G}_j} \mathbf{x}_{i\mathcal{G}_j}), \tag{11}$$

where λ is a trade-off parameter. Many efficient algorithms have been proposed to solve this problem, such as the accelerated proximal gradient descent methods (Liu and Ye, 2010; Jenatton et al., 2011b; Bach et al., 2011), block coordinate descent methods (Qin et al., 2010; Jenatton et al., 2011b) and active set methods (Bach, 2009; Roth and Fischer, 2008). However, the issues of the ℓ_1-regularization, namely the scalability issue for big data and the feature selection bias, will also happen when solving (11). More critically, when dealing with feature groups with complex structures, the number of groups can be exponential in the number of features m. As a result, solving (11) could be very expensive.

To extend AFS to group feature selection, we introduce a group scaling vector $\widehat{\mathbf{d}} = [\widehat{d}_1, \ldots, \widehat{d}_p]' \in \widehat{\mathcal{D}}$ to scale the groups, where $\widehat{\mathcal{D}} = \big\{\widehat{\mathbf{d}} \in \mathbb{R}^p \big| \sum_{j=1}^{p} \widehat{d}_j \leq B, \widehat{d}_j \in [0,1],\ j = 1, \cdots, p\big\}$. Here, without loss of generality, we first assume that there is no overlapping element among groups, namely, $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset, \forall i \neq j$. Accordingly, we have $\mathbf{w} = [\mathbf{w}'_{\mathcal{G}_1}, ..., \mathbf{w}'_{\mathcal{G}_p}]' \in \mathbb{R}^m$. By taking the shift term b into consideration, the decision function is expressed as:

$$d(\mathbf{x}) = \sum_{j=1}^{p} \sqrt{\widehat{d}_j} \mathbf{w}'_{\mathcal{G}_j} \mathbf{x}_{\mathcal{G}_j} - b,$$

By applying the squared hinge loss, the AFS based group feature selection task can be formulated as the following optimization problem:

$$\begin{aligned} \min_{\widehat{\mathbf{d}}\in\widehat{\mathcal{D}}} \min_{\mathbf{w},\boldsymbol{\xi},b} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{n} \xi_i^2 \\ \text{s.t.} \quad & y_i \left(\sum_{j=1}^{p} \sqrt{\widehat{d}_j} \mathbf{w}'_{\mathcal{G}_j} \mathbf{x}_{i\mathcal{G}_j} - b \right) \geq 1 - \xi_i, \ \ \xi_i \geq 0, \ \ i = 1, \cdots, n. \end{aligned}$$

With similar deductions in Section 2.1, this problem can be transformed into the following minimax problem:

$$\min_{\widehat{\mathbf{d}}\in\widehat{\mathcal{D}}}\max_{\boldsymbol{\alpha}\in\mathcal{A}} -\frac{1}{2}\sum_{j=1}^{p}\widehat{d}_j\left\|\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{i\mathcal{G}_j}\right\|^2 - \frac{1}{2C}\boldsymbol{\alpha}'\boldsymbol{\alpha} + \boldsymbol{\alpha}'\mathbf{1}.$$

This problem is reduced to the linear feature selection case if $|\mathcal{G}_j| = 1, \forall j = 1, ..., p$. For convenience, hereafter we drop the hat from $\widehat{\mathbf{d}}$ and $\widehat{\mathcal{D}}$. Let $\mathbf{c}_{\mathcal{G}_j}(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{i\mathcal{G}_j}$. Moreover, we define

$$f(\boldsymbol{\alpha}, \mathbf{d}) = \frac{1}{2}\sum_{j=1}^{p} d_j \left\|\mathbf{c}_{\mathcal{G}_j}(\boldsymbol{\alpha})\right\|^2 + \frac{1}{2C}\boldsymbol{\alpha}'\boldsymbol{\alpha} - \boldsymbol{\alpha}'\mathbf{1}.$$

Finally, we arrive at a unified minimax problem for both linear and group feature selections:

$$\min_{\boldsymbol{\alpha}\in\mathcal{A}}\max_{\mathbf{d}\in\mathcal{D}} \quad f(\boldsymbol{\alpha}, \mathbf{d}), \tag{12}$$

where $\mathcal{D} = \left\{\mathbf{d} \in \mathbb{R}^p \middle| \sum_{j=1}^{p} d_j \leq B,\ d_j \in [0, 1],\ j = 1, \cdots, p\right\}$. When $|\mathcal{G}_j| = 1, \forall j = 1, ..., p$, we have $p = m$, and problem (12) is reduced to problem (10).

2.3 Group Feature Selection with Complex Structures

Now we extend the above group AFS scheme to feature groups with overlapping features or even more complex structures. When dealing with groups with overlapping features, a heuristic way is to explicitly augment $\mathbf{X} = [\mathbf{f}^1, ..., \mathbf{f}^m]$ to make the overlapping groups non-overlapping by repeating the overlapping features. For example, suppose $\mathbf{X} = [\mathbf{f}^1, \mathbf{f}^2, \mathbf{f}^3]$ with groups $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2\}$, where $\mathcal{G}_1 = \{1, 2\}$ and $\mathcal{G}_2 = \{2, 3\}$, and $\mathbf{f}^2$ is an overlapping feature. To avoid the overlapping feature issue, we can repeat $\mathbf{f}^2$ to construct an augmented data set $\mathbf{X}_{au} = [\mathbf{f}^1, \mathbf{f}^2, \mathbf{f}^2, \mathbf{f}^3]$, where the group index sets become $\mathcal{G}_1 = \{1, 2\}$ and $\mathcal{G}_2 = \{3, 4\}$. This feature augmentation strategy can be extended to groups with even more complex structures, such as tree structures or graph structures (Bach, 2009). For simplicity, in this paper, we only study the tree-structured groups.

Definition 1 *Tree-structured set of groups (Jenatton et al., 2011b; Kim and Xing, 2010, 2012). A super set of groups $\mathcal{G} \triangleq \{\mathcal{G}_h\}_{\mathcal{G}_h\in\mathcal{G}}$ with $|\mathcal{G}| = p$ is said to be tree-structured in $\{1, ..., m\}$, if $\cup\mathcal{G}_h = \{1, ..., m\}$ and if for all $\mathcal{G}_g, \mathcal{G}_h \in \mathcal{G}$, $(\mathcal{G}_g \cap \mathcal{G}_h \neq \varnothing) \Rightarrow (\mathcal{G}_g \subseteq \mathcal{G}_h$ or $\mathcal{G}_h \subseteq \mathcal{G}_g)$. For such a set of groups, there exists a (non-unique) total order relation $\preceq$ such that:*

$$\mathcal{G}_g \preceq \mathcal{G}_h \Rightarrow \{\mathcal{G}_g \subseteq \mathcal{G}_h \text{ or } \mathcal{G}_g \cap \mathcal{G}_h = \varnothing\}.$$

Similar to the overlapping case, we augment the overlapping elements of all groups along the tree structures, resulting in the augmented data set $\mathbf{X}_{au} = [\mathbf{X}_{\mathcal{G}_1}, ..., \mathbf{X}_{\mathcal{G}_p}]$, where $\mathbf{X}_{\mathcal{G}_i}$ represents the data columns indexed by $\mathcal{G}_i$ and p denotes the number of all possible groups. However, this simple idea may bring great challenges for optimization, particularly when there are huge number of overlapping groups (For instance, in graph-based group structures, the number of groups p can be exponential in m (Bach, 2009)).

3. Feature Generating Machine

Under the proposed AFS scheme, both linear feature selection and group feature selection can be cast as the minimax problem in (12). By bringing in an additional variable $\theta \in \mathbb{R}$, this problem can be further formulated as a semi-infinite programming (SIP) problem (Kelley, 1960; Pee and Royset, 2010):

$$\min_{\boldsymbol{\alpha}\in\mathcal{A},\theta\in\mathbb{R}} \theta, \quad \text{s.t.} \quad \theta \geq f(\boldsymbol{\alpha}, \mathbf{d}), \quad \forall\ \mathbf{d} \in \mathcal{D}. \tag{13}$$

In (13), each nonzero $\mathbf{d} \in \mathcal{D}$ defines a quadratic constraint w.r.t. $\boldsymbol{\alpha}$. Since there are infinite $\mathbf{d}$'s in $\mathcal{D}$, problem (13) involves infinite number of constraints, thus it is very difficult to be solved.

3.1 Optimization Strategies by Feature Generation

Before solving (13), we first discuss its optimality condition. Specifically, let $\mu_h \geq 0$ be the dual variable for each constraint $\theta \geq f(\boldsymbol{\alpha}, \mathbf{d})$, the Lagrangian function of (13) can be written as:

$$\mathcal{L}(\theta, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \theta - \sum_{\mathbf{d}_h\in\mathcal{D}} \mu_h \left(\theta - f(\boldsymbol{\alpha}, \mathbf{d}_h)\right).$$

By setting its derivative w.r.t. θ to zero, we have $\sum \mu_h = 1$. Let $\mathcal{M} = \{\boldsymbol{\mu} | \sum \mu_h = 1, \mu_h \geq 0, h = 1, ..., |\mathcal{D}|\}$ be the domain of $\boldsymbol{\mu}$ and define

$$f_{\mathrm{m}}(\boldsymbol{\alpha}) = \max_{\mathbf{d}_h\in\mathcal{D}} f(\boldsymbol{\alpha}, \mathbf{d}_h).$$

The KKT conditions of (13) can be written as:

$$\sum_{\mathbf{d}_h\in\mathcal{D}} \mu_h \nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}, \mathbf{d}_h) = \mathbf{0}, \quad \text{and} \quad \sum_{\mathbf{d}_h\in\mathcal{D}} \mu_h = 1. \tag{14}$$

$$\mu_h(f(\boldsymbol{\alpha}, \mathbf{d}_h) - f_m(\boldsymbol{\alpha})) = 0, \quad \mu_h \geq 0, \quad h = 1, ..., |\mathcal{D}|. \tag{15}$$

In general, there are many constraints in problem (14). However, most of them are nonactive at the optimality if the data contain only a small number of relevant features w.r.t. the output $\mathbf{y}$. Specifically, according to condition (15), we have $\mu_h = 0$ if $f(\boldsymbol{\alpha}, \mathbf{d}_h) < f_{\mathrm{m}}(\boldsymbol{\alpha})$, which will induce the sparsity among μ_h's. Motivated by this observation, we design an efficient optimization scheme which iteratively "finds" the active constraints, and then solves a subproblem with the selected constraints only. By applying this scheme, the computational burden brought by the infinite number of constraints can be avoided. The details of the above procedure is presented in Algorithm 1, which is also known as the cutting plane algorithm (Kelley, 1960; Mutapcic and Boyd, 2009).

Algorithm 1 involves two major steps: the feature inference step (also known as the worst-case analysis) and the subproblem optimization step. Specifically, the worst-case analysis is to infer the most-violated $\mathbf{d}_t$ based on $\boldsymbol{\alpha}^{t-1}$, and add it into the active constraint set $\mathcal{C}_t$. Once an active $\mathbf{d}_t$ is identified, we update $\boldsymbol{\alpha}^t$ by solving the following subproblem with the constraints defined in $\mathcal{C}_t$:

$$\min_{\boldsymbol{\alpha}\in\mathcal{A},\theta\in\mathbb{R}} \theta, \quad \text{s.t.} \quad f(\boldsymbol{\alpha}, \mathbf{d}_h) - \theta \leq 0,\ \forall\ \mathbf{d}_h \in \mathcal{C}_t. \tag{16}$$

Algorithm 1 Cutting Plane Algorithm for Solving (13).

1: Initialize $\boldsymbol{\alpha}^0 = C\mathbf{1}$ and $\mathcal{C}_0 = \emptyset$. Set iteration index $t = 1$.
2: Feature Inference:
Do worst-case analysis to *infer* the most violated $\mathbf{d}_t$ based on $\boldsymbol{\alpha}^{t-1}$.
Set $\mathcal{C}_t = \mathcal{C}_{t-1} \bigcup \{\mathbf{d}_t\}$.
3: Subproblem Optimization:
Solve subproblem (16), obtaining the optimal solution $\boldsymbol{\alpha}^t$ and μ^t.
4: Let $t = t + 1$. Repeat step 2-3 until convergence.

For feature selection tasks, the optimization complexity of (13) can be greatly reduced, since there are only a small number of active constraints involved in problem (16).

The whole procedure iterates until some stopping conditions are achieved. As will be shown later, in general, each active $\mathbf{d}_t \in \mathcal{C}_t$ involves at most B **new features**. In this sense, we refer Algorithm 1 to as the *feature generating machine* (FGM). Recall that, at the beginning, there is no feature being selected, thus we have the empirical loss $\boldsymbol{\xi} = \mathbf{1}$. According to (8), we can initialize $\boldsymbol{\alpha}^0 = C\mathbf{1}$. Finally, once the optimal solution $\mathbf{d}^*$ to (16) is obtained, the selected features (or feature groups) are associated with the nonzero entries in $\mathbf{d}^*$. Note that, each $\mathbf{d} \in \mathcal{C}_t$ involves at most B features/groups, thus the number of selected features/groups is no more than tB after t iterations, namely $||\mathbf{d}^*||_0 \leq tB$.

3.2 Convergence Analysis

Before the introduction of the worst-case analysis and the solution to the subproblem, we first conduct the convergence analysis of Algorithm 1.

Without loss of generality, let $\mathcal{A} \times \mathcal{D}$ be the domain for problem (13). In the $(t+1)$th iteration, we find a new constraint $\mathbf{d}_{t+1}$ based on $\boldsymbol{\alpha}_t$ and add it into $\mathcal{C}_t$, i.e., $f(\boldsymbol{\alpha}_t, \mathbf{d}_{t+1}) = \max_{\mathbf{d}\in\mathcal{D}} f(\boldsymbol{\alpha}_t, \mathbf{d})$. Apparently, we have $\mathcal{C}_t \subseteq \mathcal{C}_{t+1}$. For convenience, we define

$$\beta_t = \max_{1\leq i\leq t} f(\boldsymbol{\alpha}_t, \mathbf{d}_i) = \min_{\boldsymbol{\alpha}\in\mathcal{A}} \max_{1\leq i\leq t} f(\boldsymbol{\alpha}, \mathbf{d}_i).$$

and

$$\varphi_t = \min_{1\leq j\leq t} f(\boldsymbol{\alpha}_j, \mathbf{d}_{j+1}) = \min_{1\leq j\leq t} (\max_{\mathbf{d}\in\mathcal{D}} f(\boldsymbol{\alpha}_j, \mathbf{d})),$$

First of all, we have the following lemma.

Lemma 1 *Let (α^*, θ^*) be a globally optimal solution of (13), $\{\theta_t\}$ and $\{\varphi_t\}$ as defined above, then: $\theta_t \leq \theta^* \leq \varphi_t$. With the number of iteration t increasing, $\{\theta_t\}$ is monotonically increasing and the sequence $\{\varphi_t\}$ is monotonically decreasing (Chen and Ye, 2008).*

Proof According to the definition, we have $\theta_t = \beta_t$. Moreover, $\theta^* = \min_{\boldsymbol{\alpha}\in\mathcal{A}} \max_{\mathbf{d}\in\mathcal{D}} f(\boldsymbol{\alpha}, \mathbf{d})$. For a fixed feasible $\boldsymbol{\alpha}$, we have $\max_{\mathbf{d}\in\mathcal{C}_t} f(\boldsymbol{\alpha}, \mathbf{d}) \leq \max_{\mathbf{d}\in\mathcal{D}} f(\boldsymbol{\alpha}, \mathbf{d})$, then

$$\min_{\boldsymbol{\alpha}\in\mathcal{A}} \max_{\mathbf{d}\in\mathcal{C}_t} f(\boldsymbol{\alpha}, \mathbf{d}) \leq \min_{\boldsymbol{\alpha}\in\mathcal{A}} \max_{\mathbf{d}\in\mathcal{D}} f(\boldsymbol{\alpha}, \mathbf{d}),$$

that is, $\theta_t \leq \theta^*$. On the other hand, for $\forall j = 1, \cdots, k$, $f(\boldsymbol{\alpha}_j, \mathbf{d}_{j+1}) = \max_{\mathbf{d}\in\mathcal{D}} f(\boldsymbol{\alpha}_j, \mathbf{d})$, thus $(\boldsymbol{\alpha}_j, f(\boldsymbol{\alpha}_j, \mathbf{d}_{j+1}))$ is a feasible solution of (13). Then $\theta^* \leq f(\boldsymbol{\alpha}_j, \mathbf{d}_{j+1})$ for $j = 1, \cdots, t$, and hence we have

$$\theta^* \leq \varphi_t = \min_{1\leq j\leq t} f(\boldsymbol{\alpha}_j, \mathbf{d}_{j+1}).$$

With increasing iteration t, the subset $\mathcal{C}_t$ is monotonically increasing, so $\{\theta_t\}$ is monotonically increasing while $\{\varphi_t\}$ is monotonically decreasing. The proof is completed. ■

The following theorem shows that FGM converges to a global solution of (13).

Theorem 2 *Assume that in Algorithm 1, the subproblem (16) and the worst-case analysis in step 2 can be solved. Let $\{(\boldsymbol{\alpha}_t, \theta_t)\}$ be the sequence generated by Algorithm 1. If Algorithm 1 terminates at iteration $(t+1)$, then $\{(\boldsymbol{\alpha}_t, \theta_t)\}$ is the global optimal solution of (13); otherwise, $(\boldsymbol{\alpha}_t, \theta_t)$ converges to a global optimal solution $(\boldsymbol{\alpha}^*, \theta^*)$ of (13).*

Proof We can measure the convergence of FGM by the gap difference of series $\{\theta_t\}$ and $\{\varphi_t\}$. Assume in tth iteration, there is no update of $\mathcal{C}_t$, i.e. $\mathbf{d}_{t+1} = \arg\max_{\mathbf{d}\in\mathcal{D}} f(\boldsymbol{\alpha}_t, \mathbf{d}) \in \mathcal{C}_t$, then $\mathcal{C}_t = \mathcal{C}_{t+1}$. In this case, $(\boldsymbol{\alpha}_t, \theta_t)$ is the globally optimal solution of (13). Actually, since $\mathcal{C}_t = \mathcal{C}_{t+1}$, in Algorithm 1, there will be no update of $\boldsymbol{\alpha}$, i.e. $\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t$. Then we have

$$\begin{aligned} f(\boldsymbol{\alpha}_t, \mathbf{d}_{t+1}) &= \max_{\mathbf{d}\in\mathcal{D}} f(\boldsymbol{\alpha}_t, \mathbf{d}) = \max_{\mathbf{d}\in\mathcal{C}_t} f(\boldsymbol{\alpha}_t, \mathbf{d}) = \max_{1\leq i\leq t} f(\boldsymbol{\alpha}_t, \mathbf{d}_i) = \theta_t \\ \varphi_t &= \min_{1\leq j\leq t} f(\boldsymbol{\alpha}_j, \mathbf{d}_{j+1}) \leq \theta_t. \end{aligned}$$

According to Lemma 1, we have $\theta_t \leq \theta^* \leq \varphi_t$, thus we have $\theta_t = \theta^* = \varphi_t$, and $(\boldsymbol{\alpha}_t, \theta_t)$ is the global optimum of (13).

Suppose the algorithm does not terminate in finite steps. Let $\mathcal{X} = \mathcal{A} \times [\theta_1, \theta^*]$, a limit point $(\bar{\boldsymbol{\alpha}}, \bar{\theta})$ exists for $(\boldsymbol{\alpha}_t, \theta_t)$, since $\mathcal{X}$ is compact. And we also have $\bar{\theta} \leq \theta^*$. For each t, let $\mathcal{X}_t$ be the feasible region of tth subproblem, which have $\mathcal{X}_t \subseteq \mathcal{X}$, and $(\bar{\boldsymbol{\alpha}}, \bar{\theta}) \in \cap_{t=1}^{\infty} \mathcal{X}_t \subseteq \mathcal{X}$. Then we have $f(\bar{\boldsymbol{\alpha}}, \mathbf{d}_t) - \bar{\theta} \leq 0$, $\mathbf{d}_t \in C_t$ for each given $t = 1, \cdots$.

To show $(\bar{\boldsymbol{\alpha}}, \bar{\theta})$ is global optimal of problem (13), we only need to show $(\bar{\boldsymbol{\alpha}}, \bar{\theta})$ is a feasible point of problem (13), i.e., $\bar{\theta} \geq f(\bar{\boldsymbol{\alpha}}, \mathbf{d})$ for all $\mathbf{d} \in \mathcal{D}$, so $\bar{\theta} \geq \theta^*$ and we must have $\bar{\theta} = \theta^*$. Let $v(\boldsymbol{\alpha}, \theta) = \min_{\mathbf{d}\in\mathcal{D}}(\theta - f(\boldsymbol{\alpha}, \mathbf{d})) = \theta - \max_{\mathbf{d}\in\mathcal{D}} f(\boldsymbol{\alpha}, \mathbf{d})$. Then $v(\boldsymbol{\alpha}, \theta)$ is continuous w.r.t. $(\boldsymbol{\alpha}, \theta)$. By applying the continuity property of $v(\boldsymbol{\alpha}, \theta)$, we have

$$\begin{aligned} v(\bar{\boldsymbol{\alpha}}, \bar{\theta}) &= v(\boldsymbol{\alpha}_t, \theta_t) + (v(\bar{\boldsymbol{\alpha}}, \bar{\theta}) - v(\boldsymbol{\alpha}_t, \theta_t)) \\ &= (\theta_t - f(\boldsymbol{\alpha}_t, \mathbf{d}_{t+1})) + (v(\bar{\boldsymbol{\alpha}}, \bar{\theta}) - v(\boldsymbol{\alpha}_t, \theta_t)) \\ &\geq (\theta_t - f(\boldsymbol{\alpha}_t, \mathbf{d}_{t+1})) - (\bar{\theta} - f(\bar{\boldsymbol{\alpha}}, \mathbf{d}_t)) + (v(\bar{\boldsymbol{\alpha}}, \bar{\theta}) - v(\boldsymbol{\alpha}_t, \theta_t)) \to 0 \quad (\text{when } t \to \infty), \end{aligned}$$

where we use the continuity of $v(\boldsymbol{\alpha}, \theta)$. The proof is completed. ■

4. Efficient Worst-Case Analysis

According to Theorem 2, the exact solution to the worst-case analysis is necessary for the global convergence of FGM. Fortunately, for a number of feature selection tasks, the exact worst-case analysis does exist. For simplicity, hereafter we drop the superscript t from $\boldsymbol{\alpha}^t$.

4.1 Worst-Case Analysis for Linear Feature Selection

The worst-case analysis for the linear feature selection is to solve the following maximization problem:

$$\max_{\mathbf{d}} \quad \frac{1}{2}\left\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\right\|^2, \quad \text{s.t. } \sum_{j=1}^{m} d_j \leq B, \mathbf{0} \preceq \mathbf{d} \preceq \mathbf{1}. \tag{17}$$

This problem in general is very hard to be solved. Recall that $\mathbf{c}(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i \in \mathbb{R}^m$, and we have $\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\|^2 = \|\sum_{i=1}^{n}(\alpha_i y_i \mathbf{x}_i) \odot \sqrt{\mathbf{d}}\|^2 = \sum_{j=1}^{m} c_j(\boldsymbol{\alpha})^2 d_j$. Based on this relation, we define a feature score s_j to measure the importance of features as

$$s_j = [c_j(\boldsymbol{\alpha})]^2.$$

Accordingly, problem (17) can be further formulated as a linear programming problem:

$$\max_{\mathbf{d}} \quad \frac{1}{2}\sum_{j=1}^{m} s_j d_j, \quad \text{s.t.} \quad \sum_{j=1}^{m} d_i \leq B, \quad \mathbf{0} \preceq \mathbf{d} \preceq \mathbf{1}. \tag{18}$$

The optimal solution to this problem can be obtained without any numeric optimization solver. Specifically, we can construct a feasible solution by first finding the B features with the largest feature score s_j, and then setting the corresponding d_j to 1 and the rests to 0. It is easy to verify that such a $\mathbf{d}$ is also an optimal solution to (18). Note that, as long as there are more than B features with $s_j > 0$, we have $||\mathbf{d}||_0 = B$. In other words, in general, $\mathbf{d}$ will include B features into the optimization after each worst-case analysis.

4.2 Worst-Case Analysis for Group Feature Selection

The worst-case analysis for linear feature selection can be easily extended to group feature selection. Suppose that the features are organized into groups by $\mathcal{G} = \{\mathcal{G}_1, ..., \mathcal{G}_p\}$, and there is no overlapping features among groups, namely $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset$, $\forall i \neq j$. To find the most-active groups, we just need to solve the following optimization problem:

$$\max_{\mathbf{d}\in\mathcal{D}} \quad \sum_{j=1}^{p} d_j \left\|\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{i\mathcal{G}_j}\right\|^2 = \max_{\mathbf{d}\in\mathcal{D}} \sum_{j=1}^{p} d_j \mathbf{c}'_{\mathcal{G}_j}\mathbf{c}_{\mathcal{G}_j}, \tag{19}$$

where $\mathbf{c}_{\mathcal{G}_j} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{i\mathcal{G}_j}$ for group $\mathcal{G}_j$. Let $s_j = \mathbf{c}'_{\mathcal{G}_j}\mathbf{c}_{\mathcal{G}_j}$ be the score for group $\mathcal{G}_j$. The optimal solution to (19) can be obtained by first finding the B groups with the largest s_j's, and then setting their d_j's to 1 and the rests to 0. If $|\mathcal{G}_j| = 1$, $\forall j \in \{1, ..., m\}$, problem (19) is reduced to problem (18), where $\mathcal{G} = \{\{1\}, ..., \{p\}\}$ and $s_j = [c_j(\boldsymbol{\alpha})]^2$ for $j \in \mathcal{G}$. In this sense, we unify the worst-case analysis of the two feature selection tasks in Algorithm 2.

4.3 Worst-Case Analysis for Groups with Complex Structures

Algorithm 2 can be also extended to feature groups with overlapping features or with tree-structures. Recall that $p = |\mathcal{G}|$, the worst-case analysis in Algorithm 2 takes $O(mn + p\log(B))$ cost, where the $O(mn)$ cost is for computing $\mathbf{c}$, and the $O(p\log(B))$ cost is for

Algorithm 2 Algorithm for Worst-Case Analysis.

Given $\boldsymbol{\alpha}$, B, the training set $\{\mathbf{x}_i, y_i\}_{i=1}^n$ and the group index set $\mathcal{G} = \{\mathcal{G}_1, ..., \mathcal{G}_p\}$.
1: Calculate $\mathbf{c} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$.
2: Calculate the feature score $\mathbf{s}$, where $s_j = \mathbf{c}'_{\mathcal{G}_j} \mathbf{c}_{\mathcal{G}_j}$.
3: Find the B largest s_j's.
4: Set d_j corresponding to the B largest s_j's to 1 and the rests to 0.
5: Return $\mathbf{d}$.

sorting s_j's. The second term is negligible if $p = O(m)$. However, if p is extremely large, namely $p \gg m$, the computational cost for computing and sorting s_j will be unbearable. For instance, if the feature groups are organized into a graph or a tree structure, p can become very huge, namely $p \gg m$ (Jenatton et al., 2011b).

Since we just need to find the B groups with the largest s_j's, we can address the above computational difficulty by implementing Algorithm 2 in an incremental way. Specifically, we can maintain a cache $\mathbf{c}_B$ to store the indices and scores of the B feature groups with the largest scores among those traversed groups, and then calculate the feature score s_j for each group one by one. After computing s_j for a new group $\mathcal{G}_j$, we update $\mathbf{c}_B$ if $s_j > s_B^{min}$, where s_B^{min} denotes the smallest score in $\mathbf{c}_B$. By applying this technique, the whole computational cost of the worst-case analysis can be greatly reduced to $O(n\log(m)+B\log(p))$ if the groups follow the tree-structure defined in Definition 1.

Remark 2 *Given a set of groups $\mathcal{G} = \{\mathcal{G}_1, ..., \mathcal{G}_p\}$ that is organized as a tree structure in Definition 1, suppose $\mathcal{G}_h \subseteq \mathcal{G}_g$, then $s_h < s_g$. Furthermore, $\mathcal{G}_g$ and all its decedent $\mathcal{G}_h \subseteq \mathcal{G}_g$ will not be selected if $s_g < s_B^{min}$. Therefore, the computational cost of the worst-case analysis can be reduced to $O(n\log(m) + B\log(p))$ for a balanced tree structure.*

5. Efficient Subproblem Optimization

After updating $\mathcal{C}_t$, now we tend to solve the subproblem (16). Recall that, any $\mathbf{d}_h \in \mathcal{C}_t$ indexes a set of features. For convenience, we define $\mathbf{X}_h \triangleq [\mathbf{x}_h^1, ..., \mathbf{x}_h^n]' \in \mathbb{R}^{n\times B}$, where $\mathbf{x}_h^i$ denotes the ith instance with the features indexed by $\mathbf{d}_h$.

5.1 Subproblem Optimization via MKL

Regarding problem (16), let $\mu_h \geq 0$ be the dual variable for each constraint defined by $\mathbf{d}_h$, the Lagrangian function can be written as:

$$\mathcal{L}(\theta, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \theta - \sum_{\mathbf{d}_h \in \mathcal{C}_t} \mu_t \left(\theta - f(\boldsymbol{\alpha}, \mathbf{d}_h)\right).$$

By setting its derivative w.r.t. θ to zero, we have $\sum \mu_t = 1$. Let $\boldsymbol{\mu}$ be the vector of all μ_t's, and $\mathcal{M} = \{\boldsymbol{\mu} | \sum \mu_h = 1, \mu_h \geq 0\}$ be the domain of $\boldsymbol{\mu}$. By applying the minimax saddle-point theorem (Sion, 1958), $\mathcal{L}(\theta, \boldsymbol{\alpha}, \boldsymbol{\mu})$ can be rewritten as:

$$\max_{\boldsymbol{\alpha}\in\mathcal{A}} \min_{\boldsymbol{\mu}\in\mathcal{M}} \; -\sum_{\mathbf{d}_h \in \mathcal{C}_t} \mu_h f(\boldsymbol{\alpha}, \mathbf{d}_h) = \min_{\boldsymbol{\mu}\in\mathcal{M}} \max_{\boldsymbol{\alpha}\in\mathcal{A}} \; -\frac{1}{2}(\boldsymbol{\alpha}\odot\mathbf{y})'\Big(\sum_{\mathbf{d}_h \in \mathcal{C}_t} \mu_h \mathbf{X}_h \mathbf{X}_h^{'} + \frac{1}{C}\mathbf{I}\Big)(\boldsymbol{\alpha}\odot\mathbf{y}), \quad (20)$$

where the equality holds since the objective function is concave in $\boldsymbol{\alpha}$ and convex in $\boldsymbol{\mu}$. Problem (20) is a multiple kernel learning (MKL) problem (Lanckriet et al., 2004; Rakotomamonjy et al., 2008) with $|\mathcal{C}_t|$ base kernel matrices $\mathbf{X}_h\mathbf{X}_h'$. Several existing MKL approaches can be adopted to solve this problem, such as SimpleMKL (Rakotomamonjy et al., 2008). Specifically, SimpleMKL solves the non-smooth optimization problem by applying a sub-gradient method (Rakotomamonjy et al., 2008; Nedic and Ozdaglar, 2009). Unfortunately, it is expensive to calculate the sub-gradient w.r.t. $\boldsymbol{\alpha}$ for large-scale problems. Moreover, the convergence speed of sub-gradient methods is limited. The minimax subproblem (20) can be also solved by the proximal gradient methods (Nemirovski, 2005; Tseng, 2008) or SQP methods (Pee and Royset, 2010) with faster convergence rates. However, these methods involve expensive subproblems, and they are very inefficient when n is large.

Based on the definition of $\mathbf{X}_h$, we have $\sum_{\mathbf{d}_h\in\mathcal{C}_t}\mu_h\mathbf{X}_h\mathbf{X}_h^{'} = \sum_{\mathbf{d}_h\in\mathcal{C}_t}\mu_h\mathbf{X}\mathrm{diag}(\mathbf{d}_h)\mathbf{X}^{'} = \mathbf{X}\mathrm{diag}(\sum_{\mathbf{d}_h\in\mathcal{C}_t}\mu_h\mathbf{d}_h)\mathbf{X}^{'}$ w.r.t. the linear feature selection task. Accordingly, we have

$$\mathbf{d}^* = \sum_{\mathbf{d}_h\in\mathcal{C}_t}\mu_h^*\mathbf{d}_h, \tag{21}$$

where $\boldsymbol{\mu}^* = [\mu_1^*, ..., \mu_h^*]'$ denotes the optimal solution to (20). It is easy to check that, the relation in (21) also holds for the group feature selection tasks. Since $\sum_{h=1}^{|\mathcal{C}_t|}\mu_h^* = 1$, we have $\mathbf{d}^* \in \mathcal{D} = \left\{\mathbf{d}\middle|\sum_{j=1}^m d_j \le B,\ d_j\in[0,1],\ j=1,\cdots,m\right\}$, where the nonzero entries are associated with selected features/groups.

5.2 Subproblem Optimization in the Primal

Solving the MKL problem in (20) is very expensive when n is very large. In other words, the dimension of the optimization variable $\boldsymbol{\alpha}$ in (20) is very large. Recall that, after t iterations, $\mathcal{C}_t$ includes at most tB features, where $tB \ll n$. Motivated by this observation, we propose to solve it in the primal form w.r.t. $\mathbf{w}$. Apparently, the dimension of the optimization variable $\mathbf{w}$ is much smaller than $\boldsymbol{\alpha}$, namely $||\mathbf{w}||_0 \le tB \ll n$.

Without loss of generality, we assume that $t = |\mathcal{C}_t|$ after tth iterations. Let $\mathbf{X}_h \in \mathbb{R}^{n\times B}$ denote the data with features indexed by $\mathbf{d}_h \in \mathcal{C}_t$, $\boldsymbol{\omega}_h \in \mathbb{R}^B$ denote the weight vector w.r.t. $\mathbf{X}_h$, $\boldsymbol{\omega} = [\boldsymbol{\omega}_1', ..., \boldsymbol{\omega}_t']' \in \mathbb{R}^{tB}$ be a supervector concatenating all $\boldsymbol{\omega}_h$'s, where $tB \ll n$. For convenience, we define

$$P(\boldsymbol{\omega}, b) = \frac{C}{2}\sum_{i=1}^n \xi_i^2$$

w.r.t. the squared hinge loss, where $\xi_i = \max(1 - y_i(\sum_h \boldsymbol{\omega}_h'\mathbf{x}_{ih} - b), 0)$, and

$$P(\boldsymbol{\omega}, b) = C\sum_{i=1}^n \log(1 + \exp(\xi_i)),$$

w.r.t. the logistic loss, where $\xi_i = -y_i(\sum_{h=1}^t \boldsymbol{\omega}_h'\mathbf{x}_{ih} - b)$.

Theorem 3 *Let $\mathbf{x}_{ih}$ denote the ith instance of $\mathbf{X}_h$, the MKL subproblem (20) can be equivalently addressed by solving an $\ell_{2,1}^2$-regularized problem:*

$$\min_{\boldsymbol{\omega}} \ \frac{1}{2}\left(\sum_{h=1}^t \|\boldsymbol{\omega}_h\|\right)^2 + P(\boldsymbol{\omega}, b). \tag{22}$$

Furthermore, the dual optimal solution $\boldsymbol{\alpha}^*$ can be recovered from the optimal solution ξ^*. To be more specific, $\alpha_i^* = C\xi_i^*$ holds for the square-hinge loss and $\alpha_i = \frac{C\exp(\xi_i^*)}{1+\exp(\xi_i^*)}$ holds for the logistic loss.

The proof can be found in Appendix A.

According to Theorem 3, rather than directly solving (20), we can address its primal form (22) instead, which brings great advantages for the efficient optimization. Moreover, we can recover $\boldsymbol{\alpha}^*$ by $\alpha_i^* = C\xi_i^*$ and $\alpha_i = \frac{C\exp(\xi_i^*)}{1+\exp(\xi_i^*)}$ w.r.t. the squared hinge loss and logistic loss, respectively.[2] For convenience, we define

$$F(\boldsymbol{\omega}, b) = \Omega(\boldsymbol{\omega}) + P(\boldsymbol{\omega}, b),$$

where $\Omega(\boldsymbol{\omega}) = \frac{1}{2}(\sum_{h=1}^{t} \|\boldsymbol{\omega}_h\|)^2$. $F(\boldsymbol{\omega}, b)$ is a non-smooth function w.r.t $\boldsymbol{\omega}$, and $P(\boldsymbol{\omega}, b)$ has block coordinate Lipschitz gradient w.r.t $\boldsymbol{\omega}$ and b, where $\boldsymbol{\omega}$ is deemed as a block variable. Correspondingly, let $\nabla P(\mathbf{v}) = \partial_{\mathbf{v}} P(\mathbf{v}, v_b)$ and $\nabla_b P(\mathbf{v}, v_b) = \partial_b P(\mathbf{v}, v_b)$. It is known that $F(\boldsymbol{\omega}, b)$ is at least Lipschitz continuous for both logistic loss and squared hinge loss (Yuan et al., 2011):

$$P(\boldsymbol{\omega}, b) \leq P(\mathbf{v}, v_b) + \langle \nabla P(\mathbf{v}), \boldsymbol{\omega} - \mathbf{v} \rangle + \langle \nabla_b P(v_b), b - v_b \rangle + \frac{L}{2}\|\boldsymbol{\omega} - \mathbf{v}\|^2 + \frac{L_b}{2}\|b - v_b\|^2,$$

where L and L_b denote the Lipschitz constants regarding $\boldsymbol{\omega}$ and b, respectively.

Since $F(\boldsymbol{\omega}, b)$ is separable w.r.t $\boldsymbol{\omega}$ and b, we can minimize it regarding $\boldsymbol{\omega}$ and b in a block coordinate descent manner (Tseng, 2001). For each block variable, we update it through an accelerated proximal gradient (APG) method (Beck and Teboulle, 2009; Toh and Yun, 2009), which iteratively minimizes a quadratic approximation to $F(\boldsymbol{\omega}, b)$. Specifically, given a point $[\mathbf{v}', v_b]'$, the quadratic approximation to $F(\boldsymbol{\omega}, b)$ at this point w.r.t. $\boldsymbol{\omega}$ is:

$$\begin{aligned} Q_\tau(\boldsymbol{\omega}, \mathbf{v}, v_b) &= P(\mathbf{v}, v_b) + \langle \nabla P(\mathbf{v}), \boldsymbol{\omega} - \mathbf{v} \rangle + \Omega(\boldsymbol{\omega}) + \frac{\tau}{2}\|\boldsymbol{\omega} - \mathbf{v}\|^2 \\ &= \frac{\tau}{2}\|\boldsymbol{\omega} - \mathbf{u}\|^2 + \Omega(\boldsymbol{\omega}) + P(\mathbf{v}, v_b) - \frac{1}{2\tau}\|\nabla P(\mathbf{v})\|^2, \end{aligned} \tag{23}$$

where τ is a positive constant and $\mathbf{u} = \mathbf{v} - \frac{1}{\tau}\nabla P(\mathbf{v})$. To minimize $Q_\tau(\boldsymbol{\omega}, \mathbf{v}, v_b)$ w.r.t. $\boldsymbol{\omega}$, it is reduced to solve the following Moreau projection problem (Martins et al., 2010):

$$\min_{\boldsymbol{\omega}} \quad \frac{\tau}{2}\|\boldsymbol{\omega} - \mathbf{u}\|^2 + \Omega(\boldsymbol{\omega}). \tag{24}$$

For convenience, let $\mathbf{u}_h$ be the corresponding component to $\boldsymbol{\omega}_h$, namely $\mathbf{u} = [\mathbf{u}_1', ..., \mathbf{u}_t']'$. Martins et al. (2010) has shown that, problem (24) has a unique closed-form solution, which is summarized in the following proposition.

Proposition 1 *Let $S_\tau(\mathbf{u}, \mathbf{v})$ be the optimal solution to problem (24) at point $\mathbf{v}$, then $S_\tau(\mathbf{u}, \mathbf{v})$ is unique and can be calculated as follows:*

$$[S_\tau(\mathbf{u}, \mathbf{v})]_h = \begin{cases} \frac{o_h}{\|\mathbf{u}_h\|}\mathbf{u}_h, & \text{if } o_h > 0, \\ \mathbf{0}, & \text{otherwise}, \end{cases} \tag{25}$$

2. Here the optimal dual variable $\boldsymbol{\alpha}^*$ is required in the worst-case analysis.

where $[S_\tau(\mathbf{u},\mathbf{v})]_h \in \mathbb{R}^B$ *denote the corresponding component w.r.t.* $\mathbf{u}_h$ *and* $\mathbf{o} \in \mathbb{R}^t$ *be an intermediate variable. Let* $\widehat{\mathbf{o}} = [\|\mathbf{u}_1\|, ..., \|\mathbf{u}_t\|]' \in \mathbb{R}^t$*, the intermediate vector* $\mathbf{o}$ *can be calculated via a soft-threshold operator:* $o_h = [soft(\widehat{\mathbf{o}}, \varsigma)]_h = \begin{cases} \widehat{o}_h - \varsigma, & if \quad \widehat{o}_h > \varsigma, \\ 0, & Otherwise. \end{cases}$ *. Here the threshold value* ς *can be calculated in Step 4 of Algorithm 3.*

Proof The proof can be adapted from the results in Appendix F in (Martins et al., 2010). ■

Algorithm 3 Moreau Projection $S_\tau(\mathbf{u},\mathbf{v})$.

Given an point $\mathbf{v}$, $s = \frac{1}{\tau}$ and the number of kernels t.
1: Calculate $\widehat{o}_h = \|\mathbf{g}_h\|$ for all $h = 1, ..., t$.
2: Sort $\widehat{\mathbf{o}}$ to obtain $\bar{\mathbf{o}}$ such that $\bar{o}_{(1)} \geq ... \geq \bar{o}_{(t)}$.
3: Find $\rho = \max\left\{t \middle| \bar{o}_h - \frac{s}{1+hs}\sum_{i=1}^{h}\bar{o}_i > 0, h = 1, ..., t\right\}$.
4: Calculate the threshold value $\varsigma = \frac{s}{1+\rho s}\sum_{i=1}^{\rho}\bar{o}_i$.
5: Compute $\mathbf{o} = soft(\widehat{\mathbf{o}}, \varsigma)$.
6: Compute and output $S_\tau(\mathbf{u},\mathbf{v})$ via equation (25).

Remark 3 *For the Moreau projection in Algorithm 3, the sorting takes* $O(t)$ *cost. In FGM setting,* t *in general is very small, thus the Moreau projection can be efficiently computed.*

Now we tend to minimize $F(\boldsymbol{\omega}, b)$ regarding b. Since there is no regularizer on b, it is equivalent to minimize $P(\boldsymbol{\omega}, v_b)$ w.r.t. b. The updating can be done by $b = v_b - \frac{1}{\tau_b}\nabla_b P(\mathbf{v}, v_b)$, which is essentially the steepest descent update. We can use the Armijo line search (Nocedal and Wright, 2006) to find a step size $\frac{1}{\tau_b}$ such that,

$$P(\boldsymbol{\omega}, b) \leq P(\boldsymbol{\omega}, v_b) - \frac{1}{2\tau_b}|\nabla_b P(\mathbf{v}, v_b)|^2,$$

where $\boldsymbol{\omega}$ is the minimizer to $Q_\tau(\boldsymbol{\omega}, \mathbf{v}, v_b)$. This line search can be efficiently performed since it is conducted on a single variable only.

With the calculation of $S_\tau(\mathbf{g})$ in Algorithm 3 and the updating rule of b above, we propose to solve (22) through a modified APG method in a block coordinate manner in Algorithm 4. In Algorithm 4, L_t and L_{bt} denote the Lipschitz constants of $P(\boldsymbol{\omega}, b)$ w.r.t. $\boldsymbol{\omega}$ and b at the t iteration of Algorithm 1, respectively. In practice, we estimate L_0 by $L_0 = 0.01nC$, which will be further adjusted by the line search. When $t > 0$, L_t is estimated by $L_t = \eta L_{t-1}$. Finally, a sublinear convergence rate of Algorithm 4 is guaranteed.[3]

Theorem 4 *Let* L_t *and* L_{bt} *be the Lipschitz constant of* $P(\boldsymbol{\omega}, b)$ *w.r.t.* $\boldsymbol{\omega}$ *and* b *respectively. Let* $\{(\boldsymbol{\omega}^{k'}, b^k)'\}$ *be the sequences generated by Algorithm 4 and* $L = \max(L_{bt}, L_t)$*, for any* $k \geq 1$*, we have:*

$$F(\boldsymbol{\omega}^k, b^k) - F(\boldsymbol{\omega}^*, b^*) \leq \frac{2L_t||\boldsymbol{\omega}^0 - \boldsymbol{\omega}^*||^2}{\eta(k+1)^2} + \frac{2L_{bt}(b^0 - b^*)^2}{\eta(k+1)^2} \leq \frac{2L||\boldsymbol{\omega}^0 - \boldsymbol{\omega}^*||^2}{\eta(k+1)^2} + \frac{2L(b^0 - b^*)^2}{\eta(k+1)^2}.$$

3. Regarding Algorithm 4, a linear convergence rate can be attained w.r.t. the logistic loss under mild conditions. The details can be found in Appendix C.

The proof can be found in Appendix B. The internal variables L^k and L_b^k in Algorithm 3 are useful in the proof of the convergence rate.

According to Theorem 4, if L_{bt} is very different from L_t, the block coordinate updating scheme in Algorithm 4 can achieve an improved convergence speed over the batch updating w.r.t. $(\boldsymbol{\omega}', b)'$. Moreover, the warm-start for initialization of $\boldsymbol{\omega}$ and b in Algorithm 4 is useful to accelerate the convergence speed.

Algorithm 4 Accelerated Proximal Gradient for Solving Problem (22) (**Inner Iterations**).

Initialization: Initialize the Lipschitz constant $L_t = L_{t-1}$, set $\boldsymbol{\omega}^0 = \mathbf{v}^1 = [\boldsymbol{\omega}_{t-1}', \mathbf{0}']'$ and $b^0 = v_b^1 = b_{t-1}$ by **warm start**, $\tau_0 = L_t$, $\eta \in (0,1)$, parameter $\varrho^1 = 1$ and $k = 1$.

1: Set $\tau = \eta\tau_{k-1}$.
  For $j = 0, 1, ...,$
    Set $\mathbf{u} = \mathbf{v}^k - \frac{1}{\tau}\nabla p(\mathbf{v}^k)$, compute $S_\tau(\mathbf{u}, \mathbf{v}^k)$.
    If $F(S_\tau(\mathbf{u}, \mathbf{v}^k), v_b^k) \leq Q(S_\tau(\mathbf{u}, \mathbf{v}^k), \mathbf{v}^k, v_b^k)$,
      Set $\tau_k = \tau$, stop;
    Else
      $\tau = \min\{\eta^{-1}\tau, L_t\}$.
    End
  End
2: Set $\boldsymbol{\omega}^k = S_{\tau_k}(\mathbf{u}, \mathbf{v}^k)$ and $L^k = \tau_k$.
3: Set $\tau_b = \eta\tau_k$.
  For $j = 0, 1, ...$
    Set $b = v_b^k - \frac{1}{\tau_b}\nabla_b P(\mathbf{v}, v_b^k)$.
    If $P(\boldsymbol{\omega}^k, b) \leq P(\boldsymbol{\omega}^k, v_b^k) - \frac{1}{2\tau_b}|\nabla_b P(\mathbf{v}, v_b^k)|^2$,
      Stop;
    Else
      $\tau_b = \min\{\eta^{-1}\tau_b, L_t\}$.
    End
  End
4: Set $b^k = b$ and $L_b^k = \tau_b$. Go to Step 8 if the stopping condition achieves.
5: Set $\varrho^{k+1} = \frac{1+\sqrt{1+4(\varrho^k)^2}}{2}$.
6: Set $\mathbf{v}^{k+1} = \boldsymbol{\omega}^k + \frac{\varrho^k - 1}{\varrho^{k+1}}(\boldsymbol{\omega}^k - \boldsymbol{\omega}^{k-1})$ and $v_b^{k+1} = b^k + \frac{\varrho^k - 1}{\varrho^{k+1}}(b^k - b^{k-1})$.
7: Let $k = k + 1$ and go to step 1.
8: Return and output $\boldsymbol{\omega}_t = \boldsymbol{\omega}^k$, $b_t = b^k$ and $L_t = \eta\tau_k$.

Warm Start: From Theorem 4, the number of iterations needed by APG to achieve an ϵ-solution is $O(\frac{\|\boldsymbol{\omega}^0 - \boldsymbol{\omega}^*\|}{\sqrt{\epsilon}})$. Since FGM incrementally includes a set of features into the subproblem optimization, an warm start of $\boldsymbol{\omega}^0$ can be very useful to improve its efficiency. To be more specific, when a new active constraint is added, we can use the optimal solution of the last iteration (denoted by $[\boldsymbol{\omega}_1^{*\prime}, ..., \boldsymbol{\omega}_{t-1}^{*\prime}]$) as an initial guess to the next iteration. In other words, at the tth iteration, we use $\boldsymbol{\omega}^{-1} = \boldsymbol{\omega}^0 = [\boldsymbol{\omega}_1^{*\prime}, ..., \boldsymbol{\omega}_{t-1}^{*\prime}, \mathbf{0}']'$ as the starting point.

5.3 De-biasing of FGM

Based on Algorithm 4, we show that FGM resembles the *re-training* process and can achieve de-biased solutions. For convenience, we first revisit the de-biasing process in the ℓ_1-minimization (Figueiredo et al., 2007).

De-biasing for ℓ_1-methods. To reduce the solution bias, a de-biasing process is often adopted in ℓ_1-methods. For example, in the sparse recovery problem (Figueiredo et al., 2007), after solving the ℓ_1-regularized problem, a **least-square problem** (which drops the ℓ_1-regularizer) is solved with the detected features (or supports). To reduce the feature selection bias, one can also apply this de-biasing technique to the ℓ_1-SVM for classification tasks. However, it is worth mentioning that, when dealing with classification tasks, due to the label noises, such as the rounding errors of labels, a regularizer is necessary and important to avoid the over-fitting issue. Alternatively, we can apply the standard SVM on the selected features to do the de-biasing using a relative large C, which is also referred to as the *re-training*. When C goes to infinity, it is equivalent to minimize the empirical loss without any regularizer, which, however, may cause the over-fitting problem.

De-biasing effect of FGM. Recall that, in FGM, the parameters B and the trade-off parameter C are adjusted separately. In the worst-case analysis, FGM includes B features/groups that violate the optimality condition the most. When B is sufficiently small, the selected B features/groups can be regarded as the most relevant features. After that, FGM addresses the $\ell_{2,1}^2$-regularized problem (22) w.r.t. the selected features only, which mimics the above re-training strategy for de-biasing. Specifically, we can use a relatively large C to penalize the empirical loss to reduce the solution bias. Accordingly, with a suitable C, each outer iteration of FGM can be deemed as the de-biasing process, and the de-biased solution will in turn help the worst-case analysis to select more discriminative features.

5.4 Stopping Conditions

Suitable stopping conditions of FGM are important to reduce the risk of over-fitting and improve the training efficiency. The stopping criteria of FGM include 1) the stopping conditions for the outer cutting plane iterations in Algorithm 1; 2) the stopping conditions for the inner APG iterations in Algorithm 4.

5.4.1 Stopping Conditions for Outer iterations

We first introduce the stopping conditions w.r.t. the outer iterations in Algorithm 1. Recall that the optimality condition for the SIP problem is $\sum_{\mathbf{d}_t \in \mathcal{D}} \mu_t \nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}, \mathbf{d}_t) = \mathbf{0}$ and $\mu_t(f(\boldsymbol{\alpha}, \mathbf{d}_t) - f_m(\boldsymbol{\alpha})) = 0, \forall \mathbf{d}_t \in \mathcal{D}$. A direct stopping condition can be written as:

$$f(\boldsymbol{\alpha}, \mathbf{d}) \leq f_{\mathrm{m}}(\boldsymbol{\alpha}) + \epsilon, \quad \forall \mathbf{d} \in \mathcal{D}, \tag{26}$$

where $f_{\mathrm{m}}(\boldsymbol{\alpha}) = \max_{\mathbf{d}_h \in \mathcal{C}_t} f(\boldsymbol{\alpha}, \mathbf{d}_h)$ and ϵ is a small tolerance value. To check this condition, we just need to find a new $\mathbf{d}_{t+1}$ via the worst-case analysis. If $f(\boldsymbol{\alpha}, \mathbf{d}_{t+1}) \leq f_m(\boldsymbol{\alpha}) + \epsilon$, the stopping condition in (26) is achieved. In practice, due to the scale variation of $f_{\mathrm{m}}(\boldsymbol{\alpha})$ for different problems, it is non-trivial to set the tolerance ϵ. Since we perform the subproblem optimization in the primal, and the objective value $F(\boldsymbol{\omega}_t)$ monotonically

decreases. Therefore, in this paper, we propose to use the relative function value difference as the stopping condition instead:

$$\frac{F(\boldsymbol{\omega}_{t-1}, b) - F(\boldsymbol{\omega}_t, b)}{F(\boldsymbol{\omega}_0, b)} \leq \epsilon_c, \tag{27}$$

where ϵ_c is a small tolerance value. In some applications, one may need to select a desired number of features. In such cases, we can terminate Algorithm 1 after a maximum number of T iterations with at most TB features being selected.

5.4.2 Stopping Conditions for Inner iterations

Exact and Inexact FGM: In each iteration of Algorithm 1, one needs to do the inner master problem minimization in (22). The optimality condition of (22) is $\nabla_{\boldsymbol{\omega}} F(\boldsymbol{\omega}) = \mathbf{0}$. In practice, to achieve a solution with high precision to meet this condition is expensive. Therefore, we usually achieve an ϵ-accurate solution instead.

Nevertheless, an inaccurate solution may affect the convergence. To demonstrate this, let $\widehat{\boldsymbol{\omega}}$ and $\widehat{\boldsymbol{\xi}}$ be the exact solution to (22). According to Theorem 3, the exact solution of $\widehat{\boldsymbol{\alpha}}$ to (20) can be obtained by $\widehat{\boldsymbol{\alpha}} = \widehat{\boldsymbol{\xi}}$. Now suppose $\boldsymbol{\omega}$ is an ϵ-accurate solution to (22) and $\boldsymbol{\xi}$ be the corresponding loss, then we have $\alpha_i = \widehat{\alpha}_i + \epsilon_i$, where ϵ_i is the gap between $\widehat{\boldsymbol{\alpha}}$ and $\boldsymbol{\alpha}$. When performing the worst-case analysis in Algorithm 2, we need to calculate

$$\mathbf{c} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \sum_{i=1}^{n} (\widehat{\alpha}_i + \epsilon_i) y_i \mathbf{x}_i = \widehat{\mathbf{c}} + \sum_{i=1}^{n} \epsilon_i y_i \mathbf{x}_i = \widehat{\mathbf{c}} + \Delta\widehat{\mathbf{c}},$$

where $\widehat{\mathbf{c}}$ denotes the exact feature score w.r.t. $\widehat{\boldsymbol{\alpha}}$, and $\Delta\widehat{\mathbf{c}}$ denotes the error of $\mathbf{c}$ brought by the inexact solution. Apparently, we have

$$|\widehat{c}_j - c_j| = |\Delta\widehat{c}_j| = O(\epsilon), \ \forall j = 1, ..., m.$$

Since we only need to find those significant features with the largest $|c_j|'s$, a sufficiently small ϵ is enough such that we can find the most-active constraint. Therefore, the convergence of FGM will not be affected if ϵ is sufficiently small, but overall convergence speed of FGM can be greatly improved. Let $\{\boldsymbol{\omega}^k\}$ be the inner iteration sequence, in this paper, we set the stopping condition of the inner problem as

$$\frac{F(\boldsymbol{\omega}^{k-1}) - F(\boldsymbol{\omega}^k)}{F(\boldsymbol{\omega}^{k-1})} \leq \epsilon_{in}, \tag{28}$$

where ϵ_{in} is a small tolerance value. In practice, we set $\epsilon_{in} = 0.001$, which works well for the problems that will be studied in this paper.

5.5 Cache for Efficient Implementations

The optimization scheme of FGM allows to use some cache techniques to improve the optimization efficiency.

Cache for features. Different from the cache used in kernel SVM which caches kernel entries (Fan et al., 2005), we directly cache the features in FGM. In gradient-based methods,

one needs to calculate $\mathbf{w}'\mathbf{x}_i$ for each instance to compute the gradient of the loss function, which takes $O(mn)$ cost in general. Unlike these methods, the gradient computation in the modified APG algorithm of FGM is w.r.t. the selected features only. Therefore, we can use a column-based database to store the data, and cache these features in the main memory to accelerate the feature retrieval. To cache these features, we needs $O(tBn)$ additional memory. However, the operation complexity for feature retrieval can be significantly reduced from $O(nm)$ to $O(tBn)$, where $tB \ll m$ for high dimensional problems. It is worth mentioning that, the cache for features is particularly important for the nonlinear feature selection with explicit feature mappings, where the data with expanded features can be too large to be loaded into the main memory.

Cache for inner products. The cache technique can be also used to accelerate the Algorithm 4. To make a sufficient decrease of the objective value, in Algorithm 4, a line search is performed to find a suitable step size. When doing the line search, one may need to calculate the loss function $P(\boldsymbol{\omega})$ many times, where $\boldsymbol{\omega} = S_\tau(\mathbf{g}) = [\boldsymbol{\omega}_1', ..., \boldsymbol{\omega}_t']'$. The computational cost will be very high if n is very large. However, according to equation (25), we have

$$\boldsymbol{\omega} = S_\tau(\mathbf{g}_h) = \frac{o_h}{||\mathbf{g}_h||}\mathbf{g}_h = \frac{o_h}{||\mathbf{g}_h||}(\mathbf{v}_h - \frac{1}{\tau}\nabla P(\mathbf{v}_h)),$$

where only o_h is affected by the step size. Then the calculation of $\sum_{i=1}^n \boldsymbol{\omega}'\mathbf{x}_i$ follows

$$\sum_{i=1}^n \boldsymbol{\omega}'\mathbf{x}_i = \sum_{i=1}^n \left(\sum_{h=1}^t \boldsymbol{\omega}_h'\mathbf{x}_{ih}\right) = \sum_{i=1}^n \left(\sum_{h=1}^t \frac{o_h}{||\mathbf{g}_h||}\left(\boxed{\mathbf{v}_h'\mathbf{x}_{ih}} - \frac{1}{\tau}\boxed{\nabla P(\mathbf{v}_h)'\mathbf{x}_{ih}}\right)\right).$$

According to the above calculation rule, we can make a fast computation of $\sum_{i=1}^n \boldsymbol{\omega}'\mathbf{x}_i$ by caching $\mathbf{v}_h'\mathbf{x}_{ih}$ and $\nabla P(\mathbf{v}_h)'\mathbf{x}_{ih}$ for the h^{th} group of each instance $\mathbf{x}_i$. Accordingly, the complexity of computing $\sum_{i=1}^n \boldsymbol{\omega}'\mathbf{x}_i$ can be reduced from $O(ntB)$ to $O(nt)$. That is to say, no matter how many line search steps will be conducted, we only need to scan the selected features once, which can greatly reduce the computational cost.

6. Nonlinear Feature Selection Through Kernels

By applying the kernel tricks, we can extend FGM to do nonlinear feature selections. Let $\boldsymbol{\phi}(\mathbf{x})$ be a nonlinear feature mapping that maps the input features with nonlinear relations into a high-dimensional linear feature space. To select the features, we can also introduce a scaling vector $\mathbf{d} \in \mathcal{D}$ and obtain a new feature mapping $\boldsymbol{\phi}(\mathbf{x} \odot \sqrt{\mathbf{d}})$. By replacing $(\mathbf{x} \odot \sqrt{\mathbf{d}})$ in (5) with $\boldsymbol{\phi}(\mathbf{x} \odot \sqrt{\mathbf{d}})$, the kernel version of FGM can be formulated as the following semi-infinite kernel (SIK) learning problem:

$$\min_{\boldsymbol{\alpha}\in\mathcal{A},\theta} \theta \quad : \quad \theta \geq f_{\mathbf{K}}(\boldsymbol{\alpha}, \mathbf{d}), \quad \forall\ \mathbf{d} \in \mathcal{D},$$

where $f_{\mathbf{K}}(\boldsymbol{\alpha}, \mathbf{d}) = \frac{1}{2}(\boldsymbol{\alpha}\odot\mathbf{y})'(\mathbf{K}_{\mathbf{d}} + \frac{1}{C}I)(\boldsymbol{\alpha}\odot\mathbf{y})$ and $\mathbf{K}_{\mathbf{d}}^{ij}$ is calculated as $\boldsymbol{\phi}(\mathbf{x}_i\odot\sqrt{\mathbf{d}})'\boldsymbol{\phi}(\mathbf{x}_j\odot\sqrt{\mathbf{d}})$. This problem can be solved by Algorithm 1. However, we need to solve the following optimization problem in the worst-case analysis:

$$\max_{\mathbf{d}\in\mathcal{D}} \ \frac{1}{2}\left\|\sum_{i=1}^n \boldsymbol{\alpha}_i y_i \boldsymbol{\phi}(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\right\|^2 = \max_{\mathbf{d}\in\mathcal{D}} \ \frac{1}{2}(\boldsymbol{\alpha}\odot\mathbf{y})'\mathbf{K}_{\mathbf{d}}(\boldsymbol{\alpha}\odot\mathbf{y}), \tag{29}$$

6.1 Worst Case Analysis for Additive Kernels

In general, solving problem (29) for general kernels (*e.g.*, Gaussian kernels) is very challenging. However, for **additive kernels**, this problem can be exactly solved. A kernel $\mathbf{K_d}$ is an additive kernel if it can be linearly represented by a set of base kernels $\{\mathbf{K}_j\}_{j=1}^p$ (Maji and Berg, 2009). If each base kernel $\mathbf{K}_j$ is constructed by one feature or a subset of features, we can select the optimal subset features by choosing a small subset of kernels.

Proposition 2 *The worst-case analysis w.r.t. additive kernels can be exactly solved.*

Proof Suppose that each base kernel $\mathbf{K}_j$ in an additive kernel is constructed by one feature or a subset of features. Let $\mathcal{G} = \{\mathcal{G}_1, ..., \mathcal{G}_p\}$ be the index set of features that produce the base kernel set $\{\mathbf{K}_j\}_{j=1}^p$ and $\boldsymbol{\phi}_j(\mathbf{x}_{i\mathcal{G}_j})$ be the corresponding feature map to $\mathcal{G}_j$. Similar to the group feature selection, we introduce a feature scaling vector $\mathbf{d} \in \mathcal{D} \subset \mathbb{R}^p$ to scale $\boldsymbol{\phi}_j(\mathbf{x}_{i\mathcal{G}_j})$. The resultant model becomes:

$$\begin{aligned} \min_{\mathbf{d}\in\widehat{\mathcal{D}}} \min_{\mathbf{w},\boldsymbol{\xi},b} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^n \xi_i^2 \\ \text{s.t.} \quad & y_i\left(\sum_{j=1}^p \sqrt{d_j}\mathbf{w}_{\mathcal{G}_j}'\boldsymbol{\phi}_j(\mathbf{x}_{i\mathcal{G}_j}) - b\right) \geq 1-\xi_i, \;\; \xi_i \geq 0, \;\; i = 1, \cdots, n, \end{aligned}$$

where $\mathbf{w}_{\mathcal{G}_j}$ has the same dimensionality with $\boldsymbol{\phi}_j(\mathbf{x}_{i\mathcal{G}_j})$. By transforming this problem to the SIP problem in (13), we can solve the kernel learning (selection) problem via FGM. The corresponding worst-case analysis is reduced to solve the following problem:

$$\max_{\mathbf{d}\in\mathcal{D}} \;\; \sum_{j=1}^p d_j(\boldsymbol{\alpha}\odot\mathbf{y})'\mathbf{K}_j(\boldsymbol{\alpha}\odot\mathbf{y}) = \max_{\mathbf{d}\in\mathcal{D}} \;\; \sum_{j=1}^p d_j s_j,$$

where $s_j = (\boldsymbol{\alpha}\odot\mathbf{y})'\mathbf{K}_j(\boldsymbol{\alpha}\odot\mathbf{y})$ and $\mathbf{K}_j^{i,k} = \boldsymbol{\phi}_j(\mathbf{x}_{i\mathcal{G}_j})'\boldsymbol{\phi}_j(\mathbf{x}_{k\mathcal{G}_j})$. This problem can be exactly solved by choosing the B kernels with the largest s_j's. ■

In the past decades, many additive kernels have been proposed based on specific application contexts, such as the general intersection kernel in computer vision (Maji and Berg, 2009), string kernel in text mining and ANOVA kernels (Bach, 2009). Taking the general intersection kernel for example, it is defined as: $k(\mathbf{x}, \mathbf{z}, a) = \sum_{j=1}^p \min\{|x_j|^a, |z_j|^a\}$, where $a > 0$ is a kernel parameter. When $a = 1$, it reduces to the well-known Histogram Intersection Kernel (HIK), which has been widely used in computer vision and text classifications (Maji and Berg, 2009; Wu, 2012).

It is worth mentioning that, even though we can exactly solve the worst-case analysis for additive kernels, the subproblem optimization is still very challenging for large-scale problems because of two reasons. Firstly, storing the kernel matrices takes $O(n^2)$ space complexity, which is unbearable when n is very large. Secondly, solving the MKL problem with many training points is still computationally expensive. To address these issues, we propose to a group of approximated features, such as the random features (Vedaldi and Zisserman, 2010) and the HIK expanded features (Wu, 2012), to approximate a base kernel. As a result, the MKL problem is reduced to the **group feature selection** problem. Therefore, it is scalable to big data by avoiding the storage the base kernel matrices. Moreover, the subproblem optimization can be more efficiently solved in the primal form.

6.2 Worst-Case Analysis for Ultrahigh Dimensional Big Data

Ultrahigh dimensional big data widely exist in many application contexts. Particularly, in the nonlinear classification tasks with explicit nonlinear feature mappings, the dimensionality of the feature space can be ultrahigh. If the explicit feature mapping is available, the nontrivial nonlinear feature selection task can be cast as a linear feature selection problem in the high-dimensional feature space.

Taking the polynomial kernel $k(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i' \mathbf{x}_j + r)^\upsilon$ for example, the dimension of the feature mapping exponentially increases with υ (Chang et al., 2010), where υ is referred to as the degree. When $\upsilon = 2$, the 2-degree explicit feature mapping can be expressed as

$$\boldsymbol{\phi}(\mathbf{x}) = [r, \sqrt{2\gamma r}x_1, ..., \sqrt{2\gamma r}x_m, \gamma x_1^2, ..., \gamma x_m^2, \sqrt{2}\gamma x_1 x_2, ..., \sqrt{2}\gamma x_{m-1} x_m].$$

The second-order feature mapping can capture the feature pair dependencies, thus it has been widely applied in many applications such as text mining and natural language processing (Chang et al., 2010). Unfortunately, the dimensionality of the feature space is $(m+2)(m+1)/2$ and can be ultrahigh for a median m. For example, if $m = 10^6$, the dimensionality of the feature space is $O(10^{12})$, and around 1 TB memory is required to store the weight vector $\mathbf{w}$. As a result, most of the existing methods are not applicable (Chang et al., 2010). Fortunately, this computational bottleneck can be effectively avoided by FGM since only tB features are required to be stored in the main memory. For convenience, we store the indices and scores of the selected tB features in a structured array $\mathbf{c}_B$.

Algorithm 5 Incremental Implementation of Algorithm 2 for Ultrahigh Dimensional Data.

Given $\boldsymbol{\alpha}$, B, number of data groups k, feature mapping $\boldsymbol{\phi}(\mathbf{x})$ and a structured array $\mathbf{c}_B$.

1: Split $\mathbf{X}$ into k subgroups $\mathbf{X} = [\mathbf{X}^1, ..., \mathbf{X}^k]$.

2: For $j = 1, ..., k$.

  Calculate the feature score $\mathbf{s}$ w.r.t. $\mathbf{X}^j$ according to $\boldsymbol{\phi}(\mathbf{x}_i)$.

  Sort $\mathbf{s}$ and update $\mathbf{c}_B$.

  For $i = j+1, ..., k$. (**Optional**)

    Calculate the cross feature score $\mathbf{s}$ w.r.t. $\mathbf{X}^j$ and $\mathbf{X}^i$.

    Sort $\mathbf{s}$ and update $\mathbf{c}_B$.

  End

End

3: Return $\mathbf{c}_B$.

For ultrahigh dimensional big data, it can be too huge to be loaded into the main memory, thus the worst-case analysis is still very challenging to be addressed. Motivated by the incremental worst-case analysis for complex group feature selection in Section 4.3, we propose to address the big data challenge in an incremental manner. The general scheme for the incremental implementation is presented in Algorithm 5. Particularly, we partition $\mathbf{X}$ into k small data subset of lower dimensionality as $\mathbf{X} = [\mathbf{X}^1, ..., \mathbf{X}^k]$. For each small data subset, we can load it into memory and calculate the feature scores of the features. In Algorithm 5, the inner loop w.r.t. the iteration index i is only used for the second-order feature selection, where the calculation of feature score for the **cross-features** is required. For instance, in the nonlinear feature selection using the 2-degree polynomial mapping, we need to calculate the feature score of $x_i x_j$.

7. Connections to Related Studies

In this section, we discuss the connections of proposed methods with related studies, such as the ℓ_1-regularization (Jenatton et al., 2011a), active set methods (Roth and Fischer, 2008; Bach, 2009), SimpleMKL (Rakotomamonjy et al., 2008), ℓ_q-MKL (Kloft et al., 2009, 2011; Kloft and Blanchard, 2012), infinite kernel learning (IKL) (Gehler and Nowozin, 2008), SMO-MKL (Vishwanathan et al., 2010), and so on.

7.1 Relation to ℓ_1-regularization

Recall that the ℓ_1-norm of a vector $\mathbf{w}$ can be expressed as a variational form (Jenatton et al., 2011a):

$$\|\mathbf{w}\|_1 = \sum_{j=1}^{m} |w_j| = \frac{1}{2} \min_{\mathbf{d} \succeq 0} \sum_{j=1}^{m} \frac{w_j^2}{d_j} + d_j. \tag{30}$$

It is not difficult to verify that, $d_j^* = |w_j|$ holds at the optimum, which indicates that the scale of d_j^* is proportional to $|w_j|$. Therefore, it is meaningless to impose an additional ℓ_1-constraint $||\mathbf{d}||_1 \leq B$ or $||\mathbf{w}||_1 \leq B$ in (30) since both $\mathbf{d}$ and $\mathbf{w}$ are scale-sensitive. As a result, it is not so easy for the ℓ_1-norm methods to control the number of features to be selected as FGM does. On the contrary, in AFS, we bound $\mathbf{d} \in [0, 1]^m$.

To demonstrate the connections of AFS to the ℓ_1-norm regularization, we need to make some transformations. Let $\widehat{w}_j = w_j \sqrt{d_j}$ and $\widehat{\mathbf{w}} = [\widehat{w}_1, ..., \widehat{w}_m]'$, the variational form of the problem (5) can be equivalently written as

$$\begin{aligned} \min_{\mathbf{d} \in \mathcal{D}} \min_{\widehat{\mathbf{w}}, \boldsymbol{\xi}, b} \quad & \frac{1}{2} \sum_{j=1}^{m} \frac{\widehat{w}_j^2}{d_j} + \frac{C}{2} \sum_{i=1}^{n} \xi_i^2 \\ \text{s.t.} \quad & y_i(\widehat{\mathbf{w}}' \mathbf{x}_i - b) \geq 1 - \xi_i, \quad i = 1, \cdots, n. \end{aligned}$$

For simplicity, hereby we drop the hat from $\widehat{\mathbf{w}}$ and define a new regularizer $\|\mathbf{w}\|_B^2$ as

$$\|\mathbf{w}\|_B^2 = \min_{\mathbf{d} \succeq \mathbf{0}} \sum_{j=1}^{m} \frac{w_j^2}{d_j}, \quad \text{s.t.} \quad ||\mathbf{d}||_1 \leq B, \quad \mathbf{d} \in [0, 1]^m. \tag{31}$$

This new regularizer has the following properties.

Proposition 3 *Given a vector $\mathbf{w} \in \mathbb{R}^m$ with $\|\mathbf{w}\|_0 = \kappa > 0$, where κ denotes the number of nonzero entries in $\mathbf{w}$. Let $\mathbf{d}^*$ be the minimizer of (31), we have: (I) $d_j^* = 0$ if $|w_j| = 0$. (II) If $\kappa \leq B$, then $d_j^* = 1$ for $|w_j| > 0$; else if $\frac{\|\mathbf{w}\|_1}{\max\{|w_j|\}} \geq B$ and $\kappa > B$, then we have $\frac{|w_j|}{d_j^*} = \frac{\|\mathbf{w}\|_1}{B}$ for all $|w_j| > 0$. (III) If $\kappa \leq B$, then $\|\mathbf{w}\|_B = \|\mathbf{w}\|_2$; else if $\frac{\|\mathbf{w}\|_1}{\max\{|w_j|\}} \geq B$ and $\kappa > B$, $\|\mathbf{w}\|_B = \frac{\|\mathbf{w}\|_1}{\sqrt{B}}$.*

The proof can be found in Appendix D.

According to Proposition 3, if $B < \kappa$, $\|\mathbf{w}\|_B$ is equivalent to the ℓ_1-norm regularizer. However, no matter how large the magnitude of $|w_j|$ is, d_j in $\|\mathbf{w}\|_B^2$ is always upper bounded

by 1, which lead to two advantages of $\|\mathbf{w}\|_B^2$ over the ℓ_1-norm regularizer. Firstly, by using $\|\mathbf{w}\|_B^2$, the sparsity and the over-fitting problem can be controlled separately by FGM. Specifically, one can choose a proper C to reduce the feature selection bias, and a proper stopping tolerance ϵ_c in (27) or a proper parameter B to adjust the number of features to be selected. Conversely, in the ℓ_1-norm regularized problems, the number of features is determined by the regularization parameter C, but the solution bias may happen if we intend to select a small number of features with a small C. Secondly, by transforming the resultant optimization problem into an SIP problem, a feature generating paradigm has been developed. By iteratively infer the most informative features, this scheme is particularly suitable for dealing with ultrahigh dimensional big data that are infeasible for the existing ℓ_1-norm methods, as shown in Section 6.2.

Proposition 3 can be easily extended to the group feature selection cases and multiple kernel learning cases. For instance, given a $\mathbf{w} \in \mathbb{R}^m$ with p groups $\{\mathcal{G}_1, ..., \mathcal{G}_p\}$, we have $\sum_{j=1}^p ||\mathbf{w}_{\mathcal{G}_j}||_2 = \|\mathbf{v}\|_1$, where $\mathbf{v} = [||\mathbf{w}_{\mathcal{G}_1}||, ..., ||\mathbf{w}_{\mathcal{G}_p}||]' \in \mathbb{R}^p$. Therefore, the above two advantages are also applicable to FGM for group feature selection and multiple kernel learning.

7.2 Connection to Existing AFS Schemes

The proposed AFS scheme is very different from the existing AFS schemes (e.g., Weston et al., 2000; Chapelle et al., 2002; Grandvalet and Canu, 2002; Rakotomamonjy, 2003; Varma and Babu, 2009; Vishwanathan et al., 2010). In existing works, the scaling vector $\mathbf{d} \succeq \mathbf{0}$ $\mathbf{d}$ is not upper bounded. For instance, in the SMO-MKL method (Vishwanathan et al., 2010), the AFS problem is reformulated as the following problem:

$$\min_{\mathbf{d} \succeq \mathbf{0}} \max_{\boldsymbol{\alpha} \in \mathcal{A}} \mathbf{1}'\boldsymbol{\alpha} - \frac{1}{2} \sum_{j=1}^p d_j (\boldsymbol{\alpha} \odot \mathbf{y})' \mathbf{K}_j (\boldsymbol{\alpha} \odot \mathbf{y}) + \frac{\lambda}{2} (\sum_j d_j^q)^{\frac{2}{q}},$$

where $\mathcal{A} = \{\boldsymbol{\alpha} | 0 \preceq \boldsymbol{\alpha} \preceq C\mathbf{1}, \mathbf{y}'\boldsymbol{\alpha} = 0\}$ and $\mathbf{K}_j$ denote a sub-kernel. When $0 \leq q \leq 1$, it induces sparse solutions, but results in non-convex optimization problems. Moreover, the sparsity of the solution is still determined by the regularization parameter λ. Consequently, the solution bias inevitably exists in the SMO-MKL formulation.

A more related work is the ℓ_1-MKL (Bach et al., 2004; Sonnenburg et al., 2006) or the SimpleMKL problem (Rakotomamonjy et al., 2008), which tries to learn a linear combination of kernels. The variational regularizer of SimpleMKL can be written as:

$$\min_{\mathbf{d} \succeq 0} \sum_{j=1}^p \frac{||\mathbf{w}_j||^2}{d_j}, \quad \text{s.t.} \quad ||\mathbf{d}||_1 \leq 1,$$

where p denotes the number of kernels and $\mathbf{w}_j$ represents the parameter vector of the jth kernel in the context of MKL (Kloft et al., 2009, 2011; Kloft and Blanchard, 2012). Correspondingly, the regularizer $||\mathbf{w}||_B^2$ regarding kernels can be expressed as:

$$\min_{\mathbf{d} \succeq 0} \sum_{j=1}^p \frac{||\mathbf{w}_j||^2}{d_j}, \quad \text{s.t.} \quad ||\mathbf{d}||_1 \leq B, \ \ \mathbf{d} \in [0, 1]^p. \tag{32}$$

To illustrate the difference between (32) and the ℓ_1-MKL, we divide the two constraints in (32) by B, and obtain

$$\sum_{j=1}^{p} \frac{d_j}{B} \leq 1, \quad 0 \leq \frac{d_j}{B} \leq \frac{1}{B}, \forall j \in \{1, ..., p\}.$$

Clearly, the box constraint $\frac{d_j}{B} \leq \frac{1}{B}$ makes (32) different from the variational regularizer in ℓ_1-MKL. Actually, the ℓ_1-norm MKL is only a special case of $||\mathbf{w}||_B^2$ when $B = 1$. Moreover, by extending Proposition 3, we can obtain that if $B > \kappa$, we have $||\mathbf{w}||_B^2 = \sum_{j=1}^{p} ||\mathbf{w}_j||^2$, which becomes a non-sparse regularizer. Another similar work is the ℓ_q-MKL, which generalizes the ℓ_1-MKL to ℓ_q-norm ($q > 1$) (Kloft et al., 2009, 2011; Kloft and Blanchard, 2012). Specifically, the variational regularizer of ℓ_q-MKL can be written as

$$\min_{\mathbf{d} \succeq 0} \sum_{j=1}^{p} \frac{||\mathbf{w}_j||^2}{d_j}, \quad \text{s.t.} \quad ||\mathbf{d}||_q^2 \leq 1.$$

We can see that, the box constraint $0 \leq \frac{d_j}{B} \leq \frac{1}{B}, \forall j \in \{1, ..., p\}$ is missing in the ℓ_q-MKL. However, when $q > 1$, the ℓ_q-MKL cannot induce sparse solutions, and thus cannot discard non-important kernels or features. Therefore, the underlying assumption for ℓ_q-MKL is that, most of the kernels are relevant for the classification tasks. Finally, it is worth mentioning that, when doing multiple kernel learning, both ℓ_1-MKL and ℓ_q-MKL require to compute and involve all the base kernels. Consequently the computational cost is unbearable for large-scale problems with many kernels.

An infinite kernel learning method is introduced to deal with infinite number of kernels ($p = \infty$) (Gehler and Nowozin, 2008). Specifically, IKL adopts the ℓ_1-MKL formulation (Bach et al., 2004; Sonnenburg et al., 2006), thus it can be considered as a special case of FGM when setting $B = 1$. Due to the infinite number of possible constraints, IKL also adopts the cutting plane algorithm to address the resultant problem. However, it can only include one kernel per iteration; while FGM can include B kernels per iteration. In this sense, IKL is also analogous to the active set methods (Roth and Fischer, 2008; Bach, 2009). For both methods, the worst-case analysis for large-scale problems usually dominates the overall training complexity. For FGM, since it is able to include B kernels per iteration, it obviously reduces the number of worst-case analysis steps, and thus has great computational advantages over IKL. Finally, it is worth mentioning that, based on the IKL formulation, it is non-trivial for IKL to include B kernels per iteration.

7.3 Connection to Multiple Kernel Learning

In FGM, each subproblem is formulated as a SimpleMKL problem (Rakotomamonjy et al., 2008), and any SimpleMKL solver can be used to solve it. For instance, an approximate solution can be also efficiently obtained by a sequential minimization optimization (SMO) (Bach et al., 2004; Vishwanathan et al., 2010). Sonnenburg et al. (2006) proposed a semi-infinite linear programming formulation for MKL which allows MKL to be iteratively solved with SVM solver and linear programming. Xu et al. (2009b) proposed an extended level method to improve the convergence of MKL. More recently, an online ultra-fast MKL algorithm,

called as the UFO-MKL, was proposed by Orabona and Jie (2011). However, its $O(1/\epsilon)$ convergence rate is only guaranteed when a strongly convex regularizer $\Omega(\mathbf{w})$ is added to the objective. Without the strongly convex regularizer, its convergence is unclear.

In summary, FGM is different from MKL in several aspects. At first, FGM iteratively includes B new kernels through the worst-case analysis. Particularly, these B kernels will be formed as a base kernel for the MKL subproblem of FGM. From the kernel learning view, FGM provides a new way to construct base kernels. Secondly, since FGM tends to select a subset of kernels, it is especially suitable for MKL with many kernels. Thirdly, to scale MKL to big data, we propose to use the approximated features (or explicit feature mappings) for kernels. As a result, the MKL problem is reduced to a group feature selection problem, and we can solve the subproblem in its primal form.

7.4 Connection to Active Set Methods

Active set methods have been widely applied to address the challenges of large number of features or kernels (Roth and Fischer, 2008; Bach, 2009). Basically, active set methods iteratively include a variable that violates the optimality condition of the sparsity-induced problems. In this sense, active methods can be considered as a special case of FGM with $B = 1$. However, FGM is different from active set methods. Firstly, their motivations are different: active set methods start from the Lagrangian duality of the sparsity-induced problems; while FGM starts from the proposed AFS scheme, solves an SIP problem. Secondly, active set methods only include one active feature/group/kernel at each iteration. Regarding this algorithm, when the desired number of kernels or groups becomes relatively large, active set methods will be very computationally expensive. On the contrary, FGM allows to add B new features/groups/kernels per iteration, which can greatly improve the training efficiency by reducing the number of worst-case analysis. Thirdly, a sequence of $\ell_{2,1}^2$-regularized non-smooth problems are solved in FGM, which is very different from the active set methods. Finally, the de-biasing of solutions is not investigated in the active set methods (Bach, 2009; Roth and Fischer, 2008).

8. Experiments

We compare the performance of FGM with several state-of-the-art baseline methods on three learning tasks, namely the linear feature selection, the ultrahigh dimensional nonlinear feature selection and the group feature selection.[4]

The experiments are organized as follows. Firstly, in Section 8.1, we present the general experimental settings. After that in Section 8.2, we conduct synthetic experiments to study the performance of FGM on the linear feature selection. Moreover, in Section 8.3, we study

4. In the experiments, some aforementioned methods, such as NMMKL, QCQP-SSVM and SVM-RFE, are not included for comparison due to the high computational cost for the optimization or sub-optimality for the feature selection. Interested readers can refer to (Tan et al., 2010) for the detailed comparisons. We also do not include the ℓ_q-MKL (Kloft et al., 2009, 2011; Kloft and Blanchard, 2012) for comparison since it cannot induce sparse solutions. Instead, we include an ℓ_q-variant, i.e., UFO-MKL (Orabona and Jie, 2011), for comparison. Finally, since IKL is a special case of FGM with $B = 1$, we study its performance through FGM with $B = 1$ instead. Since it is analogous to the active set methods, its performance can be also observed from the results of active set method.

the performance of FGM with the shift of hyperplane. In Section 8.4, we conduct real-world experiments on linear feature selection. In Section 8.5, we conduct ultrahigh dimensional nonlinear feature selection experiments with polynomial feature mappings. Finally, we demonstrate the efficacy of FGM on the group feature selection in Section 8.6.

8.1 Data Sets and General Experimental Settings

Several large-scale and high dimensional real-world data sets are used to verify the performance of different methods. General information of these data sets, such as the average nonzero features per instance, is listed in Table 1.[5] Among them, `epsilon`, `Arxiv astro-ph`, `rcv1.binary` and `kddb` data sets have been split into training set and testing set. For `real-sim`, `aut-avn` and `news20.binary`, we randomly split them into training and testing sets, as shown in Table 1.

Data set	m	n_{train}	n_{test}	# nonzeros per instance	Parameter Range		
					l1-SVM (C)	l1-LR(C)	SGD-SLR(λ_1)
epsilon	2,000	400,000	100,000	2,000	[5e-4, 1e-2]	[2e-3, 1e-1]	[1e-4, 8e-3]
aut-avn	20,707	40,000	22,581	50	–	–	–
real-sim	20,958	32,309	40,000	52	[5e-3, 3e-1]	[5e-3, 6e-2]	[1e-4, 8e-3]
rcv1	47,236	677,399	20,242	74	[1e-4, 4e-3]	[5e-5, 2e-3]	[1e-4, 8e-3]
astro-ph	99,757	62,369	32,487	77	[5e-3, 6e-2]	[2e-2, 3e-1]	[1e-4, 8e-3]
news20	1,355,191	9,996	10,000	359	[5e-3, 3e-1]	[5e-2, 2e1]	[1e-4, 8e-3]
kddb	29,890,095	19,264,097	748,401	29	[5e-6, 3e-4]	[3e-6, 1e-4]	[1e-4, 8e-3]

Table 1: Statistics of the data sets used in the experiments. Parameter Range lists the ranges of the parameters for various ℓ_1-methods to select different number of features. The data sets `rcv1` and `aut-avn` will be used in group feature selection tasks.

On the linear feature selection task, comparisons are conducted between FGM and the ℓ_1-regularized methods, including ℓ_1-SVM and ℓ_1-LR. For FGM, we study FGM with SimpleMKL solver (denoted by MKL-FGM)[6] (Tan et al., 2010), FGM with APG method for the squared hinge loss (denoted by PROX-FGM) and the logistic loss (denoted by PROX-SLR), respectively.

Many efficient batch training algorithms have been developed to solve ℓ_1-SVM and ℓ_1-LR, such as the interior point method, fast iterative shrinkage-threshold algorithm (FISTA), block coordinate descent (BCD), Lassplore method (Liu and Ye, 2010), generalized linear model with elastic net (GLMNET) and so on (Yuan et al., 2010, 2011). Among them, LIBLinear, which adopts the coordinate descent to solve the non-smooth optimization problem, has demonstrated state-of-the-art performance in terms of training efficiency (Yuan et al.,

5. Among these data sets, `epsilon`, `real-sim`, `rcv1.binary`, `news20.binary` and `kddb` can be downloaded at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`, `aut-avn` can be downloaded at `http://vikas.sindhwani.org/svmlin.html` and `Arxiv astro-ph` is from Joachims (2006).
6. For the fair comparison, we adopt the LIBLinear (e.g., CD-SVM) as the SVM solver in SimpleMKL when performing linear feature selections. The source codes of MKL-FGM are available at `http://www.tanmingkui.com/fgm.html`.

2010). In LIBLinear, by taking the advantages of data sparsity, it achieves very fast convergence speed for sparse data sets (Yuan et al., 2010, 2011). In this sense, we include the LIBLinear solver for comparison[7]. Besides, we take the standard SVM and LR classifier of LIBLinear with all features as the baselines, denoted by CD-SVM and CD-LR, respectively. We use the default stopping criteria of LIBLinear for ℓ_1-SVM, ℓ_1-LR, CD-SVM and CD-LR.

SGD methods have gained great attention for solving large-scale problems (Langford et al., 2009; Shalev-Shwartz and Zhang, 2013). In this experiment, we include the proximal stochastic dual coordinate ascent with logistic loss for comparison (which is denoted by SGD-SLR). SGD-SLR has shown the state-of-the-art performance among various SGD methods (Shalev-Shwartz and Zhang, 2013).[8] In SGD-SLR, there are three important parameters, namely λ_1 to penalize $||\mathbf{w}||_1$, λ_2 to penalize $||\mathbf{w}||_2^2$, and the stopping criterion *min.dgap*. Suggested by the package, in the following experiment, we fix λ_2 = 1e-4 and *min.dgap*=1e-5, and change λ_1 to obtain different levels of sparsity. All the methods are implemented in C++.

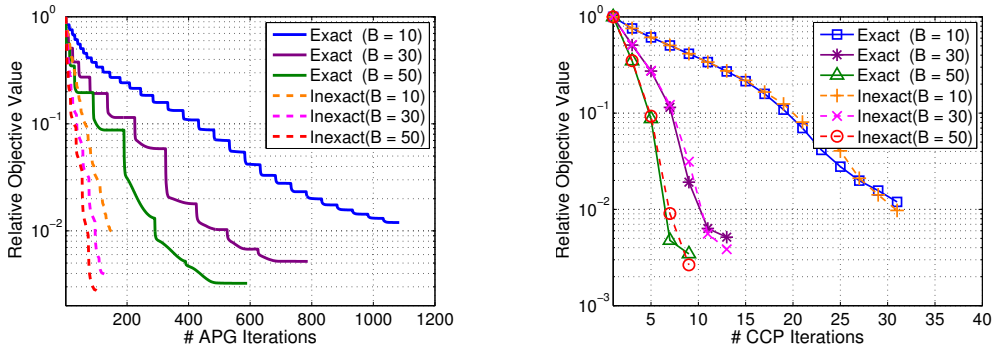
On group feature selection tasks, we compare FGM with four recently developed group lasso solvers: FISTA (Liu and Ye, 2010; Jenatton et al., 2011b; Bach et al., 2011), block coordinate descent method (denoted by BCD) (Qin et al., 2010), active set method (denoted by ACTIVE) (Bach, 2009; Roth and Fischer, 2008) and UFO-MKL (Orabona and Jie, 2011). Among them, FISTA has been thoroughly studied by several researchers (Liu and Ye, 2010; Jenatton et al., 2011b; Bach et al., 2011), and we adopt the implementation of SLEP package[9], where an improved line search is used (Liu and Ye, 2010). We implement the block coordinate descent method proposed by Qin et al. (2010), where each subproblem is formulated as a trust-region problem and solved by a Newton's root-finding method (Qin et al., 2010). For UFO-MKL, it is an online optimization method,[10] and we stop the training after 20 epochs. Finally, we implement ACTIVE method based on the SLEP solver. All the methods for group feature selection are implemented in MATLAB for fair comparison.

All the comparisons are performed on a 2.27GHZ Intel(R)Core(TM) 4 DUO CPU running windows sever 2003 with 24.0GB main memory.

8.2 Synthetic Experiments on Linear Feature Selection

In this section, we compare the performance of different methods on two toy data sets of different scales, namely $\mathbf{X} \in \mathbb{R}^{4,096\times 4,096}$ and $\mathbf{X} \in \mathbb{R}^{8,192\times 65,536}$. Here each $\mathbf{X}$ is a Gaussian random matrix with each entry sampled from the i.i.d. Gaussian distribution $\mathcal{N}(0,1)$. To produce the output $\mathbf{y}$, we first generate a sparse vector $\mathbf{w}$ with 300 nonzero entries, with each nonzero entry sampled from the i.i.d. Uniform distribution $\mathcal{U}(0,1)$. After that, we produce the output by $\mathbf{y} = \text{sign}(\mathbf{X}\mathbf{w})$. Since only the nonzero w_i contributes to the output $\mathbf{y}$, we consider the corresponding feature as a relevant feature regarding $\mathbf{y}$. Similarly, we generate the testing data set $\mathbf{X}_{\text{test}}$ with output labels $\mathbf{y}_{\text{test}} = \text{sign}(\mathbf{X}_{\text{test}}\mathbf{w})$. The number of testing points for both cases is set to 4,096.

7. Sources are available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`.
8. Sources are available at `http://stat.rutgers.edu/home/tzhang/software.html`.
9. Sources are available at `http://www.public.asu.edu/~jye02/Software/SLEP/index.htm`.
10. Sources are available at `http://dogma.sourceforge.net/`.



(a) Relative objective values w.r.t. APG iterations.

(b) Relative objective values w.r.t. outer iterations.

Figure 1: Convergence of Inexact FGM and Exact FGM on the synthetic data set.

8.2.1 Convergence Comparison of Exact and Inexact FGM

In this experiment, we study the convergence of the *Exact* and *Inexact* FGM on the small scale data set. To study the *Exact* FGM, for simplicity, we set the stopping tolerance $\epsilon_{in} = 1.0 \times 10^{-6}$ in equation (28) for APG algorithm; while for *Inexact* FGM, we set $\epsilon_{in} = 1.0 \times 10^{-3}$. We set $C = 10$ and test different B's from $\{10, 30, 50\}$. In this experiment, only the squared hinge loss is studied. In Figure 1(a), we report the relative objective values w.r.t. all the APG iterations for both methods; In Figure 1(b), we report the relative objective values w.r.t. the outer iterations. We have the following observations from Figures 1(a) and 1(b).

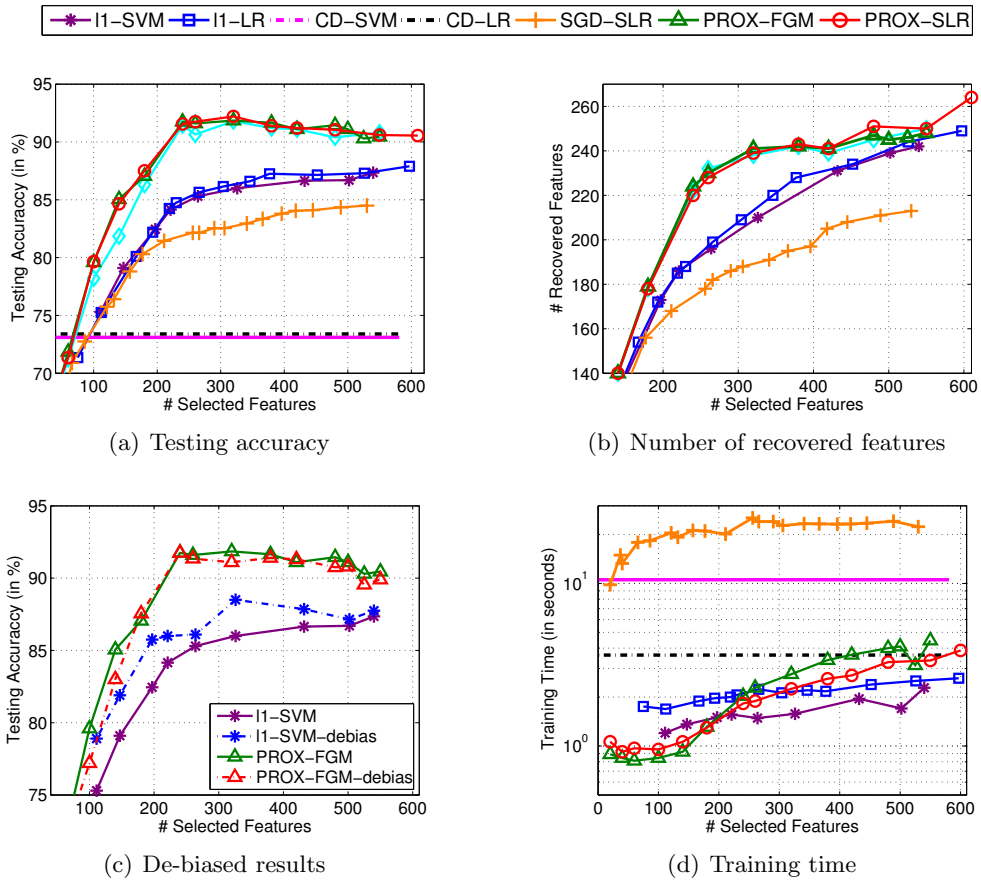
Firstly, from Figure 1(a), for each comparison method, the function value sharply decreases at some iterations, where an active constraint is added. For the *Exact* FGM, it requires more APG iterations under the tolerance $\epsilon_{in} = 1.0 \times 10^{-6}$, but the function value does not show significant decrease after several APG iterations. On the contrary, from Figure 1(a), the *Inexact* FGM, which uses a relatively larger tolerance $\epsilon_{in} = 1.0 \times 10^{-3}$, requires much fewer APG iterations to achieve the similar objective values to *Exact* FGM under the same parameter B. Particularly, from Figure 1(b), the *Inexact* FGM achieves the similar objective values to *Exact* FGM after each outer iteration. According to these observations, on one hand, ϵ_{in} should be small enough such that the subproblem can be sufficiently optimized. On the other hand, a relatively large tolerance (e.g. $\epsilon_{in} = 1.0 \times 10^{-3}$) can greatly accelerate the convergence speed without degrading the performance.

Moreover, according to Figure 1(b), PROX-FGM with a large B in general converges faster than that with a small B. Generally speaking, by using a large B, less number of outer iterations and worst-case analysis are required, which is critical when dealing with big data. However, if B is too large, some non-informative features may be mistakenly included, and the solution may not be exactly sparse.

8.2.2 Experiments on Small-Scale Synthetic Dataset

In this experiment, we evaluate the performance of different methods in terms of testing accuracies w.r.t. different number of selected features. Specifically, to obtain sparse solutions

of different sparsities, we vary $C \in [0.001, 0.007]$ for l1-SVM, $C \in$ [5e-3, 4e-2] for l1-LR and $\lambda_1 \in$ [7.2e-4, 2.5e-3] for SGD-SLR.[11] On contrary to these methods, we fix $C = 10$ and choose even numbers in $\{2, 4, ..., 60\}$ for B to obtain different number of features. It can be seen that, it is much easier for FGM to control the number of features to be selected. Specifically, the testing accuracies and the number of recovered ground-truth features w.r.t. the number of selected features are reported in Figure 2(a) and Figure 2(b), respectively. The training time of different methods is listed in Figure 2(d).



(a) Testing accuracy

(b) Number of recovered features

(c) De-biased results

(d) Training time

Figure 2: Experimental results on the small data set, where CD-SVM and CD-LR denote the results of standard SVM and LR with all features, respectively. The training time of MKL-FGM is about 1,500 seconds, which is up to 1,000 times slower than APG solver. We did not report it in the figures due to presentation issues.

For convenience of presentation, let m_s and m_g be the number of selected features and the number of ground-truth features, respectively. From Figure 2(a) and Figure 2(b), FGM

11. Here, we carefully choose C or λ_1 for these three ℓ_1-methods such that the numbers of selected features uniformly spread over the range [0, 600]. Since the values of C and λ_1 change a lot for different problems, hereafter we only give their ranges. Under this experimental setting, the results of ℓ_1-methods cannot be further improved through parameter tunings.

based methods demonstrate better testing accuracy than all ℓ_1-methods when $m_s > 100$. Correspondingly, from Figure 2(b), under the same number of selected features, FGM based methods include more ground-truth features than ℓ_1-methods when $m_s \geq 100$. For SGD-SLR, it shows the worst testing accuracy among the comparison methods, and also recovers the least number of ground-truth features.

One of the possible reasons for the inferior performance of the ℓ_1-methods, as mentioned in the Introduction section, is the solution bias brought by the ℓ_1-regularization. To demonstrate this, we do re-training to reduce the bias using CD-SVM with $C = 20$ with the selected features, and then do the prediction using the de-biased models. The results are reported in Figure 2(c), where l1-SVM-debias and PROX-FGM-debias denote the de-biased counterparts for l1-SVM and PROX-FGM, respectively. In general, *if there was no feature selection bias, both FGM and l1-SVM should have the similar testing accuracy to their de-biased counterparts*. However, from Figure 2(c), l1-SVM-debias in general has much better testing accuracy than l1-SVM; while PROX-FGM has similar or even better testing accuracy than PROX-FGM-debias and l1-SVM-debias. These observations indicate that: 1) the solution bias indeed exists in the ℓ_1-methods and affects the feature selection performance; 2) FGM can reduce the feature selection bias.

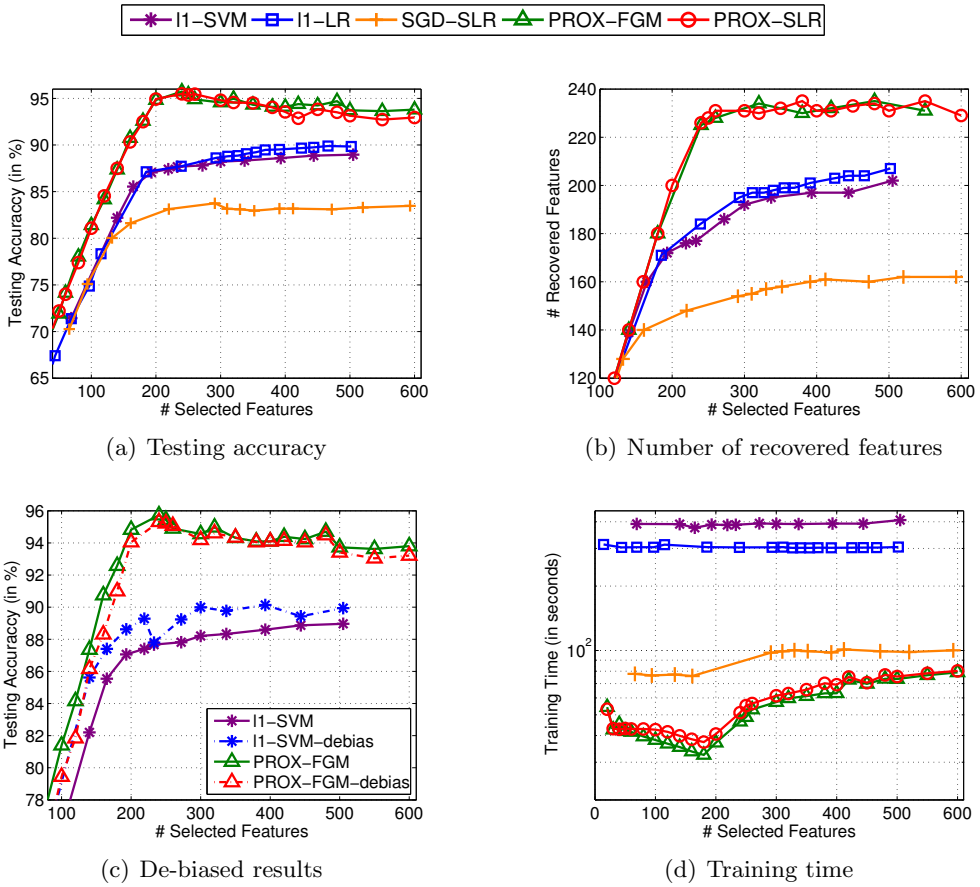
From Figure 2(d), on this small-scale data set, PROX-FGM and PROX-SLR achieve comparable efficiency with the LIBlinear solver. On the contrary, SGD-SLR, which is a typical stochastic gradient method, spends the longest training time. This observation indicates that SGD-SLR method may not be suitable for small-scale problems. Finally, as reported in the caption of Figure 2(d), PROX-FGM and PROX-SLR are up to 1,000 times faster than MKL-FGM using SimpleMKl solver. The reason is that, SimpleMKl uses the subgradient methods to address the non-smooth optimization problem with n variables; While in PROX-FGM and PROX-SLR, the subproblem is solved in the primal problem w.r.t. a small number of selected variables.

Finally, from Figure 2, if the number of selected features is small ($m_s < 100$), the testing accuracy is worse than CD-SVM and CD-LR with all features. However, if sufficient number ($m_s > 200$) of features are selected, the testing accuracy is much better than CD-SVM and CD-LR with all features, which verifies the importance of the feature selection.

8.2.3 Experiments on Large-scale Synthetic Dataset

To demonstrate the scalability of FGM, we conduct an experiment on a large-scale synthetic data set, namely $\mathbf{X} \in \mathbb{R}^{8,192 \times 65,536}$. Here, we do not include the comparisons with MKL-FGM due to its high computational cost. For PROX-FGM and PROX-SLR, we follow their experimental settings above. For l1-SVM and l1-LR, we vary $C \in [0.001, 0.004]$ and $C \in [0.005, 0.015]$ to determine the number of features to be selected, respectively. The testing accuracy, the number of recovered ground-truth features, the de-biased results and the training time of the compared methods are reported in Figure 3(a), 3(b), 3(c) and 3(d), respectively.

From Figure 3(a) and 3(b) and 3(c), both PROX-FGM and PROX-SLR outperform l1-SVM, l1-LR and SGD-SLR in terms of both testing accuracy and the number of recovered ground-truth features. From Figure 3(d), PROX-FGM and PROX-SLR show better training efficiency than the coordinate based methods (namely, LIBlinear) and the SGD



(a) Testing accuracy (b) Number of recovered features

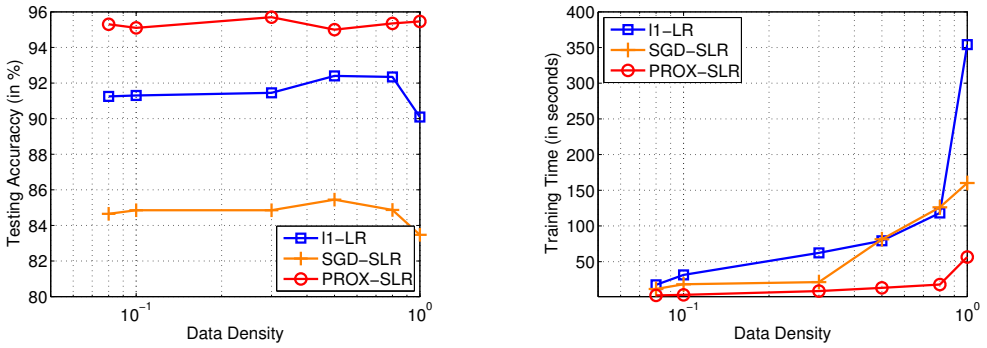
(c) De-biased results (d) Training time

Figure 3: Performance comparison on the large-scale synthetic data set.

based method (namely SGD-SLR). Basically, FGM solves a sequence of small optimization problems of $O(ntB)$ cost, and spends only a small number of iterations to do the worst-case analysis of $O(mn)$ cost. On the contrary, the ℓ_1-methods may take many iterations to converge, and each iteration takes $O(mn)$ cost. On this large-scale data set, SGD-SLR shows faster training speed than LIBlinear, but it has much inferior testing accuracy over other methods.

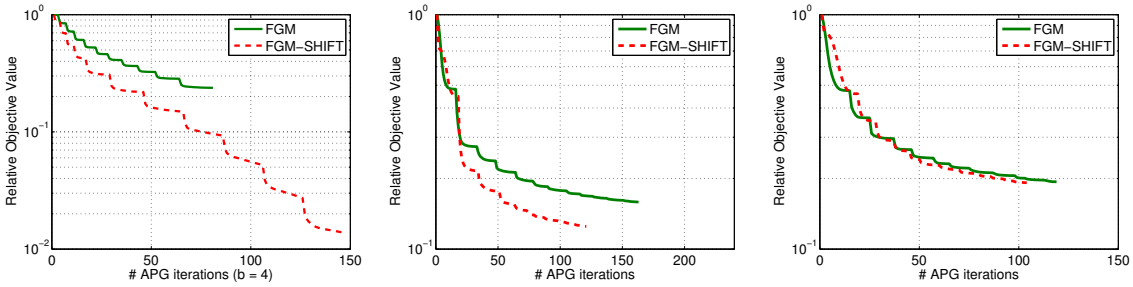
In LIBlinear, the efficiency has been improved by taking the advantage of the data sparsity. Considering this, we investigate the sensitivity of the referred methods to the data density. To this end, we generate data sets of different data densities by sampling the entries from $\mathbf{X}^{8{,}192\times 65{,}656}$ with different data densities in $\{0.08, 0.1, 0.3, 0.5, 0.8, 1\}$, and study the influence of the data density on different learning algorithms. For FGM, only the logistic loss is studied (e.g. PROX-SLR). We use the default experimental settings for PROX-SLR, and watchfully vary $C \in [0.008, 5]$ for l1-LR and $\lambda_1 \in$ [9.0e-4, 3e-3] for SGD-SLR. For the sake of brevity, we only report the **best accuracy** obtained over all parameters, and the corresponding training time of l1-LR, SGD-SLR and PROX-SLR in Figure 4.

From Figure 4(a), under different data densities, PROX-SLR always outperforms l1-SVM and SGD-SLR in terms of the **best accuracy**. From Figure 4(b), l1-SVM shows



(a) Testing accuracy (b) Training time

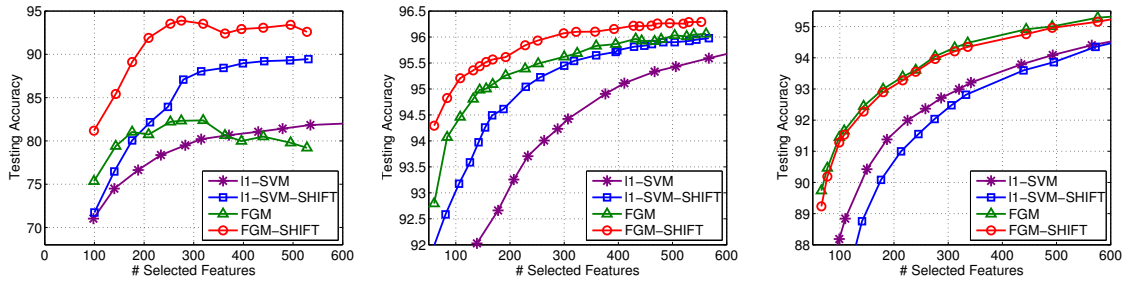
Figure 4: Performance comparison on the synthetic data set ($n = 8,192$, $m = 65,536$) with different data densities in $\{0.08, 0.1, 0.3, 0.5, 0.8, 1\}$.



(a) Synthetic data set ($b = 4$) (b) `astro-ph` (c) `real-sim`

Figure 5: Relative objective values regarding each APG iteration, where $b = 4$ in the caption of Figure 5(a) denotes the ground-truth shift of the hyperplane from the origin.

comparable efficiency with PROX-SLR on data sets of low data density. However, on relative denser data sets, PROX-SLR is much more efficient than l1-SVM, which indicates that FGM has a better scalability than l1-SVM on dense data.



(a) Synthetic data set ($b = 4$) (b) `astro-ph` (c) `real-sim`

Figure 6: Testing accuracy of different methods on the three data data sets.

8.3 Feature Selection with Shift of Hyperplane

In this section, we study the effectiveness of the shift version of FGM (denoted by FGM-SHIFT) on a synthetic data set and two real-world data sets, namely `real-sim` and `astro-ph`. We follow the data generation in Section 7.1 to generate the synthetic data set except that we include a shift term b for the hyperplane when generating the output $\mathbf{y}$. Specifically, we produce $\mathbf{y}$ by $\mathbf{y} = \text{sign}(\mathbf{X}\mathbf{w} - b\mathbf{1})$, where $b = 4$. The shift version of ℓ_1-SVM by LIBlinear (denoted by l1-SVM-SHIFT) is adopted as the baseline. In Figure 5, we report the relative objective values of FGM and FGM-SHIFT w.r.t. the APG iterations on three data sets. In Figure 6, we report the testing accuracy versus different number of selected features.

From Figure 5, FGM-SHIFT indeed achieves much lower objective values than FGM on the synthetic data set and `astro-ph` data set, which demonstrates the effectiveness of FGM-SHIFT. On the `real-sim` data set, FGM and FGM-SHIFT achieve similar objective values, which indicates that the shift term on `real-sim` is not significant. As a result, FGM-SHIFT may not significantly improve the testing accuracy.

From Figure 6, on the synthetic data set and `astro-ph` data set, FGM-SHIFT shows significant better testing accuracy than the baseline methods, which coincides with the better objective values of FGM-SHIFT in Figure 5. l1-SVM-SHIFT also shows better testing accuracy than l1-SVM, which verifies the importance of shift consideration for l1-SVM. However, on the `real-sim` data set, the methods with shift show similar or even inferior performances over the methods without shift consideration, which indicates that the shift of the hyperplane from the origin is not significant on the `real-sim` data set. Finally, FGM and FGM-SHIFT are always better than the counterparts of l1-SVM.

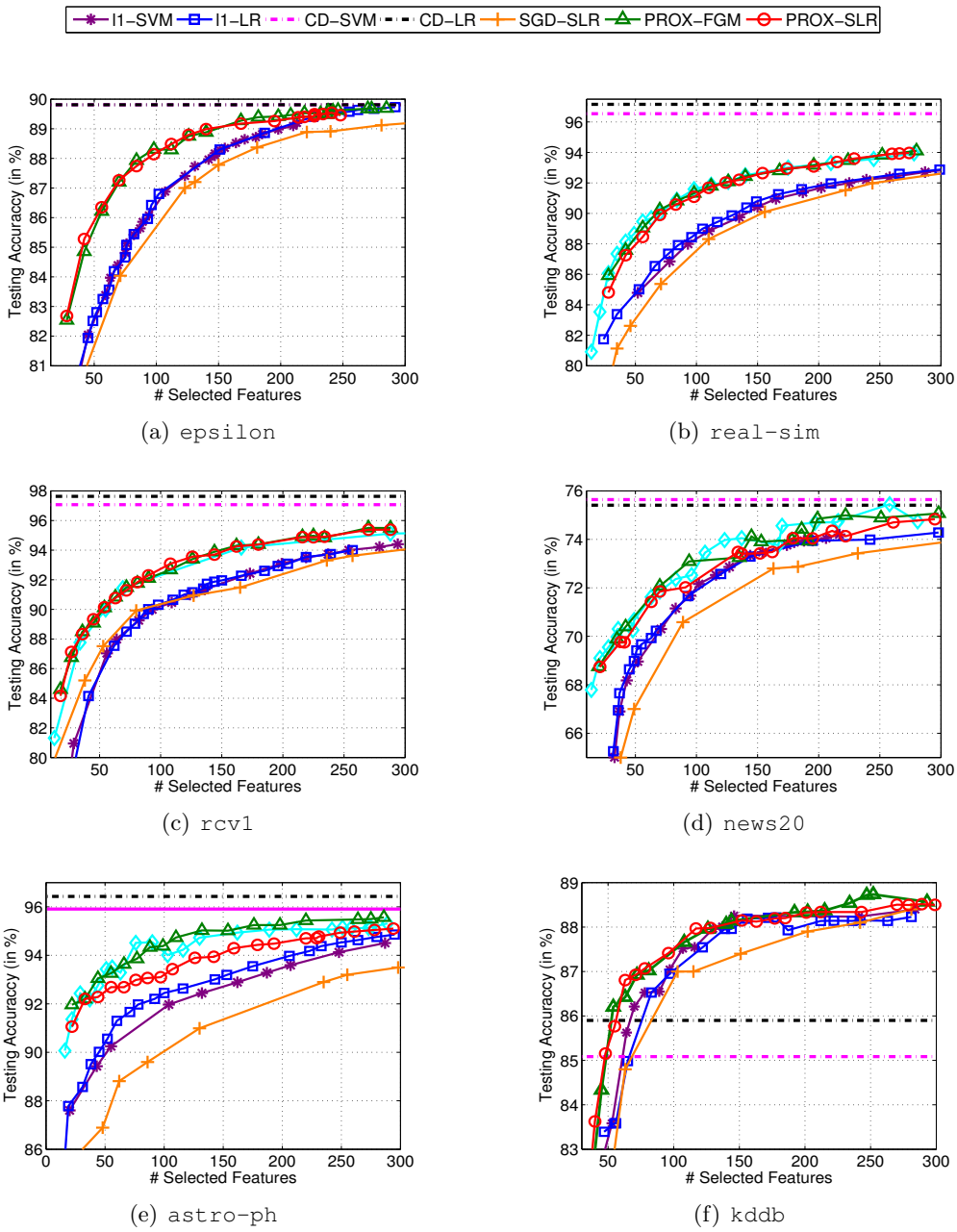
8.4 Performance Comparison on Real-World Data Sets

In this section, we conduct three experiments to compare the performance of FGM with the referred baseline methods on real-world data sets. In Section 8.4.1, we compare the performance of different methods on six real-world data sets. In Section 8.4.2, we study the feature selection bias issue. Finally, in Section 8.4.3, we conduct the sensitivity study of parameters for FGM.

8.4.1 Experimental Results on Real-World Data Sets

On real-world data sets, the number of ground-truth features is unknown. We only report the testing accuracy versus different number of selected features. For FGM, we fix $C = 10$, and vary $B \in \{2, 4, ..., 60\}$ to select different number of features. For the ℓ_1-methods, we watchfully vary the regularization parameter to select different number of features. The ranges of C and λ_1 for ℓ_1-methods are listed in Table 1.

The testing accuracy and training time of different methods against the number of selected features are reported in Figure 7 and Figure 8, respectively. From Figure 7, on all data sets, FGM (including PROX-FGM, PROX-SLR and MKL-FGM) obtains comparable or better performance than the ℓ_1-methods in terms of testing accuracy within 300 features. Particularly, FGM shows much better testing accuracy than ℓ_1-methods on five of the studied data sets, namely `epsilon`, `real-sim`, `rcv1.binary`, `Arxiv astro-ph` and `news20`.



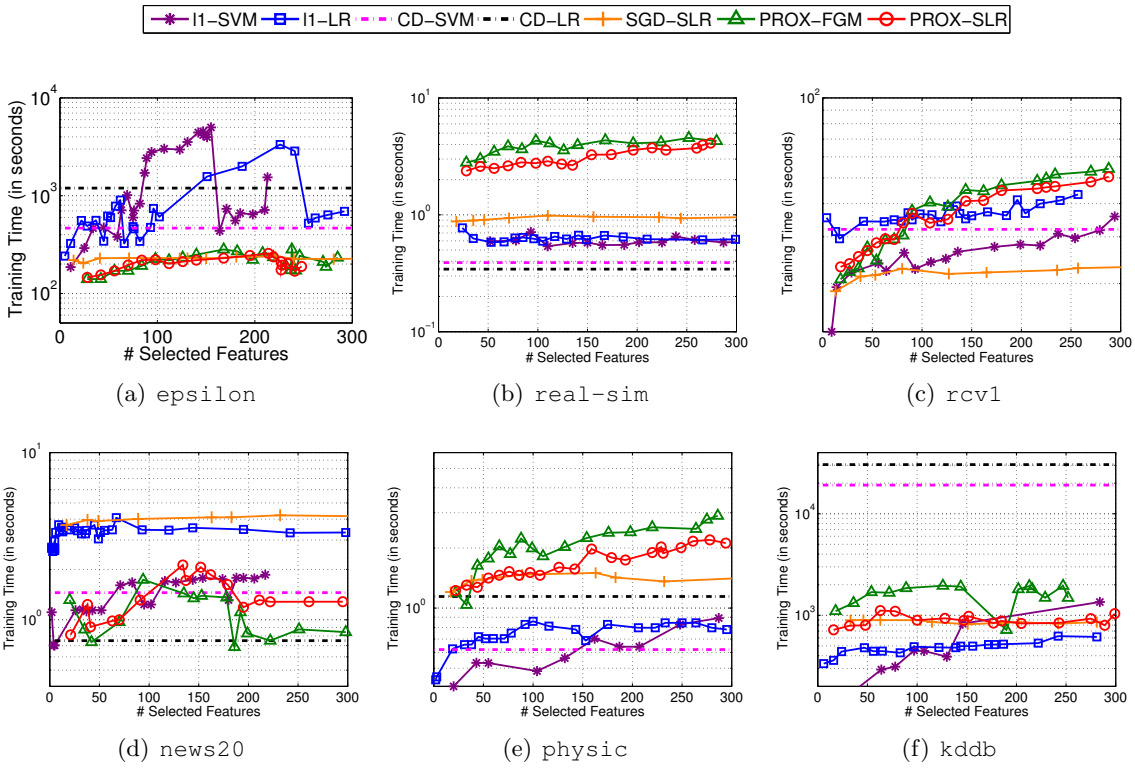
(a) epsilon (b) real-sim

(c) rcv1 (d) news20

(e) astro-ph (f) kddb

Figure 7: Testing accuracy on various data sets.

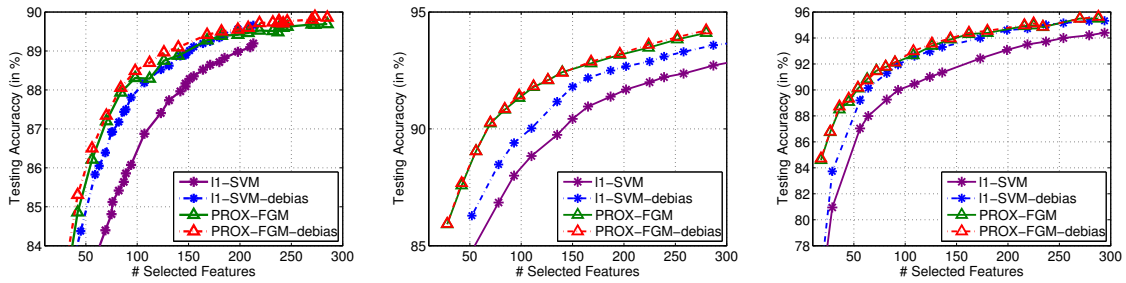
From Figure 8, PROX-FGM and PROX-SLR show competitive training efficiency with the ℓ_1-methods. Particularly, on the large-scale dense epsilon data set, PROX-FGM and PROX-SLR are much efficient than the LIBlinear ℓ_1-solvers. For SGD-SLR, although it demonstrates comparable training efficiency with PROX-FGM and PROX-SLR, it attains much worse testing accuracy. In summary, FGM based methods in general obtain better



(a) epsilon (b) real-sim (c) rcv1

(d) news20 (e) physic (f) kddb

Figure 8: Training time on various data sets.



(a) epsilon (b) real-sim (c) rcv1

Figure 9: De-biased results on real-world data sets.

feature subsets with competitive training efficiency with the considered baselines on real-world data sets.

8.4.2 De-biasing Effect of FGM

In this experiment, we demonstrate the de-biasing effect of FGM on three real-world data sets, namely epsilon, real-sim and rcv1. Here, only the squared hinge loss (namely PFOX-FGM) is studied. The de-biased results are reported in Figure 9, where PROX-

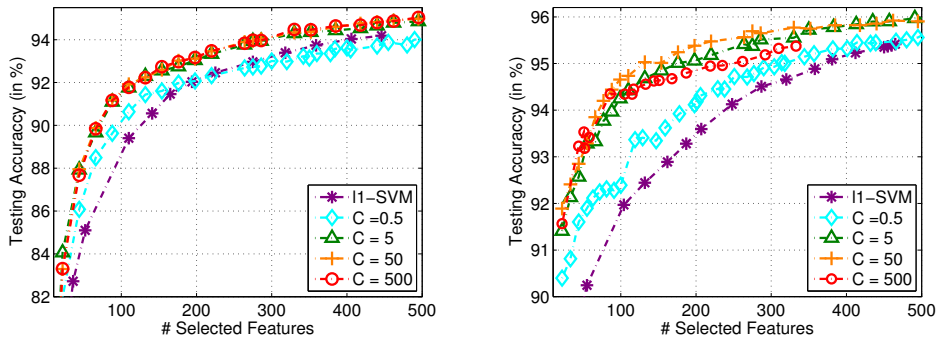
FGM-debias and l1-SVM-debias denote the de-biased results of PROX-FGM and l1-SVM, respectively.

From Figure 9, l1-SVM-debias shows much better results than l1-SVM, indicating that the feature selection bias issue exists in l1-SVM on these real-world data sets. On the contrary, PROX-FGM achieves close or even better results compared with its de-biased counterparts, which verifies that PROX-FGM itself can reduce the feature selection bias. Moreover, on these data sets, FGM shows better testing accuracy than the de-biased l1-SVM, namely l1-SVM-debias, which indicates that the features selected by FGM are more relevant than those obtained by l1-SVM due to the reduction of feature selection bias.

8.4.3 Sensitivity Study of Parameters

In this section, we conduct the sensitivity study of parameters for PROX-FGM. There are two parameters in FGM, namely the sparsity parameter B and the regularization parameter C. In this experiments, we study the sensitivity of these two parameters on `real-sim` and `astro-ph` data sets. l1-SVM is adopted as the baseline.

In the first experiment, we study the sensitivity of C. FGM with suitable C can reduce the feature selection bias. However, C is too large, the over-fitting problem may happen. To demonstrate this, we test $C \in \{0.5, 5, 50, 500\}$. The testing accuracy of FGM under different C's is reported in Figure 10. From Figure 10, the testing accuracy with small a C in general is worse than that with a large C. The reason is that, when C is small, feature selection bias may happen due to the under-fitting problem. However, when C is sufficient large, increasing C may not necessarily improve the performance. More critically, if C is too large, the over-fitting problem may happen. For example, on the `astro-ph` data set, FGM with $C = 500$ in general performs much worse than FGM with $C = 5$ and $C = 50$. Another important observation is that, on both data sets, FGM with different C's generally performs better than the l1-SVM.

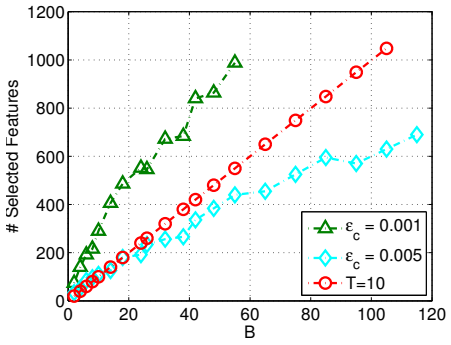


(a) `real-sim` (b) `astro-ph`

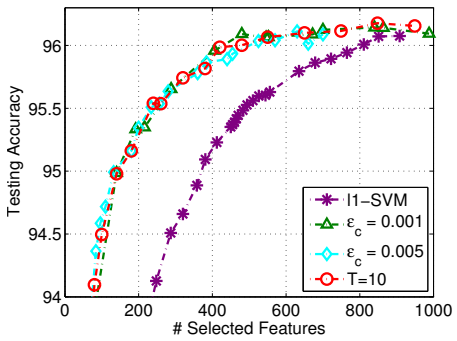
Figure 10: Sensitivity of the parameter C for FGM on `real-sim` and `astro-ph` data sets.

Recall that, a large C may lead to slower convergence speed due to the increasing of the Lipschitz constant of $F(\boldsymbol{\omega}, b)$. In practice, we suggest choosing C in the range of

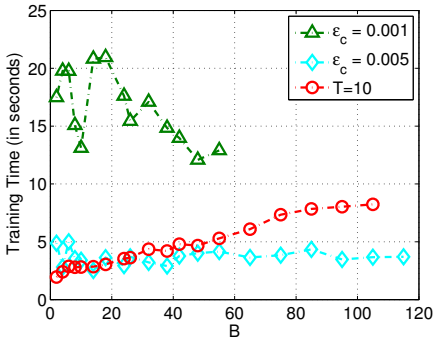
$[1, 100]$. In Section 8.4, we have set $C = 10$ for all data sets. Under this setting, FGM has demonstrated superb performance over the competing methods. On the contrary, choosing the regularization parameter for ℓ_1-methods is more difficult. In other words, FGM is more convenient to do model selections.



(a) # Selected Features Versus B



(b) Testing Accuracy Versus B



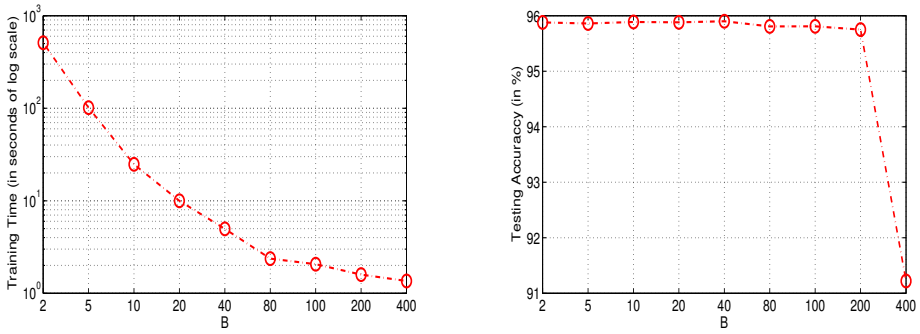
(c) Training Time (in seconds)

Figure 11: Sensitivity of the parameter B for FGM on `astro-ph` data set, where FGM is stopped once $(F(\boldsymbol{\omega}_{t-1}, b) - F(\boldsymbol{\omega}_t, b))/F(\boldsymbol{\omega}_0, b) \leq \epsilon_c$.

In the second experiment, we study the sensitivity of parameter B for FGM under two stopping conditions: (1) the condition $(F(\boldsymbol{\omega}_{t-1}, b) - F(\boldsymbol{\omega}_t, b))/F(\boldsymbol{\omega}_0, b) \leq \epsilon_c$ is achieved; (2) a maximum T iterations is achieved, where $T = 10$. Here, we test two values of ϵ_c, namely $\epsilon_c = 0.005$ and $\epsilon_c = 0.001$. The number of selected features, the testing accuracy and the training time versus different B are reported in Figure 11(a), 11(b) and 11(c), respectively.

In Figure 11(a), given the number of selected feature # features, the number of required iterations is about $\lceil \frac{\#\text{ features}}{B} \rceil$ under the first stopping criterion. In this sense, FGM with $\epsilon_c = 0.001$ takes more than 10 iterations to terminate, thus will choose more features. As a result, it needs more time for the optimization with the same B, as shown in Figure 11(c). On the contrary, FGM with $\epsilon_c = 0.005$ requires fewer number of iterations (smaller than 10 when $B > 20$). Surprisingly, as shown in Figure 11(b), FGM with fewer iterations (where $\epsilon_c = 0.005$ or $T = 10$) obtain similar testing accuracy with FGM using $\epsilon_c = 0.001$, but has much better training efficiency. This observation indicates that, we can set a small number

outer iterations (for example $5 \leq T \leq 20$) to trade-off the training efficiency and the feature selection performance.



(a) Training Time (in seconds) Versus B (b) Testing Accuracy Versus B

Figure 12: Sensitivity of the parameter B for FGM on `astro-ph` data set. Given a parameter B, we stop FGM once 400 features are selected.

In the third experiment, we study the influence of the parameter B on the performance of FGM on the `astro-ph` data set. For convenience of comparison, we stop FGM once 400 features are selected w.r.t. different B's.

The training time and testing accuracy w.r.t. different B's are shown in Figure 12(a) and 12(b), respectively. From Figure 12(a), choosing a large B in general leads to better training efficiency. Particularly, FGM with $B = 40$ is about 200 times faster than FGM with $B = 2$. Recall that, active set methods can be considered as special cases of FGM with $B = 1$ (Roth and Fischer, 2008; Bach, 2009). Accordingly, we can conclude that, FGM with a properly selected B can be much faster than active set methods. However, it should be pointed that, if B is too large, the performance may degrade. For instance, if we choose $B = 400$, the testing accuracy dramatically degrades, which indicates that the selected 400 features are not the optimal ones. In summary, choosing a suitable B (e.g. $B \leq 100$) can much improve the efficiency while maintaining promising generalization performance.

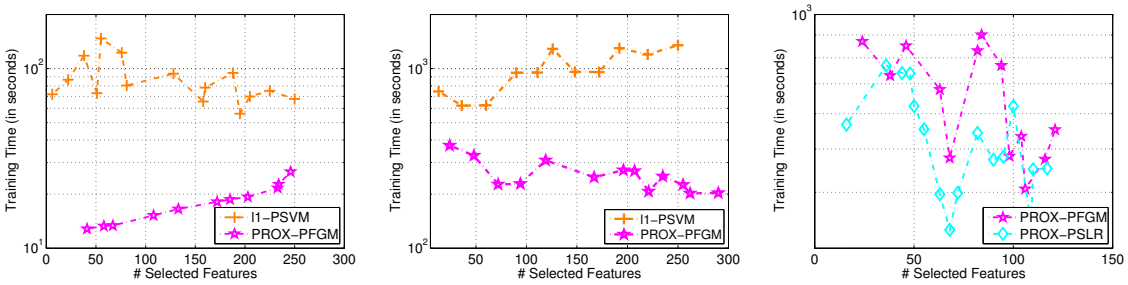
8.5 Ultrahigh Dimensional Feature Selection via Nonlinear Feature Mapping

In this experiment, we compare the efficiency of FGM and ℓ_1-SVM on nonlinear feature selections using polynomial feature mappings on two medium dimensional data sets and a high dimensional data set. The comparison methods are denoted by PROX-PFGM, PROX-PSLR and l1-PSVM, respectively.[12] The details of the studied data sets are shown in Table 2, where m_{Poly} denotes the dimension of the polynomial mappings and γ is the polynomial kernel parameter used in this experiment. The `mnist38` data set consists of the digital images of 3 and 8 from the `mnist` data set.[13] For the `kddb` data set, we only use the first 10^6 instances. Finally, we change the parameter C for l1-PSVM to obtain different number of features.

12. The codes of l1-PSVM are available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/#fast_training_testing_for_degree_2_polynomial_mappings_of_data`.
13. The data set is available from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

Data set	m	m_{Poly}	n_{train}	n_{test}	γ
mnist38	784	$O(10^5)$	40,000	22,581	4.0
real-sim	20,958	$O(10^8)$	32,309	40,000	8.0
kddb	4,590,807	$O(10^{14})$	1000, 000	748,401	4.0

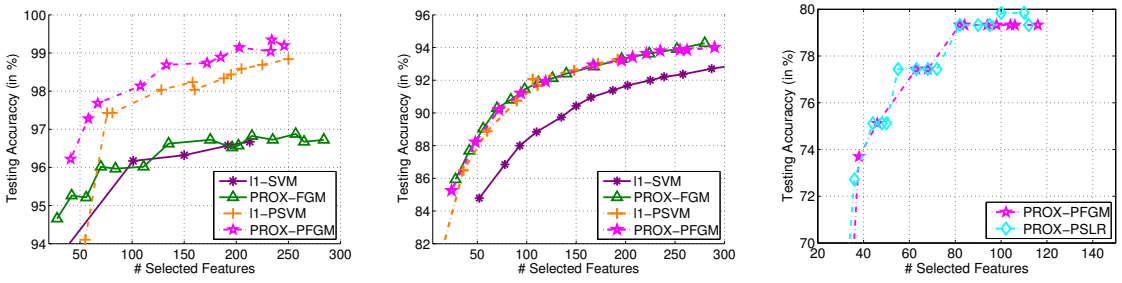
Table 2: Details of data sets using polynomial feature mappings.



(a) mnist38 (b) real-sim (c) kddb

Figure 13: Training time of different methods on nonlinear feature selection using polynomial mappings.

The training time and testing accuracy on different data sets are reported in Figure 13 and 14, respectively. Both PROX-PFGM and l1-PSVM can address the two medium dimensional problems. However, PROX-PFGM shows much better efficiency than l1-PSVM. Moreover, l1-PSVM is infeasible on the kddb data set due to the ultrahigh dimensionality. Particularly, in l1-PSVM, it needs more than 1TB memory to store a dense $\mathbf{w}$, which is infeasible for a common PC. Conversely, this difficulty can be effectively addressed by FGM. Specifically, PROX-PFGM completes the training within 1000 seconds.



(a) mnist38 (b) real-sim (c) kddb

Figure 14: Testing accuracy of different methods on nonlinear feature selection using polynomial mappings.

From the figures, the testing accuracy on mnist38 data set with polynomial mapping is much better than that of linear methods, which demonstrate the usefulness of the nonlinear

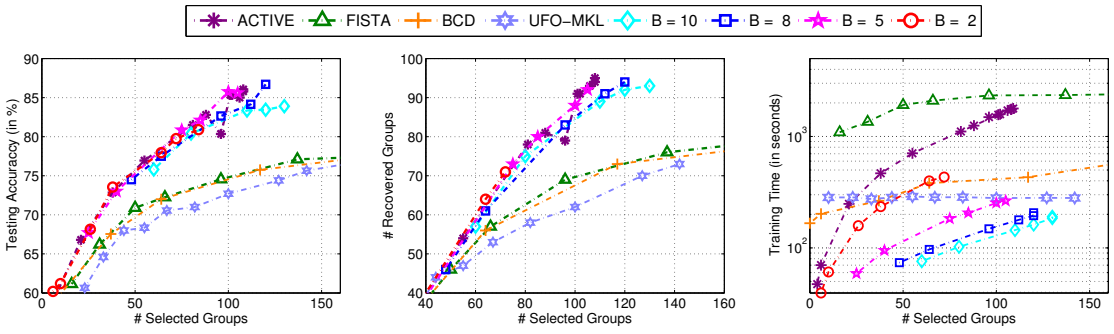
feature expansions. On the `real-sim` and `kddb` data sets, however, the performance with polynomial mapping does not show significant improvements. A possible reason is that these two data sets are linearly separable.

8.6 Experiments for Group Feature Selection

In this section, we study the performance of FGM for group feature selection on a synthetic data set and two real-world data sets. Here only the logistic loss is studied since it has been widely used for group feature selections on classification tasks (Roth and Fischer, 2008; Liu and Ye, 2010). To demonstrate the sensitivity of the parameter C to FGM, we vary C to select different number of groups under the stopping tolerance $\epsilon_c = 0.001$. For each C, we test $B \in \{2, 5, 8, 10\}$. The tradeoff parameter λ in SLEP is chosen from $[0, 1]$, where a larger lambda leads to more sparse solutions (Liu and Ye, 2010). Specifically, we set λ in $[0.002, 0.700]$ for FISTA and ACTIVE, and set λ in $[0.003, 0.1]$ for BCD.

8.6.1 Synthetic Experiments on Group Feature Selection

In the synthetic experiment, we generate a random matrix $\mathbf{X} \in \mathbb{R}^{4,096 \times 400,000}$ with each entry sampled from the i.i.d. Gaussian distribution $\mathcal{N}(0, 1)$. After that, we directly group the 400,000 features into 40,000 groups of equal size (Jenatton et al., 2011b), namely each feature group contains 10 features. We randomly choose 100 groups of them as the ground-truth informative groups. To this end, we generate a sparse vector $\mathbf{w}$, where only the entries of the selected groups are nonzero values sampled from the i.i.d. Gaussian distribution $\mathcal{N}(0, 1)$. Finally, we produce the output labels by $\mathbf{y} = \text{sign}(\mathbf{X}\mathbf{w})$. We generate 2,000 testing points in the same manner.



(a) Testing accuracy (b) Recovered features (c) Training time

Figure 15: Results of group feature selection on the synthetic data set.

The testing accuracy, training time and number of recovered ground-truth groups are reported in Figure 15(a), 15(b) and 15(c), respectively. Here only the results within 150 groups are included since we only have 100 informative ground-truth groups. From Figure 15(a), FGM achieves better testing accuracy than FISTA, BCD and UFO-MKL. The reason is that, FGM can reduce the group feature selection bias. From Figure 15(c), in general, FGM is much more efficient than FISTA and BCD. Interestingly, the active set method (denoted by ACTIVE) also shows good testing accuracy compared with FISTA

and BCD, but from Figure 15(c), its efficiency is limited since it only includes one element per iteration. Accordingly, when selecting a large number of groups on big data, its computational cost becomes unbearable. For UFO-MKL, although its training speed is fast, its testing accuracy is generally worse than others. Finally, with a fixed B for FGM, the number of selected groups will increase when C becomes large. This is because, with a larger C, one imposes more importance on the training errors, more groups are required to achieve lower empirical errors.

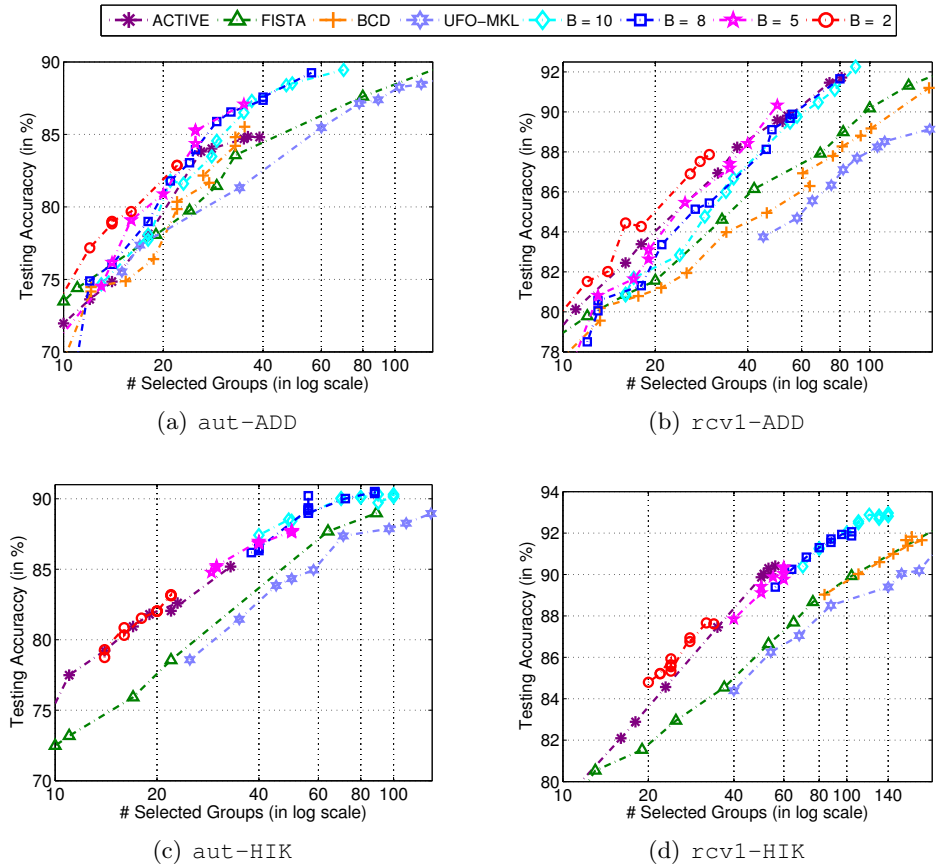
Data set	m	n_{train}	Size of training set (GB)			n_{test}	Size of testing set(GB)		
			Linear	ADD	HIK		Linear	ADD	HIK
aut	20,707	40, 000	0.027	0.320	0.408	22,581	0.016	0.191	0.269
rcv1	47,236	677,399	0.727	8.29	9.700	20,242	0.022	0.256	0.455

Table 3: Details of data sets used for HIK kernel feature expansion and Additive kernel feature expansion. For HIK kernel feature expansion, each original feature is represented by a group of 100 features; while for Additive kernel feature expansion, each original feature is represented by a group of 11 features.

8.6.2 Experiments on Real-World Data Sets

In this section, we study the effectiveness of FGM for group feature selection on two real-world data sets, namely `aut-avn` and `rcv1`. In real-world applications, the group prior of features comes in different ways. In this paper, we produce the feature groups using the explicit kernel feature expansions (Wu, 2012; Vedaldi and Zisserman, 2010), where each original feature is represented by a group of approximated features. Such expansion can vastly improve the training efficiency of kernel methods while keeping good approximation performance in many applications, such as in computer vision (Wu, 2012). For simplicity, we only study the HIK kernel expansion (Wu, 2012) and the additive Gaussian kernel expansion (Vedaldi and Zisserman, 2010). In the experiments, for fair comparisons, we pre-generate the explicit features for two data sets. The details of the original data sets and the expanded data sets are listed in Table 3. We can observe that, after the feature expansion, the storage requirements dramatically increase.

Figure 16 and 17 report the testing accuracy and training time of different methods, respectively. From Figure 16, FGM and the active set method achieve superior performance over FISTA, BCD and UFO-MKL in terms of testing accuracy. Moreover, from Figure 17, FGM gains much better efficiency than the active set method. It is worth mentioning that, due to the unbearable storage requirement, the feature expansion cannot be explicitly stored when dealing with ultrahigh dimensional big data. Accordingly, FISTA and BCD, which require the explicit presentation of data, cannot work in such cases. On the contrary, the proposed feature generating paradigm can effectively address this computational issue since it only involves a sequence of small-scale optimization problems.



(a) aut-ADD (b) rcv1-ADD

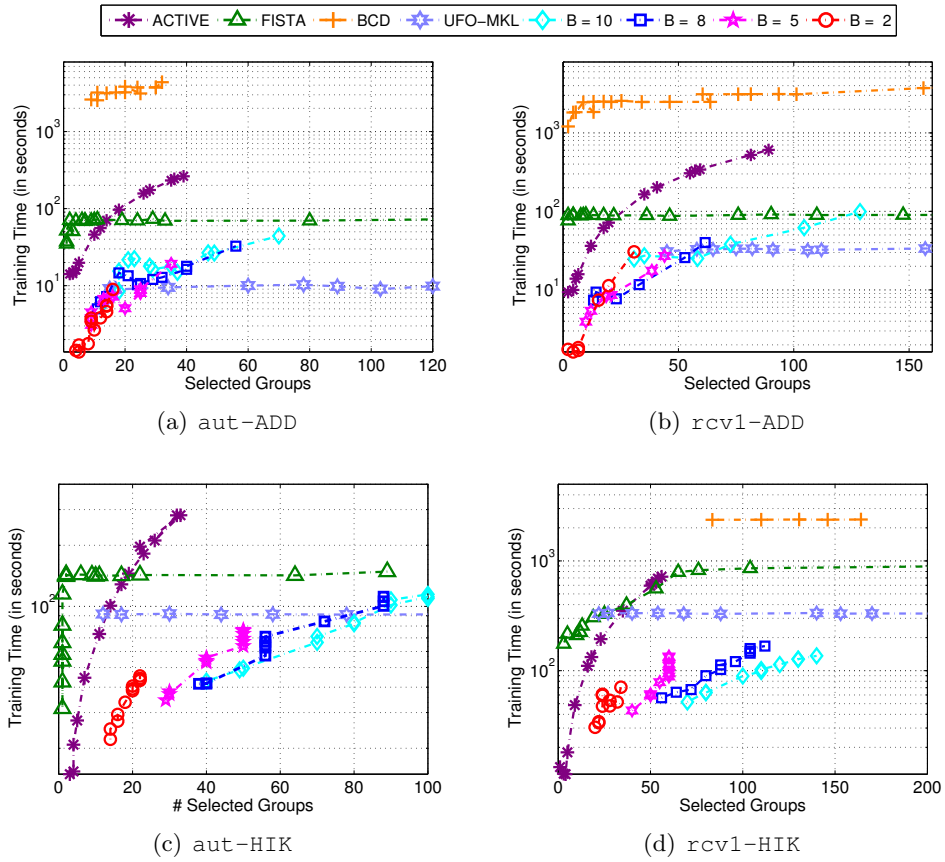
(c) aut-HIK (d) rcv1-HIK

Figure 16: Testing accuracy on group feature selection tasks. The groups are generated by HIK or additive feature mappings. The results of BCD on aut-HIK is not reported due to the heavy computational cost.

9. Conclusions

In this paper, an adaptive feature scaling (AFS) scheme has been proposed to conduct feature selection tasks. Specifically, to explicitly control the number features to be selected, we first introduce a vector $\mathbf{d} \in [0,1]^m$ to scale the input features, and then impose an ℓ_1-norm constraint $||\mathbf{d}||_1 \leq B$, where B represents the least number of features to be selected. Although the resultant problem is non-convex, we can transform it into an equivalent convex SIP problem. After that, a feature generating machine (FGM) is proposed to solve the SIP problem, which essentially includes B informative features per iteration and solves a sequence of much reduced MKL subproblems. The global convergence of FGM has been verified. Moreover, to make FGM scalable to big data, we propose to solve the primal form of the MKL subproblem through a modified APG method. Some efficient cache techniques are also developed to further improve the training efficiency. Finally, FGM has been extended to perform group feature selection and multiple kernel learning w.r.t. additive kernels.

FGM has two major advantages over the ℓ_1-norm methods and other existing feature selection methods. Firstly, with a separate control of the model complexity and sparsity,



(a) aut-ADD (b) rcv1-ADD

(c) aut-HIK (d) rcv1-HIK

Figure 17: Training time on group feature selections.

FGM can effectively handle the feature selection bias issue. Secondly, since only a small subset of features or kernels are involved in the subproblem optimization, FGM is particularly suitable for the ultrahigh dimensional feature selection task on big data, for which most of the existing methods are infeasible. It is worth mentioning that, unlike most of the existing methods, FGM avoids the storing of all base kernels or the full explicit feature mappings. Therefore, it can vastly reduce the unbearable memory demands of MKL with many base kernels or the nonlinear feature selection with ultrahigh-dimensional feature mappings.

Comprehensive experiments have been conducted to study the performance of the proposed methods on both linear feature selection and group feature selection tasks. Extensive experiments on synthetic data sets and real-world data sets have demonstrated the superior performance of FGM over the baseline methods in terms of both training efficiency and testing accuracy.

In this paper, the proposed methods have tackled big data problems with million training examples ($O(10^7)$) and 100 trillion features ($O(10^{14})$). Recall that the subproblems of FGM can be possibly addressed through SGD methods, we will explore SGD methods in the future to further improve the training efficiency over bigger data with ultra-large sample size.

Acknowledgments

We would like to acknowledge the valuable comments and useful suggestions by the Action Editor and the four anonymous reviewers. We would like to express our gratitude to Dr. Xinxing Xu and Dr. Shijie Xiao for the proofreading and comments. This research was partially supported by the Nanyang Technological University, the ASTAR Thematic Strategic Research Programme (TSRP) Grant No. 1121720013, and the Australian Research Council Future Fellowship FT130100746.

Appendix A. Proof of Theorem 3

Proof The proof parallels the results of Bach et al. (2004), and is based on the conic duality theory. Let $\Omega(\boldsymbol{\omega}) = \frac{1}{2}\left(\|\boldsymbol{\omega}_h\|\right)^2$ and define the cone $\mathcal{Q}_B = \{(\mathbf{u}, v) \in \mathbb{R}^{B+1}, \|\mathbf{u}\|_2 \leq v\}$. Furthermore, let $z_h = \|\boldsymbol{\omega}_h\|$, we have $\Omega(\boldsymbol{\omega}) = \frac{1}{2}\left(\sum_{h=1}^{t}\|\boldsymbol{\omega}_t\|\right)^2 = \frac{1}{2}z^2$ with $z = \sum_{h=1}^{t} z_h$. Apparently, we have $z_h \geq 0$ and $z \geq 0$. Finally, problem (22) can be transformed to the following problem:

$$\min_{z,\boldsymbol{\omega}} \quad \frac{1}{2}z^2 + P(\boldsymbol{\omega}, b), \quad \text{s.t. } \sum_{h=1}^{t} z_h \leq z, \quad (\boldsymbol{\omega}_t, z_h) \in \mathcal{Q}_B, \tag{33}$$

where $\boldsymbol{\omega} = [\boldsymbol{\omega}_1', ..., \boldsymbol{\omega}_t']'$. The Lagrangian function of (33) regarding the squared hinge loss can be written as:

$$\begin{aligned}&\mathcal{L}(z, \boldsymbol{\omega}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \gamma, \boldsymbol{\zeta}, \boldsymbol{\varpi})\\ &= \frac{1}{2}z^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 - \sum_{i=1}^{n}\alpha_i\left(y_i(\sum \boldsymbol{\omega}_h'\mathbf{x}_{ih} - b) - 1 + \xi_i\right) + \gamma(\sum_{h=1}^{t} z_h - z) - \sum_{h=1}^{t}(\boldsymbol{\zeta}_h'\boldsymbol{\omega}_h + \varpi_h z_h),\end{aligned}$$

where $\boldsymbol{\alpha}$, γ, $\boldsymbol{\zeta}_t$ and ϖ_t are the Lagrangian dual variables to the corresponding constraints. The KKT condition can be expressed as

$$\begin{array}{ll}
\nabla_z\mathcal{L} = z - \gamma = 0 & \Rightarrow z = \gamma;\\
\nabla_{z_h}\mathcal{L} = \gamma - \varpi_h = 0 & \Rightarrow \varpi_h = \gamma;\\
\nabla_{\boldsymbol{\omega}_h}\mathcal{L} = -\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{ih} - \boldsymbol{\zeta}_h = 0 & \Rightarrow \boldsymbol{\zeta}_h = -\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{ih};\\
\nabla_{\xi_i}\mathcal{L} = C\xi_i - \alpha_i = 0 & \Rightarrow \xi_i = \frac{\alpha_i}{C};\\
\|\boldsymbol{\zeta}_h\| \leq \varpi_h & \Rightarrow \|\boldsymbol{\zeta}_h\| \leq \gamma;\\
\nabla_b\mathcal{L} = 0 & \Rightarrow \sum_{i=1}^{n}\alpha_i y_i = 0.
\end{array}$$

By substituting the above equations into the Lagrangian function, we have

$$\mathcal{L}(z, \boldsymbol{\omega}, \boldsymbol{\alpha}, \gamma, \boldsymbol{\zeta}, \boldsymbol{\varpi}) = -\frac{1}{2}\gamma^2 - \frac{1}{2C}\boldsymbol{\alpha}'\boldsymbol{\alpha} + 1'\boldsymbol{\alpha}.$$

Hence the dual problem of the $\ell_{2,1}^2$-regularized problem regarding squared hinge loss can be written as:

$$\begin{aligned} \max_{\gamma,\boldsymbol{\alpha}} \quad & -\frac{1}{2}\gamma^2 - \frac{1}{2C}\boldsymbol{\alpha}'\boldsymbol{\alpha} + 1'\boldsymbol{\alpha} \\ \text{s.t} \quad & \Big\| \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_{ih} \Big\| \leq \gamma, \quad h = 1, \cdots, t, \\ & \sum_{i=1}^{n} \alpha_i y_i = 0, \alpha_i \geq 0, \quad i = 1, \cdots, n. \end{aligned}$$

Let $\theta = \frac{1}{2}\gamma^2 + \frac{1}{2C}\boldsymbol{\alpha}'\boldsymbol{\alpha} - \boldsymbol{\alpha}'\mathbf{1}$, $\boldsymbol{\omega}_h = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_{ih}$ and $f(\boldsymbol{\alpha}, \mathbf{d}_h) = \frac{1}{2}\|\boldsymbol{\omega}_h\|^2 + \frac{1}{2C}\boldsymbol{\alpha}'\boldsymbol{\alpha} - \boldsymbol{\alpha}'\mathbf{1}$, we have

$$\begin{aligned} \max_{\theta,\boldsymbol{\alpha}} \quad & -\theta, \\ \text{s.t} \quad & f(\boldsymbol{\alpha}, \mathbf{d}_h) \leq \theta, \quad h = 1, \cdots, t, \\ & \sum_{i=1}^{n} \alpha_i y_i = 0, \alpha_i \geq 0, \quad i = 1, \cdots, n. \end{aligned}$$

which indeed is in the form of problem (16) by letting $\mathcal{A}$ be the domain of $\boldsymbol{\alpha}$. This completes the proof and brings the connection between the primal and dual formulation.

By defining $0\log(0) = 0$, with the similar derivation above, we can obtain the dual form of (33) regarding the logistic loss. Specifically, the Lagrangian function of (33) w.r.t. the logistic loss is:

$$\begin{aligned} &\mathcal{L}(z, \boldsymbol{\omega}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \gamma, \boldsymbol{\zeta}, \boldsymbol{\varpi}) \\ &= \frac{1}{2}z^2 + C\sum_{i=1}^{n} \log(1 + \exp(\xi_i)) - \sum_{i=1}^{n} \alpha_i \left(y_i(\sum \boldsymbol{\omega}_h' \mathbf{x}_{ih} - b) + \xi_i \right) + \gamma(\sum_{h=1}^{t} z_h - z) - \sum_{h=1}^{t} (\boldsymbol{\zeta}_h' \boldsymbol{\omega}_h + \varpi_h z_h), \end{aligned}$$

where $\boldsymbol{\alpha}$, γ, $\boldsymbol{\zeta}_t$ and ϖ_t are the Lagrangian dual variables to the corresponding constraints. The KKT condition can be expressed as

$$\begin{array}{ll} \nabla_z \mathcal{L} = z - \gamma = 0 & \Rightarrow z = \gamma; \\ \nabla_{z_h} \mathcal{L} = \gamma - \varpi_h = 0 & \Rightarrow \varpi_h = \gamma; \\ \nabla_{\boldsymbol{\omega}_h} \mathcal{L} = -\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_{ih} - \boldsymbol{\zeta}_h = 0 & \Rightarrow \boldsymbol{\zeta}_h = -\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_{ih}; \\ \nabla_{\xi_i} \mathcal{L} = \frac{C\exp(\xi_i)}{1+\exp(\xi_i)} - \alpha_i = 0 & \Rightarrow \exp(\xi_i) = \frac{\alpha_i}{C - \alpha_i}; \\ \|\boldsymbol{\zeta}_h\| \leq \varpi_h & \Rightarrow \|\boldsymbol{\zeta}_h\| \leq \gamma; \\ \nabla_b \mathcal{L} = 0 & \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0. \end{array}$$

By substituting all the above results into the Lagrangian function, we have

$$\mathcal{L}(z, \boldsymbol{\omega}, \boldsymbol{\alpha}, \gamma, \boldsymbol{\zeta}, \boldsymbol{\varpi}) = -\frac{1}{2}\gamma^2 - \sum_{i=1}^{n} (C - \alpha_i)\log(C - \alpha_i) - \sum_{i=1}^{n} \alpha_i \log(\alpha_i).$$

The dual form of the $\ell_{2,1}^2$-regularized problem regarding logistic loss can be written as:

$$\begin{aligned}\max_{\gamma,\boldsymbol{\alpha}} \quad & -\frac{1}{2}\gamma^2 - \sum_{i=1}^{n}(C-\alpha_i)\log(C-\alpha_i) - \sum_{i=1}^{n}\alpha_i\log(\alpha_i) \\ \text{s.t.} \quad & \Big\|\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{ih}\Big\| \leq \gamma, \quad h=1,\cdots,t, \\ & \sum_{i=1}^{n}\alpha_i y_i = 0, \alpha_i \geq 0, \quad i=1,\cdots,n.\end{aligned}$$

Let $\theta = \frac{1}{2}\gamma^2 + \sum_{i=1}^{n}(C-\alpha_i)\log(C-\alpha_i) + \sum_{i=1}^{n}\alpha_i\log(\alpha_i)$, $\boldsymbol{\omega}_h = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_{ih}$, $f(\boldsymbol{\alpha}, \mathbf{d}_h) = \frac{1}{2}\|\boldsymbol{\omega}_h\|^2 + \sum_{i=1}^{n}(C-\alpha_i)\log(C-\alpha_i) + \sum_{i=1}^{n}\alpha_i\log(\alpha_i)$, then we have

$$\begin{aligned}\max_{\theta,\boldsymbol{\alpha}} \quad & -\theta, \\ \text{s.t.} \quad & f(\boldsymbol{\alpha}, \mathbf{d}_h) \leq \theta, \quad h=1,\cdots,t, \\ & \sum_{i=1}^{n}\alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i=1,\cdots,n.\end{aligned}$$

Finally, according to the KKT condition, we can easily recover the dual variable $\boldsymbol{\alpha}$ by $\alpha_i = \frac{C\exp(\xi_i)}{1+\exp(\xi_i)}$. This completes the proof. ■

Appendix B. Proof of Theorem 4

The proof parallels the results of Beck and Teboulle (2009), and includes several lemmas. First of all, we define a one variable function $Q_{\tau_b}(\mathbf{v}, b, v_b)$ w.r.t. b as

$$Q_{\tau_b}(\mathbf{v}, b, v_b) \quad = \quad P(\mathbf{v}, v_b) + \langle \nabla_b P(\mathbf{v}, v_b), b - v_b\rangle + \frac{\tau_b}{2}\|b - v_b\|^2, \tag{34}$$

where we abuse the operators $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$ for convenience.

Lemma 4 *$S_\tau(\mathbf{u}, \mathbf{v}) = \arg\min_{\boldsymbol{\omega}} Q_\tau(\boldsymbol{\omega}, \mathbf{v}, v_b)$ is the minimizer of problem (23) at point $\mathbf{v}$, if and only if there exists $g(S_\tau(\mathbf{u}, \mathbf{v})) \in \partial\Omega(S_\tau(\mathbf{u}, \mathbf{v}))$, the subgradient of $\Omega(\boldsymbol{\omega})$ at $S_\tau(\mathbf{u}, \mathbf{v})$, such that*

$$g(S_\tau(\mathbf{u}, \mathbf{v})) + \tau(S_\tau(\mathbf{u}, \mathbf{v}) - \mathbf{v}) + \nabla P(\mathbf{v}) = \mathbf{0}.$$

Proof The proof can be completed by the optimality condition of $Q_\tau(\boldsymbol{\omega}, \mathbf{v}, v_b)$ w.r.t. $\boldsymbol{\omega}$. ■

Lemma 5 *Let $S_\tau(\mathbf{u}, \mathbf{v}) = \arg\min_{\boldsymbol{\omega}} Q_\tau(\boldsymbol{\omega}, \mathbf{v}, v_b)$ be the minimizer of problem (23) at point $\mathbf{v}$, and $S_{\tau_b}(b) = \arg\min_b Q_{\tau_b}(\mathbf{v}, b, v_b)$ be the minimizer of problem (34) at point v_b. Due to the line search in Algorithm 4, we have*

$$\begin{aligned}& F(S_\tau(\mathbf{u}, \mathbf{v}), v_b) \leq Q_\tau(S_\tau(\mathbf{u}, \mathbf{v}), \mathbf{v}, v_b). \\ & P(\mathbf{v}, S_{\tau_b}(v_b)) \leq Q_{\tau_b}(\mathbf{v}, S_{\tau_b}(v_b), v_b).\end{aligned}$$

and

$$F(S_\tau(\mathbf{u},\mathbf{v}),S_{\tau_b}(b)) \le Q_\tau(S_\tau(\mathbf{u},\mathbf{v}),\mathbf{v},v_b) + \langle \nabla_b P(\mathbf{v},v_b), S_{\tau_b}(b)-v_b\rangle + \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b\|^2. \quad (35)$$

Furthermore, for any $(\boldsymbol{\omega}',b)'$ *we have*

$$\begin{aligned} F(\boldsymbol{\omega},b) - F(S_\tau(\mathbf{u},\mathbf{v}),S_{\tau_b}(b)) \;\ge\; & \tau_b\langle S_{\tau_b}(b)-v_b, v_b-b\rangle + \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b\|^2 \\ & +\tau\langle S_\tau(\mathbf{u},\mathbf{v})-\mathbf{v},\mathbf{v}-\boldsymbol{\omega}\rangle + \frac{\tau}{2}\|S_\tau(\mathbf{u},\mathbf{v})-\mathbf{v}\|^2. \quad (36)\end{aligned}$$

Proof We only prove the inequality (35) and (36). First of all, recall that in Algorithm 4, we update $\boldsymbol{\omega}$ and b separately. It follows that

$$\begin{aligned} & F(S_\tau(\mathbf{u},\mathbf{v}),S_{\tau_b}(b)) \\ = \;& \Omega(S_\tau(\mathbf{u},\mathbf{v})) + P(S_\tau(\mathbf{u},\mathbf{v}),S_{\tau_b}(v_b)) \\ \le \;& \Omega(S_\tau(\mathbf{u},\mathbf{v})) + Q_{\tau_b}(S_\tau(\mathbf{u},\mathbf{v}),S_{\tau_b}(b),v_b) \\ = \;& \Omega(S_\tau(\mathbf{u},\mathbf{v})) + P(S_\tau(\mathbf{u},\mathbf{v}),v_b) + \langle \nabla_b P(\mathbf{v},v_b), S_{\tau_b}(b)-v_b\rangle + \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b\|^2 \\ = \;& F(S_\tau(\mathbf{u},\mathbf{v}),v_b) + \langle \nabla_b P(\mathbf{v},v_b), S_{\tau_b}(b)-v_b\rangle + \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b\|^2 \\ \le \;& Q_\tau(S_\tau(\mathbf{u},\mathbf{v}),\mathbf{v},v_b) + \langle \nabla_b P(\mathbf{v},v_b), S_{\tau_b}(b)-v_b\rangle + \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b\|^2. \end{aligned}$$

This proves the inequality in (35).

Now we prove the inequality (36). First of all, since both $P(\boldsymbol{\omega},b)$ and $\Omega(\boldsymbol{\omega})$ are convex functions, we have

$$\begin{aligned} & P(\boldsymbol{\omega},b) \ge P(\mathbf{v},v_b) + \langle \nabla P(\mathbf{v}), \boldsymbol{\omega}-\mathbf{v}\rangle + \langle \nabla_b P(\mathbf{v},v_b), b-v_b\rangle, \\ & \Omega(\boldsymbol{\omega}) \ge \Omega(S_\tau(\mathbf{u},\mathbf{v})) + \langle \boldsymbol{\omega}-S_\tau(\mathbf{u},\mathbf{v}), g(S_\tau(\mathbf{g},\mathbf{v}))\rangle, \end{aligned}$$

where $g(S_\tau(\mathbf{u},\mathbf{v}))$ be the subgradient of $\Omega(\boldsymbol{\omega})$ at point $S_\tau(\mathbf{u},\mathbf{v})$. Summing up the above inequalities, we obtain

$$\begin{aligned} & F(\boldsymbol{\omega},b) \\ \ge \;& P(\mathbf{v},v_b) + \langle \nabla P(\mathbf{v}), \boldsymbol{\omega}-\mathbf{v}\rangle + \langle \nabla_b P(\mathbf{v},v_b), b-v_b\rangle + \Omega(S_\tau(\mathbf{u},\mathbf{v})) + \langle \boldsymbol{\omega}-S_\tau(\mathbf{u},\mathbf{v}), g(S_\tau(\mathbf{g},\mathbf{v}))\rangle, \end{aligned}$$

In addition, we have

$$\begin{aligned} & F(\boldsymbol{\omega},b) - \Big(Q_\tau(S_\tau(\mathbf{u},\mathbf{v}),\mathbf{v},v_b) + \langle \nabla_b P(\mathbf{v},v_b), S_{\tau_b}(b)-v_b\rangle + \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b\|^2\Big) \\ = \;& P(\mathbf{v},v_b) + \langle \nabla P(\mathbf{v}), \boldsymbol{\omega}-\mathbf{v}\rangle + \langle \nabla_b P(\mathbf{v},v_b), b-v_b\rangle + \Omega(S_\tau(\mathbf{u},\mathbf{v})) + \langle \boldsymbol{\omega}-S_\tau(\mathbf{u},\mathbf{v}), g(S_\tau(\mathbf{g},\mathbf{v}))\rangle, \\ & -\Big(Q_\tau(S_\tau(\mathbf{u},\mathbf{v}),\mathbf{v},v_b) + \langle \nabla_b P(\mathbf{v},v_b), S_{\tau_b}(b)-v_b\rangle + \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b\|^2\Big) \\ = \;& P(\mathbf{v},v_b) + \langle \nabla P(\mathbf{v}), \boldsymbol{\omega}-\mathbf{v}\rangle + \langle \nabla_b P(\mathbf{v},v_b), b-v_b\rangle + \Omega(S_\tau(\mathbf{u},\mathbf{v})) + \langle \boldsymbol{\omega}-S_\tau(\mathbf{u},\mathbf{v}), g(S_\tau(\mathbf{g},\mathbf{v}))\rangle, \\ & -\Big(P(\mathbf{v},v_b) + \langle \nabla P(\mathbf{v}), S_\tau(\mathbf{u},\mathbf{v})-\mathbf{v}\rangle + \Omega(S_\tau(\mathbf{u},\mathbf{v})) + \frac{\tau}{2}\|S_\tau(\mathbf{u},\mathbf{v})-\mathbf{v}\|^2 + \langle \nabla_b P(\mathbf{v},v_b), S_{\tau_b}(b)-v_b\rangle \\ & \quad + \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b\|^2\Big) \\ = \;& \langle \nabla P(\mathbf{v}) + g(S_\tau(\mathbf{g},\mathbf{v})), \boldsymbol{\omega}-S_\tau(\mathbf{u},\mathbf{v})\rangle - \frac{\tau}{2}\|S_\tau(\mathbf{u},\mathbf{v})-\mathbf{v}\|^2 \\ & +\langle \nabla_b P(\mathbf{v},v_b), b-S_{\tau_b}(b)\rangle - \frac{\tau_b}{2}\|S_{\tau_b}(b)-v_b\|^2. \end{aligned}$$

With the relation $S_{\tau_b}(b) = b - \frac{\nabla_b P(\mathbf{v}, v_b)}{\tau_b}$ and Lemma 4, we obtain

$$\begin{aligned}
&F(\boldsymbol{\omega}, b) - F(S_\tau(\mathbf{u}, \mathbf{v}), S_{\tau_b}(b)) \\
&\geq F(\boldsymbol{\omega}, b) - \left(Q_\tau(S_\tau(\mathbf{u}, \mathbf{v}), \mathbf{v}, v_b) + \langle \nabla_b P(\mathbf{v}, v_b), S_{\tau_b}(b) - v_b \rangle + \frac{\tau_b}{2} \|S_{\tau_b}(b) - v_b\|^2 \right). \\
&\geq \langle \nabla P(\mathbf{v}) + g(S_\tau(\mathbf{g}, \mathbf{v})), \boldsymbol{\omega} - S_\tau(\mathbf{u}, \mathbf{v}) \rangle - \frac{\tau}{2} \|S_\tau(\mathbf{u}, \mathbf{v}) - \mathbf{v}\|^2 \\
&\quad + \langle \nabla_b P(\mathbf{v}, v_b), b - S_{\tau_b}(b) \rangle - \frac{\tau_b}{2} \|S_{\tau_b}(b) - v_b\|^2. \\
&= \tau \langle \mathbf{v} - S_\tau(\mathbf{u}, \mathbf{v}), \boldsymbol{\omega} - S_\tau(\mathbf{u}, \mathbf{v}) \rangle - \frac{\tau}{2} \|S_\tau(\mathbf{u}, \mathbf{v}) - \mathbf{v}\|^2 \\
&\quad + \tau_b \langle v_b - S_{\tau_b}(b), b - S_{\tau_b}(b) \rangle - \frac{\tau_b}{2} \|S_{\tau_b}(b) - v_b\|^2. \\
&= \tau \langle S_\tau(\mathbf{u}, \mathbf{v}) - \mathbf{v}, \mathbf{v} - \boldsymbol{\omega} \rangle + \frac{\tau}{2} \|S_\tau(\mathbf{u}, \mathbf{v}) - \mathbf{v}\|^2 \\
&\quad + \tau_b \langle S_{\tau_b}(b) - v_b, v_b - b \rangle + \frac{\tau_b}{2} \|S_{\tau_b}(b) - v_b\|^2.
\end{aligned}$$

This completes the proof. ■

Lemma 6 *Let $L_{bt} = \sigma L_t$, where $\sigma > 0$. Furthermore, let us define*

$$\begin{aligned}
\mu^k &= F(\boldsymbol{\omega}^k, b^k) - F(\boldsymbol{\omega}^*, b^*), \\
\boldsymbol{\nu}^k &= \rho^k \boldsymbol{\omega}^k - (\rho^k - 1)\boldsymbol{\omega}^{k-1} - \boldsymbol{\omega}^*, \\
\upsilon^k &= \rho^k b^k - (\rho^k - 1) b^{k-1} - b^*,
\end{aligned}$$

and then the following relation holds:

$$\frac{2(\rho^k)^2 \mu^k}{L^k} - \frac{(\rho^{k+1})^2 \mu^{k+1}}{L^{k+1}} \geq \left(\|\boldsymbol{\nu}^{k+1}\|^2 - \|\boldsymbol{\nu}^k\|^2 \right) + \sigma \left((\upsilon^{k+1})^2 - (\upsilon^k)^2 \right).$$

Proof Note that we have $\boldsymbol{\omega}^{k+1} = S_\tau(\mathbf{u}, \mathbf{v}^{k+1})$ and $b^{k+1} = S_{\tau_b}(v_b^{k+1})$. By applying Lemma 5, let $\boldsymbol{\omega} = \boldsymbol{\omega}^k$, $\mathbf{v} = \mathbf{v}^{k+1}, \tau = L^{k+1}$, $b = b^k$, $v_b = v_b^{k+1}, \tau_b = L_b^{k+1}$, we have

$$\begin{aligned}
2(\mu^k - \mu^{k+1}) \quad &\geq \quad L^{k+1} \left(\|\boldsymbol{\omega}^{k+1} - \mathbf{v}^{k+1}\|^2 + 2\langle \boldsymbol{\omega}^{k+1} - \mathbf{v}^{k+1}, \mathbf{v}^{k+1} - \boldsymbol{\omega}^k \rangle \right) \\
&\quad + L_b^{k+1} \left(\|b^{k+1} - v_b^{k+1}\|^2 + 2\langle b^{k+1} - v_b^{k+1}, v_b^{k+1} - b^k \rangle \right).
\end{aligned}$$

Multiplying both sides by $(\rho^{k+1} - 1)$, we obtain

$$\begin{aligned}
2(\rho^{k+1} - 1)(\mu^k - \mu^{k+1}) \quad &\geq \quad L^{k+1}(\rho^{k+1} - 1) \left(\|\boldsymbol{\omega}^{k+1} - \mathbf{v}^{k+1}\|^2 + 2\langle \boldsymbol{\omega}^{k+1} - \mathbf{v}^{k+1}, \mathbf{v}^{k+1} - \boldsymbol{\omega}^k \rangle \right) \\
&\quad + L_b^{k+1}(\rho^{k+1} - 1) \left(\|b^{k+1} - v_b^{k+1}\|^2 + 2\langle b^{k+1} - v_b^{k+1}, v_b^{k+1} - b^k \rangle \right).
\end{aligned}$$

Also, let $\boldsymbol{\omega} = \boldsymbol{\omega}^*$, $\mathbf{v} = \mathbf{v}^{k+1}, \tau = L^{k+1}$, $b = b^k$, $v_b = v_b^{k+1}$, and $\tau_b = L_b^{k+1}$, we have

$$\begin{aligned}
-2\mu^{k+1} \quad &\geq \quad L^{k+1} \left(\|\boldsymbol{\omega}^{k+1} - \mathbf{v}^{k+1}\|^2 + 2\langle \boldsymbol{\omega}^{k+1} - \mathbf{v}^{k+1}, \mathbf{v}^{k+1} - \boldsymbol{\omega}^* \rangle \right) \\
&\quad + L_b^{k+1} \left(\|b^{k+1} - v_b^{k+1}\|^2 + 2\langle b^{k+1} - v_b^{k+1}, v_b^{k+1} - b^* \rangle \right).
\end{aligned}$$

Summing up the above two inequalities, we get

$$\begin{aligned}
&2\left((\rho^{k+1} - 1)\mu^k - \rho^{k+1}\mu^{k+1} \right) \\
&\geq \quad L^{k+1} \left(\rho^{k+1} \|\boldsymbol{\omega}^{k+1} - \mathbf{v}^{k+1}\|^2 + 2\langle \boldsymbol{\omega}^{k+1} - \mathbf{v}^{k+1}, \rho^{k+1}\mathbf{v}^{k+1} - (\rho^{k+1} - 1)\boldsymbol{\omega}^k - \boldsymbol{\omega}^* \rangle \right) \\
&\quad + L_b^{k+1} \left(\rho^{k+1} \|b^{k+1} - v_b^{k+1}\|^2 + 2\langle b^{k+1} - v_b^{k+1}, \rho^{k+1} v_b^{k+1} - (\rho^{k+1} - 1) b^k - b^* \rangle \right).
\end{aligned}$$

Multiplying both sides by ρ^{k+1}, we obtain

$$\begin{aligned}
&2\left(\rho^{k+1}(\rho^{k+1}-1)\mu^k-(\rho^{k+1})^2\mu^{k+1}\right)\\
&\quad\geq\ L^{k+1}\left((\rho^{k+1})^2||\boldsymbol{\omega}^{k+1}-\mathbf{v}^{k+1}||^2+2\rho^{k+1}\langle\boldsymbol{\omega}^{k+1}-\mathbf{v}^{k+1},\rho^{k+1}\mathbf{v}^{k+1}-(\rho^{k+1}-1)\boldsymbol{\omega}^k-\boldsymbol{\omega}^*\rangle\right)\\
&\qquad+L_b^{k+1}\left((\rho^{k+1})^2||b^{k+1}-v_b^{k+1}||^2+2\rho^{k+1}\langle b^{k+1}-v_b^{k+1},\rho^{k+1}v_b^{k+1}-(\rho^{k+1}-1)b^k-b^*\rangle\right).
\end{aligned}$$

Since $(\rho^k)^2=(\rho^{k+1})^2-\rho^{k+1}$, it follows that

$$\begin{aligned}
&2\left((\rho^k)^2\mu^k-(\rho^{k+1})^2\mu^{k+1}\right)\\
&\quad\geq\ L^{k+1}\left((\rho^{k+1})^2||\boldsymbol{\omega}^{k+1}-\mathbf{v}^{k+1}||^2+2\rho^{k+1}\langle\boldsymbol{\omega}^{k+1}-\mathbf{v}^{k+1},\rho^{k+1}\mathbf{v}^{k+1}-(\rho^{k+1}-1)\boldsymbol{\omega}^k-\boldsymbol{\omega}^*\rangle\right)\\
&\qquad+L_b^{k+1}\left((\rho^{k+1})^2||b^{k+1}-v_b^{k+1}||^2+2\rho^{k+1}\langle b^{k+1}-v_b^{k+1},\rho^{k+1}v_b^{k+1}-(\rho^{k+1}-1)b^k-b^*\rangle\right).
\end{aligned}$$

By applying the equality $||\mathbf{u}-\mathbf{v}||^2+2\langle\mathbf{u}-\mathbf{v},\mathbf{v}-\mathbf{w}\rangle=||\mathbf{u}-\mathbf{w}||^2-||\mathbf{v}-\mathbf{w}||^2$, we have

$$\begin{aligned}
&2\left((\rho^k)^2\mu^k-(\rho^{k+1})^2\mu^{k+1}\right)\\
&\quad\geq\ L^{k+1}\left(||\rho^{k+1}\boldsymbol{\omega}^{k+1}-(\rho^{k+1}-1)\boldsymbol{\omega}^k-\boldsymbol{\omega}^*||^2-||\rho^{k+1}\mathbf{v}^{k+1}-(\rho^{k+1}-1)\boldsymbol{\omega}^k-\boldsymbol{\omega}^*||^2\right)\\
&\qquad+L_b^{k+1}\left(||\rho^{k+1}b^{k+1}-(\rho^{k+1}-1)b^k-b^*||^2-||\rho^{k+1}v_b^{k+1}-(\rho^{k+1}-1)b^k-b^*||^2\right).
\end{aligned}$$

With $\rho^{k+1}\mathbf{v}^{k+1}=\rho^{k+1}\boldsymbol{\omega}^k+(\rho^k-1)(\boldsymbol{\omega}^k-\boldsymbol{\omega}^{k-1})$, $\rho^{k+1}v_b^{k+1}=\rho^{k+1}b^k+(\rho^k-1)(b^k-b^{k-1})$ and the definition of $\boldsymbol{\nu}^k$, it follows that

$$2\left((\rho^k)^2\mu^k-(\rho^{k+1})^2\mu^{k+1}\right)\geq L^{k+1}\left(||\boldsymbol{\nu}^{k+1}||^2-||\boldsymbol{\nu}^k||^2\right)+L_b^{k+1}\left((\upsilon^{k+1})^2-(\upsilon^k)^2\right).$$

Assuming that there exists a $\sigma>0$ such that $L_b^{k+1}=\sigma L^{k+1}$, we get

$$\frac{2\left((\rho^k)^2\mu^k-(\rho^{k+1})^2\mu^{k+1}\right)}{L^{k+1}}\geq\left(||\boldsymbol{\nu}^{k+1}||^2-||\boldsymbol{\nu}^k||^2\right)+\sigma\left((\upsilon^{k+1})^2-(\upsilon^k)^2\right).$$

Since $L^{k+1}\geq L^k$ and $L_b^{k+1}\geq L_b^k$, we have

$$\frac{2(\rho^k)^2\mu^k}{L^k}-\frac{(\rho^{k+1})^2\mu^{k+1}}{L^{k+1}}\geq\left(||\boldsymbol{\nu}^{k+1}||^2-||\boldsymbol{\nu}^k||^2\right)+\sigma\left((\upsilon^{k+1})^2-(\upsilon^k)^2\right).$$

This completes the proof. ■

Finally, with Lemma 6, following the proof of Theorem 4.4 in (Beck and Teboulle, 2009), we have

$$F(\boldsymbol{\omega}^k,b^k)-F(\boldsymbol{\omega}^*,b^*)\leq\frac{2L^k||\boldsymbol{\omega}^0-\boldsymbol{\omega}^*||^2}{(k+1)^2}+\frac{2\sigma L^k(b^0-b^*)^2}{(k+1)^2}\leq\frac{2L_t||\boldsymbol{\omega}^0-\boldsymbol{\omega}^*||^2}{\eta(k+1)^2}+\frac{2L_{bt}(b^0-b^*)^2}{\eta(k+1)^2}.$$

This completes the proof.

Appendix C: Linear Convergence of Algorithm 4 for the Logistic Loss

In Algorithm 4, by fixing $\varrho^k=1$, it is reduced to the proximal gradient method (Nesterov, 2007), and it attains a linear convergence rate for the logistic loss, if $\mathbf{X}$ satisfies the following *Restricted Eigenvalue Condition* (Zhang, 2010b):

Definition 2 *(Zhang, 2010b) Given an integer $\kappa > 0$, a design matrix $\mathbf{X}$ is said to satisfy the Restricted Eigenvalue Condition at sparsity level κ, if there exists positive constants $\gamma_-(\mathbf{X}, \kappa)$ and $\gamma_+(\mathbf{X}, \kappa)$ such that*

$$\gamma_-(\mathbf{X}, \kappa) = \inf \left\{ \frac{\boldsymbol{\omega}^\top \mathbf{X}^\top \mathbf{X} \boldsymbol{\omega}}{\boldsymbol{\omega}^\top \boldsymbol{\omega}}, \boldsymbol{\omega} \neq \mathbf{0}, ||\boldsymbol{\omega}||_0 \leq \kappa \right\},$$
$$\gamma_+(\mathbf{X}, \kappa) = \sup \left\{ \frac{\boldsymbol{\omega}^\top \mathbf{X}^\top \mathbf{X} \boldsymbol{\omega}}{\boldsymbol{\omega}^\top \boldsymbol{\omega}}, \boldsymbol{\omega} \neq \mathbf{0}, ||\boldsymbol{\omega}||_0 \leq \kappa \right\}.$$

Remark 7 *For the logistic loss, if $\gamma_-(\mathbf{X}, tB) \geq \tau > 0$, Algorithm 4 with $\varrho^k = 1$ attains a linear convergence rate.*

Proof Let $\xi_i = -y_i(\sum_{h=1}^t \boldsymbol{\omega}_h' \mathbf{x}_{ih} - b)$, the Hessian matrix for the logistic loss can be calculated by (Yuan et al., 2011):

$$\nabla^2 P(\boldsymbol{\omega}) = C \mathbf{X}' \boldsymbol{\Delta} \mathbf{X},$$

where $\boldsymbol{\Delta}$ is a diagonal matrix with diagonal element $\boldsymbol{\Delta}_{i,i} = \frac{1}{1+\exp(\xi_i)}(1 - \frac{1}{1+\exp(\xi_i)}) > 0$. Apparently, $\nabla^2 P(\boldsymbol{\omega}, b)$ is upper bounded on a compact set due to the existence of $\gamma_+(\mathbf{X}, \kappa)$. Let $\sqrt{\boldsymbol{\Delta}}$ be the square root of $\boldsymbol{\Delta}$. Then if $\gamma_-(\mathbf{X}, tB) \geq \tau > 0$, we have $\gamma_-(\sqrt{\boldsymbol{\Delta}}\mathbf{X}, tB) > 0$ due to $\boldsymbol{\Delta}_{i,i} > 0$. In other words, the logistic loss is strongly convex if $\gamma_-(\mathbf{X}, tB) > 0$. Accordingly, the linear convergence rate can be achieved (Nesterov, 2007). ■

Appendix D: Proof of Proposition 3

Proof Proof of argument (I): We prove it by contradiction. Firstly, suppose $\mathbf{d}^*$ is a minimizer and there exists an $l \in \{1 \dots m\}$, such that $w_l = 0$ but $d_l^* > 0$. Let $0 < \epsilon < d_l^*$, and choose one $j \in \{1 \dots m\}$ where $j \neq l$, such that $|w_j| > 0$. Define new solution $\widehat{\mathbf{d}}$ in the following way:

$$\widehat{d}_j = d_j^* + d_l^* - \epsilon, \quad \widehat{d}_l = \epsilon, \quad \text{and,}$$
$$\widehat{d}_k = d_k^*, \quad \forall k \in \{1 \dots m\} \backslash \{j, l\}.$$

Then it is easy to check that

$$\sum_{j=1}^m \widehat{d}_j = \sum_{j=1}^m d_j^* \leq B.$$

In other words, $\widehat{\mathbf{d}}$ is also a feasible point. However, since $\widehat{d}_j = d_j^* + d_l^* - \epsilon \geq d_j^*$, it follows that

$$\frac{w_j^2}{\widehat{d}_j} < \frac{w_j^2}{d_j^*}.$$

Therefore, we have

$$\sum_{j=1}^m \frac{w_j^2}{\widehat{d}_j} < \sum_{j=1}^m \frac{w_j^2}{d_j^*},$$

which contradict the assumption that $\mathbf{d}^*$ is the minimizer.

On the other hand, if $|w_j| > 0$ and $d_j^* = 0$, by the definition, $\frac{x_j^2}{0} = \infty$. As we expect to get the finite minimum, so if $|w_j| > 0$, we have $d_j^* > 0$.

(II): First of all, the argument holds trivially when $\|\mathbf{w}\|_0 = \kappa \leq B$.

If $\|\mathbf{w}\|_0 = \kappa > B$, without loss of generality, we assume $|w_j| > 0$ for the first κ elements. From the argument (I), we have $1 \geq d_j > 0$ for $j \in \{1 \dots \kappa\}$ and $\sum_{j=1}^{\kappa} d_j \leq B$. Note that $\sum_{j=1}^{\kappa} \frac{w_j^2}{d_j}$ is convex regarding $\mathbf{d}$. The minimization problem can be written as:

$$\min_{\mathbf{d}} \sum_{j=1}^{\kappa} \frac{w_j^2}{d_j}, \qquad \text{s.t.} \quad \sum_{j=1}^{\kappa} d_j \leq B, \quad d_j > 0, \quad 1 - d_j \geq 0. \tag{37}$$

The KKT condition of this problem can be written as:

$$\begin{aligned}
&-w_j^2/d_j^2 + \gamma - \zeta_j + \nu_j = 0, \\
&\zeta_j d_j = 0, \\
&\nu_j(1 - d_j) = 0, \qquad (38) \\
&\gamma(B - \sum_{j=1}^{\kappa} d_j) = 0, \qquad (39) \\
&\gamma \geq 0, \zeta_j \geq 0, \nu_j \geq 0, \forall\ j \in \{1 \dots \kappa\},
\end{aligned}$$

where γ, ζ_j and ν_j are the dual variables for the constraints $\sum_{j=1}^{\kappa} d_j \leq B$, $d_j > 0$ and $1 - d_j \geq 0$ respectively. For those $d_j > 0$, we have $\zeta_j = 0$ for $\forall j \in \{1 \dots \kappa\}$ due to the KKT condition. Accordingly, by the first equality in KKT condition, we must have

$$d_j = |w_j|/\sqrt{\gamma + \nu_j}, \quad \forall j \in \{1 \dots \kappa\}.$$

Moreover, since $\sum_{j=1}^{\kappa} d_j \leq B < \kappa$, there must exist some $d_j < 1$ with $\nu_j = 0$ (otherwise $\sum_{j=1}^{\kappa} d_j$ will be greater than B). Here $\nu_j = 0$ because of the condition (38). This observation implies that $\gamma \neq 0$ since each d_j is bounded. Since $d_j \leq 1$, the condition $\sqrt{\gamma + \nu_j} \geq \max\{|w_j|\}$ must hold for $\forall j \in \{1 \dots \kappa\}$. Furthermore, by the complementary condition (39), we must have

$$\sum_{j=1}^{\kappa} d_j = B.$$

By substituting $d_j = |w_j|/\sqrt{\gamma + \nu_j}$ back to the objective function of (37), it becomes

$$\sum_{j=1}^{\kappa} |w_j| \sqrt{\gamma + \nu_j}.$$

To get the minimum of the above function, we are required to set the nonnegative ν_j as small as possible.

Now we complete the proof with the assumption $\|\mathbf{w}\|_1 / \max\{|w_j|\} \geq B$. When setting $\nu_j = 0$, we get $d_j = \frac{|w_j|}{\sqrt{\gamma}}$ and $\sum_{j=1}^{\kappa} \frac{|w_j|}{\sqrt{\gamma}} = B$. It is easy to check that $\sqrt{\gamma} = \|\mathbf{w}\|_1 / B \geq \max\{|w_j|\}$ and $d_j = B|w_j|/\|\mathbf{w}\|_1 \leq 1$, which satisfy the KKT condition. Therefore, the above $\mathbf{d}$ is an optimal solution. This completes the proof of the argument (II).

(III): With the results of (II), if $\kappa \leq B$, we have $\sum_{j=1}^{m} \frac{w_j^2}{d_j} = \sum_{j=1}^{\kappa} w_j^2$. Accordingly, we have $\|\mathbf{w}\|_B = \|\mathbf{w}\|_2$. And if $\kappa > B$ and $\|\mathbf{w}\|_1 / \max\{w_j\} \geq B$, we have

$$\sum \frac{w_j^2}{d_j} = \sum \frac{|w_j|}{d_j}|w_j| = \frac{\|\mathbf{w}\|_1}{B} \sum |w_j| = \frac{(\|\mathbf{w}\|_1)^2}{B}.$$

Hence we have $\|\mathbf{w}\|_B = \sqrt{\sum \frac{w_j^2}{d_j}} = \frac{\|\mathbf{w}\|_1}{\sqrt{B}}$. This completes the proof. ■

References

F. R. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Convex optimization with sparsity-inducing norms. In *Optimization for Machine Learning*. S. Sra, S. Nowozin, S. J. Wright., 2011.

F. R. Bach. High-dimensional non-linear variable selection through hierarchical kernel learning. Technical report, 2009.

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*, 2004.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. on Imaging Sciences*, 2(1):183–202, 2009.

B . Blum, M. I. Jordan, D. E. Kim, R. Das, P. Bradley, and D. Baker. Feature selection methods for improving protein structure prediction with rosetta. In *NIPS*, 2007.

P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *ICML*, 1998.

A. B. Chan, N. Vasconcelos, and G. R. G. Lanckriet. Direct convex relaxations of sparse SVM. In *ICML*, 2007.

Y. W. Chang, C. J. Hsieh, K. W. Chang, M. Ringgaard, and C. J. Lin. Training and testing low-degree polynomial data mappings via linear SVM. *JMLR*, 11:1471–1490, 2010.

O. Chapelle and S. S. Keerthi. Multi-class feature selection with support vector machines. In *Proceedings of the American Statistical Association*, 2008.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Mach. Learn.*, 46(1):131–159, 2002.

J. Chen and J. Ye. Training SVM with indefinite kernels. In *ICML*, 2008.

A. Dasgupta, P. Drineas, and B. Harb. Feature selection methods for text classification. In *KDD*, 2007.

J. Deng, A. C. Berg, and F. Li. Hierarchical semantic indexing for large scale image retrieval. In *CVPR*, pages 785–792. IEEE, 2011.

J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *JMLR*, 10:2899–2934, 2009.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd ed.)*. Hoboken, NJ: Wiley-Interscience, 2000.

R. Fan, P. Chen, and C.-J. Lin. Working set selection using second order information for training SVM. *JMLR*, 6:1889–1918, 2005.

M Figueiredo, R. Nowak, and S. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE J. Sel. Top. Sign. Proces.: Special Issue on Convex Optimization Methods for Signal Processing*, 1(4):586–597, 2007.

G. M. Fung and O. L. Mangasarian. A feature selection newton method for support vector machine classification. *Comput. Optim. Appl.*, 28:185–202, 2004.

P. Gehler and S. Nowozin. Infinite kernel learning. Technical report, 2008.

T. R. Golub, D. K. Slonim, and P. Tamayo. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 7:286–531, 1999.

Y. Grandvalet and S. Canu. Adaptive scaling for feature selection in svms. In *NIPS*, 2002.

Q. Gu, Z. Li, and J. Han. Correlated multi-label feature selection. In *CIKM*, 2011a.

Q. Gu, Z. Li, and J. Han. Generalized fisher score for feature selection. In *UAI*, 2011b.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *JMLR*, 3: 1157–1182, 2003.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46:389–422, 2002.

C. J. Hsieh, K. W. Chang, C. J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *ICML*, 2008.

A. Jain, S.V.N. Vishwanathan, and M. Varma. SPF-GMKL: generalized multiple kernel learning with a million kernels. In *KDD*, 2012.

R. Jenatton, J. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. Technical report, 2011a.

R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for hierarchical sparse coding. *JMLR*, 12:2297–2334, 2011b.

T. Joachims. Training linear SVMs in linear time. In *KDD*, 2006.

J. E. Kelley. The cutting plane method for solving convex programs. *J. Soc. Ind. Appl. Math.*, 8(4):703 –712, 1960.

S. Kim and E. P. Xing. Tree-guided group lasso for multi-response regression with structured sparsity, with applications to eQTL mapping. *Ann. Statist.*, Forthcoming, 2012.

S. Kim and E. P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML 2010*, 2010.

S. Kim and S. Boyd. On general minimax theorems. *Pacific J. Math.*, 1958, 8(1):171–176, 1958.

M. Kloft and G. Blanchard. On the convergence rate of ℓ_p-norm multiple kernel learning. *JMLR*, 13:2465–2501, 2012.

M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K. Müller, and A. Zien. Efficient and accurate ℓ_p-norm multiple kernel learning. *NIPS*, 22(22):997–1005, 2009.

M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien. Lp-norm multiple kernel learning. *JMLR*, 12:953–997, 2011.

R. Kohavi and G. John. Wrappers for feature subset selection. *Artif. Intell.*, 97:273–324, 1997.

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *JMLR*, 5:27–72, 2004.

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *JMLR*, 10:777–801, 2009.

P. Li, A. Shrivastava, J. Moore, and A. C. Konig. Hashing algorithms for large-scale learning. In *NIPS*, 2011.

P. Li, A. Owen, and C. H. Zhang. One permutation hashing. In *NIPS*, 2012.

D. Lin, D. P. Foster, and L. H. Ungar. A risk ratio comparison of ℓ_0 and ℓ_1 penalized regressions. Technical report, University of Pennsylvania, 2010.

J. Liu and J. Ye. Efficient l1/lq norm regularization. Technical report, 2010.

A. C. Lozano, G. Swirszcz, and N. Abe. Group orthogonal matching pursuit for logistic regression. In *AISTATS*, 2011.

S. Maji and A. C. Berg. Max-margin additive classifiers for detection. In *ICCV*, 2009.

Q. Mao and I. W. Tsang. A feature selection method for multivariate performance measures. *IEEE Trans. Pattern Anal. Mach.*, 35(9):2051–2063, 2013.

A. F. T. Martins, M. A. T. Figueiredo, P. M. Q. Aguiar, N. A. Smith, and E. P. Xing. Online multiple kernel learning for structured prediction. Technical report, 2010.

L. Meier, S. Van De Geer, and P. Bühlmann. The group lasso for logistic regression. *J. Roy. Stat. Soc. B.*, 70(1):53–71, 2008.

A. Mutapcic and S. Boyd. Cutting-set methods for robust convex optimization with pessimizing oracles. *Optim. Method Softw.*, 24(3):381?06, 2009.

A. Nedic and A. Ozdaglar. Subgradient methods for saddle-point problems. *J. Optimiz. Theory App.*, 142(1):205–228, 2009.

A. Nemirovski. Prox-method with rate of convergence o(1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM J. Opt.*, 15:229–251, 2005.

Y. Nesterov. Gradient methods for minimizing composite objective function. Technical report, 2007.

A. Y. Ng. On feature selection: Learning with exponentially many irrelevant features as training examples. In *ICML*, 1998.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

F. Orabona and L. Jie. Ultra-fast optimization algorithm for sparse multi kernel learning. In *ICML*, 2011.

E. Y. Pee and J. O. Royset. On solving large-scale finite minimax problems using exponential smoothing. *J. Optimiz. Theory App.*, online, 2010.

Z. Qin, K. Scheinberg, and D. Goldfarb. Efficient block-coordinate descent algorithms for the group lasso. Technical report, 2010.

A. Rakotomamonjy. Variable selection using svm-based criteria. *JMLR*, 3:1357–1370, 2003.

A. Rakotomamonjy, F. R. Bach, Y. Grandvalet, and S. Canu SimpleMKL. *JMLR*, 9: 2491–2521, 2008.

M. Rastegari, C. Fang, and L. Torresani. Scalable object-class retrieval with approximate and top-ranking. In *ICCV*, 2011.

V. Roth and B. Fischer. The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In *ICML*, 2008.

D. Sculley, G. M. Wachman, and C. E. Brodley. Spam filtering using inexact string matching in explicit feature space with on-line linear classifiers. In *The Fifteenth Text REtrieval Conference (TREC) Proceedings*, 2006.

S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *JMLR*, 14:567–599, 2013.

S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large Scale Multiple Kernel Learning. *JMLR*, 7:1531–1565, 2006.

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

M. Tan, I. W. Tsang, and L. Wang. Learning sparse svm for feature selection on very high dimensional data sets. In *ICML*, 2010.

M. Tan, I. W. Tsang, and L. Wang. Minimax sparse logistic regression for very high-dimensional feature selection. *IEEE Trans. Neural Netw. Learning Syst.*, 24(10):1609–1622, 2013.

A. Tewari, P. Ravikumar, and I. S. Dhillon. Greedy algorithms for structurally constrained high dimensional problems. In *NIPS*, 2011.

K. C. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. Technical report, 2009.

P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. Technical report, University of Washington, 2008.

P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optimiz. Theory App.*, 109(3):475–494, 2001.

M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *ICML*, 2009.

A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010.

S. V. N. Vishwanathan, Z. Sun, N. Theera-Ampornpunt, and M. Varma. Multiple kernel learning and the SMO algorithm. In *NIPS*, December 2010.

K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *ICML*, 2009.

J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. In *NIPS*, 2000.

J. Wu. Efficient hik svm learning for image classification. *IEEE Trans. Image Process*, 21(10):4442–4453, 2012.

L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. In *NIPS*, 2009.

Z. Xu, R. Jin, Ye J., Michael R. Lyu, and King I. Non-monotonic feature selection. In *ICML*, 2009a.

Z. Xu, R. Jin, I. King, and M.R. Lyu. An extended level method for efficient multiple kernel learning. In *NIPS*. 2009b.

Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *ICML*, 2010.

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. *JMLR*, 11:3183–3236, 2010.

G. X. Yuan, C. H. Ho, and C. J. Lin. An improved GLMNET for l1-regularized logistic regression and support vector machines. *JMLR*, 13:1999–2030, 2012.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. Roy. Stat. Soc. B.*, 68(1):49–67, 2006.

C. H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *Ann. Statist.*, 38(2):894–942, 2010a.

C. H. Zhang and J. Huang. The sparsity and bias of the lasso selection in high-dimensional linear regression. *Ann. Statist.*, 36(4):1567–1594, 2008.

D. Zhang and W. S. Lee. Extracting key-substring-group features for text classification. In *KDD*, 2006.

T. Zhang. Analysis of multi-stage convex relaxation for sparse regularization. *JMLR*, 11: 1081–1107, 2010b.

Z. Zhao and H. Liu. Spectral feature selection for supervised and unsupervised learning. In *ICML*, 2007.

J. Zhu, S. Rossett, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *NIPS*, 2003.

Journal of Machine Learning Research 15 (2014) 1431-1453 Submitted 1/13; Revised 12/13; Published 4/14

Adaptive Sampling for Large Scale Boosting

Charles Dubout CHARLES@DUBOUT.CH
François Fleuret FRANCOIS.FLEURET@IDIAP.CH
Computer Vision and Learning Group
Idiap Research Institute
CH-1920 Martigny, Switzerland

Editor: Nicolas Vayatis

Abstract

Classical boosting algorithms, such as AdaBoost, build a strong classifier without concern for the computational cost. Some applications, in particular in computer vision, may involve millions of training examples and very large feature spaces. In such contexts, the training time of off-the-shelf boosting algorithms may become prohibitive. Several methods exist to accelerate training, typically either by sampling the features or the examples used to train the weak learners. Even if some of these methods provide a guaranteed speed improvement, they offer no insurance of being more efficient than any other, given the same amount of time.

The contributions of this paper are twofold: (1) a strategy to better deal with the increasingly common case where features come from multiple sources (for example, color, shape, texture, etc., in the case of images) and therefore can be partitioned into meaningful subsets; (2) new algorithms which balance at every boosting iteration the number of weak learners and the number of training examples to look at in order to maximize the expected loss reduction. Experiments in image classification and object recognition on four standard computer vision data sets show that the adaptive methods we propose outperform basic sampling and state-of-the-art bandit methods.

Keywords: boosting, large scale learning, feature selection

1. Introduction

Boosting is a simple and efficient machine learning algorithm which provides state-of-the-art performance on many tasks. It consists of building a strong classifier as a linear combination of weak learners, by adding them one after another in a greedy manner.

It has been repeatedly demonstrated that combining multiple kind of features addressing different aspects of the signal is an extremely efficient strategy to improve performance (Opelt et al., 2006; Gehler and Nowozin, 2009; Fleuret et al., 2011; Dubout and Fleuret, 2011a,b). As shown by our experimental results, vanilla boosting of stumps over multiple image features such as HOG, LBP, color histograms, etc., usually reaches close to state-of-the-art performance. However, such techniques entails a considerable computational cost, which increases with the number of features considered during training.

The critical operations contributing to the computational cost of a boosting iteration are the computations of the features and the selection of the weak learner. Both depend on the number of features and the number of training examples taken into account. While

textbook AdaBoost repeatedly selects each weak learner using all the features and all the training examples for a predetermined number of rounds, one is not obligated to do so and can instead choose to look only at a subset of both.

Since performance increases with both, one needs to balance the two to keep the computational cost under control. As boosting progresses, the performance of the candidate weak learners degrades, and they start to behave more and more similarly. While a small number of training examples is initially sufficient to characterize the good ones, as the learning problems become more and more difficult, optimal values for a fixed computational cost tend to move towards smaller number of features and larger number of examples.

In this paper, we present three new families of algorithms to explicitly address these issues: (1) Tasting (see Section 4 on page 1434) uses a small number of features sampled prior to learning to adaptively bias the sampling towards promising subsets at every step; (2) Maximum Adaptive Sampling (see Section 5.3 on page 1439) models the distribution of the weak learners' performance and the noise in order to determine the optimal trade-off between the number of weak learners and the number of examples to look at; and (3) Laminating (see Section 5.4 on page 1440) iteratively refines the learner selection using more and more examples.

2. Related Works

AdaBoost and similar boosting algorithms estimate for each candidate weak learner a score dubbed "edge", which requires to loop through every training example and take into account its weight, which reflects its current importance in the loss reduction. Reducing this computational cost is crucial to cope with high-dimensional feature spaces or very large training sets. This can be achieved through two main strategies: sampling the training examples, or the feature space, since there is a direct relation between features and weak learners.

Sampling the training set was introduced historically to deal with weak learners which cannot be trained with weighted examples (Freund and Schapire, 1996). This procedure consists of sampling examples from the training set according to their boosting weights, and of approximating a weighted average over the full set by a non-weighted average over the sampled subset. It is related to Bootstrapping as similarly the training algorithm will sample harder and harder examples based on the performance of the previous weak learners. See Section 3 for formal details. Such a procedure has been re-introduced recently for computational reasons (Bradley and Schapire, 2007; Duffield et al., 2007; Kalal et al., 2008; Fleuret and Geman, 2008), since the number of sampled examples controls the trade-off between statistical accuracy and computational cost.

Sampling the feature space is the central idea behind LazyBoost (Escudero et al., 2000), and simply consists of replacing the brute-force exhaustive search over the full feature set by an optimization over a subset produced by sampling uniformly a predefined number of features. The natural redundancy of most type of features makes such a procedure generally efficient. However, if a subset of important features is too small, it may be overlooked during training.

Recently developed algorithms rely on multi-arms bandit methods to balance properly the exploitation of features known to be informative, and the exploration of new features

(Busa-Fekete and Kegl, 2009, 2010). The idea behind those methods is to associate a bandit arm to every feature, and to see the loss reduction as a reward. Maximizing the overall reduction is achieved with a standard bandit strategy such as UCB (Auer et al., 2002), or Exp3.P (Auer et al., 2003).

These techniques suffer from two important drawbacks. First they make the assumption that the quality of a feature — the expected loss reduction of a weak learner using it — is stationary. This goes against the underpinning of boosting, which is that at any iteration the performance of the weak learners is relative to the boosting weights, which evolve over the training (Exp3.P does not make such an assumption explicitly, but still rely exclusively on the history of past rewards). Second, without additional knowledge about the feature space, the only structure they can exploit is the stationarity of individual features. Hence, improvement over random selection can only be achieved by sampling again *the exact same features* already seen in the past. In our experiments, we therefore only use those methods in a context where features can be partitioned into subsets of different types. This allows us to model the quality, and thus to bias the sampling, at a higher level than individual features.

All those approaches exploit information about features to bias the sampling, hence making it more efficient, and reducing the number of weak learners required to achieve the same loss reduction. However, they do not explicitly aim at controlling the computational cost. In particular, there is no notion of varying the number of examples used for the estimation of the loss reduction.

3. Preliminaries

We first present in this section some analytical results to approximate a standard round of AdaBoost — or other similar boosting algorithms — by sampling both the training examples and the features used to build the weak learners. We then precise more formally what we mean by subset of features or weak learners.

3.1 Standard Boosting

Given a binary training set

$$(x_n, y_n) \in \mathcal{X} \times \{-1, 1\}, n = 1, \dots, N,$$

where $\mathcal{X}$ is the space of the "visible" signal, and a set $\mathcal{H}$ of weak learners of the form $h : \mathcal{X} \to \{-1, 1\}$, the standard boosting procedure consists of building a strong classifier

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x),$$

by choosing the terms $\alpha_t \in \mathbb{R}$ and $h_t \in \mathcal{H}$ in a greedy manner so as to minimize a loss (for example the empirical exponential loss in the case of AdaBoost) estimated over the training examples. At every iteration, choosing the optimal weak learner boils down to finding the one with the largest edge ϵ, which is the derivative of the loss reduction w.r.t. the weak learner weight α. The higher this value, the more the loss can be reduced locally, and thus

the better the weak learner. The edge is a linear function of the responses of the weak learner over the training examples

$$\epsilon(h) = \sum_{n=1}^{N} \omega_n y_n h(x_n),$$

where the weights ω_n's depend on the loss function (usually either the exponential or logistic loss) and on the current responses of f over the x_n's. We consider without loss of generality that they have been normalized such that $\sum_n \omega_n = 1$. We can therefore consider the weights ω_n's as a distribution over the training examples and rewrite the edge as an expectation over them,

$$\epsilon(h) = \mathrm{E}_{N \sim \omega_n}[y_N h(x_N)], \tag{1}$$

where $N \sim \omega_n$ stands for $\mathrm{P}(N = n) = \omega_n$. The idea of weighting-by-sampling (Fleuret and Geman, 2008) consists of replacing the expectation in Equation (1) with an approximation obtained by sampling. Let $N_1, \ldots, N_S$, be i.i.d. random variables distributed according to the discrete probability density distribution defined by the ω's, we define the approximated edge as

$$\hat{\epsilon}(h) = \frac{1}{S} \sum_{s=1}^{S} y_{N_s} h(x_{N_s}), \tag{2}$$

which follows a binomial distribution centered on the true edge, with a variance decreasing with the number of sampled examples S. It is accurately modeled by the Gaussian

$$\hat{\epsilon}(h) \approx \mathcal{N}\left(\epsilon(h), \frac{1}{S}\right), \tag{3}$$

as the approximation holds asymptotically and the magnitude of the weak learners' edges is typically small, such that $(1 + \epsilon(h))(1 - \epsilon(h)) \approx 1$.

3.2 Feature Subsets

It frequently happens that the features making up the signal space $\mathcal{X}$ can be divided into meaningful disjoint subsets $\mathcal{F}_k$ such that $\mathcal{X} = \cup_{k=1}^{K} \mathcal{F}_k$. This division can for example be the result of the features coming from different sources or some natural clustering of the feature space. In such a case it makes sense to use this information during training, as features coming from the same subset $\mathcal{F}_k$ can typically be expected to be more homogeneous than features coming from different subsets.

4. Tasting

We describe here our approach called Tasting (Dubout and Fleuret, 2011a) which biases the sampling toward promising subsets of features. Tasting in its current form is limited to deal with weak learners looking at a single feature, such as decision stumps. Extending it to deal efficiently with weak learners looking at multiple features is outside of the scope of this work.

4.1 Main Algorithm

The core idea of Tasting is to sample a small number R of features from every subset before starting the training *per se* and, at every boosting step, in using these few features together with the current boosting weights to get an estimate of the best subset(s) $\mathcal{F}_k$(s) to use.

We cannot stress enough that these R features are not the ones used to build the classifier, they are only used to figure out what is/are the best subset(s) at any time during training. As those sampled features are independent and identically distributed samples of the feature response vectors, we can compute the empirical mean of any functional of the said response vectors, in particular the expected loss reduction.

At any boosting step, Tasting requires, for any feature subset, an estimate of the expectation of the edge of the best weak learner we would obtain by sampling uniformly Q features from this subset and picking the best weak learner using one of them,

$$\mathrm{E}_{F_1,\ldots,F_Q \sim \mathcal{U}(\mathcal{F}_k)} \left[\max_{q=1}^{Q} \max_{h \in \mathcal{H}_{F_q}} \epsilon(h) \right], \tag{4}$$

where $\mathcal{F}_k$ are the indices of the features belonging to the k-th subset and $\mathcal{H}_F$ is the space of weak learners looking solely at feature F. Hence $\max_{h \in \mathcal{H}_{F_q}} \epsilon(h)$ is the best weak learner looking solely at feature F_q, and $\max_{q=1}^{Q} \max_{h \in \mathcal{H}_{F_q}} \epsilon(h)$ is the best weak learner looking solely at one of the Q features $F_1, \ldots, F_Q$.

We can build an approximation of this quantity using the R features we have stored. Let $\epsilon_1, \ldots, \epsilon_R$ be the edges of the best R weak learners built from these features. We make the assumption without loss of generality that $\epsilon_1 \leq \epsilon_2 \leq \cdots \leq \epsilon_R$. Let $R_1, \ldots, R_Q$ be independent and identically distributed, uniform over $\{1, \ldots, R\}$. We approximate the quantity in Equation (4) with

$$\begin{aligned}
\mathrm{E}\left[\max_{q=1}^{Q} \epsilon_{R_q} \right] &= \sum_{r=1}^{R} \mathrm{P}\left(\max_{q=1}^{Q} R_q = r \right) \epsilon_r \\
&= \sum_{r=1}^{R} \left[\mathrm{P}\left(\max_{q=1}^{Q} R_q \leq r \right) - \mathrm{P}\left(\max_{q=1}^{Q} R_q \leq r-1 \right) \right] \epsilon_r \\
&= \frac{1}{R^Q} \sum_{r=1}^{R} \left[r^Q - (r-1)^Q \right] \epsilon_r .
\end{aligned} \tag{5}$$

4.2 Tasting Variants

We propose two versions of the Tasting procedure, which differ in the number of feature subsets they visit at every iteration. Either one for Tasting 1.Q or up to Q for Tasting Q.1.

In Tasting 1.Q (Algorithm 1), the selection of the optimal subset k^* from which to sample the Q features is accomplished by estimating for every subset the expected maximum edge, which is directly related to the expected loss reduction, if we were sampling from that subset only. The computation is done over the R features saved before starting training, which serve as a representation of the full set $\mathcal{F}_k$.

In Tasting Q.1 (see Algorithm 2), it is not one but several feature subsets which can be selected, as the algorithm picks the best subset k_q^* for every one of the Q features to

Algorithm 1 The Tasting 1.Q algorithm first samples uniformly R features from every feature subset $\mathcal{F}_k$. It uses these features at every boosting step to find the optimal feature subset k^* from which to sample. After the selection of the Q features, the algorithm continues like AdaBoost.

Input: $\mathcal{F}, Q, R, T$

Initialize: $\forall k \in \{1, \dots, K\}, \forall r \in \{1, \dots, R\}, f_r^k \leftarrow \text{sample}(\mathcal{U}(\mathcal{F}_k))$

for $t = 1, \dots, T$ **do**

$\forall k \in \{1, \dots, K\}, \forall r \in \{1, \dots, R\}, \epsilon_r^k \leftarrow \max_{h \in \mathcal{H}_{f_r^k}} \epsilon(h)$

$k^* \leftarrow \operatorname*{argmax}_k \mathrm{E}\left[\max_{q=1}^{Q} \epsilon_{R_q}^k\right]$ `# Computed using equation (5)`

$\forall q \in \{1, \dots, Q\}, F_q \leftarrow \text{sample}(\mathcal{U}(\mathcal{F}_{k^*}))$

$h_t \leftarrow \operatorname*{argmax}_{h \in \cup_q \mathcal{H}_{F_q}} \epsilon(h)$

$\dots$

end for

Algorithm 2 The Tasting Q.1 algorithm first samples uniformly R features from every feature subset $\mathcal{F}_k$. It uses them to find the optimal subset k_q^* for every one of the Q features to sample at every boosting step. After the selection of the Q features, the algorithm continues like AdaBoost.

Input: $\mathcal{F}, Q, R, T$

Initialize: $\forall k \in \{1, \dots, K\}, \forall r \in \{1, \dots, R\}, f_r^k \leftarrow \text{sample}(\mathcal{U}(\mathcal{F}_k))$

for $t = 1, \dots, T$ **do**

$\forall k \in \{1, \dots, K\}, \forall r \in \{1, \dots, R\}, \epsilon_r^k \leftarrow \max_{h \in \mathcal{H}_{f_r^k}} \epsilon(h)$

$\epsilon^* \leftarrow 0$

for $q = 1, \dots, Q$ **do**

$k_q^* \leftarrow \operatorname*{argmax}_k \mathrm{E}\left[\max\left(\epsilon^*, \epsilon^k\right)\right]$

$F_q \leftarrow \text{sample}\left(\mathcal{U}\left(\mathcal{F}_{k_q^*}\right)\right)$

$\epsilon^* \leftarrow \max\left(\epsilon^*, \max_{h \in \mathcal{H}_{F_q}} \epsilon(h)\right)$

end for

$h_t \leftarrow \operatorname*{argmax}_{h \in \cup_q \mathcal{H}_{F_q}} \epsilon(h)$

$\dots$

end for

sample, given the best edge ϵ^* achieved so far. Again the computation is done only over the R features saved before starting training.

4.3 Relation with Bandit Methods

The main strength of boosting is its ability to spot and combine complementary features. If the loss has already been reduced in a certain "functional direction", the scores of weak learners in the same direction will be low, and they will be rejected. For instance, the firsts learners for a face detector may use color-based features to exploit the skin color. After a few boosting steps using this modality, color would be exhausted as a source of information, and only examples with a non-standard face color would have large weights. Other features, for instance edge-based, would become more informative, and be picked.

Uniform sampling of features accounts poorly for such behavior since it simply discards the boosting weights, and hence has no information whatsoever about the directions which have "already been exploited" and which should be avoided. In practice, this means that the rejection of bad feature can only be done at the level of the boosting itself, which may end up with a majority of useless features.

Bandit methods (described in Section 6.4) are slightly more adequate, as they model the performance of every feature from previous iterations. However, this modeling takes into account the boosting weights very indirectly, as they make the assumption that the distributions of loss reduction are stationary, while they are precisely not. Coming back to our face-detector example, bandit methods would go on believing that color is informative since it was in the previous iterations, even if the boosting weights have specifically accumulated on faces where color is now totally useless. While the estimate of loss reduction may asymptotically converge to an adequate model, it is a severe weakness while the boosting weights are still evolving.

Tasting addresses this weakness by keeping the ability to properly estimate the performance of every feature subset, *given the current boosting weights*, hence the ability to discard feature subsets redundant with features already picked. In some sense, Tasting can be seen as boosting done at a the subset level.

5. Maximum Adaptive Sampling and Laminating

The algorithms in this section sample both the weak learners and the training examples at every iteration in order to maximize the expectation of the loss reduction, under a strict computational cost constraint.

5.1 Edge estimation

At every iteration they model the expectation of the edge of the selected weak learner. Let $\epsilon_1, \ldots, \epsilon_Q$ stand for the true edges of Q independently sampled weak learners. Let $\Delta_1, \ldots, \Delta_Q$ be a series of independent random variables standing for the noise in the estimation of the edges due to the sampling of only S training examples. Finally $\forall q$, let $\hat{\epsilon}_q = \epsilon_q + \Delta_q$ be the approximated edge. With these definitions, $\operatorname{argmax}_q \hat{\epsilon}_q$ is the selected weak learner. We define ϵ^* as the true edge of the selected weak learner, that is the one

with the highest approximated edge

$$\epsilon^* = \epsilon_{\text{argmax}_q \hat{\epsilon}_q}. \tag{6}$$

This quantity is random due to both the sampling of the weak learners, and the sampling of the training examples. The quantity we want to optimize is $\mathrm{E}[\epsilon^*]$, the expectation of the true edge of the selected learner over all weak learners and over all training examples, which increases with both Q and S. A higher Q increases the number of terms in the maximization of Equation (6), while a higher S reduces the variance of the Δ's, ensuring that ϵ^* is closer to $\max_q \epsilon_q$. In practice, if the variance of the Δ's is of the order of, or higher than, the variance of the ϵ's, the maximization is close to a random selection, and looking at many weak learners is useless. Assuming that the $\hat{\epsilon}_q$'s are all different we have,

$$\begin{aligned}
\mathrm{E}[\epsilon^*] &= \mathrm{E}\Big[\epsilon_{\text{argmax}_q \hat{\epsilon}_q}\Big] \\
&= \sum_{q=1}^{Q} \mathrm{E}\left[\epsilon_q \prod_{i \neq q} \mathbf{1}_{\{\hat{\epsilon}_i < \hat{\epsilon}_q\}}\right] \\
&= \sum_{q=1}^{Q} \mathrm{E}\left[\mathrm{E}\left[\epsilon_q \prod_{i \neq q} \mathbf{1}_{\{\hat{\epsilon}_i < \hat{\epsilon}_q\}} \,\middle|\, \hat{\epsilon}_q\right]\right] \\
&= \sum_{q=1}^{Q} \mathrm{E}\left[\mathrm{E}[\epsilon_q \mid \hat{\epsilon}_q] \prod_{i \neq q} \mathrm{E}\big[\mathbf{1}_{\{\hat{\epsilon}_i < \hat{\epsilon}_q\}} \mid \hat{\epsilon}_q\big]\right],
\end{aligned}$$

where the last equality follows from the independence of the weak learners.

5.2 Modeling the True Edge

If the distributions of the ϵ_q's and the Δ_q's are Gaussians or mixtures of Gaussians, we can derive analytical expressions for both $\mathrm{E}[\epsilon_q \mid \hat{\epsilon}_q]$ and $\mathrm{E}\big[\mathbf{1}_{\{\hat{\epsilon}_i < \hat{\epsilon}_q\}} \mid \hat{\epsilon}_q\big]$, and compute the value of $\mathrm{E}[\epsilon^*]$ efficiently. In the case where weak learners can be partitioned into meaningful subsets, it makes sense to model the distributions of the edges separately for each subset.

As an illustrative example, we consider here the case where the ϵ_q's, the Δ_q's, and hence also the $\hat{\epsilon}_q$'s all follow Gaussian distributions. We take $\epsilon_q \sim \mathcal{N}(0,1)$ and $\Delta_q \sim \mathcal{N}(0,\sigma^2)$ and obtain,

$$\begin{aligned}
\mathrm{E}[\epsilon^*] &= Q\ \mathrm{E}\left[\mathrm{E}[\epsilon_1 \mid \hat{\epsilon}_1] \prod_{i \neq 1} \mathrm{E}\big[\mathbf{1}_{\{\hat{\epsilon}_i < \hat{\epsilon}_1\}} \mid \hat{\epsilon}_1\big]\right] \\
&= Q\ \mathrm{E}\left[\frac{\hat{\epsilon}_1}{\sigma^2+1} \Phi\left(\frac{\hat{\epsilon}_1}{\sqrt{\sigma^2+1}}\right)^{Q-1}\right] \\
&= \frac{Q}{\sqrt{\sigma^2+1}}\ \mathrm{E}\Big[\epsilon_1 \Phi(\epsilon_1)^{Q-1}\Big] \\
&= \frac{1}{\sqrt{\sigma^2+1}}\ \mathrm{E}\left[\max_{1 \leq q \leq Q} \epsilon_q\right],
\end{aligned}$$

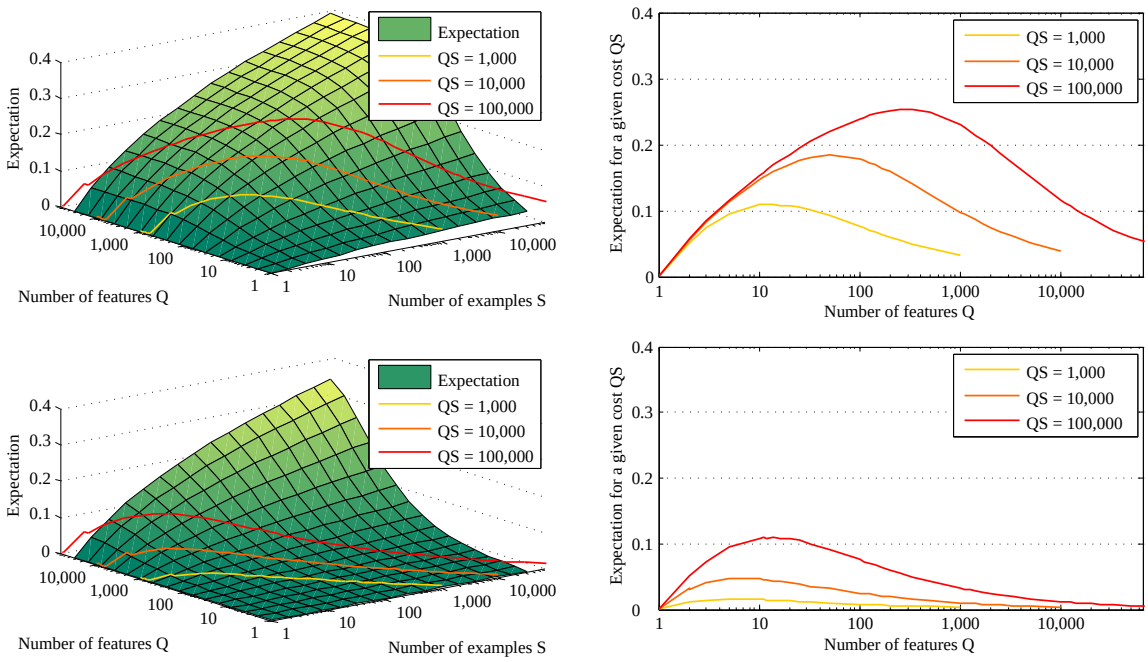


Figure 1: Simulation of the expectation of ϵ^* in the case where both the ϵ_q's and the Δ_q's follow Gaussian distributions. Top: $\epsilon_q \sim \mathcal{N}(0, 10^{-2})$. Bottom: $\epsilon_q \sim \mathcal{N}(0, 10^{-4})$. In both simulations $\Delta_q \sim \mathcal{N}(0, \frac{1}{S})$. Left: expectation of ϵ^* vs. the number of sampled weak learners Q and the number of examples S. Right: same value as a function of Q alone, for different fixed costs (product of Q and S). As these graphs illustrate, the optimal value for Q is greater for larger variances of the ϵ_q's. In such a case the ϵ_q's are more spread out, and identifying the largest one can be done despite a large noise in the estimations, hence with a limited number of training examples.

where Φ stands for the cumulative distribution function of the unit Gaussian, and σ^2 is of order $\frac{1}{S}$. See Figure 1 for an illustration of the behavior of $\mathrm{E}[\epsilon^*]$ for two different variances of the ϵ_q's and a cost proportional to QS, the total number of features computed.

There is no reason to expect the distribution of the ϵ_q's to be Gaussian, contrary to the Δ_q's, as shown in Equation (3), but this is not a problem as it can usually be approximated by a mixture, for which we can still derive analytical expressions, even if the ϵ_q's or the Δ_q's have different distributions for different q's.

5.3 M.A.S. Variants

We created three algorithms modeling the distribution of the ϵ_q's with a Gaussian mixture model, and $\Delta_q = \hat{\epsilon}_q - \epsilon_q$ as a Gaussian. The first one, M.A.S. naive, is described in Algorithm 3, and fits the model to the edges estimated at the previous iteration.

Algorithm 3 The M.A.S. naive algorithm models the current edge distribution with a Gaussian mixture model fitted on the edges estimated at the previous iteration. It uses this density model to compute the pair (Q^*, S^*) maximizing the expectation of the true edge of the selected learner $\mathrm{E}[\epsilon^*]$, and then samples the corresponding number of weak learners and training examples, before keeping the weak learner with the highest approximated edge. After the selection of the Q features, the algorithm continues like AdaBoost.

Input: $gmm, Cost$

for $t = 1, \ldots, T$ **do**

$(Q^*, S^*) \leftarrow \underset{\mathrm{cost}(Q,S) \leq Cost}{\mathrm{argmax}} \mathrm{E}_{gmm}[\epsilon^*]$

$\forall q \in \{1, \ldots, Q^*\}, H_q \leftarrow \mathrm{sample}(\mathcal{U}(\mathcal{H}))$

$\forall s \in \{1, \ldots, S^*\}, N_s \leftarrow \mathrm{sample}(\mathcal{U}(\{1, \ldots, N\}))$

$\forall q \in \{1, \ldots, Q^*\}, \hat{\epsilon}_q \leftarrow \frac{1}{S^*} \sum_{s=1}^{S^*} y_{N_s} H_q(x_{N_s})$ `# Similar to equation (2)`

$h_t \leftarrow H_{\mathrm{argmax}_q \hat{\epsilon}_q}$

$gmm \leftarrow \mathrm{fit}(\hat{\epsilon}_1, \ldots, \hat{\epsilon}_{Q^*})$

...

end for

The second one, M.A.S. 1.Q, takes into account the decomposition of the weak learners into K subsets, associated to different kind of features. It models the distributions of the ϵ_q's separately for each subset, estimating the distribution of each on a small number of weak learners and examples sampled at the beginning of each boosting iteration, chosen so as to account for 10% of the total computational cost. From these models, it optimizes Q, S, and the index k of the subset to sample from. Unlike M.A.S. naive, it has to draw a small number of weak learners and examples in order to fit the model since the edges estimated at the previous iterations came from a unique subset.

Finally M.A.S. Q.1 similarly models the distributions of the ϵ_q's, but it optimizes $Q_1, \ldots, Q_K$ greedily, starting from $Q_1 = 0, \ldots, Q_K = 0$, and iteratively incrementing one of the Q_k so as to maximize $\mathrm{E}[\epsilon^*]$. This greedy procedure is repeated for different values of S and ultimately the $Q_1, \ldots, Q_K, S$ leading to the maximum expectation are selected.

5.4 Laminating

The last algorithm we have developed tries to reduce the requirement for a density model of the ϵ_q's. At every boosting iteration it iteratively reduces the number of considered weak learners, and increases the number of examples taken into account.

Given fixed Q and S, at every boosting iteration, the Laminating algorithm first samples Q weak learners and S training examples. Then, it computes the approximated edges and keeps the $\frac{Q}{2}$ best learners. If more than one remains, it samples $2S$ examples, and re-iterates. The whole process is described in Algorithm 4. The number of iterations is bounded by $\lceil \log_2(Q) \rceil$.

Algorithm 4 The Laminating algorithm starts by sampling Q weak learners and S examples at the beginning of every boosting iteration, and refine those by successively halving the number of learners and doubling the number of examples until only one learner remains. After the selection of the Q features, the algorithm continues like AdaBoost.

Input: Q, S

for $t = 1, \ldots, T$ **do**

$\forall q \in \{1, \ldots, Q\}, h_q \leftarrow \text{sample}(\mathcal{U}(\mathcal{H}))$

while $Q > 1$ **do**

$\forall s \in \{1, \ldots, S\}, N_s \leftarrow \text{sample}(\mathcal{U}(\{1, \ldots, N\}))$

$\forall q \in \{1, \ldots, Q\}, \hat{\epsilon}_q \leftarrow \frac{1}{S} \sum_{s=1}^{S} y_{N_s} H_q(x_{N_s})$ `# Similar to equation (2)`

$\text{sort}(h_1, \ldots, h_Q)$ s.t. $\hat{\epsilon}_1 \geq \cdots \geq \hat{\epsilon}_Q$ `# Order the weak learners s.t.`

$Q \leftarrow \frac{Q}{2}$ `# the best half comes first`

$S \leftarrow 2S$

end while

...

end for

We have the following results on the accuracy of this Laminating procedure (the proof is given in Appendix A):

Lemma 1 *Let $q^\star = \text{argmax}_q \epsilon_q$ and $\delta > 0$. The probability for an iteration of the Laminating algorithm to retain only weak learners with edges below or equal to $\epsilon_{q^\star} - \delta$ is*

$$\mathrm{P}\left(\left|\left\{q : \epsilon_q \leq \epsilon_{q^\star} - \delta,\, \hat{\epsilon}_q \geq \max_{r:\epsilon_r \geq \epsilon_{q^\star} - \delta} \hat{\epsilon}_r\right\}\right| \geq \frac{Q}{2}\right) \leq 4 \exp\left(-\frac{\delta^2 S}{2}\right).$$

This holds regardless of the independence of the ϵ_q's and/or the Δ_q's.

Since at each iteration the number of examples S doubles the lemma implies the following theorem:

Theorem 1 *The probability for the full Laminating procedure starting with Q weak learners and S examples to end up with a learner with an edge below or equal to $\epsilon_{q^*} - \delta$ (the edge of the optimal weak learner at the start of the procedure minus δ) is upper bounded by (the proof is given in Appendix B)*

$$\frac{4}{1 - \exp\left(-\frac{\delta^2 S}{69}\right)} - 4.$$

The theorem shows that as the number of samples grows, the probability to retain a bad weak learner eventually goes down exponentially with the number of training examples, as in this case the bound can be approximated by $4 \exp\left(-\frac{\delta^2 S}{69}\right)$. This confirms the usual relation between the number of examples and the complexity of the space of predictors in learning theory.

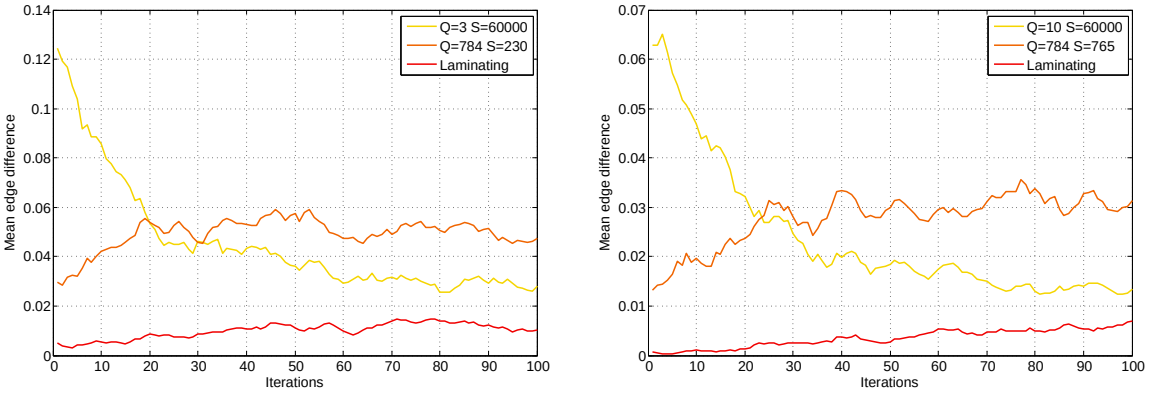


Figure 2: Difference between the maximum edge and the best edge found by 3 different sampling strategies on the MNIST data set using the original features. The algorithm used is AdaBoost.MH using $T = 100$ decision stumps as weak learners, and the results were averaged over 10 randomized runs. The first strategy samples uniformly a small number of features Q and determines the best one using all $S = 60,000$ training examples. The second strategy samples all $Q = 784$ features and determines the best one using a small number of training examples S. The third strategy is Laminating, starting from all the features and a suitable number of training examples chosen so as to have the same cost as the first two strategies. The cost is the product of Q and S and is set to $QS = 180,000$ for the left figure and $QS = 600,000$ for the right one.

In practice the difference between the maximum edge $\epsilon_{q^\star}$ and the edge of the final weak learner selected by Laminating is typically smaller than the difference with the edge of a learner selected by a strategy looking at a fixed number of weak learners and training examples, as can be observed in Figure 2.

6. Experiments

We demonstrate the effectiveness of our approaches on four standard image classification and object detection data sets, using 19 kinds of features (33 on Caltech 101) divided in as many subsets. We used the AdaBoost.MH algorithm (Schapire and Singer, 1999) with decision stumps as weak learners to be able to use all methods in the same conditions.

6.1 Features

The features used in our experiments with all but the Caltech 101 data set can be divided into three categories. (1) Image transforms: identity, grayscale conversion, Fourier and Haar transforms, gradient image, local binary patterns (ILBP/LBP). (2) Intensity histograms: sums of the intensities in random image patches, grayscale and color histograms of the entire image. (3) Gradient histograms: histograms of (oriented and non oriented) gradients, Haar-like features.

The features from the first category typically have a large dimensionality, usually proportional to the number of pixels in the image. Some of them do not pre-process the images (identity, grayscale conversion, LBP, etc.) while some pre-transform them to another space, prior to accessing any feature (typically the Fourier and Haar transforms).

Features from the second and third categories being histograms, they are usually much smaller (containing typically of the order of a few hundreds to a few thousands coefficients), but require some pre-processing to build the histograms.

For the Caltech 101 data set we used the same features as (Gehler and Nowozin, 2009) in their experiments. They used five type of features: PHOG shape descriptors, appearance (SIFT) descriptors, region covariance, local binary patterns, and V1S+, which are normalized Gabor filters. More details can be found in the referenced paper. Those features are computed in a spatial pyramid, where each scale of the pyramid is considered as being part of a different subset, leading to a total of 33 features. The number of features used in our experiments (33) differ from (Gehler and Nowozin, 2009) as they also compute a 'subwindow-kernel' of SIFT features which we did not use.

6.2 Data sets

Figure 3: Example images from the four data sets used for the experimental validation. Top left: first image of every digit taken from the MNIST database. Top right: images from the INRIA Person data set. Bottom left: random images from the Caltech 101 data set. Bottom right: some of the first images of the CIFAR-10 data set.

The first data set that we used is the MNIST handwritten digits database (LeCun et al., 1998). It is composed of 10 classes and its training and testing sets consist respectively of 60,000 and 10,000 grayscale images of resolution 28×28 pixels (see the upper left part of Figure 3 for some examples). The total number of features on this data set is 16,775.

The second data set that we used is the INRIA Person data set (Dalal and Triggs, 2005). It is composed of a training and a testing set respectively of 2,418 and 1,126 color images of

pedestrians of dimensions 64×128 pixels cropped from real-world photographs, along with 1,219 and 453 "background" images not containing any people (see the upper right part of Figure 3 for some examples). We extracted 10 negative samples from each one of the background image, following the setup of (Dalal and Triggs, 2005). The total number of features on this data set is 230,503.

The third data set that we used is Caltech 101 (Fei-Fei et al., 2004) due to its wide usage and the availability of already computed features (Gehler and Nowozin, 2009). It consists of color images of various dimensions organized in 101 object classes (see the bottom left part of Figure 3 for some examples). We sampled 15 training examples and 20 distinct test examples from every class, as advised on the data set website. The total number of features on this data set is 360,630.

The fourth and last data set that we used is CIFAR-10 (Krizhevsky, 2009). It is a labeled subset of the 80 tiny million images data set. It is composed of 10 classes and its training and testing sets consist respectively of 50,000 and 10,000 color images of size 32×32 pixels (see the bottom left part of Figure 3 for some examples). The total number of features on this data set is 29,879.

6.3 Uniform Sampling Baselines

A naive sampling strategy would pick the Q features uniformly in $\cup_k \mathcal{F}_k$. However, this does not distribute the sampling properly among the $\mathcal{F}_k$'s. In the extreme case, if one of the $\mathcal{F}_k$ had a far greater cardinality than the others, all features would come from it. And in most contexts, mixing features from the different $\mathcal{F}_k$'s in an equilibrate manner is critical to benefit from their complementarity. We propose the four following baselines to pick a good feature at every boosting step:

- **Best subset** picks Q features at random in a fixed subset, the one with the smallest final boosting loss.
- **Uniform Naive** picks Q features at random, uniformly in $\cup_k \mathcal{F}_q$.
- **Uniform 1.Q** picks one of the feature subsets at random, and then samples the Q features from that single subset.
- **Uniform Q.1** picks at random, uniformly, Q subsets of features (with replacement if $Q > K$), and then picks one feature uniformly in each subset.

The cost of running **Best subset** is K times higher than running the other three strategies since the subset leading to the smallest final boosting loss is not known a priori and requires to redo the training K times. Also, since it makes use of one subset only we can expect its final performance to be lower than the others. It was included for comparison only.

6.4 Bandit Sampling Baselines

The strategies of the previous section are purely random and do not exploit any kind of information to bias their sampling. Smarter strategies to deal with the problem of exploration-exploitation trade-off were first introduced in (Busa-Fekete and Kegl, 2009), and extended in (Busa-Fekete and Kegl, 2010). The driving idea of these papers is to

entrust a multi-armed bandits (MAB) algorithm (respectively UCB in Auer et al. (2002) and Exp3.P in Auer et al. (2003)) with the mission to sample useful features.

The multi-armed bandits problem is defined as follows: there are M gambling machines (the "arms" of the bandits), and at every time-step t the gambler chooses an arm j_t, pulls it, and receives a reward $r^t_{j_t} \in [0,1]$. The goal of the algorithm is to minimize the weak-regret, that is the difference between the reward obtained by the gambler and the best fixed arm, retrospectively.

The first weakness of these algorithms in the context of accelerating boosting, that we have identified in Section 2, is the assumption of stationarity of the loss reduction, which cannot be easily dealt with. Even though the Exp3.P algorithm does not make such an assumption explicitly, it still ignores the boosting weights, and thus can only rely on the history of past rewards.

The second weakness, the application context, can be addressed in our setting by learning the usefulness of the subsets instead of individual features.

A third weakness is that in boosting one aims at minimizing the loss (which translates into maximizing the sum of the rewards for the bandit algorithm), and not at minimizing the weak-regret.

Finally, another issue arises when trying to use those algorithms in practice. As they use some kind of confidence intervals, the scale of the rewards matters greatly. For example, if all the rewards obtained are very small ($\forall t, r^t \leq \epsilon \ll 1$), the algorithms will not learn anything, as they expect rewards to make full use of the range $[0,1]$.

For this reason we set the bandit baselines' meta-parameters to the ones leading to the lowest loss a posteriori, as explained in Section 6.5, and use a third multi-armed bandit algorithm in our experiments, ϵ-greedy (Auer et al., 2002), which does not suffer from this problem.

Hence, we use in our experiments the three following baselines, using the same reward as in (Busa-Fekete and Kegl, 2010):

- **UCB** picks Q features from the subset that maximizes $\bar{r}_j + \sqrt{(2\log n)/n_j}$, where $\bar{r}_j$ is the current average reward of subset j, n_j is the number of times subset j was chosen so far, and n is the current boosting round.

- **Epx3.P** maintains a distribution of weights over the feature subsets, and at every round picks one subset accordingly, obtains a reward, and updates the distribution. For the precise definition of the algorithm, see (Auer et al., 2003; Busa-Fekete and Kegl, 2010).

- **ϵ-greedy** picks Q features from the subset with the highest current average reward with probability $1-\epsilon_n$, or from a random subset with probability ϵ_n, where $\epsilon_n = \frac{cK}{d^2 n}$, and c and d are parameters of the algorithm.

6.5 Results

We tested all the proposed methods of Sections 4, 5.3, and 5.4 against the baselines described in Sections 6.3 and 6.4 on the four benchmark data sets described above in Section 6.2 using the standard train/test cuts and all the features of Section 6.1. We report the results of doing

MNIST								
Methods	T (# boosting steps)							
	10		100		1,000		10,000	
	loss	test error	loss	test error	loss	test error	loss	test error
Best family*	-0.43	36.48	-0.95	5.77	-1.84	1.47	-4.84	0.92
Uniform Naive	-0.38	45.3	-0.85	7.79	-1.74	1.64	-5.37	0.93
Uniform 1.Q	-0.36	49.4	-0.75	10.8	-1.51	2.18	-3.90	1.08
Uniform Q.1	-0.38	43.0	-0.86	7.40	-1.72	1.71	-5.06	0.97
UCB*	-0.40	41.9	-0.79	9.67	-1.64	1.86	-5.54	0.94
Exp3.P*	-0.36	47.9	-0.77	10.3	-1.66	1.79	-5.42	0.92
ϵ**-greedy***	-0.37	45.9	-0.88	7.04	-1.78	1.57	-5.45	**0.88**
Tasting 1.Q	-0.43	36.1	-0.96	5.38	-1.91	1.41	-5.90	0.92
Tasting Q.1	-0.44	34.8	-0.97	5.31	-1.91	1.36	**-5.91**	0.94
M.A.S. Naive	-0.51	26.3	-1.01	4.78	-1.80	1.54	-5.06	0.96
M.A.S. 1.Q	-0.48	29.9	-0.98	5.21	-1.74	1.63	-4.15	1.04
M.A.S. Q.1	-0.43	35.7	-0.98	5.21	-1.78	1.68	-4.51	1.01
Laminating	**-0.55**	**21.9**	**-1.10**	**3.85**	**-2.00**	**1.35**	-5.87	0.96

Table 1: Mean boosting loss ($\log_{10}$) and test error (%) after various number of boosting steps on the MNIST data set with all families of features. Methods highlighted with a $\star$ require the tuning of meta-parameters, which have been optimized by training fully multiple times.

INRIA								
Methods	T (# boosting steps)							
	10		100		1,000		10,000	
	loss	test error	loss	test error	loss	test error	loss	test error
Best family*	-0.34	12.2	-0.93	3.29	-3.72	1.20	-26.9	1.00
Uniform Naive	-0.31	13.4	-0.86	4.87	-3.92	1.27	-31.9	0.53
Uniform 1.Q	-0.30	14.2	-0.95	3.99	-4.26	0.89	-34.3	0.37
Uniform Q.1	-0.30	14.0	-1.01	3.92	-4.86	0.69	-40.0	0.33
UCB*	-0.35	12.1	-1.08	3.17	-5.47	0.61	-49.3	0.30
Exp3.P*	-0.31	13.6	-0.91	4.09	-4.53	0.79	-44.7	0.32
ϵ**-greedy***	-0.34	12.9	-1.11	2.89	-5.92	0.54	-49.3	0.34
Tasting 1.Q	-0.39	10.9	-1.30	2.14	-6.44	0.49	-54.1	0.30
Tasting Q.1	-0.40	11.2	-1.30	2.33	-6.54	0.57	-55.1	0.32
M.A.S. Naive	-0.46	8.80	-1.50	1.66	-7.23	0.44	-60.4	**0.27**
M.A.S. 1.Q	-0.41	10.1	-1.45	1.82	-6.87	0.50	-55.9	0.28
M.A.S. Q.1	-0.44	9.43	-1.51	1.65	-6.97	0.42	-57.1	**0.27**
Laminating	**-0.56**	**6.85**	**-2.05**	**1.12**	**-11.2**	**0.39**	**-99.8**	0.30

Table 2: Mean boosting loss ($\log_{10}$) and test error (%) after various number of boosting steps on the INRIA data set with all families of features. Methods highlighted with a $\star$ require the tuning of meta-parameters, which have been optimized by training fully multiple times.

Caltech 101								
Methods	T (# boosting steps)							
	10		100		1,000		10,000	
	loss	test error	loss	test error	loss	test error	loss	test error
Best family*	-0.80	95.2	-1.44	79.4	-7.17	56.7	-65.5	41.9
Uniform Naive	-0.79	95.8	-1.40	80.3	-6.81	55.6	-61.8	38.8
Uniform 1.Q	-0.79	95.9	-1.36	79.0	-5.84	54.2	-49.6	40.8
Uniform Q.1	-0.81	94.2	-1.44	76.5	-6.74	51.8	-59.2	37.6
UCB*	-0.81	94.2	-1.40	78.6	-6.46	52.6	-61.6	37.0
Exp3.P*	-0.79	95.8	-1.34	80.3	-5.89	54.7	-54.4	40.6
ϵ**-greedy***	-0.81	94.8	-1.42	76.7	-7.26	50.6	-67.1	37.4
Tasting 1.Q	**-0.82**	**93.8**	**-1.50**	**74.2**	**-7.47**	50.7	**-68.1**	**35.3**
Tasting Q.1	**-0.82**	93.9	**-1.50**	74.5	-7.46	**50.5**	**-68.1**	35.5
M.A.S. Naive	-0.80	94.3	-1.43	76.2	-6.70	51.8	-59.1	37.9
M.A.S. 1.Q	-0.78	96.4	-1.01	90.5	-2.04	85.9	-29.5	53.6
M.A.S. Q.1	-0.79	95.8	-1.21	85.6	-5.01	58.7	-42.7	44.5
Laminating	-0.81	94.3	-1.43	77.0	-6.33	53.0	-54.4	38.4

Table 3: Mean boosting loss ($\log_{10}$) and test error (%) after various number of boosting steps on the Caltech 101 data set with all families of features. Methods highlighted with a $\star$ require the tuning of meta-parameters, which have been optimized by training fully multiple times.

CIFAR-10								
Methods	T (# boosting steps)							
	10		100		1,000		10,000	
	loss	test error	loss	test error	loss	test error	loss	test error
Best family	-0.27	73.6	-0.33	57.4	-0.43	44.8	-0.67	40.2
Uniform Naive	-0.26	74.9	-0.34	55.9	-0.48	38.9	-0.93	32.2
Uniform 1.Q	-0.26	76.6	-0.33	57.5	-0.47	39.9	-0.84	31.3
Uniform Q.1	-0.27	74.3	-0.34	53.8	-0.49	37.6	-0.91	30.9
UCB*	-0.27	73.3	-0.34	56.2	-0.49	37.7	-0.90	30.6
Exp3.P*	-0.26	77.2	-0.33	58.0	-0.47	38.9	-0.86	30.3
ϵ-greedy*	-0.26	75.8	-0.35	53.4	-0.49	37.09	-0.88	**30.0**
Tasting 1.Q	-0.28	72.6	-0.36	50.9	**-0.50**	**36.2**	**-0.95**	31.7
Tasting Q.1	-0.28	71.8	-0.36	50.9	**-0.50**	36.3	**-0.95**	31.5
M.A.S. Naive	-0.28	71.9	-0.35	52.5	-0.49	37.5	-0.91	31.0
M.A.S. 1.Q	-0.28	70.7	-0.35	53.3	-0.45	40.5	-0.63	33.8
M.A.S. Q.1	-0.28	71.4	-0.35	52.7	-0.45	40.4	-0.62	34.1
Laminating	**-0.29**	**67.8**	**-0.37**	**49.1**	**-0.50**	36.8	-0.88	31.5

Table 4: Mean boosting loss ($\log_{10}$) and test error (%) after various number of boosting steps on the CIFAR-10 data set with all families of features. Methods highlighted with a $\star$ require the tuning of meta-parameters, which have been optimized by training fully multiple times.

up to 10,000 boosting rounds averaged through ten randomized runs in Tables 1—4. We used as cost for all the algorithms the number of evaluated features, that is for each boosting iteration QS, the number of sampled features times the number of sampled examples. For the Laminating algorithm we multiplied this cost by the number of iterations $\lceil \log_2(Q) \rceil$. We set the maximum cost of all the algorithms to $10N$, setting $Q = 10$ and $S = N$ for the baselines, as this configuration leads to the best results after 10,000 boosting rounds.

The parameters of the baselines—namely the scale of the rewards for UCB and Exp3.P, and the c/d^2 ratio of ϵ-greedy—were optimized by trying all values of the form $2^n, n = \{0, 1, ..., 11\}$, and keeping the one leading to the smallest final boosting loss on the training set, which is unfair to the uniform baselines as well as our methods. We set the values of the parameters of Exp3.P to $\eta = 0.3$ and $\lambda = 0.15$ as recommended in (Busa-Fekete and Kegl, 2010).

These results illustrate the efficiency of the proposed methods. Up to 1,000 boosting rounds, the Laminating algorithms is the clear winner on three out of the four data sets. Then come the M.A.S. and the Tasting procedures, still performing far better than the baselines. On the Caltech 101 data set the situation is different. Since it contains a much smaller number of training examples compared to the other data sets (1515 versus several tens of thousands), there is no advantage in sampling examples. It even proves detrimental as the M.A.S. and Laminating methods are beaten by the baselines after 1,000 iterations.

The performance of all the methods tends to get similar for 10,000 stumps, which is unusually large. The Tasting algorithm appears to fare the best, sampling examples offering no speed gain for such a large number of boosting steps, except on the INRIA data set. On this data set the Laminating algorithm still dominates, although its advantage in loss reduction does not translate into a lower test error anymore.

7. Conclusion

We have improved boosting by modeling the statistical behavior of the weak learners' edges. This allowed us to maximize the loss reduction under strict control of the computational cost. Experiments demonstrate that the algorithms perform well on real-world pattern recognition tasks.

Extensions of the proposed methods could be investigated along two axes. The first one is to merge the best two methods by adding a Tasting component to the Laminating procedure, in order to bias the sampling towards promising feature subsets. The second is to add a bandit-like component to the methods by adding a variance term related to the lack of samples, and their obsolescence in the boosting process. This would account for the degrading density estimation when subsets have not been sampled for a while, and induce an exploratory sampling which may be missing in the current algorithms.

Acknowledgments

We are grateful to Gilles Blanchard for suggesting an improvement of Theorem 1 to make the bound independent of Q.

Charles Dubout was supported by the Swiss National Science Foundation under grant 200021-124822 — VELASH, and François Fleuret was supported in part by the European Community's 7th Framework Programme under grant agreement 247022 — MASH.

Appendix A

Proof of Lemma 1:

Since

$$\max_{r:\epsilon_r \geq \epsilon_{q^\star} - \delta} \hat{\epsilon}_r \geq \hat{\epsilon}_{q^\star} \tag{7}$$

Defining $B_q = \mathbf{1}_{\{\Delta_q - \Delta_{q^\star} \geq \delta\}}$, we have

$$\begin{aligned}
&\mathrm{P}\left(\left|\left\{q:\ \epsilon_q \leq \epsilon_{q^\star} - \delta,\ \hat{\epsilon}_q \geq \max_{r:\epsilon_r \geq \epsilon_{q^\star} - \delta} \hat{\epsilon}_r\right\}\right| \ \geq\ \frac{Q}{2}\right) \\
\leq &\mathrm{P}\left(\left|\{q:\ \epsilon_q \leq \epsilon_{q^\star} - \delta,\ \hat{\epsilon}_q \geq \hat{\epsilon}_{q^\star}\}\right| \ \geq\ \frac{Q}{2}\right) && (8)\\
\leq &\mathrm{P}\left(\left|\{q : \epsilon_q \leq \epsilon_{q^\star} - \delta, \Delta_q - \Delta_{q^\star} \geq \delta\}\right| \geq \frac{Q}{2}\right) && (9)\\
\leq &\mathrm{P}\left(\sum_{q,\, \epsilon_q \leq \epsilon_{q^\star} - \delta} B_q \geq \frac{Q}{2}\right) && (10)\\
\leq &\mathrm{P}\left(\sum_q B_q \geq \frac{Q}{2}\right) && (11)\\
\leq &\mathrm{P}\left(\frac{2\sum_q B_q}{Q} \geq 1\right) && (12)\\
\leq &\frac{2\,\mathrm{E}\left[\sum_q B_q\right]}{Q} && (13)\\
\leq &2\,\mathrm{E}[B_q] && (14)\\
\leq &2\,\mathrm{E}\left[\mathbf{1}_{\{(\Delta_q \geq \frac{\delta}{2}) \cup (\Delta_{q^\star} \leq -\frac{\delta}{2})\}}\right] && (15)\\
\leq &4\exp\left(-\frac{\delta^2 S}{2}\right). && (16)
\end{aligned}$$

■

Equation (7) is true since $q^\star$ is among the $\{r : \epsilon_r \geq \epsilon_{q^\star} - \delta\}$ and δ is positive. Equations (8) to (12) are true since we relax conditions on the event. Equation (13) is true since $\mathrm{P}(X \geq 1) \leq \mathrm{E}(X)$ for $X \geq 0$. Equations (14) and (15) are true analytically, and equation (16) follows from Hœffding's inequality.

Appendix B

Proof of Theorem 1:

Defining $\delta_k = \frac{1}{C}\delta\sqrt{\frac{k}{2^{k-1}}}$ where $C = \sum_{k=1}^{\lceil \log_2(Q) \rceil} \sqrt{\frac{k}{2^{k-1}}}$ is a normalization constant such that the δ_k's sum to the original δ, i.e. $\sum_{k=1}^{\lceil \log_2(Q) \rceil} \delta_k = \delta$.

We apply Lemma 1 with constant δ_k for each of the k Laminating iterations, $1 \leq k \leq \lceil \log_2(Q) \rceil$. Since each iteration samples twice as many training examples as the previous one, and the δ_k's sum to the original δ, the probability to end up with a weak learner with an edge below or equal to $\epsilon_{q^*} - \delta$ is upper bounded by

$$
\begin{aligned}
& 4 \sum_{k=1}^{\lceil \log_2(Q) \rceil} \exp\left(-\frac{\delta_k^2 \, S \, 2^{k-1}}{2}\right) \\
\leq & 4 \sum_{k=1}^{\infty} \exp\left(-\frac{\delta^2 S k}{2C^2}\right) && (17) \\
\leq & 4 \left(\frac{1}{1 - \exp\left(-\frac{\delta^2 S}{2C^2}\right)} - 1\right) && (18) \\
\leq & 4 \left(\frac{1}{1 - \exp\left(-\frac{\delta^2 S}{69}\right)} - 1\right). && (19)
\end{aligned}
$$

■

Equation (17) is true analytically, Equation (18) follows from the formula for geometric series, and Equation (19) is true due to the fact that the constant C is upper bounded by $\sqrt{2}$ times the polylogarithm $\mathrm{Li}_{-\frac{1}{2}}\left(\frac{1}{\sqrt{2}}\right) = \sum_{k=1}^{\infty} \sqrt{\frac{k}{2^k}} \approx 4.15$.

References

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256, 2002.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2003.

J. Bradley and R. Schapire. Filterboost: Regression and classication on large datasets. In *Neural Information Processing Systems*, 2007.

R. Busa-Fekete and B. Kegl. Accelerating AdaBoost using UCB. *JMLR W&CP*, 2009.

R. Busa-Fekete and B. Kegl. Fast Boosting using adversarial bandits. In *ICML*, 2010.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition*, 2005. `http://pascal.inrialpes.fr/data/human/`.

C. Dubout and F. Fleuret. Tasting families of features for image classification. In *International Conference on Computer Vision*, 2011a.

C. Dubout and F. Fleuret. Boosting with maximum adaptive sampling. In *Neural Information Processing Systems*, 2011b.

N. Duffield, C. Lund, and M. Thorup. Priority sampling for estimation of arbitrary subset sums. *J. ACM*, 54, December 2007.

G. Escudero, L. Màrquez, and G. Rigau. Boosting applied to word sense disambiguation. *Machine Learning: ECML 2000*, pages 129–141, 2000.

L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In *CVPR, Workshop on Generative-Model Based Vision*, 2004. `http://www.vision.caltech.edu/Image_Datasets/Caltech101/`.

F. Fleuret and D. Geman. Stationary features and cat detection. *Journal of Machine Learning Research*, 9:2549–2578, 2008.

F. Fleuret, T. Li, C. Dubout, E. K. Wampler, S. Yantis, and D. Geman. Comparing machines and humans on a visual categorization test. *Proceedings of the National Academy of Sciences*, 108(43):17621–17625, 2011.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.

P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *International Conference on Computer Vision*, 2009.

Z. Kalal, J. Matas, and K. Mikolajczyk. Weighted sampling for large-scale Boosting. *British machine vision conference*, 2008.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. `http://www.cs.toronto.edu/~kriz/cifar.html`.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86(11), pages 2278–2324, 1998. `http://yann.lecun.com/exdb/mnist/`.

A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with Boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:416–431, 2006.

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999.

Journal of Machine Learning Research 15 (2014) 1455-1459 Submitted 8/13; Revised 1/14; Published 4/14

Manopt, a Matlab Toolbox for Optimization on Manifolds

Nicolas Boumal NICOLASBOUMAL@GMAIL.COM
Department of Mathematical Engineering,
Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium

Bamdev Mishra BAMDEVM@GMAIL.COM
Department of Electrical Engineering and Computer Science
Université de Liège, B-4000 Liège, Belgium

P.-A. Absil ABSIL@INMA.UCL.AC.BE
Department of Mathematical Engineering
Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium

Rodolphe Sepulchre R.SEPULCHRE@ENG.CAM.AC.UK
Department of Engineering (Control Group)
University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK

Editor: Antti Honkela

Abstract

Optimization on manifolds is a rapidly developing branch of nonlinear optimization. Its focus is on problems where the smooth geometry of the search space can be leveraged to design efficient numerical algorithms. In particular, optimization on manifolds is well-suited to deal with rank and orthogonality constraints. Such structured constraints appear pervasively in machine learning applications, including low-rank matrix completion, sensor network localization, camera network registration, independent component analysis, metric learning, dimensionality reduction and so on.

The Manopt toolbox, available at `www.manopt.org`, is a user-friendly, documented piece of software dedicated to simplify experimenting with state of the art Riemannian optimization algorithms. By dealing internally with most of the differential geometry, the package aims particularly at lowering the entrance barrier.

Keywords: Riemannian optimization, nonlinear programming, non convex, orthogonality constraints, rank constraints, optimization with symmetries, rotation matrices

1. Introduction

Optimization on manifolds, or Riemannian optimization, is a fast growing research topic in the field of nonlinear optimization. Its purpose is to provide efficient numerical algorithms to find (at least local) optimizers for problems of the form

$$\min_{x \in \mathcal{M}} f(x), \tag{1}$$

where the search space $\mathcal{M}$ is a smooth space: a differentiable manifold which can be endowed with a Riemannian structure. In a nutshell, this means $\mathcal{M}$ can be linearized locally at each point x as a tangent space $\mathrm{T}_x\mathcal{M}$ and an inner product $\langle \cdot, \cdot \rangle_x$ which smoothly depends on x is available on $\mathrm{T}_x\mathcal{M}$. For example, when $\mathcal{M}$ is a submanifold of $\mathbb{R}^{n \times m}$, a typical inner product is $\langle H_1, H_2 \rangle_X = \mathrm{trace}(H_1^\top H_2)$. Many smooth search spaces arise often in applications.

For example, the **oblique manifold** $\mathcal{M} = \{X \in \mathbb{R}^{n\times m} : \operatorname{diag}(X^\top X) = \mathbb{1}_m\}$ is a product of spheres. That is, $X \in \mathcal{M}$ if each column of X has unit 2-norm in $\mathbb{R}^n$. Absil and Gallivan (2006) show how independent component analysis can be cast on this manifold as non-orthogonal joint diagonalization.

When furthermore it is only the product $Y = X^\top X$ which matters, matrices of the form QX are equivalent for all orthogonal Q. This suggests a quotient geometry for the **fixed-rank elliptope** $\mathcal{M} = \{Y \in \mathbb{R}^{m\times m} : Y = Y^\top \succeq 0, \operatorname{rank}(Y) = n, \operatorname{diag}(Y) = \mathbb{1}_m\}$. Grubišić and Pietersz (2007) optimize over this set to produce low-rank approximations of covariance matrices.

The (compact) **Stiefel manifold** is the Riemannian submanifold of orthonormal matrices, $\mathcal{M} = \{X \in \mathbb{R}^{n\times m} : X^\top X = I_m\}$. Theis et al. (2009) formulate independent component analysis with dimensionality reduction as optimization over the Stiefel manifold. Journée et al. (2010b) frame sparse principal component analysis over this manifold as well.

The **Grassmann manifold** $\mathcal{M} = \{\operatorname{col}(X) : X \in \mathbb{R}^{n\times m}_*\}$, where X is a full-rank matrix and $\operatorname{col}(X)$ denotes the subspace spanned by its columns, is the set of subspaces of $\mathbb{R}^n$ of dimension m. Among other things, optimization over the Grassmann manifold is useful in low-rank matrix completion, where it is observed that if one knows the column space spanned by the sought matrix, then completing the matrix according to a least-squares criterion is easy (Boumal and Absil, 2011; Keshavan et al., 2010).

The **special orthogonal group** $\mathcal{M} = \{X \in \mathbb{R}^{n\times n} : X^\top X = I_n \text{ and } \det(X) = 1\}$ is the group of rotations, typically considered as a Riemannian submanifold of $\mathbb{R}^{n\times n}$. Optimization problems involving rotation matrices occur in robotics and computer vision, when estimating the attitude of vehicles or the pose of cameras (Boumal et al., 2013).

The set of **fixed-rank matrices** $\mathcal{M} = \{X \in \mathbb{R}^{n\times m} : \operatorname{rank}(X) = k\}$ admits a number of different Riemannian structures. Vandereycken (2013) proposes an embedded geometry for $\mathcal{M}$ and exploits Riemannian optimization on that manifold to address the low-rank matrix completion problem. Shalit et al. (2012) use the same geometry to address similarity learning. Mishra et al. (2012) further cover a number of quotient geometries.

The set of **symmetric, positive semidefinite, fixed-rank matrices** is also a manifold, $\mathcal{M} = \{X \in \mathbb{R}^{n\times n} : X = X^\top \succeq 0, \operatorname{rank}(X) = k\}$. Meyer et al. (2011) exploit this to propose low-rank algorithms for metric learning. This space is tightly related to the space of **Euclidean distance matrices** X such that X_{ij} is the squared distance between two fixed points $x_i, x_j \in \mathbb{R}^k$. Mishra et al. (2011) leverage this geometry to formulate efficient low-rank algorithms for Euclidean distance matrix completion.

The rich geometry of Riemannian manifolds makes it possible to define gradients and Hessians of cost functions f, as well as systematic procedures (called *retractions*) to move on the manifold starting at a point x, along a specified tangent direction at x. Those are sufficient ingredients to generalize standard nonlinear optimization methods such as gradient descent, conjugate gradients, quasi-Newton, trust-regions, etc.

Building upon many earlier results not reviewed here, the recent monograph by Absil et al. (2008) sets an algorithmic framework to analyze problems of the form (1) when f is a smooth function, with a strong emphasis on building a theory that leads to efficient numerical algorithms on special manifolds. In particular, it describes the necessary ingredients to design first- and second-order algorithms on Riemannian submanifolds and quotient manifolds of linear spaces. These algorithms come with numerical costs and convergence

guarantees essentially matching those of the Euclidean counterparts they generalize. For example, the Riemannian trust-region method converges globally toward critical points and converges locally quadratically when the Hessian of f is available.

The maturity of the theory of smooth Riemannian optimization, its widespread applicability and its excellent track record performance-wise prompted us to build the Manopt toolbox: a user-friendly piece of software to help researchers and practitioners experiment with these tools. Code and documentation are available at `www.manopt.org`.

2. Architecture and features of Manopt

The toolbox architecture is based on a separation of the manifolds, the solvers and the problem descriptions. For basic use, one only needs to pick a manifold from the library, describe the cost function (and possible derivatives) on this manifold and pass it on to a solver. Accompanying tools help the user in common tasks such as numerically checking whether the cost function agrees with its derivatives up to the appropriate order, approximating the Hessian based on the gradient of the cost, etc.

Manifolds in Manopt are represented as structures and are obtained by calling a factory. The manifold descriptions include projections on tangent spaces, retractions, helpers to convert Euclidean derivatives (gradient and Hessian) to Riemannian derivatives, etc. All the manifolds mentioned in the introduction work out of the box, and more can be added. Cartesian products of known manifolds are supported too.

Solvers in Manopt are functions that implement generic Riemannian minimization algorithms. Solvers log standard information at each iteration and comply with standard stopping criteria. Users may provide callbacks to log extra information or check custom stopping criteria. Currently available solvers include Riemannian trust-regions—based on work by Absil et al. (2007)—and conjugate gradients (both with preconditioning), as well as steepest descent and a couple of derivative-free schemes. More solvers can be added.

An optimization problem in Manopt is represented as a problem structure. The latter includes a field which contains a manifold, as obtained from a factory. Additionally, the problem structure hosts function handles for the cost function f and (possibly) its derivatives. An abstraction layer at the interface between the solvers and the problem description offers great flexibility in the cost function description. As the needs grow during the lifecycle of the toolbox and new ways of describing f become necessary (subdifferentials, partial gradients, etc.), it will be sufficient to update this interface.

Computing $f(x)$ typically produces intermediate results which can be reused in order to compute the derivatives of f at x. To prevent redundant computations, Manopt incorporates an (optional) caching system, which becomes useful when transitioning from a proof-of-concept draft of the algorithm to a convincing implementation.

3. Example: the maximum cut problem

Given an undirected graph with n nodes and weights $w_{ij} \geq 0$ on the edges such that $W \in \mathbb{R}^{n\times n}$ is the weighted adjacency matrix and $D \in \mathbb{R}^{n\times n}$ is the diagonal degree matrix with $D_{ii} = \sum_j w_{ij}$, the graph Laplacian is the positive semidefinite matrix $L = D - W$. The max-cut problem consists in building a partition $s \in \{+1, -1\}^n$ of the nodes in two

classes such that $\frac{1}{4}s^\top Ls = \sum_{i<j} w_{ij}\frac{(s_i-s_j)^2}{4}$, that is, the sum of the weights of the edges connecting the two classes, is maximum. Let $X = ss^\top$. Then, max-cut is equivalent to:

$$\begin{aligned}\max_{X\in\mathbb{R}^{n\times n}} \ & \operatorname{trace}(LX)/4 \\ \text{s.t. } & X = X^\top \succeq 0, \operatorname{diag}(X) = \mathbb{1}_n \text{ and } \operatorname{rank}(X) = 1.\end{aligned}$$

Goemans and Williamson (1995) proposed and analyzed the famous relaxation of this problem which consists in dropping the rank constraint, yielding a semidefinite program. Alternatively relaxing the rank constraint to be $\operatorname{rank}(X) \leq r$ for some $1 < r < n$ yields a tighter but nonconvex relaxation. Journée et al. (2010a) observe that fixing the rank with the constraint $\operatorname{rank}(X) = r$ turns the search space into a smooth manifold, the fixed-rank elliptope, which can be optimized over using Riemannian optimization. In Manopt, simple code for this reads (with $Y \in \mathbb{R}^{n\times r}$ such that $X = YY^\top$):

```
% The problem structure hosts a manifold structure as well as function handles
% to define the cost function and its derivatives (here provided as Euclidean
% derivatives, which will be converted to their Riemannian equivalent).
problem.M = elliptopefactory(n, r);
problem.cost  = @(Y)    -trace(Y'*L*Y)/4;
problem.egrad = @(Y)    -(L*Y)/2;
problem.ehess = @(Y, U) -(L*U)/2;           % optional

% These diagnostics tools help make sure the gradient and Hessian are correct.
checkgradient(problem); pause;
checkhessian(problem);  pause;

% Minimize with trust-regions, a random initial guess and default options.
Y = trustregions(problem);
```

Randomly projecting Y yields a cut: `s = sign(Y*randn(r, 1))`. The Manopt distribution includes advanced code for this example, where the caching functionalities are used to avoid redundant computations of the product LY in the cost and the gradient, and the rank r is increased gradually to obtain a global solution of the max-cut SDP (and hence a formal upperbound), following a procedure described by Journée et al. (2010a).

Acknowledgments

NB and BM are research fellows with the FNRS (aspirants). This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office. It was financially supported by the Belgian FRFC.

References

P.-A. Absil and K.A. Gallivan. Joint diagonalization on the oblique manifold for independent component analysis. In *Acoustics, Speech and Signal Processing, ICASSP 2006. IEEE International Conference on*, volume 5, 2006. doi: 10.1109/ICASSP.2006.1661433.

P.-A. Absil, C. G. Baker, and K. A. Gallivan. Trust-region methods on Riemannian manifolds. *Foundations of Computational Mathematics*, 7(3):303–330, July 2007.

P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

N. Boumal and P.-A. Absil. RTRMC: A Riemannian trust-region method for low-rank matrix completion. In J. Shawe-Taylor et al., editors, *Advances in Neural Information Processing Systems 24 (NIPS)*, pages 406–414. 2011.

N. Boumal, A. Singer, and P.-A. Absil. Robust estimation of rotations from relative measurements by maximum likelihood. *Proceedings of the 52nd Conference on Decision and Control, CDC 2013*, 2013.

M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

I. Grubišić and R. Pietersz. Efficient rank reduction of correlation matrices. *Linear algebra and its applications*, 422(2):629–653, 2007.

M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010a.

M. Journée, Y. Nesterov, P. Richtárik, and R. Sepulchre. Generalized power method for sparse principal component analysis. *The Journal of Machine Learning Research*, 11: 517–553, 2010b.

R.H. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. *The Journal of Machine Learning Research*, 99:2057–2078, 2010.

G. Meyer, S. Bonnabel, and R. Sepulchre. Regression on fixed-rank positive semidefinite matrices: a Riemannian approach. *The Journal of Machine Learning Research*, 12:593–625, 2011.

B. Mishra, G. Meyer, and R. Sepulchre. Low-rank optimization for distance matrix completion. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 4455–4460. IEEE, 2011.

B. Mishra, G. Meyer, S. Bonnabel, and R. Sepulchre. Fixed-rank matrix factorizations and Riemannian low-rank optimization. *Arxiv preprint arXiv:1209.0430*, 2012.

U. Shalit, D. Weinshall, and G. Chechik. Online learning in the embedded manifold of low-rank matrices. *The Journal of Machine Learning Research*, 13:429–458, 2012.

F.J. Theis, T.P. Cason, and P.-A. Absil. Soft dimension reduction for ICA by joint diagonalization on the Stiefel manifold. In *Independent Component Analysis and Signal Separation*, pages 354–361. Springer, 2009.

B. Vandereycken. Low-rank matrix completion by Riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013. doi: 10.1137/110845768.

Journal of Machine Learning Research 15 (2014) 1461-1492 Submitted 2/13; Revised 12/13; Published 4/14

Training Highly Multiclass Classifiers

Maya R. Gupta MAYAGUPTA@GOOGLE.COM
Samy Bengio BENGIO@GOOGLE.COM
Google Inc.
1600 Amphitheatre Pkwy
Mountain View, CA 94301, USA

Jason Weston JASEWESTON@GMAIL.COM
Google Inc.
76 9th Avenue,
New York, NY 10011 USA

Editor: Koby Crammer

Abstract

Classification problems with thousands or more classes often have a large range of class-confusabilities, and we show that the more-confusable classes add more noise to the empirical loss that is minimized during training. We propose an online solution that reduces the effect of highly confusable classes in training the classifier parameters, and focuses the training on pairs of classes that are easier to differentiate at any given time in the training. We also show that the adagrad method, recently proposed for automatically decreasing step sizes for convex stochastic gradient descent, can also be profitably applied to the non-convex joint training of supervised dimensionality reduction and linear classifiers as done in Wsabie. Experiments on ImageNet benchmark data sets and proprietary image recognition problems with 15,000 to 97,000 classes show substantial gains in classification accuracy compared to one-vs-all linear SVMs and Wsabie.

Keywords: large-scale, classification, multiclass, online learning, stochastic gradient

1. Introduction

Problems with many classes abound: from classifying a description of a flower as one of the over 300,000 known flowering plants (Paton et al., 2008), to classifying a whistled tune as one of the over 30 million recorded songs (Eck, 2013). Many practical multiclass problems are labelling images, for example face recognition, or tagging locations in vacation photos. In practice, the more classes considered, the greater the chance that some classes will be easy to separate, but that some classes will be highly confusable.

When training a discriminative multi-class classifier, the true goal is to minimize expected error on future samples, but in practice we minimize empirical error on samples we already have. In this paper, we show that classes that are more confusable add more noise to the empirical loss. To address this, we propose approximating the expected error with a different empirical loss we term the *empirical class-confusion loss*. For the large-scale online training, we show that an *online empirical class-confusion loss* can be implemented for stochastic gradient descent by simply ignoring stochastic gradients corresponding to a repeated confusion between classes. This proposed strategy also automatically implements

a form of *curriculum learning*, that is, of learning to distinguish easy classes before focusing on learning to distinguish hard classes (Bengio et al., 2009).

In this paper, we focus on classifiers that use a linear discriminant or a single prototypical feature vector to represent each class. Linear classifiers are a popular approach to highly multiclass problems because they are efficient in terms of memory and inference and can provide good performance (Perronnin et al., 2012; Lin et al., 2011; Sanchez and Perronnin, 2011). Class prototypes offer similar memory/efficiency advantages. The last layer of a deep belief network classifier is often linear or soft-max discriminant functions (Bengio, 2009), and the proposed ideas for adapting online loss functions should be applicable in that context as well.

We apply the proposed loss function adaptation to the multiclass linear classifier called Wsabie (Weston et al., 2011). We also simplify Wsabie's weighting of stochastic gradients, and employ a recent advance in automatic step-size adaptation called *adagrad* (Duchi et al., 2011). The resulting proposed Wsabie^{++} classifier almost doubles the classification accuracy on benchmark Imagenet data sets compared to Wsabie, and shows substantial gains over one-vs-all SVMs.

The rest of the article is as follows. After establishing notation in Section 2, we explain in Section 3 how different class confusabilities can distort the standard empirical loss. We then review loss functions for jointly training multiclass linear classifiers in Section 4, and stochastic gradient descent variants for large-scale learning in Section 5. In Section 6, we propose a practical online solution to adapt the empirical loss to account for the variance of class confusability. We describe our adagrad implementation in Section 7. Experiments are reported on benchmark and proprietary image classification data sets with 15,000-97,000 classes in Section 8 and 9. We conclude with some notes about the key issues and unresolved questions.

2. Notation And Assumptions

We take as given a set of training data $\{(x_t, \mathcal{Y}_t)\}$ for $t = 1, \ldots, n$, where $x_t \in \mathbb{R}^d$ is a feature vector and $\mathcal{Y}_t \subset \{1, 2, \ldots, G\}$ is the subset of the G class labels that are known to be correct labels for x_t. For example, an image might be represented by set of features x_t and have known labels $\mathcal{Y}_t = \{\texttt{dolphin}, \texttt{ocean}, \texttt{Half Moon Bay}\}$. We assume a discriminant function $f(x; \beta_g)$ has been chosen with class-specific parameters β_g for each class with $g = 1, \ldots, G$. The class discriminant functions are used to classify a test sample x as the class label that solves

$$\arg\max_g f(x; \beta_g). \tag{1}$$

Most of this paper applies equally well to "learning to rank," in which case the output might be a top-ranked or ranked-and-thresholded list of classes for a test sample x. For simplicity, we restrict our discussion and metrics to the classification paradigm given by (1).

Many of the ideas in this paper can be applied to any choice of discriminant function $f(x, \beta_g)$, but in this paper we focus on efficiency in terms of test-time and memory, and so we focus on class discriminants that are parameterized by a d-dimensional vector per class. Two such functions are: the inner product $f(x; \beta_g) = \beta_g^T x$, and the squared ℓ_2 norm $f(x; \beta_g) = -(\beta_g - x)^T(\beta_g - x)$. We also refer to these as linear discriminants and

Euclidean distance discriminants, respectively. For example, one-vs-all linear SVMs use a linear discriminant, where the β_g are each trained to maximize the margin between samples from the gth class and all samples from all other classes. The nearest means classifier (Hastie et al., 2001) uses an Euclidean distance discriminant where each class prototype β_g is set to be the mean of all the training samples labelled with class g. Both the linear and nearest-prototype functions produce linear decision boundaries between classes. And with either the linear or Euclidean discriminants, the classifier has a total of $G \times d$ parameters, and testing as per (1) scales as $O(Gd)$.

To reduce memory and test time, and also as a regularizer, it may be useful for the classifier to include a dimensionality reduction matrix (sometimes called an embedding matrix) $W \in \mathbb{R}^{m \times d}$, and then use linear or Euclidean discriminants in the reduced dimensionality space, for example $f(x; W, \beta_g) = \beta_g^T W x$ or $f(x; W, \beta_g) = -(\beta_g - x)^T W (\beta_g - x)$.

3. The Problem with a Large Variance in Class Confusability

The underlying goal when discriminatively training a classifier is to minimize expected classification error, but this goal is often approximated by the empirical classification errors on a given data set. In this section, we show that the expected error does not count errors between confusable classes (like `dolphin` and `porpoise`) the same as errors between separable classes (like `cat` and `dolphin`), whereas the empirical error counts all errors equally. Consequently, more confusable classes add more noise to the standard empirical error approximation of the expected error, and this confusable-class noise can adversely affect training.

Then in Section 6, we propose addressing this issue by changing the way we measure empirical loss to reduce the impact of errors between more-confusable classes.

3.1 Expected Classification Error Depends on Class Confusability

Define a classifier c as a map from an input feature vector x to a class such that $c : \mathbb{R}^d \to 1, 2, \ldots, G$. Let I be the indicator function, and assume there exists a joint probability distribution $P_{X,Y}$ on the random feature vector $X \in \mathbb{R}^d$ and class $Y \in \{1, 2, \ldots, G\}$. Then the expected classification error of classifier c is:

$$
\begin{aligned}
E_{X,Y}[I_{Y \neq c(X)}] &= E_X\left[E_{Y|X}[I_{Y \neq c(X)}]\right] && (2)\\
&= E_X\left[P_{Y|X}(Y \neq c(X))\right] \text{ because } I \text{ is a Bernoulli random variable} && (3)\\
&\approx \frac{1}{n}\sum_{t=1}^{n} P_{Y|x_t}(Y \neq c(x_t)), \text{law of large numbers approximation} && (4)\\
&\approx \frac{1}{n}\sum_{t=1}^{n} I_{y_t \neq c(x_t)}, && (5)
\end{aligned}
$$

where the approximation in (4) replaces the expectation with an average over n samples, an approximation that is asymptotically correct as $n \to \infty$ by the law of large numbers (LLN). The final approximation given in (5) produces the standard empirical error.

Equations (3) and (4) show that the expected error depends on the *probability* that a given feature vector x_t has corresponding random class label Y_t equal to the classifier's decision $c(x_t)$. For example, suppose that sample x_t is equally likely to be class 1 or class 2, but no other class. If the classifier labels x_t as $c(x_t) = 1$, one should add $P_{Y|x_t}(Y \neq (c(x_t) = 1)) = 1/2$ to the approximate error given by (4). On the other hand, suppose that for another sample x_j, the probability of class 1 is .99 and the probability of class 3 is .01. Then if the classifier calls $c(x_j) = 1$ we should add $P_{Y|x_t}(Y \neq (c(x_t) = 1)) = .01$ to the loss, whereas if the classifier calls $c(x_j) = 3$ we should add $P_{Y|x_t}(Y \neq (c(x_t) = 1)) = .99$ to the loss. This relative weighting based on class confusions is in contrast to the standard empirical loss given in (5) that simply counts all errors equally.

One can interpret the standard empirical loss given in (5) as the maximum likelihood approximation to (2) that estimates $P_{Y|x_t}$ given the training sample pair (x_t, y_t) as $\hat{P}_{Y|x_t}(y_t) = 1$ and $\hat{P}_{Y|x_t}(g) = 0$ for all other classes g. This maximum likelihood estimate converges asymptotically to (4), but for a finite number of training samples may produce a poor approximation. For binary classifiers, the approximation (5) may be quite good. We argue that (5) is generally a worse approximation as the number of classes increases. The key issue is that while the one-or-zero error approximation in (5) asymptotically converges to $P_{Y|x_t}$, it *converges more slowly* when classes are *more confusable*, and thus more-confusable classes add more "noise" to the approximation than less-confusable classes, biasing the empirical loss to overfit the noise of the more confusable class confusions.

Let us characterize this difference in noise. For any feature vector x and classifier c the true class label Y is a random variable, and thus the indicator $I_{Y \neq c(x_t)}$ in (5) is a random indicator with a binomial distribution with parameter $p = P_{Y|x_t}(Y \neq c(x_t))$. The variance of the random indicator $I_{Y \neq c(x_t)}$ is $p(1-p)$, and thus the more confusable the classes, the more variance there will be in the corresponding samples' contribution to the empirical loss.

Beyond noting the variance of the empirical errors is quadratic in p, it is not straightforward to formally characterize the distribution of the empirical loss for binomials with different p for finite n (see for example Brown et al., 2001). However we can emphasize this point with a histogram of simulated empirical errors in Figure 1. The figure shows histograms of 1000 different simulations of the empirical error, calculated by averaging either 10 random samples (top) or 100 random samples (bottom) that either have $p = P_{Y|x_t}(Y \neq c(x_t)) = 0.5$ (left) or $p = P_{Y|x_t}(Y \neq c(x_t)) = 0.01$ (right).

The left-hand side of Figure 1 corresponds to samples x that are equally likely to be one of two classes, and so even the Bayes classifier is wrong half the time, such that $p = P_{Y|x_t}(Y \neq c(x_t)) = 0.5$. The empirical error of such samples will eventually converge to the true error 0.5, but we see (top left) that the empirical error of ten such samples varies greatly! Even one hundred such samples (bottom left) are often a full .1 away from their converged value. This is in contrast to the right-hand examples corresponding to samples x_t that are easily classified such that $p = P_{Y|x_t}(Y \neq c(x_t)) = 0.01$. Their empirical error is generally much closer to the correct .01. Thus the more-confusable classes add more noise to the standard empirical loss approximation (5).

in the special case that the Bayes error is zero and the classifier c is the Bayes classifier the standard empirical loss approximation (5) is exact. For practical classification problems with many classes, we argue that at least some classes will be very confusable, and thus the Bayes error will not be zero, and (5) can be a dangerous approximation to use for training.

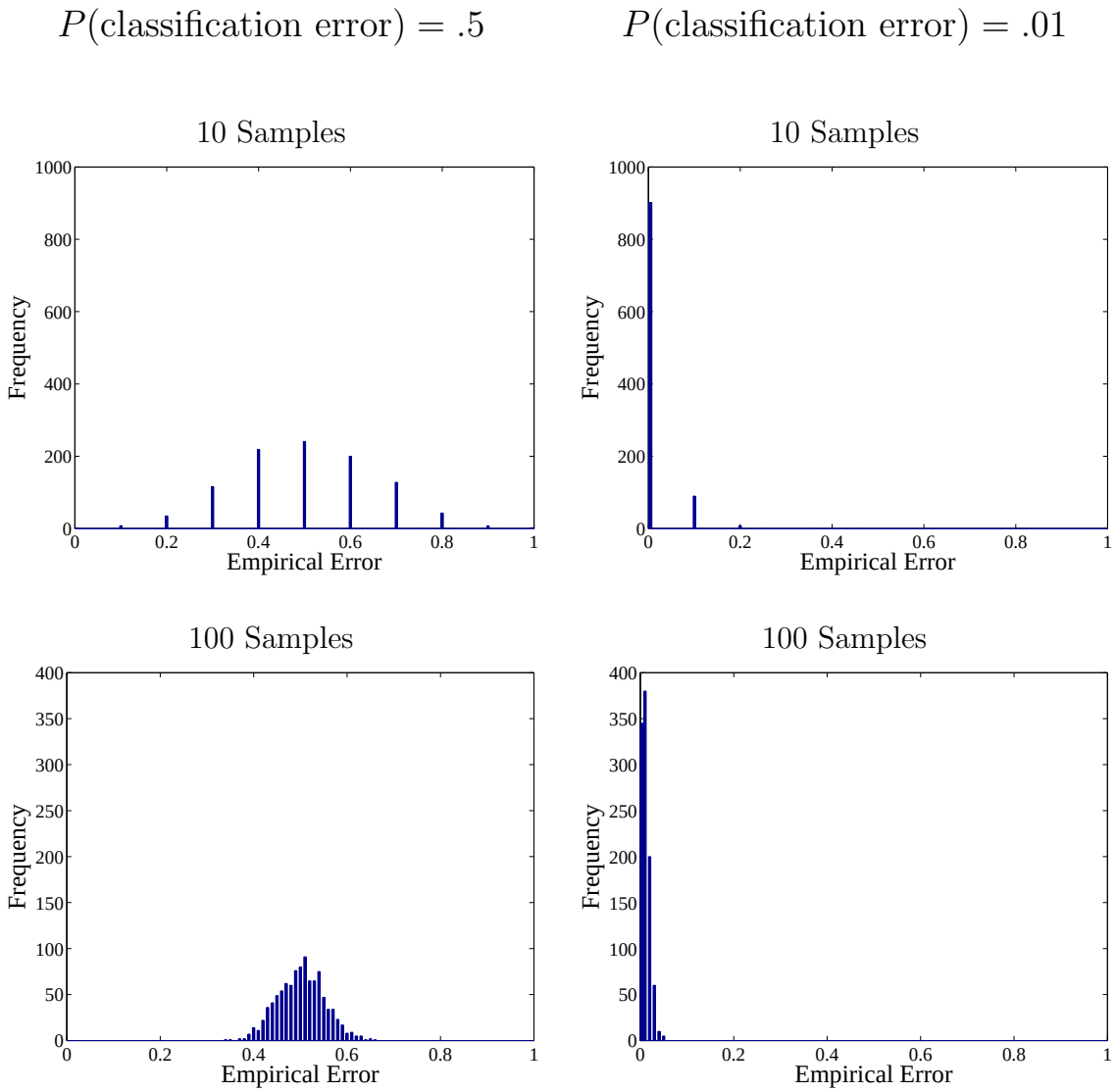
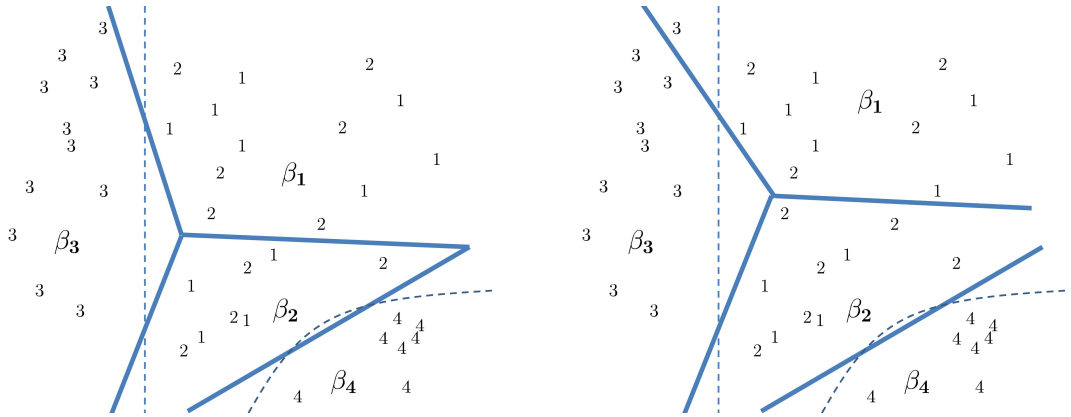


Figure 1: Histograms of the empirical error of 10 random samples (top) or 100 random samples (bottom). As the number of samples averaged grows, the empirical error will converge to the true probability of an error, either .5 (left) or .01 (right). But given a finite sample, the empirical error may be quite noisy, and when the true error is high (left) the empirical error can be much noisier than when the true error is low (right).



Empirical errors: 10 Empirical errors: 9

Figure 2: Two classifiers and the same draw of random training samples from four classes. Dotted lines correspond to the Bayes decision boundaries, and indicate that class 1 and class 2 are indistinguishable (same Bayes decision regions). Solid lines correspond to the classifier decision boundaries, determined by which class prototype $\beta_1, \beta_2, \beta_3$, or β_4 is closest. The two figures differ in the placement of β_1, which produces different classifier decision boundaries. In this case, because of the randomness of the given training samples, the empirical error is higher for the left classifier than the right classifier, but the left classifier is closer to the optimal Bayes classifier.

Figure 2 shows an example of empirical error being overfit to noise between confusable classes. The figure compares two classifiers. Each classifier uses a Euclidean discriminant function, that is, the gth class is represented by a prototype vector $\{\beta_g\}$, and a feature vector is classified as the nearest prototype with respect to Euclidean distance. Thus the decision boundaries are formed by the Voronoi diagram denoted with the thick lines, and the decision boundary between any two classes is linear.

The two classifiers in Figure 2 differ only in the placement of β_1. One sees that decision boundaries produced are not independent of each other: the right classifier has moved β_1 up to reduce empirical errors between class 1 and class 2, but this also changes the decision boundary between classes 1 and 3, and incurs a new empirical error of a class 3 sample. The left classifier in Figure 2 is actually closer to the Bayes decision boundaries (shown by the dotted lines), and would have lower error on a test set (on average).

If the feature dimension is high enough, then as the number of classes G grows, the number of decision boundaries between classes can grow at a worst-case rate of G^2, and yet the ability of these efficient classifiers to describe decision boundaries is fixed at Gd degrees of freedom. And with high-dimensional feature spaces, many classes are next to each other. This interdependence of the pairwise class decision boundaries is why simply minimizing the total empirical error is a bad strategy: it is too sensitive to the empirical error noise of the more-confusable classes.

3.2 Two Factors We Mostly Ignore In This Discussion

Throughout this paper we ignore the dependence of $P_{Y|x}(Y \neq c(x))$ on the particular feature vector x, and focus instead on how confusable a particular class $y = c(X)$ is averaged over X. For example, pictures of porpoises may on average be confused with pictures of dolphins, even though a particular image of a porpoise may be more or less confusable.

Also, we have thus far ignored the fact that discriminative training usually makes a further approximation of (5) by replacing the indicator by a convex approximation like the hinge loss. Such convex relaxations do not avoid the issues described in the previous section, though they may help. For example, the hinge loss increases the weight of an error that is made farther from the decision boundary. To the extent that classes are less confusable farther away from the decision boundary, the hinge loss may be a better approximation than the indicator to the probabilistic weighting of (4). However, if the features are high-dimensional, the distribution of distances from the decision boundary may be less variable than one would expect from two-dimensional intuition (see for example Hall and Marron, 2005).

3.3 A Different Approximation for the Empirical Loss

We argued above that when computing the empirical test error, approximating $P_{Y|x_t}(Y \neq c(x_t))$ by 1 if $y_t = c(x_t)$ and by 0 otherwise adds preferentially more label noise from more confusable classes.

Here we propose a different approximation for $P_{Y|x_t}(Y \neq c(x_t))$. As usual, if $c(x_t) = y_t$, we approximate $P_{Y|x_t}(Y \neq c(x_t))$ by 0. But if $c(x_t) \neq y_t$, then we use the empirical probability that a training sample that has label y_t is not classified as $c(x_t)$:

$$P_{Y|x_t}(Y \neq c(x_t)) \approx E_{X|Y=y_t}\left[P(c(X) \neq c(x_t)\right] \tag{6}$$

$$\approx \left(\frac{\sum_{\tau=1}^{n} I_{y_\tau = y_t} I_{c(x_\tau) \neq c(x_t)}}{\sum_{\tau=1}^{n} I_{y_\tau = y_t}} \right) I_{y_t \neq c(x_t)}. \tag{7}$$

This approximation depends on how consistently feature vectors with training label y_t are classified as class $c(x_t)$, and counts common class-confusions less. For example, consider the right-hand classifier $c(x)$ in Figure 2. There is just one training sample labeled `3` that is incorrectly classified as class `1`. The cost of that error according to (7) is 10/11, because there are eleven class `3` examples, and ten of those are classified as class `3`. On the other hand, the cost of incorrectly labeling a sample of class `1` as a sample of class `2` would be only 7/11, because seven of the eleven class `1` samples are not labeled as class `2`.

Thus this approximation generally has the desired effect of counting confusions between confusable classes relatively less than confusions between easy-to-separate classes. This approximation is more exact for "good" classifiers $c(x)$ that are more similar to the Bayes classifier, and more exact if the feature vectors X are equally predictive for each class label so that averaging over X in (6) is a good approximation for most realizations x_t.

One could implement this approximation in a sequential process: first train a classifier, then compute the empirical class-confusion probability matrix, and then re-train a classifier

using the approximation (7) for the empirical loss. Shamir and Dekel (2010) proposed a related but more extreme two-step approach for highly multi-class problems: first train a classifier on all classes, and then delete classes that are poorly estimated by the classifier.

To be more practical, we propose continuously evolving the classifier to ignore the currently highly-confusable classes by implementing (6) in an online fashion with SGD. This simple variant can be interpreted as implementing curriculum learning (Bengio et al., 2009), a topic we discuss further in Section 6.2.1. But before detailing the proposed simple online strategy in Section 6, we need to review related work in loss functions for multi-class classifiers.

4. Related Work in Loss Functions for Multiclass Classifiers

In this section, we review loss functions for multiclass classifiers, and discuss recent work adapting such loss functions to the online setting for large-scale learning.

One of the most popular classifiers for highly multiclass learning is one-vs-all linear SVMs, which have only $O(Gd)$ parameters to learn and store, and $O(Gd)$ time needed for testing. A clear advantage of one-vs-all is that the G class discriminant functions $\{f_g(x)\}$ can be trained independently. An alternate parallelizable approach is to train all G^2 one-vs-one SVMs, and let them vote for the best class (also known as round-robin and all-vs-all). Binary classifiers can also be combined using error-correcting code approaches (Dietterich and Bakiri, 1995; Allwein et al., 2000; Crammer and Singer, 2002). A well-regarded experimental study of multiclass classification approaches by Rifkin and Klatau (2004) showed that one-vs-all SVMs performed "just as well" on a set of ten benchmark data sets with 4-49 classes as one-vs-one or error-correcting code approaches.

A number of researchers have independently extended the two-class SVM optimization problem to a joint multiclass optimization problem that maximizes pairwise margins subject to the training samples being correctly classified, with respect to pairwise slack variables (Vapnik, 1998; Weston and Watkins, 1998, 1999; Bredensteiner and Bennet, 1999).[1] These extensions have been shown to be essentially equivalent quadratic programming problems (Guermeur, 2002). The minimized empirical loss can be stated as the sum of the pairwise errors:

$$L_{\text{pairwise}}(\{\beta_g\}) = \sum_{t=1}^{n} \quad \frac{1}{|\mathcal{Y}_t|} \sum_{y^+ \in \mathcal{Y}_t} \frac{1}{|\mathcal{Y}_t^C|} \quad \sum_{y^- \in \mathcal{Y}_t^C} |b - f(x_t; \beta_{y^+}) + f(x_t; \beta_{y^-})|_+, \qquad (8)$$

where $|\cdot|_+$ is short-hand for $\max(0, \cdot)$, b is a margin parameter, $\mathcal{Y}_t^C$ is the complement set of $\mathcal{Y}_t$, and we added normalizers to account for the case that a given x_t may have more than one positive label such that $|\mathcal{Y}_t| > 1$.

Crammer and Singer (2001) instead suggested taking the maximum hinge loss over all the negative classes:

$$L_{\max \text{ loss}}(\{\beta_g\}) = \sum_{t=1}^{n} \quad \frac{1}{|\mathcal{Y}_t|} \sum_{y^+ \in \mathcal{Y}_t} \quad \max_{y^- \in \mathcal{Y}_t^C} |b - f(x_t; \beta_{y^+}) + f(x_t; \beta_{y^-})|_+. \qquad (9)$$

1. See also the work of Herbrich et al. (2000) for a related pairwise loss function for ranking rather than classification.

This maximum hinge-loss is sometimes called *multiclass SVM*, and can be derived from a margin-bound (Mohri et al., 2012). Daniely et al. (2012) theoretically compared multiclass SVM with one-vs-all, one-vs-one, tree-based linear classifiers and error-correcting output code linear classifiers. They showed that the hypothesis class of multiclass SVM contains that of one-vs-all and tree-classifiers, which strictly contain the hypothesis class of one-vs-one classifiers. Thus the potential performance with multiclass SVM is larger. However they also showed that the approximation error of one-vs-one is smallest, with multiclass SVM next smallest.

Statnikov et al. (2005) compared eight multiclass classifiers including that of Weston and Watkins (1999) and Crammer and Singer (2001) on nine cancer classification problems with 3 to 26 classes and less than 400 samples per problem. On these small-scale data sets, they found the Crammer and Singer (2001) classifier was best (or tied) on 2/3 of the data sets, and the pairwise loss given in (8) performed almost as well.

Lee et al. (2004) prove in their Lemma 2 that previous approaches to multiclass SVMs are not guaranteed to be asymptotically consistent. For more on consistency of multiclass classification loss functions, see Rifkin and Klatau (2004), Tewari and Bartlett (2007), Zhang (2004), and Mroueh et al. (2012). Lee et al. (2004) proposed a multiclass loss function that is consistent. They force the class discriminants to sum to zero such that $\sum_g f(x;\beta_g) = 0$ for all x, and define the loss:

$$L_{\text{total loss}}(\{\beta_g\}) = \sum_{t=1}^{n} \sum_{y^- \in \mathcal{Y}_t^C} |f(x_t;\beta_{y^-}) - \frac{1}{G-1}|_+. \tag{10}$$

This loss function jointly trains the class discriminants so that the total sum of wrong class discriminants for each training sample is small. Minimizing this loss can be expressed as a constrained quadratic program. The experiments of Lee et al. (2004) on a few small data sets did not show much difference between the performance of (10) and (8).

5. Online Loss Functions for Training Large-scale Multiclass Classifiers

If there are a large number of training samples n, then computing the loss for each candidate set of classifier parameters becomes computationally prohibitive. The usual solution is to minimize the loss in an online fashion with stochastic gradient descent, but exactly how to sample the stochastic gradients becomes a key issue. Next, we review two stochastic gradient approaches that correspond to different loss functions: AUC sampling (Grangier and Bengio, 2008) and the WARP sampling used in the Wsabie classifier (Weston et al., 2011).

5.1 AUC Sampling

For a large number of training samples n, Grangier and Bengio (2008) proposed optimizing (8) by sequentially uniformly sampling from each of the three sums in (8):

1. draw one training sample x_t,
2. draw one correct class y^+ from $\mathcal{Y}_t$,

3. draw one incorrect class y^- from $\mathcal{Y}_t^C$,

4. compute the corresponding stochastic gradient of the loss in (8),

5. update the classifier parameters.

This sampling strategy, referred to as *area under the curve* (AUC) *sampling*, is inefficient because most randomly drawn incorrect class samples will have zero hinge loss and thus not produce an update to the classifier.

5.2 WARP Sampling

The *weighted approximately ranked pairwise* (WARP) sampling was introduced by Weston et al. (2011) to make the stochastic gradient sampling more efficient than AUC sampling, and was evolved from the weighted pairwise classification loss of Usunier et al. (2009). Unlike AUC sampling, WARP sampling focuses on sampling from negative classes that produce non-zero stochastic gradients for a given training example.

To explain WARP sampling, we first define the *WARP loss*:

$$L_{\text{WARP}}(\{\beta_g\}) = \sum_{t=1}^{n} \quad \frac{1}{|\mathcal{Y}_t|} \sum_{y^+ \in \mathcal{Y}_t} \quad \frac{1}{|\mathcal{V}_{t,y^+}|} \sum_{y^v \in \mathcal{V}_{t,y^+}} w(y^+) \quad |b - f(x_t; \beta_{y^+}) + f(x_t; \beta_{y^v})|_+, \tag{11}$$

where $w(y^+)$ is a weight on the correct label, and $\mathcal{V}_{t,y^+}$ is the set of *violating classes* defined:

$$\mathcal{V}_{t,y^+} = \{y^v \text{ s.t. } |b - f(x_t; \beta_{y^+}) + f(x_t; \beta_{y^v})|_+ > 0\}. \tag{12}$$

As in Usunier et al. (2009), Weston et al. (2011) suggest using a weight function $w(y^+)$ that is an increasing function of the number of violating classes. Because the number of violating classes defines the rank of the correct class y^+, they denote the number of violating classes for a training sample with training class y^+ as $r(y^+)$. They suggest using the truncated harmonic series for the weight function,

$$w(y^+) = \sum_{j=1}^{r(y^+)} \frac{1}{j}. \tag{13}$$

Weston et al. (2011) proposed *WARP sampling* which sequentially uniformly samples from each of the three sums in (11):

1. draw one training sample x_t,

2. draw one correct class y^+ from $\mathcal{Y}_t$,

3. draw one violating class y^v from $\mathcal{V}_{t,y^+}$ if one exists,

4. compute the corresponding stochastic gradient of the loss in (11),

5. update the classifier parameters.

To sample a violating class from $\mathcal{V}_{t,y^+}$, the negative classes in $\mathcal{Y}_t^C$ are uniformly randomly sampled until a class that satisfies the violation constraint (12) is found, or the number of allowed such trials (generally set to be G) is exhausted. The rank $r(y^+)$ needed to calculate the weight in (13) is estimated to be $(G-1)$ divided by the number of negative classes $y^- \in \mathcal{Y}_t$ that had to be tried before finding a violating class y^v from $\mathcal{V}_{t,y^+}$.

5.3 Some Notes Comparing AUC and WARP Loss

We note that WARP sampling is more likely than AUC sampling to update the parameters of a training sample's positive class y^+ if y^+ has few violating classes, that is if (x_t, y_t) is already highly-ranked by (1). Specifically, suppose a training pair (x_t, y_t) is randomly sampled, and suppose $H > 0$ of the G classes are violators such that their hinge-loss is non-zero with respect to (x, y^+). WARP sampling will draw random classes until it finds a violator and makes an update, but AUC will only make an update if the one random class it draws happens to be a violator, so only $H/(G-1)$ of the time. By definition, the higher-ranked the correct class y_t is for x_t, the smaller the number of violating classes H, and the less likely AUC sampling will update the classifier to learn from (x_t, y_t).

In this sense, WARP sampling is more focused on fixing class parameters that are almost right already, whereas AUC sampling is more focused on improving class parameters that are very wrong. At test time, classifiers choose only the highest-ranked class discriminant as a class label, and thus the fact that AUC sampling updates more often on lower-ranked classes is likely the key reason that WARP sampling performs so much better in practice (see the experiments of Weston et al. (2011) and the experiments in this paper). Even in ranking, it is usually only the top ranked classes that are of interest. However, the WARP weight $w(y^+)$ given in (13) partly counteracts this difference by assigning greater weights (equivalently a larger step-size to the stochastic gradient) if the correct class has many violators, as then its rank is lower. In this paper, one of the proposals we make is to use constant weights $w(y^+) = 1$, so the training is even more focused on improving classes that are already highly-ranked.

AUC sampling is also inefficient because so many of the random samples result in a zero gradient. In fact, we note that the probability that AUC sampling will update the classifier *decreases* if there are more classes. Specifically, suppose for a given training sample pair (x_t, y_t) there are H classes that violate it, and that there are G classes in total. Then the probability that AUC sampling updates the classifier for (x_t, y_t) is $H/(G-1)$, which linearly decreases as the number of classes G is increased.

5.4 Online Versions of Other Multiclass Losses

WARP sampling implements an online version of the pairwise loss given in (8) (Weston et al., 2011). One can also interpret the WARP loss sampling as an online approximation of the maximum hinge loss given in (9), where the maximum violating class is approximated by the sampled violating class. This interpretation does not call for a rank-based weighting $w(y^+)$, and in fact we found that setting $w(y^+) = 1$ improved accuracy by roughly 20% on a large-scale image annotation task (see Table 6). A better approximation of (9) would require

sampling multiple violating classes and then taking the class with the worst discriminant, we did not try this due to the expected time needed to find multiple violating classes. Further, we hypothesize that choosing the class with the largest violation as the violating class could actually perform poorly for practical highly multiclass problems like Imagenet because the worst discriminant may belong to a class that is a missing correct label, rather than an incorrect label.

An online version of the loss proposed by Lee et al. (2004) and given in (10) would be more challenging to implement because the G class discriminants are required to be normalized; we do not know of any such experiments.

5.5 The Wsabie Classifier

Weston et al. (2011) combined the WARP sampling with online learning of a supervised linear dimensionality reduction. They learn an embedding matrix $W \in \mathbb{R}^{m \times d}$ that maps a given d-dimensional feature vector x to an m-dimensional "embedded" vector $Wx \in \mathbb{R}^m$, where $m \leq d$, and then the G class-specific discriminants of dimension m are trained to separate classes in the embedding space. Weston et al. (2011) referred to this combination as the *Wsabie* classifier. This changes the WARP loss given in (11) to the non-convex Wsabie loss, defined:

$$L_{\text{Wsabie}}(W, \{\beta_g\}) = \sum_{t=1}^{n} \sum_{y^+ \in \mathcal{Y}_t} \sum_{y^v \in \mathcal{V}_{t,y^+}} w(y^+) \quad |b - f(Wx_t; \beta_{y^+}) + f(Wx_t; \beta_{y^v})|_+.$$

Adding the embedding matrix W changes the number of parameters from Gd to $Gm + md$. For a large number of classes G and a small embedding dimension m (the case of interest here) this reduces the overall parameters, and so the addition of the embedding matrix W acts as a regularizer, reduces memory, and reduces testing time.

6. Online Adaptation of the Empirical Loss to Reduce Impact of Highly Confusable Classes

In this section, we present a simple and memory-efficient online implementation of the empirical class-confusion loss we proposed in Section 3.3 that reduces the impact of highly confusable classes on the standard empirical loss. First, we describe the batch variant of this proposal and quantify its effect. Then in Section 6.2 we describe a sampling version. In Section 6.3, we propose a simple extension that experimentally increases the accuracy of the resulting classifier, without using additional memory. We show the proposed online strategy works well in practice in Section 9.

6.1 Reducing the Effect of Highly Confusable Classes By Ignoring Last Violators

We introduce the key idea of a *last violator class* with a simple example before a formal definition. Suppose during online training the hundredth training sample x_{100} has label $y_{100} =$ `tiger`, and that the last training sample we saw labelled `tiger` was x_5. And suppose `lion` was a violating class for that training sample pair $(x_5, \texttt{tiger})$, that is $|1 -$

$f(x_5; \theta_{\texttt{tiger}}) + f(x_5; \theta_{\texttt{lion}})|_+ > 0$. Then for sample $(x_{100}, \texttt{tiger})$ we call the class `lion` the *last violator class*.

Formally, we call class v_{t,y^+} a *last violator* for the training sample pair (x_t, y^+) if x_τ was the last training sample for which y^+ was the sampled positive class and v_{t,y^+} was a violator for (x_τ, y^+), that is, $v_{t,y^+} \in \mathcal{V}_{\tau,y^+}$. The *set of violators* $\mathcal{V}_{\tau,y^+}$ becomes the *set of last violators of* (x_t, y^+), which we denote $\tilde{\mathcal{V}}_{t,y^+}$.

In order to decrease the effect of highly confusable classes on training the classifier, we propose to ignore losses for any violator class that was also a last violator class. The reasoning is that if the last violator class and a current violator class are the same, it indicates that the class y_t and that violator class are consistently confused (for example `tiger` and `lion`). And if two classes are consistently confused, we would like to reduce their impact on the empirical loss, as discussed in Section 3.

For example, say sequential training samples that were labelled `cat` had the following violating classes:

`dog` and `pig`, `dog` and `pig`, none, `dog`, `dog`, none, `dog`, pig.

The proposed approach ignores any violator that was also a last violator:

`dog` and `pig`, ~~`dog`~~ and ~~`pig`~~, none, `dog`, ~~`dog`~~, none, `dog`, `pig`.

Mathematically, to ignore last violators we simply add an indicator function $\mathbb{I}$ to the loss function. For example, ignoring last violators with the WARP loss from (11) can be written:

$$L_{\text{proposed}}(\{\theta_g\}) = \sum_{t=1}^{n} \sum_{y^+ \in \mathcal{Y}_t} \sum_{y^v \in \mathcal{V}_{t,y^+}} w(y^+) \quad |b - f(x_t; \theta_{y^+}) + f(x_t; \theta_{y^v})|_+ \quad \mathbb{I}_{y^v \notin \tilde{\mathcal{V}}_{t,y^+}}. \tag{14}$$

That is, instead of forming one estimate of the correct empirical loss as we proposed in Section 3.3, here we approximate the correct empirical loss as the average of the series of Bernoulli random variables represented by the extra indicator in (14). In fact, the proposal to ignore last violators implements the proposed approximation (7): the probability that an error is not-counted is the probability that the violating class is confused with the training class:

Proposition 1: Suppose there are n samples labelled class g, and each such sample has probability p of being classified as class h. Then the expected number of losses summed in (11) is np, but the expected number of losses summed in (14) is $(n-1)p(1-p) + p$.

This reduction of the empirical error from np to $np(1-p)$ is the same as in the earlier proposal (6), where in Proposition 1 the probability $1-p$ is the same as the expectation in (6).

6.2 Ignoring Sampled Last Violators for Online Learning

Building on the WARP sampling proposed by Weston et al. (2011) and reviewed in Section 5.2, we propose an online sampling implementation of (14), where for each class y^+ we store one sampled last violator and only update the classifier if the current violator is not the same as the last violator. Specifically,

1. draw one training sample x_t,
2. draw one correct class y^+ from $\mathcal{Y}_t$,
3. if there is no last violator v_{t,y^+} or if v_{t,y^+} exists but is not a violator for (x_t, y^+), then draw and store one violating class y^v from $\mathcal{V}_{y^+}$ and
   (a) compute the corresponding stochastic gradient of the loss in (8)
   (b) update the parameters.

Table 1 re-visits the same example as earlier, and illustrates for eight sequential training examples whose training label was `cat` what the last violator class is, whether the last violator is a current violator (in which case the current error is ignored), or if not ignored, which of the current violators is randomly sampled for the classifier update.

Throughout the training, the state of the sampled last violator for any class y^+ can be viewed as a Markov chain. We illustrate this for the class y^+ and two possible violating classes g and h in Figure 3.

In the experiments to follow, we couple the proposed online empirical class-confusion loss sampling strategy with an embedding matrix as in the Wsabie algorithm for efficiency and regularization, and refer to this as Wsabie^{++}. A complete description of Wsabie^{++} is given in Table 3, including the adagrad stepsize updates described in the next section. The memory needed to implement this discounting is $O(G)$ because only one last violator class is stored for each of the G classes.

Set of `Cat` Violators	`Cat`'s LV	`Cat`'s LV Violates?	New Violator Sampled?
1: `dog` and `pig`	none	-	`dog`
2: `dog` and `pig`	`dog`	yes	ignored
3: `dog`	`dog`	yes	ignored
4: `dog` and `pig`	`dog`	yes	ignored
5: `pig`	`dog`	no	`pig`
6: no violators	`pig`	no	none
7: `dog`	none	-	`dog`
8: `dog`	`dog`	yes	ignored

Table 1: Example of ignoring sampled last violators for eight sequential samples (one per row) whose training label is `cat`.

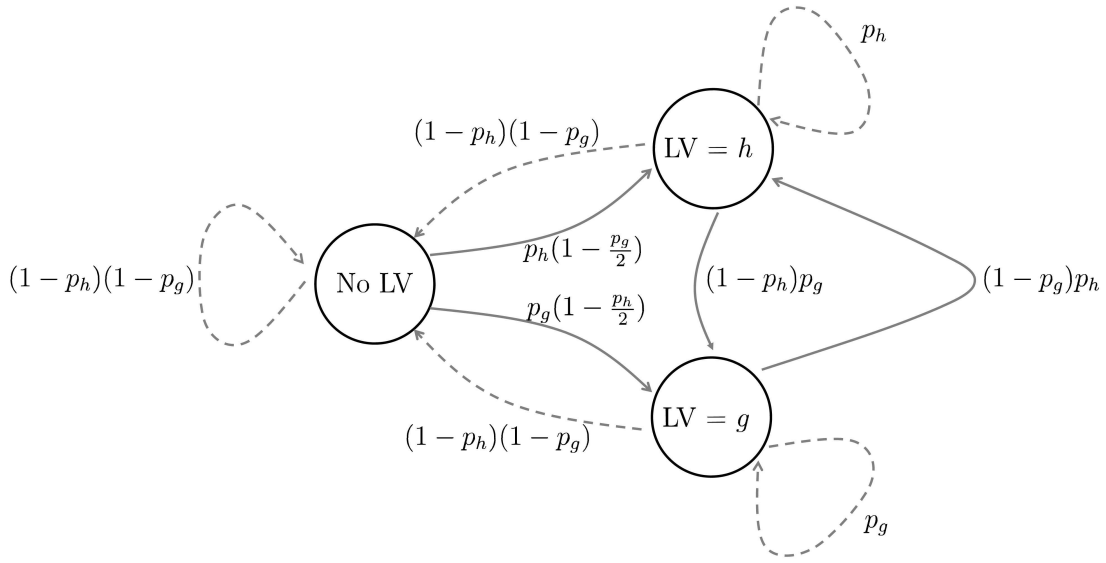


Figure 3: In the proposed sampling strategy for the online empirical class-confusion loss, the state of the last violator (LV) for a class y^+ can be interpreted as a Markov chain where a transition occurs for each training sample. The figure illustrates the case where there are just two possible violating classes, class g and class h, which violate samples of class y^+ with probability p_g and p_h respectively. Then the last violator for class y^+ is always in one of three possible states: no last violator, class h is the last violator, or class g is the last violator. Solid lines indicate a violation that is counted; dotted lines indicate a violation that is ignored. The three states have stationary distribution:

$$
\begin{aligned}
P(\text{No LV}) &= \frac{1}{Z}, \\
P(\text{LV } = g) &= \frac{1}{Z}\frac{p_g(2+p_h-p_gp_h)}{2(p_g-1)(p_gp_h-1)}, \\
P(\text{LV } = h) &= \frac{1}{Z}\frac{p_h(2+p_g-p_gp_h)}{2(p_h-1)(p_gp_h-1)},
\end{aligned}
$$

where Z is the normalizer that makes the stationary distribution sum to 1.

True Class		Last Violator Class
`tiger`	→	`lion`
`lion`	→	`cat`
`cat`	→	`kitten`
`kitten`	→	`panther`
`panther`	→	`cat`

Table 2: Example chain of five classes and their sampled last violator.

6.2.1 Curriculum Learning

We have primarily motivated ignoring last violators as a better approximation for the expected classification error. However, because this approach is online, it has a second practical effect of changing the distribution of classifier updates as the classifier improves during training. Consider the classifier at some fixed point during training. At that point, classes that are better separated from all other classes are less likely to have a last violator stored, and thus more likely to be trained on. This increases the chance that the classifier first learns to separate easy-to-separate classes. At each point in time, the classifier is less likely to be updated to separate classes it finds most confusable. Bengio et al. (2009) have argued that this kind of easy-to-hard learning is natural and useful, particularly when optimizing non-convex loss functions as is the case when one jointly learns an embedding matrix W for efficiency and regularization.

6.3 Extending the Discounting Loss to Multiple Last Violators

Table 2 shows an example of five classes and what their last violator class might be at some point in the online training. For example, Table 2 suggests that `tiger` and `lion` are highly confusable, and that `lion` and `cat` are highly confusable, and thus we suspect that `tiger` and `cat` may also be highly confusable. To further reduce the impact of these sets of highly confusable classes, we extend the above approach to ignoring a training sample if it is currently violated by its last violator class's last violator class, and so on. The longer the chain of last violators we choose to ignore, the more training samples are ignored, and the ignored training samples are preferentially those belonging to clusters of classes that are highly-confusable with each other.

Formally, let v^2_{t,y^+} denote the last violator of the last violator of y^+, that is $v^2_{t,y^+} = v_{t,v_{t,y^+}}$. For the example given in Table 2, if y^+ is `tiger`, then its last violator is $v_{t,y^+} =$ `lion`, and $v^2_{t,y^+} =$ `cat`. More generally, let v^Q_{t,y^+} be the Qth-order last violator, for example $v^3_{t,\texttt{tiger}} =$ `kitten`.

Let $\tilde{\mathcal{V}}^Q_{t,y^+}$ be the set of last violators up through order Q for positive class y^+ and the tth sample. We extend (14) to ignore this larger set of likely highly-confusable classes:

$$L_{\text{proposed-}Q}(\{\theta_g\}) = \sum_{t=1}^{n} \sum_{y^+ \in \mathcal{Y}_t} \sum_{y^v \in \mathcal{V}_{t,y^+}} |b - f(x_t; y^+) + f(x_t; y^v)|_+ \quad \mathbb{I}_{y^v \notin \tilde{\mathcal{V}}^Q_{t,y^+}}. \qquad (15)$$

To use (15) in an online setting, each time a training sample and its positive class are drawn, we check if any q-th order last violator v^q_{t,y^+} for any $q \leq Q$ is a current violator, and if so, we ignore that training sample and move directly to the next training sample without updating the classifier parameters.

Table 3 gives the complete proposed sampling and updating algorithm for Euclidean discriminant functions, including the adaptive adagrad step-size explained in Section 7 which follows. For Euclidean discriminant functions we did not find (experimentally) that we needed any constraints or additional regularizers on W or $\{\beta_g\}$, though if desired a regularization step can be added.

7. Adagrad For Learning Rate

Convergence speed of stochastic gradient methods is sensitive to the choice of stepsizes. Recently, Duchi et al. (2011) proposed a parameter-dependent learning rate for stochastic gradient methods. They proved that their approach has strong theoretical regret guarantees for convex objective functions, and experimentally it produced better results than comparable methods such as regularized dual averaging (Xiao, 2010) and the passive-aggressive method (Crammer et al., 2006). In our experiments, we applied adagrad both to the convex training of the one-vs-all SVMs and AUC sampling, as well as to the non-convex Wsabie^{++} training. Inspired by our preliminary results using adagrad for non-convex optimization, Dean et al. (2012) also tried adagrad for non-convex training of a deep belief network, and also found it produced substantial improvements in practice.

The main idea behind adagrad is that each parameter gets its own stepsize, and each time a parameter is updated its stepsize is decreased to be proportional to the running sum of the magnitude of all previous updates. For simplicity, we limit our description to the case where the parameters being optimized are unconstrained, which is how we implemented it. For memory and computational efficiency, Duchi et al. (2011) applying adagrad separately for each parameter (as opposed to modeling correlations between parameters).

We applied adagrad to adapt the stepsize for the G classifier discriminants $\{\beta_g\}$ and the $m \times d$ embedding matrix W. We found that we could save memory without affecting experimental performance by averaging the adagrad learning rate over the embedding dimensions such that we keep track of one scalar adagrad weight per class. That is, let $\Delta_{g,\tau}$ denote the stochastic gradient for β_g at time τ, then we update β_g as follows:

$$\beta_{g,t+1} = \beta_{g,t} - \eta \left(\sum_{\tau=0}^{t} \left(\frac{\Delta^T_{\tau,g} \Delta_{\tau,g}}{d} \right) \right)^{-1/2} \Delta_{g,\tau}. \tag{16}$$

Analogously, we found it experimentally effective and more memory efficient to keep track of one averaged scalar adagrad weight for each of the m rows of the embedding matrix W.

There are two main effects to using adagrad. First, suppose there are two classes that are updated equally often, then the class with larger stochastic gradients $\{\Delta_\tau\}$ will experience a faster-decaying learning rate. Second, and we believe the more relevant issue for our use, is that some classes are updated frequently, and some classes rarely. Suppose that all stochastic gradients $\{\Delta_\tau\}$ have the same magnitude, then the classes that are updated more rarely experience relatively larger updates. In our experiments the second effect was

Model:

Training Data Pairs: $(x_t, \mathcal{Y}_t)$ for $t = 1, 2, \ldots, n$
Embedded Euclidean Discriminant: $f(Wx; \beta_g) = -(\beta_g - Wx)^T(\beta_g - Wx)$

Hyperparameters:

Embedding Dimension: m
Stepsize: $\lambda \in \mathbb{R}_+$
Margin: $b \in \mathbb{R}_+$
Depth of last violator chain: $Q \in \mathbb{N}$

Initialize:

$W_{j,r}$ set randomly to -1 or 1 for $j = 1, 2, \ldots, m$, $r = 1, 2, \ldots, d$
$\beta_g = \mathbf{0}$ for all $g = 1, 2, \ldots, G$
$\alpha_g = \mathbf{0}$ for all $g = 1, 2, \ldots, G$
$\alpha_{W_j} = \mathbf{0}$ for all $j = 1, 2, \ldots, m$
v_{y^+} = empty set for all y^+

While Not Converged:

Sample x_t uniformly from $\{x_1, \ldots, x_n\}$.
Sample y^+ uniformly from $\mathcal{Y}_t$.
If $|b - f(Wx_t; \beta_{y^+}) + f(Wx_t; \beta_{v^q_{y^+}})|_+ > 0$ for any $q = 1, 2, \ldots, Q$, **continue**.
Set foundViolator = false.
For count = 1 to G:
Sample y^- uniformly from $\mathcal{Y}_t^C$.
If $|b - f(Wx_t; \beta_{y^+}) + f(Wx_t; \beta_{y^-})|_+ > 0$,
set foundViolator = true and **break**.
If foundViolator = false, set v_{y^+} to the empty set and **continue**.
Set $v_{y^+} = y^-$.
Compute the stochastic gradients:

$$\Delta_{y^+} = 2(\beta_{y^+} - Wx_t)$$
$$\Delta_{y^-} = -2(\beta_{y^-} - Wx_t)$$
$$\Delta_W = (\beta_{y^-} - \beta_{y^+})(Wx_t)^T.$$

Update the adagrad parameters:

$$\alpha_{y^+} = \alpha_{y^+} + \tfrac{1}{d}\Delta^T_{y^+}\Delta_{y^+}$$
$$\alpha_{y^-} = \alpha_{y^-} + \tfrac{1}{d}\Delta^T_{y^-}\Delta_{y^-}$$
$$\alpha_{W_j} = \alpha_{W_j} + \tfrac{1}{d}\Delta^T_{W_j}\Delta_{W_j} \text{ for } j = 1, 2, \ldots, m.$$

Update the classifier parameters:

$$\beta_{y^+} = \beta_{y^+} - \frac{\lambda}{\sqrt{\alpha_{y^+}}}\Delta_{y^+}$$
$$\beta_{y^-} = \beta_{y^-} - \frac{\lambda}{\sqrt{\alpha_{y^-}}}\Delta_{y^-}$$
$$W_j = W_j - \frac{\lambda}{\sqrt{\alpha_{W_j}}}\Delta_{W_j} \text{ for } j = 1, 2, \ldots, m.$$

Table 3: Wsabie^{++} training (for Euclidean discriminants).

predominant, which we tested by setting the learning rate for each parameter proportional to the inverse square root of the number of times that parameter has been updated. This "counting adagrad" produced results that were not statistically different using (16). (The experimental results in this paper are reported using adagrad proper as per (16).)

We use α to refer to the running sum of gradient magnitudes in the complete Wsabie^{++} algorithm description given in Table 3.

8. Experiments

We first detail the data sets used. Then in Section 8.2 we describe the features. In Section 8.3 we describe the different classifiers compared and how the parameters and hyperparameters were set.

8.1 Data Sets

Experiments were run with four data sets, as summarized in Table 4 and detailed below.

	16k ImageNet	22k ImageNet	21k Web Data	97k Web Data
Number of Classes	15,589	21,841	21,171	96,812
Number of Samples	9 million	14 million	9 million	40 million
Number of Features	1024	479	1024	1024

Table 4: Data sets.

8.1.1 ImageNet Data Sets

ImageNet (Deng et al., 2009) is a large image data set organized according to WordNet (Fellbaum, 1998). Concepts in WordNet, described by multiple words or word phrases, are hierarchically organized. ImageNet is a growing image data set that attaches one of these concepts to each image using a quality-controlled human-verified labeling process.

We used the spring 2010 and fall 2011 releases of the Imagenet data set. The spring 2010 version has around 9M images and 15,589 classes (16k ImageNet). The *fall 2011* version has about 14M images and 21,841 classes (22k ImageNet). For both data sets, we separated out 10% of the examples for validation, 10% for test, and the remaining 80% was used for training.

8.1.2 Web Data Sets

We also had access to a large proprietary set of images taken from the web, together with a noisy annotation based on anonymized users' click information. We created two data sets from this corpus that we refer to as 21k Web Data and 97k Web Data. The 21k Web Data contains about 9M images, divided into 20% for validation, 20% for test, and 60% for train, and the images are labelled with 21,171 distinct classes. The 97k Web Data contains about 40M images, divided into 10% for validation, 10% for test, and 80% for train, and the images are labelled with 96,812 distinct classes.

There are five main differences between the Web Data and ImageNet. First, the types of labels found in Imagenet are more *academic*, following the strict structure of WordNet.

In contrast, the Web Data labels are taken from a set of popular queries that were the input to a general-purpose image search engine, so it includes people, brands, products, and abstract concepts. Second, the number of images per label in Imagenet is artificially forced to be somewhat uniform, while the Web Data distribution of number of images per label is generated by popularity with users, and is thus more exponentially distributed. Third, because of the popular origins of the Web data sets, classes may be translations of each other, plural vs. singular concepts, or synonyms (for examples, see Table 7). Thus we expect more highly-confusable classes for the Web Data than ImageNet. A fourth key difference is Imagenet disambiguates polysemous labels whereas Web Data does not, for example, an image labeled `palm` might look like the palm of a hand or like a palm tree. The fifth difference is that there may be multiple given positive labels for some of the Web samples, for example, the same image might be labelled `mountain`, `mountains`, `Himalaya`, and `India`.

Lastly, classes may be at different and overlapping precision levels, for example the class `cake` and the class `wedding cake`.

8.2 Features

We do not focus on feature extraction in this work, although features certainly can have a big impact on performance. For example, Sanchez and Perronnin (2011) recently achieved a 160% gain in accuracy on the 10k ImageNet datatset by changing the features but not the classification method.

In this paper we use features, similar to those used in Weston et al. (2011). We first combined multiple spatial (Grauman and Darrell, 2007) and multiscale color and texton histograms (Leung and Malik, 1999) for a total of about 5×10^5 dimensions. The descriptors are somewhat sparse, with about 50000 non-zero weights per image. Some of the constituent histograms are normalized and some are not. We then perform kernel PCA (Schoelkopf et al., 1999) on the combined feature representation using the intersection kernel (Barla et al., 2003) to produce a 1024-dimensional or 479-dimensional input vector per image (see Tab. 4), which is then used as the feature vectors for the classifiers.

8.3 Classifiers Compared and Hyperparameters

We experimentally compared the following linear classifiers: nearest means, one-vs-all SVMs, AUC, Wsabie, and the proposed Wsabie^{++} classifiers. Table 5 compares these methods as they were implemented for the experiments.

The nearest means classifier is the most efficient to train of the compared methods as it only passes over the training samples once and computes the mean of the training feature vectors for each class (and there are no hyperparameters).

Like the nearest means classifier, we implemented Wsabie^{++} with Euclidean discriminants (as detailed in Table 3) and as such it can be considered a *discriminative* nearest means classifier. Testing with Euclidean discriminants can easily be made faster by applying exact or approximate fast k-NN methods, where the class prototypes $\{\beta_g\}$ play the role of the neighbors. Further, Euclidean discriminants lend themselves more naturally to visualization than the inner product, as each class is represented by a prototype.

	One-vs-all SVMs 1+:1-	One-vs-all SVMs	AUC Sampling	Wsabie	Wsabie^{++}
Loss Function	separable	separable	pairwise	pairwise	pairwise
Regularization	ℓ_2 constraint	ℓ_2 constraint	ℓ_2 constraint	embedding matrix ℓ_2 constraint	embedding matrix
# Negative Examples Per Positive Example in Training	1	chosen using validation set	1	as many as needed to find a violator	as many as needed to find a violator
Learning rate adaptation	adagrad	adagrad	adagrad	based on estimated rank	adagrad
Empirical Class Confusion Loss	no	no	no	no	yes
Discriminant Function $f_g(x)$	$\beta_g^T x$	$\beta_g^T x$	$\beta_g^T x$	$\beta_g^T W x$	$\|\beta_g - Wx\|_2^2$

Table 5: Comparison of the different stochastic gradient methods implemented in the experiments

One-vs-all linear SVMs are the most popular choice for large-scale classifiers due to studies showing their good performance, their parallelizable training, relatively small memory, and fast test-time (Rifkin and Klatau, 2004; Deng et al., 2010; Sanchez and Perronnin, 2011; Perronnin et al., 2012; Lin et al., 2011). Perronnin et al. (2012) highlights the importance of getting the right balance of negative to positive examples used to train the one-vs-all linear SVMs. As in their paper, we cross-validate the expected number of negative examples per positive example; the allowable choices were powers of 2. In contrast, earlier published results by Weston et al. (2011) that compared Wsabie to one-vs-all SVMs used one negative example per positive example, analogous to the AUC classifier. We included this comparison, which we labelled One-vs-all SVMs 1+:1- in the tables.

Both Wsabie and Wsabie^{++} jointly train an embedding matrix W as described in Section 3.5. The embedding dimension d was chosen on the validation set from the choices $d = \{32, 64, 96, 128, 192, 256, 384, 512, 768, 1024\}$ embedding dimensions. In addition, we created *ensemble* Wsabie and Wsabie^{++} classifiers by concatenating $\lfloor \frac{m}{d} \rfloor$ such d-dimensional models to produce a classifier with a total of m parameters to compare classifiers that require the same memory and test-time.

All hyperparameters were chosen based on the accuracy on a held-out validation set. Step-size, margin, and regularization constant hyperparameters were varied by powers of ten. The order Q of the last violators was varied by powers of 2. Chosen hyperparameters are recorded in Table 9. Both the pairwise loss and Wsabie classifier are implemented with standard ℓ_2 constraints on the class discriminants (and for Wsabie, on the rows of the embedding matrix). We did not use any regularization constraints for Wsabie^{++}.

We initialized the Wsabie parameters and SVM parameters uniformly randomly within the constraint set. We initialize the proposed training by setting all β_g to the origin, and all components of the embedding matrix are equally likely to be -1 or 1. Experiments with different initialization schemes for these different classifiers showed that different (reasonable) initializations gave very similar results.

With the exception of nearest means, all classifiers were trained online with stochastic gradients. We also used adagrad for the convex optimizations of both one-vs-all SVMs and the AUC sampling, which increased the speed of convergence.

Recently, Perronnin et al. (2012) showed good results with one-vs-all SVM classifiers and the WARP loss where they also cross-validated an early-stopping criterion. Adagrad reduces step sizes over time, and this removed the need to worry about early stopping. In fact, we did not see any obvious overfitting with any of the classifier training (validation set and test set errors were statistically similar). Each algorithm was allowed to train on up to 100 loops through the entire training set or until the validation set performance had not changed in 24 hours. Even those runs that ran the entire 100 loops appeared to have essentially converged. Implemented in C++ without parallelization, all algorithms (except nearest means) took around one week to train the 16k Imagenet data set, around two weeks to train the 21k and 22k data sets, and around one month to train the 97k data set. Also in all cases roughly 80% of the validation accuracy was achieved in roughly the first 20% of the training time.

Because stochastic gradient descent uses random sampling of the training samples, multiple runs will produce slightly different results. To address this randomness, we ran five runs of each classifier for each set of candidate parameters, and reported the test accuracy

and parameters for the run that had the best accuracy on the validation set. For one-vs-all SVMs with its convex objective, the five runs usually differed by .1% (absolute), whereas optimizing the nonconvex objectives of Wsabie and Wsabie^{++} produced much greater randomness within five runs, as much as .5% (absolute). Cross-validating substantially more runs of the training would probably produce classifiers with slightly better accuracy, but cross-validating between too many runs could just lead to overfitting. We did not explore this issue carefully.

8.4 Metrics

Each classifier outputs the class it considers the one best prediction for a given test sample. We measure the accuracy of these predictions averaged over all the samples in the test set. For some data sets, such as the Web data sets, samples may have more than one correct class, and are counted as correct if the classifier picks any one of the correct classes. Note that some results published for Imagenet use a slightly different metric: classification accuracy averaged over the G classes (Deng et al., 2010).

9. Results

We first give some illustrative results showing the effect of the three proposed differences between Wsabie^{++} and Wsabie. Then we compare Wsabie^{++} to four different efficient classifiers on four large-scale data sets.

9.1 Comparison of Different Aspects of Wsabie^{++}

Wsabie^{++} as detailed in Table 2 differs from the Wsabie classifier (Weston et al., 2011) in the following respects:

1. ignores last violators
2. weights all stochastic gradients equally, that is, $w(r(y^+)) = 1$ in (11),
3. uses adagrad to adapt the learning rates,
4. uses Euclidean discriminants and no parameter regularization, rather than linear discriminants and ℓ_2 parameter regularization as done in Wsabie.

Table 6 shows how each of these first three differences increases the classification accuracy on the 21k Web data set. For the results in this table, the embedding dimension was fixed at $d = 100$, but all other classifier parameters were chosen to maximize accuracy on the validation set.

In addition, for simplicity we used Euclidean discriminants rather than linear discriminants: with Euclidean discriminants we found we did not need any additional parameter regularization, and it is simpler to apply adagrad when the parameters are unconstrained.

The results show that either adagrad or 10 last violators alone improves accuracy by 35%. Weighting all updates equally ($w(r(y^+)) = 1$) alone also improves accuracy by 10%. In combination, these changes complement each other, almost doubling the accuracy from 3.7% to 7.1%.

Classifier	Test Accuracy
Wsabie (Weston et al., 2011)	3.7%
Wsabie + 10 last violators	5.0%
Wsabie + adagrad	5.0%
Wsabie + $w(r(y^+)) = 1$ in (11)	4.1%
Wsabie + adagrad + 10 last violators	5.9%
Wsabie + adagrad + $w(r(y^+)) = 1$ in (11)	6.0%
Wsabie + adagrad + $w(r(y^+)) = 1$ in (11) + 1 last violator	6.3%
Wsabie + adagrad + $w(r(y^+)) = 1$ in (11) + 10 last violators	7.1%
Wsabie + adagrad + $w(r(y^+)) = 1$ in (11) + 100 last violators	6.8%

Table 6: Effect of the proposed differences compared to Wsabie for a $d = 100$ dimensional embedding space on 21k Web Data.

Table 7 gives examples of the classes corresponding to neighboring $\{\beta_g\}$ in the embedded feature space after the Wsabie^{++} training.

Class	1-NN	2-NN	3-NN	4-NN	5-NN
`poodle`	`caniche`	`pudel`	`labrador`	`puppies`	`cocker spaniel`
`dolphin`	`dauphin`	`delfin`	`dolfinjnen`	`delfiner`	`dolphins`
`San Diego`	`Puerto Madero`	`Sydney`	`Vancouver`	`Kanada`	`Tripoli`
`mountain`	`mountains`	`montagne`	`Everest`	`Alaska`	`Himalaya`
`router`	`modem`	`switch`	`server`	`lan`	`network`
`calligraphy`	`fonts`	`Islamic calligraphy`	`borders`	`quotes`	`network`

Table 7: For each of the classes on the left, the table shows the five nearest (in terms of Euclidean distance) class prototypes $\{\beta_g\}$ in the proposed discriminatively trained embedded feature space for 21k Web Data set. Because these classes originated as web queries, some class names are translations of each other, for example `dolphin` and `dauphin` (French for dolphin). While these may seem like exact synonyms, in fact different language communities often have slightly different visual notions of the same concept. Similarly, the classes `Obama` and `President Obama` are expected to be largely overlapping, but their class distributions differ in the formality and context of the images.

Lastly, we illustrate how the Wsabie^{++} test accuracy depends on the number of embedding dimensions. These results are for the 21k Web Data set, with the step-size and margin

parameters chosen using the validation set, and 10 last violators:

Number of Embedding Dimensions:	128	192	256	384	512	768	1024
Wsabie^{++} Test Accuracy:	7.4%	7.7%	8.3%	7.9%	7.1%	6.7%	6.5%

9.2 Comparison of Different Classifiers

Table 8 compares the accuracy of the different classifiers, where all hyperparameters were cross-validated. The validated parameter choices are reported in Table 9.[2]

Wsabie^{++} was consistently most accurate, followed by the one-vs-all SVMs with the average number of negative samples per positive sample chosen by validation. The row labelled Wsabie^{++} was $2-26\%$ more accurate and $2-4\times$ more efficient ($2-4\times$ smaller model size) than the one-vs-all SVMs because the validated embedding dimension was 192 or 256 dimensions, down from the original 479 or 1024 features.

Like Perronnin et al. (2012), our experiments showed that choosing the hyperparameter of how many negative samples per positive sample for the one-vs-all SVMs made an impressive difference to its performance. The one-vs-all SVMs 1+:1-, which used a fixed ratio of one negative sample per positive sample was $2-4$ times worse!

The row labelled Wsabie^{++} Ensemble is a concatenation of 2-4 Wsabie^{++} classifiers trained on different random samplings so that the total number of parameters is roughly the same as the SVM (the embedding matrices W add slightly to the total storage and efficiency calculations). With efficiency thus roughly controlled, the accuracy gain increased slightly to $3-28\%$.

The least improvement was seen on the 21k Web Data set. Our best hypothesis as to why that is that the classifiers are already close to the best performance possible with linear separators, and so there is little headroom for improvement. Some support for this hypothesis is the tiny gain the ensemble of multiple Wsabie^{++} classifiers versus the Wsabie^{++}.

One surprise was that Wsabie^{++} performed almost as well on the 97k Web Data as on the 21k Web Data set even though there were four times as many classes. We have two main hypotheses of why this happened. First, there were more training samples in the 97k Web Data for the classes that were already present in the 21k Web Data. Second, the added classes had fewer samples but were often quite specific, and the samples from a specific class can be easier to distinguish than samples from a more generic class. For example, samples from the more specific class of `diamond earrings` are easier to distinguish than samples from the more generic class `jewelry`. Likewise, samples from the class `beer foam` are easier to correctly classify than samples from the class `beer`.

10. Discussion, Hypotheses and Key Issues

This paper focused on how highly confusable classes can distort the empirical loss used in discriminative training of multi-class classifiers. We proposed a lightweight online approach to reduce the effect of highly-confusable classes on the empirical loss, and showed that it can

2. Parameters for step-size and margin were not independent, with larger margins working better with larger step-sizes. We hypothesize that one of these parameters could be fixed and only the other cross-validated. We did not see any overfitting: scores on the validation set were statistically similar to scores on the test sets for all the compared methods.

	16k ImageNet	22k ImageNet	21k Web Data	97k Web Data
Nearest Means	4.4%	2.7%	2.6%	2.3%
One-vs-all SVMs 1+:1-	4.1%	3.5%	2.1%	1.6%
One-vs-all SVMs	9.4%	8.2%	8.3%	6.8%
AUC Sampling	4.7%	5.1%	2.8%	3.1%
Wsabie	6.5%	6.6%	4.5%	2.8%
Wsabie Ensemble	8.1%	7.0%	6.0%	3.4%
Wsabie^{++}	11.2%	10.3%	8.5%	8.2%
Wsabie^{++} Ensemble	**11.9%**	**10.5%**	**8.6%**	**8.3%**

Table 8: Image classification test accuracy

substantially increase performance in practice. Experimentally, we also showed that using adagrad to evolve the learning rates in the stochastic gradient descent is effective despite the nonconvexity of the loss (due to the joint learning of the linear dimensionality reduction and linear classifiers).

We argued that when there are many classes it is suboptimal to measure performance by simply counting errors, because this overemphasizes the noise of highly-confusable classes. Yet our test error is measured in the standard way: by counting how many samples were classified incorrectly. A better approach to measuring test error would be subjective judgements of error. In fact, we have verified that for the image classification problems considered in the experiments, subjects are less critical about confusions of classes they consider more confusable (for example confusing `dolphin` and `porpoise`), but very critical of confusions between classes they do not consider confusable (for example `dolphin` and `Statue of Liberty`). Thus, suppose you had two candidate classifiers, each of which made 100 errors, but one made all 100 errors between `dolphin` and `porpoise` and the other made 100 more random errors. Standard test error of summing the errors would consider these classifiers equal, but users would generally prefer the first classifier. Weston et al. (2011) provide one approach to addressing this issue with a *sibling precision* measure.

A related issue is that it is known that the experimental data sets used are not tagged with the complete set of correct class labels for each image. For example, in the 21k Web Data, an image of a red heart might be labelled `love` and `red`, but not happen to be labelled `heart`, even though that would be considered a correct label in a subjective evaluation. We hypothesize that the proposed approach of probabilistically ignoring samples with consistent confusions helps reduce the impact of such missing positive labels.

We built on WARP, which finds a violating class per each training sample if one exists. This strategy works much better in practice than the AUC sampling that samples one positive class and one negative class per training sample. We believe this is because WARP sampling focuses on improving the parameters of classes that are already quite good, rather than focusing on parameters for classes that are very confused. Analogously, and in agreement with results by Perronnin et al. (2012) on very different features, we saw that sampling one negative sample per positive sample for one-vs-all SVMs performed very poorly compared to sampling a validated number of negatives per positive. Inspired by these performance differences due to the choice of negative:positive ratios, we also considered validating a hyperparameter for Wsabie^{++} that would determine how many negative classes

	Stepsize	Margin	Embedding dimension	Balance β	# LVs
One-vs-all SVM 1+:1-					
16k ImageNet	.01	.1			
21k ImageNet	.01	.1			
21k Web Data	.1	1			
97k Web Data	.01	.1			
One-vs-all SVM					
16k ImageNet	.01	.1		64	
21k ImageNet	.01	.1		64	
21k Web Data	.1	1		64	
97k Web Data	.01	.1		128	
AUC Sampling					
16k ImageNet	.01	.1			
21k ImageNet	.01	.1			
21k Web Data	.001	.01			
97k Web Data	.001	.01			
Wsabie					
16k ImageNet	.01	.1	128		
21k ImageNet	.001	.1	128		
21k Web Data	.001	.1	256		
97k Web Data	.0001	.1	256		
Wsabie^{++}:					
16k ImageNet	10	10,000	192		8
21k ImageNet	10	10,000	192		8
21k Web Data	10	10,000	256		8
97k Web Data	10	10,000	256		32

Table 9: Classifier parameters chosen using validation set

to consider for each positive class, rather than the WARP sampling which draws negative classes until it finds a violator. Preliminary results showed that the validation set chose the largest parameter choice which was almost the same as the number of classes. Thus that approach required another hyperparameter but seemed to be doing exactly what WARP sampling does and appeared to offer no accuracy improvement over WARP sampling.

This research focused on accurate and efficient classification, and not on the issue of training time. With the exception of nearest means, the methods compared were implemented with stochastic gradient descent for efficient online training to deal with the large number of training samples n. As implemented, the methods took roughly equally long to train. However, parallel training of the G one-vs-all SVMs would have been roughly G times as fast. While not as naturally parallelizable, we have had some success in parallelizing the

WARP sampling strategy across multiple cores and multiple machines, but the details are outside the scope of this work.

Our experiments were some of the largest image labeling experiments ever performed, and were carefully implemented and executed. But our experiments were narrow and limited in the sense that only image labeling problems were considered, and that the feature derivations were similar and all dense. The presented theory and motivation was not limited however, and we hypothesize that similar results would hold up for other applications, different features, and sparser features.

We focused in this paper only on classifiers that use linear (or Euclidean) discriminant functions because they are popular for large-scale highly multiclass problems due to their efficiency and reasonably good performance. However, for a given feature set, the best performance on large data sets such as ImageNet may well be achieved with exact k-NN (Deng et al., 2010; Weston et al., 2013b) or a more sophisticated lazy classifier (Garcia et al., 2010), or with a deep network (Krizhevsky et al., 2012; Dean et al., 2012). However, for many real-life large-scale problems these methods may require infeasible memory and test-time, and so linear methods are of interest at least for their efficiency, and may be used to filter candidates to a smaller set for secondary evaluation by a more flexible classifier. In addition, the last layer of a neural network is often a linear or other high-model-bias classifier, and the proposed approaches may thus be useful in training a deep network as well.

Label trees and label partitioning can be even more efficient at inference (Bengio et al., 2010; Deng et al., 2011; Weston et al., 2013a). We did not take advantage or impose a hierarchical structure on the classes, which can be a fruitful approach to efficiently implementing highly multiclass classification. Other research in large-scale classification takes advantage of the natural hierarchy of classes in real-world classification problems such as labeling images (Deng et al., 2010; Griffin and Perona, 2008; Nister and Stewenius, 2006). For example, in Web Data one class is `wedding cake`, which could fit into the broader class of `cake`, and the even broader class of `food`. One problem with leveraging such hierarchies may be that they are not strict trees; `wedding cake` also falls under the broader class of `wedding`, or there may be no natural hierarchy. Some of the theory and strategy of this paper should be complementary to such hierarchical approaches.

Acknowledgments

We thank Mouhamadou Cisse, Gal Chechik, Andrew Cotter, Koby Crammer, John Duchi, Bela Frigyik, and Yoram Singer for helpful discussions.

Appendix: Proof of Proposition 1

Let Z_t be a Bernoulli random variable with parameter p that models the event that the tth sample of class y^+ is classified as class h. Then for n trials the expected number of times a class y^+ sample is classified as class h is $E[\sum_t Z_t] = \sum_t E[Z_t] = nE[Z_t] = np$ the Z_t are independent and identically distributed.

Let O_t be another Bernoulli random variable such that $O_t = 1$ if the tth sample of class y^+ is counted in the loss given by (14), and $O_t = 0$ otherwise. Note that $O_1 = Z_1$, and for

$t > 1$, $O_t = 1$ iff $Z_t = 1$ and $Z_{t-1} = 0$. Thus,

$$E[O_t] = E[Z_t = 1 \text{ and } Z_{t-1} = 0] = P(Z_t = 1, Z_{t-1} = 0) = P(Z_t = 1)P(Z_t = 0) = p(1-p)$$

by the independence of the Bernoulli random variables Z_t and Z_{t-1}. Then the expected number of confusions counted by (14) is $E[\sum_t O_t] = \sum_t E[O_t]$ by linearity, which can be expanded: $E[O_1] + \sum_{t=2}^{n} E[O_t] = p + (n-1)p(1-p)$.

References

E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal Machine Learning Research*, 1:113–141, 2000.

A. Barla, F. Odone, and A. Verri. Histogram intersection kernel for image classification. *Intl. Conf. Image Processing (ICIP)*, 3:513–516, 2003.

S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *NIPS*, 2010.

Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

Y. Bengio, J. Laouradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009.

E. J. Bredensteiner and K. P. Bennet. Multicategory classification by support vector machines. *Computational Optimization and Applications*, 12:53–79, 1999.

L. D. Brown, T. T. Cai, and A. DasGupta. Interval estimation for a binomial proportion. *Statistical Science*, 16(2):101–117, 2001.

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

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2):201–233, 2002.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive aggressive algorithms. *Journal Machine Learning Research*, 7:551–585, 2006.

A. Daniely, S. Sabato, and S. Shalev-Shwartz. Multiclass learning approaches: A theoretical comparison with implications. In *NIPS*, 2012.

J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senoir, P. Tucker, K. Yang, and A. Ng. Large scale distributed deep networks. In *NIPS*, 2012.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conf. Computer Vision Pattern Recognition (CVPR)*, 2009.

J. Deng, A. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *European Conf. Computer Vision (ECCV)*, 2010.

J. Deng, S. Satheesh, A. C. Berg, and L. Fei-Fei. Fast and balanced: Efficient label tree learning for large scale object recognition. In *NIPS*, 2011.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal Artificial Intelligence Research*, 2:264–286, 1995.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal Machine Learning Research*, 12:2121–2159, 2011.

D. Eck. Personal communication from Google music expert Douglas Eck. 2013.

C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

E. K. Garcia, S. Feldman, M. R. Gupta, and S. Srivastava. Completely lazy learning. *IEEE Trans. Knowledge and Data Engineering*, 22(9):1274–1285, 2010.

D. Grangier and S. Bengio. A discriminative kernel-based model to rank images from text queries. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30:1371–1384, 2008.

K. Grauman and T. Darrell. The pyramid match kernel: Efficient learning with sets of features. *Journal Machine Learning Research*, 8:725–760, April 2007.

G. Griffin and P. Perona. Learning and using taxonomies for fast visual categorization. In *CVPR*, 2008.

Y. Guermeur. Combining discriminant models with new multi-class SVMs. *Pattern Analysis and Applications*, 5:168–179, 2002.

P. Hall and J. S. Marron. Geometric representation of high dimension, low sample size data. *J. R. Statst. Soc. B*, 67(3):427–444, 2005.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, New York, 2001.

R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, 2000.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines. *Journal American Statistical Association*, 99(465):67–81, 2004.

T. Leung and J. Malik. Recognizing surface using three-dimensional textons. *Intl. Conf. Computer Vision (ICCV)*, 1999.

Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang. Large-scale image classification: Fast feature extraction and SVM training. In *CVPR*, 2011.

M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT Press, Cambridge, MA, 2012.

Y. Mroueh, T. Poggio, L. Rosasco, and J-J E. Slotine. Multiclass learning with simplex coding. In *NIPS*, 2012.

D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

A. Paton, N. Brummitt, R. Govaerts, K. Harman, S. Hinchcliffe, B. Allkin, and E. Lughadha. Target 1 of the global strategy for plant conservation: a working list of all known plant speciesprogress and prospects. *Taxon*, 57:602–611, 2008.

F. Perronnin, Z. Akata, Z. Harchaoui, and C. Schmid. Towards good practice in large-scale learning for image classification. In *CVPR*, 2012.

R. Rifkin and A. Klatau. In defense of one-vs-all classification. *Journal Machine Learning Research*, 5:101–141, 2004.

J. Sanchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *CVPR*, 2011.

B. Schoelkopf, A. J. Smola, and K. R. Müller. Kernel principal component analysis. *Advances in kernel methods: support vector learning*, pages 327–352, 1999.

O. Shamir and O. Dekel. Multiclass-multilabel classication with more classes than examples. In *Proc. AISTATS*, 2010.

A. Statnikov, C. F. Aliferis, I. Tsamardinos, D. Hardin, and S. Levy. A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics*, 21(5):631–643, 2005.

A. Tewari and P. L. Bartlett. On the consistency of multiclass classification methods. *Journal Machine Learning Research*, 2007.

N. Usunier, D. Buffoni, and P. Gallinari. Ranking with ordered weighted pairwise classification. In *ICML*, 2009.

V. Vapnik. *Statistical Learning Theory*. 1998.

J. Weston and C. Watkins. Multi-class support vector machines. *Technical Report CSD-TR-98-04 Dept. Computer Science, Royal Holloway, University London*, 1998.

J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proc. European Symposium on Artificial Neural Networks*, 1999.

J. Weston, S. Bengio, and N. Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *Intl. Joint Conf. Artificial Intelligence, (IJCAI)*, pages 2764–2770, 2011.

J. Weston, A. Makadia, and H. Yee. Label partitioning for sublinear ranking. In *ICML*, 2013a.

J. Weston, R. Weiss, and H. Yee. Affinity weighted embedding. In *Proc. International Conf. Learning Representations*, 2013b.

L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Microsoft Research Technical Report MSR-TR-2010-23*, 2010.

T. Zhang. Statistical analysis of some multi-category large margin classification methods. *Journal Machine Learning Research*, 5:1225–1251, 2004.

Journal of Machine Learning Research 15 (2014) 1493-1522 Submitted 6/13; Revised 2/14; Published 4/14

Locally Adaptive Factor Processes for Multivariate Time Series

Daniele Durante DURANTE@STAT.UNIPD.IT
Bruno Scarpa SCARPA@STAT.UNIPD.IT
Department of Statistical Sciences
University of Padua
Padua 35121, Italy

David B. Dunson DUNSON@STAT.DUKE.EDU
Department of Statistical Science
Duke University
Durham, NC 27708-0251, USA

Editor: Robert E. McCulloch

Abstract

In modeling multivariate time series, it is important to allow time-varying smoothness in the mean and covariance process. In particular, there may be certain time intervals exhibiting rapid changes and others in which changes are slow. If such time-varying smoothness is not accounted for, one can obtain misleading inferences and predictions, with over-smoothing across erratic time intervals and under-smoothing across times exhibiting slow variation. This can lead to mis-calibration of predictive intervals, which can be substantially too narrow or wide depending on the time. We propose a locally adaptive factor process for characterizing multivariate mean-covariance changes in continuous time, allowing locally varying smoothness in both the mean and covariance matrix. This process is constructed utilizing latent dictionary functions evolving in time through nested Gaussian processes and linearly related to the observed data with a sparse mapping. Using a differential equation representation, we bypass usual computational bottlenecks in obtaining MCMC and online algorithms for approximate Bayesian inference. The performance is assessed in simulations and illustrated in a financial application.

Keywords: Bayesian nonparametrics, locally varying smoothness, multivariate time series, nested Gaussian process, stochastic volatility

1. Introduction

In analyzing multivariate time series data, collected in financial applications, monitoring of influenza outbreaks and other fields, it is often of key importance to accurately characterize dynamic changes over time in not only the mean of the different elements (e.g., assets, influenza levels at different locations) but also the covariance. As shown in Figure 1, it is typical in many domains to cycle irregularly between periods of rapid and slow change; most statistical models are insufficiently flexible to capture such locally varying smoothness in assuming a single bandwidth parameter. Inappropriately restricting the smoothness to be constant can have a major impact on the quality of inferences and predictions, with over-

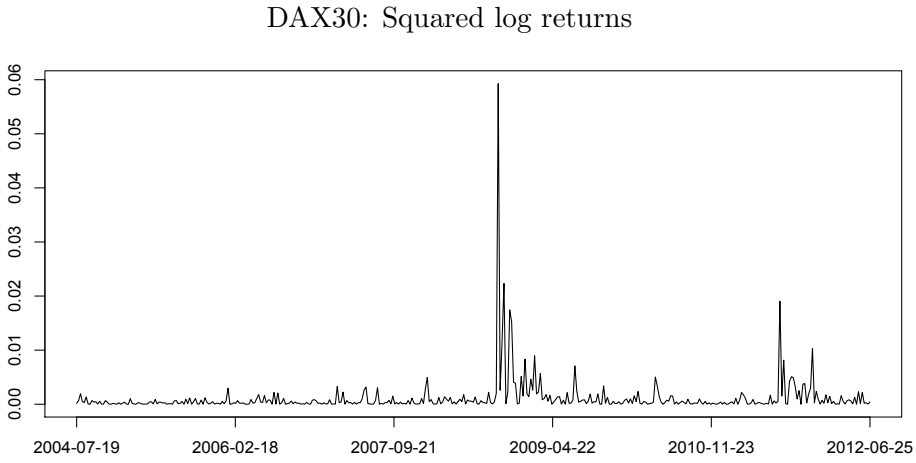


Figure 1: Squared log returns of DAX30. Weekly data from 19/07/2004 to 25/06/2012.

smoothing during times of rapid change. This leads to an under-estimation of uncertainty during such volatile times and an inability to accurately predict risk of extremal events.

Let $y_t = (y_{1t}, \ldots, y_{pt})^T$ denote a random vector at time t, with $\mu(t) = \mathrm{E}(y_t)$ and $\Sigma(t) = \mathrm{cov}(y_t)$. Our focus is on Bayesian modeling and inference for the multivariate mean-covariance stochastic process, $\Gamma = \{\mu(t), \Sigma(t), t \in \mathcal{T}\}$ with $\mathcal{T} \subset \Re^+$. Of particular interest is allowing locally varying smoothness, meaning that the rate of change in the $\{\mu(t), \Sigma(t)\}$ process is varying over time. To our knowledge, there is no previous proposed stochastic process for a coupled mean-covariance process, which allows locally varying smoothness. A key to our construction is the use of latent processes, which have time-varying smoothness. This results in a *locally adaptive factor* (LAF) process. We review the relevant literature below and then describe our LAF formulation.

1.1 Relevant Literature

There is a rich literature on modeling a $p \times 1$ time-varying mean vector $\mu(t)$, covering multivariate generalizations of autoregressive models (VAR) (see, e.g., Tsay, 2005), Kalman filtering (Kalman, 1960), nonparametric mean regression via Gaussian processes (GP) (Rasmussen and Williams, 2006), polynomial spline (Huang et al., 2002), smoothing spline (Hastie and Tibshirani, 1990) and kernel smoothing methods (Silverman, 1984). Such approaches perform well for slowly-changing trajectories with constant bandwidth parameters regulating implicitly or explicitly global smoothness; however, our interest is allowing smoothness to vary locally in continuous time. Possible extensions for local adaptivity include free knot splines (MARS) (Friedman, 1991), which perform well in simulations but the different strategies proposed to select the number and the locations of knots via stepwise knot selection (Friedman, 1991), Bayesian knot selection (Smith and Kohn, 1996) or MCMC methods (George and McCulloch, 1993), prove to be computationally intractable for moderately large p. Other flexible approaches include wavelet shrinkage (Donoho and

Johnstone, 1995), local polynomial fitting via variable bandwidth (Fan and Gijbels, 1995) and linear combination of kernels with variable bandwidths (Wolpert et al., 2011).

There is a separate literature on estimating a time-varying covariance matrix $\Sigma(t)$. This is particularly of interest in applications where volatilities and co-volatilities evolve through non constant paths. One popular approach estimates $\Sigma(t)$ via an exponentially weighted moving average (EWMA); see, e.g., Tsay (2005). This approach uses a single time-constant smoothing parameter $0 < \lambda < 1$, with extensions to accommodate locally varying smoothness not straightforward due to the need to maintain positive semidefinite $\Sigma(t)$ at every time. To allow for higher flexibility in the dynamics of the covariances, generalizations of EWMA have been proposed including the diagonal vector ARCH model (DVEC), (Bollerslev et al., 1988) and its variant, the BEKK model (Engle and Kroner, 1995). These models are computationally demanding and are not designed for moderate to large p. DCC-GARCH (Engle, 2002) improves the computational tractability of the previous approaches through a two-step formulation. However, the univariate GARCH assumed for the conditional variances of each time series and the higher level GARCH models with the same parameters regulating the evolution of the time-varying conditional correlations, restrict the evolution of the variance and covariance matrices. PC-GARCH (Ding, 1994; Burns, 2005) and O-GARCH (Alexander, 2001) perform dimensionality reduction through a latent factor formulation; see also van der Weide (2002). However, time-constant factor loadings and uncorrelated latent factors constrain the evolution of $\Sigma(t)$.

Such models fall far short of our goal of allowing $\Sigma(t)$ to be fully flexible with the dependence between $\Sigma(t)$ and $\Sigma(t + \Delta)$ varying with not just the time-lag Δ but also with time. In addition, these models do not handle missing data easily and tend to require long series for accurate estimation (Burns, 2005). Accommodating changes in continuous time is important in many applications, and avoids having the model be critically dependent on the time scale, with inconsistent models obtained as time units are varied.

Wilson and Ghahramani (2010) join machine learning and econometrics efforts by proposing a model for both mean and covariance regression in multivariate time series, improving previous work of Bru (1991) on Wishart processes in terms of computational tractability and scalability, allowing a more complex structure of dependence between $\Sigma(t)$ and $\Sigma(t + \Delta)$. Specifically, they propose a continuous time generalised Wishart process (GWP), which defines a collection of positive semi-definite random matrices $\Sigma(t)$ with Wishart marginals. Nonparametric mean regression for $\mu(t)$ is also considered via GP priors; however, the trajectories of means and covariances inherit the smooth behavior of the underlying Gaussian processes, limiting the flexibility of the approach across times exhibiting sharp changes.

Even for iid observations from a multivariate normal model with a single time stationary covariance matrix, there are well known problems with Wishart priors motivating a rich literature on dimensionality reduction techniques based on factor and graphical models. There has been abundant recent interest in applying such approaches to dynamic settings. Refer to Lopes and Carvalho (2007), Nakajima and West (2013) and the references cited therein for recent literature on Bayesian dynamic factor models for multivariate stochastic volatility. The Markov switching assumption for the levels of the common factor volatilities in Lopes and Carvalho (2007) improves flexibility, but may be restrictive in some applied fields and requires the additional choice of the number of possible regimes. Nakajima and West (2013) allow the factor loadings to evolve dynamically over time, while including sparsity through

a latent thresholding approach, leading to apparently improved performance in portfolio allocation. They assume a time-varying discrete-time autoregressive model, which allows the dependence in the covariance matrices $\Sigma(t)$ and $\Sigma(t+\Delta)$ to vary as a function of both t and Δ. However, the result is a richly parameterized and computationally challenging model, with selection of the number of factors proceeding by cross validation. Our emphasis is instead on developing continuous time stochastic processes for $\Sigma(t)$ and $\mu(t)$, which accommodate locally varying smoothness and provide relatively efficient MCMC computations based on a Gibbs sampler.

Fox and Dunson (2011) propose an alternative Bayesian covariance regression (BCR) model, which defines the covariance matrix as a regularized quadratic function of time-varying loadings in a latent factor model, characterizing the latter as a sparse combination of a collection of unknown Gaussian process dictionary functions. Although their approach provides a continuous time and highly flexible model that accommodates missing data and scales to moderately large p, there are two limitations motivating this article. Firstly, their proposed covariance stochastic process assumes a stationary dependence structure, and hence tends to under-smooth during periods of stability and over-smooth during periods of sharp changes. Secondly, the well known computational problems with usual GP regression are inherited, leading to difficulties in scaling to long series and issues in mixing of MCMC algorithms for posterior computation.

1.2 Contribution and Outline

Our proposed LAF process instead includes dictionary functions that are generated from nested Gaussian processes (nGP) (Zhu and Dunson, 2013), representing recently proposed priors which exploit stochastic differential equations (SDEs) to enforce GP priors for the function's mth order derivatives and favor local adaptivity by centering the latter on an higher level GP instantaneous mean. Such nGP reduces the GP computational burden involving matrix inversions from $O(T^3)$ to $O(T)$, with T denoting the length of the time series, while also allowing flexible locally varying smoothness. Marginalizing out the latent factors, we obtain a stochastic process that inherits these advantages. We also develop a different and more computationally efficient approach under this new model and propose an online implementation, which can accommodate streaming data. In Section 2, we describe the LAF structure with particular attention to prior specification. Section 3 explores the main features of the Gibbs sampler for posterior computation and outlines the steps for a fast online updating approach. In Section 4 we compare our model to BCR and to some of the most widely used models for multivariate stochastic volatility, through simulation studies. Finally in Section 5 an application to National Stock Market Indices across countries is examined.

2. Locally Adaptive Factor Processes

Our focus is on defining a novel locally adaptive factor process for $\Gamma = \{\mu(t), \Sigma(t), t \in \mathcal{T}\}$. In particular, taking a Bayesian approach, we define a prior $\Gamma \sim P$, where P is a probability measure over the space $\mathcal{P}$ of p-variate mean-covariance processes on $\mathcal{T}$. In particular, each element of $\mathcal{P}$ corresponds to a realization of the stochastic process Γ, and the measure P assigns probabilities to a σ-algebra of subsets of $\mathcal{P}$.

Although the proposed class of LAF processes can be used much more broadly, in conducting inferences in this article, we focus on the simple case in which data consist of vectors $y_i = (y_{1i}, \ldots, y_{pi})^T$ collected at times t_i, for $i = 1, \ldots, T$. These times can be unequally-spaced, or collected under an equally-spaced design with missing observations. An advantage of using a continuous-time process is that it is trivial to allow unequal spacing, missing data, and even observation times across which only a subset of the elements of y_i are observed. We additionally make the simplifying assumption that

$$y_i \sim \mathrm{N}_p(\mu(t_i), \Sigma(t_i)).$$

It is straightforward to modify the methodology to accommodate substantially different observation models.

2.1 LAF Specification

A common strategy in modeling of large p matrices is to rely on a lower-dimensional factorization, with factor analysis providing one possible direction. Sparse Bayesian factor models have been particularly successful in challenging cases, while having advantages over frequentist competitors in incorporating a probabilistic characterization of uncertainty in the number of factors as well as the parameters in the loadings and residual covariance. For recent articles on Bayesian sparse factor analysis for a single large covariance matrix, refer to Bhattacharya and Dunson (2011), Pati et al. (2012) and the references cited therein.

In our setting, we are instead interested in letting the mean vector and the covariance matrix vary flexibly over time. Extending the usual factor analysis framework to this setting, we say that $\Gamma = \{\mu(t), \Sigma(t), t \in \mathcal{T}\} \sim \mathrm{LAF}_{L,K}(\Theta, \Sigma_0, \Sigma_\xi, \Sigma_A, \Sigma_\psi, \Sigma_B)$ if

$$\mu(t) = \Theta\xi(t)\psi(t), \tag{1a}$$
$$\Sigma(t) = \Theta\xi(t)\xi(t)^T\Theta^T + \Sigma_0, \tag{1b}$$

where Θ is a $p \times L$ matrix of constant coefficients, $\Sigma_0 = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_p^2)$, while $\xi(t)_{L\times K}$ and $\psi(t)_{K\times 1}$ are matrices comprising continuous dictionary functions evolving in time via nGP, $\xi_{lk}(t) \sim \mathrm{nGP}([\Sigma_\xi]_{lk} = \sigma^2_{\xi_{lk}}, [\Sigma_A]_{lk} = \sigma^2_{A_{lk}})$ and $\psi_k(t) \sim \mathrm{nGP}([\Sigma_\psi]_k = \sigma^2_{\psi_k}, [\Sigma_B]_k = \sigma^2_{B_k})$.

Restricting our attention on the generic element $\xi_{lk}(t) : \mathcal{T} \to \Re$ of the matrix $\xi(t)_{L\times K}$ (the same holds for $\psi_k(t) : \mathcal{T} \to \Re$), the nGP provides a highly flexible stochastic process on the dictionary functions whose smoothness, explicitly modeled by their mth order derivatives $D^m\xi_{lk}(t)$ via stochastic differential equations, is expected to be centered on a local instantaneous mean function $A_{lk}(t)$, which represents a higher-level Gaussian process, that induces adaptivity to locally varying smoothing. Specifically, we let

$$D^m\xi_{lk}(t) = A_{lk}(t) + \sigma_{\xi_{lk}}W_{\xi_{lk}}(t), \quad m \in N, \quad m \geq 2, \tag{2a}$$
$$D^nA_{lk}(t) = \sigma_{A_{lk}}W_{A_{lk}}(t), \quad n \in N, \quad n \geq 1, \tag{2b}$$

where $\sigma_{\xi_{lk}} \in \Re^+$, $\sigma_{A_{lk}} \in \Re^+$, $W_{\xi_{lk}}(t) : \mathcal{T} \to \Re$ and $W_{A_{lk}}(t) : \mathcal{T} \to \Re$ are independent Gaussian white noise processes with mean $\mathrm{E}[W_{\xi_{lk}}(t)] = \mathrm{E}[W_{A_{lk}}(t)] = 0$, for all $t \in \mathcal{T}$, and covariance function $\mathrm{E}[W_{\xi_{lk}}(t)W_{\xi_{lk}}(t^*)] = \mathrm{E}[W_{A_{lk}}(t)W_{A_{lk}}(t^*)] = 1$ if $t = t^*$, 0 otherwise. This formulation naturally induces a stochastic process for $\xi_{lk}(t)$ with varying smoothness,

where $\mathrm{E}[D^m \xi_{lk}(t)|A_{lk}(t)] = A_{lk}(t)$, and initialization at t_1 based on the assumption

$$\begin{aligned}[\xi_{lk}(t_1), D^1\xi_{lk}(t_1), \ldots, D^{m-1}\xi_{lk}(t_1)]^T &\sim \mathrm{N}_m(0, \sigma^2_{\mu_{lk}} I_m),\\ [A_{lk}(t_1), D^1 A_{lk}(t_1), \ldots, D^{n-1}A_{lk}(t_1)]^T &\sim \mathrm{N}_n(0, \sigma^2_{\alpha_{lk}} I_n).\end{aligned}$$

The Markovian property implied by SDEs in (2a) and (2b) represents a key advantage in terms of computational tractability as it allows a simple state space formulation. In particular, referring to Zhu and Dunson (2013) for $m = 2$ and $n = 1$ (this can be easily extended for higher m and n), and for $\delta_i = t_{i+1} - t_i$ sufficiently small, the process for $\xi_{lk}(t)$ along with its first order derivative $\xi'_{lk}(t)$ and the local instantaneous mean $A_{lk}(t)$ follow the approximated state equation

$$\begin{bmatrix} \xi_{lk}(t_{i+1}) \\ \xi'_{lk}(t_{i+1}) \\ A_{lk}(t_{i+1}) \end{bmatrix} = \begin{bmatrix} 1 & \delta_i & 0 \\ 0 & 1 & \delta_i \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \xi_{lk}(t_i) \\ \xi'_{lk}(t_i) \\ A_{lk}(t_i) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{i,\xi_{lk}} \\ \omega_{i,A_{lk}} \end{bmatrix}, \tag{3}$$

where $[\omega_{i,\xi_{lk}}, \omega_{i,A_{lk}}]^T \sim \mathrm{N}_2(0, V_{i,lk})$, with $V_{i,lk} = \mathrm{diag}(\sigma^2_{\xi_{lk}}\delta_i, \sigma^2_{A_{lk}}\delta_i)$.

Similarly to the nGP specification for the elements in $\xi(t)$, we can represent the nested Gaussian process for $\psi_k(t)$ with the following state equation:

$$\begin{bmatrix} \psi_k(t_{i+1}) \\ \psi'_k(t_{i+1}) \\ B_k(t_{i+1}) \end{bmatrix} = \begin{bmatrix} 1 & \delta_i & 0 \\ 0 & 1 & \delta_i \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \psi_k(t_i) \\ \psi'_k(t_i) \\ B_k(t_i) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{i,\psi_k} \\ \omega_{i,B_k} \end{bmatrix}, \tag{4}$$

for $k = 1, \ldots, K$, where $[\omega_{i,\psi_k}, \omega_{i,B_k}]^T \sim \mathrm{N}_2(0, S_{i,k})$, with $S_{i,k} = \mathrm{diag}(\sigma^2_{\psi_k}\delta_i, \sigma^2_{B_k}\delta_i)$. Similarly to $\xi_{lk}(t)$, we let

$$\begin{aligned}[\psi_k(t_1), D^1\psi_k(t_1), \ldots, D^{m-1}\psi_k(t_1)]^T &\sim \mathrm{N}_m(0, \sigma^2_{\mu_k} I_m),\\ [B_k(t_1), D^1 B_k(t_1), \ldots, D^{n-1}B_k(t_1)]^T &\sim \mathrm{N}_n(0, \sigma^2_{\alpha_k} I_n).\end{aligned}$$

There are two crucial aspects to highlight. Firstly, this formulation is defined at every point over a subset of the real line and allows an irregular grid of observations over t by relating the latent states at $i + 1$ to those at i through the distance between t_{i+1} and t_i where i represents a discrete order index and $t_i \in \mathcal{T}$ the time value related to the ith observation. Secondly, compared to Zhu and Dunson (2013) our approach represents an important generalization in: (i) extending the analysis to the multivariate case (i.e. y_i is a p-dimensional vector instead of a scalar) and (ii) accommodating locally adaptive smoothing not only on the mean but also on the time-varying covariance functions.

2.2 LAF Interpretation

Model (1a)–(1b) can be induced by marginalizing out the K-dimensional latent factors vector η_i, in the model

$$y_i = \Lambda(t_i)\eta_i + \epsilon_i, \quad \epsilon_i \sim \mathrm{N}_p(0, \Sigma_0), \tag{5}$$

where $\eta_i = \psi(t_i) + \nu_i$ with $\nu_i \sim \mathrm{N}_K(0, I_K)$ and elements $\psi_k(t) \sim \mathrm{nGP}(\sigma^2_{\psi_k}, \sigma^2_{B_k})$ for $k = 1, \ldots, K$. In LAF formulation we assume moreover that the time-varying factor loadings

matrix $\Lambda(t)$ is a sparse linear combination, with respect to the weights of the $p \times L$ matrix Θ, of a much smaller set of continuous nested Gaussian processes $\xi_{lk}(t) \sim \text{nGP}(\sigma^2_{\xi_{lk}}, \sigma^2_{A_{lk}})$ comprising the $L \times K$, with $L << p$, matrix $\xi(t)$. As a result

$$\Lambda(t_i) = \Theta\xi(t_i). \tag{6}$$

Such a decomposition plays a crucial role in further reducing the number of nested Gaussian processes to be modeled from $p \times K$ to $L \times K$ leading to a more computationally tractable formulation in which the induced $\Gamma = \{\mu(t), \Sigma(t), t \in \mathcal{T}\}$ follows a locally adaptive factor $\text{LAF}_{L,K}(\Theta, \Sigma_0, \Sigma_\xi, \Sigma_A, \Sigma_\psi, \Sigma_B)$ process where

$$\mu(t_i) = \text{E}(y_i \mid t = t_i) = \Theta\xi(t_i)\psi(t_i), \tag{7a}$$

$$\Sigma(t_i) = \text{cov}(y_i \mid t = t_i) = \Theta\xi(t_i)\xi(t_i)^T\Theta^T + \Sigma_0. \tag{7b}$$

There is a literature on using Bayesian factor analysis with time-varying loadings, but essentially all the literature assumes discrete-time dynamics on the loadings while our focus is instead on allowing the loadings, and hence the induced $\Gamma = \{\mu(t), \Sigma(t), t \in \mathcal{T}\}$ processes, to evolve flexibly in continuous time. Hence, we are most closely related to the literature on Gaussian process latent factor models for spatial and temporal data; refer, for example, to Lopes et al. (2008) and Lopes et al. (2011). In these models, the factor loadings matrix characterizes spatial dependence, with time-varying factors accounting for dynamic changes.

Fox and Dunson (2011) instead allow the loadings matrix to vary through a continuous time stochastic process built from latent $\text{GP}(0, c)$ dictionary functions independently for all $l = 1, \ldots, L$ and $k = 1, \ldots, K$, with c the squared exponential correlation function having $c(t, t^*) = \exp(-\kappa||t - t^*||_2^2)$. In our work we follow the lead of Fox and Dunson (2011) in using a nonparametric latent factor model as in (5)–(6), but induce fundamentally different behavior on $\Gamma = \{\mu(t), \Sigma(t), t \in \mathcal{T}\}$ by carefully modifying the stochastic processes for the dictionary functions.

Note that the above decomposition of $\Gamma = \{\mu(t), \Sigma(t), t \in \mathcal{T}\}$ is not unique. Potentially we could constrain the loadings matrix to enforce identifiability (Geweke and Zhou, 1996), but this approach induces an undesirable order dependence among the responses (Aguilar and West, 2000; West, 2003; Lopes and West, 2004; Carvalho et al., 2008). Given our focus on estimation of Γ we follow Ghosh and Dunson (2009) in avoiding identifiability constraints, as such constraints are not necessary to ensure identifiability of the induced mean $\mu(t)$ and covariance $\Sigma(t)$. The characterization of the class of time-varying covariance matrices $\Sigma(t)$ is proved by Lemma 2.1 of Fox and Dunson (2011) which states that for K and L sufficiently large, any covariance regression can be decomposed as in (1b). Similar results are obtained for the mean process.

2.3 Prior Specification

We adopt a hierarchical prior specification approach to induce a prior P on $\Gamma = \{\mu(t), \Sigma(t), t \in \mathcal{T}\}$ with the goal of maintaining simple computation and allowing both covariances and means to evolve flexibly over continuous time. Specifically

- $\Gamma|\Theta, \Sigma_0, \Sigma_\xi, \Sigma_A, \Sigma_\psi, \Sigma_B \sim \text{LAF}_{L,K}(\Theta, \Sigma_0, \Sigma_\xi, \Sigma_A, \Sigma_\psi, \Sigma_B)$.

- Recalling the assumption $\xi_{lk}(t) \sim \text{nGP}(\sigma^2_{\xi_{lk}}, \sigma^2_{A_{lk}})$ within LAF representation, we assume for each each element $[\Sigma_\xi]_{lk}$ and $[\Sigma_A]_{lk}$ of the $L \times K$ matrices Σ_ξ and Σ_A respectively, the following priors

$$\begin{aligned} \sigma^2_{\xi_{lk}} &\sim \text{InvGa}(a_\xi, b_\xi), \\ \sigma^2_{A_{lk}} &\sim \text{InvGa}(a_A, b_A), \end{aligned}$$

  independently for each (l, k); where $\text{InvGa}(a, b)$ denotes the Inverse Gamma distribution with shape a and scale b.

- Similarly, the variances $[\Sigma_\psi]_k = \sigma^2_{\psi_k}$ and $[\Sigma_B]_k = \sigma^2_{B_k}$ in the state equation representation of the nGP for each $\psi_k(t) \sim \text{nGP}(\sigma^2_{\psi_k}, \sigma^2_{B_k})$ are assumed

$$\begin{aligned} \sigma^2_{\psi_k} &\sim \text{InvGa}(a_\psi, b_\psi), \\ \sigma^2_{B_k} &\sim \text{InvGa}(a_B, b_B), \end{aligned}$$

  independently for each k.

- To address the issue related to the selection of the number of dictionary elements a shrinkage prior is proposed for Θ. In particular, following Bhattacharya and Dunson (2011) we assume

$$\theta_{jl} | \phi_{jl}, \tau_l \sim \text{N}(0, \phi^{-1}_{jl} \tau^{-1}_l), \quad \phi_{jl} \sim \text{Ga}(3/2, 3/2),$$

$$\vartheta_1 \sim \text{Ga}(a_1, 1), \quad \vartheta_h \sim \text{Ga}(a_2, 1), \; h \geq 2, \quad \tau_l = \prod_{h=1}^{l} \vartheta_h. \tag{8}$$

  Note that if $a_2 > 1$ the expected value for ϑ_h is greater than 1. As a result, as l goes to infinity, τ_l tends to infinity, shrinking θ_{jl} towards zero. This leads to a flexible prior for θ_{jl} with a local shrinkage parameter ϕ_{jl} and a global column-wise shrinkage factor τ_l which allows many elements of Θ being close to zero as L increases. Our formulation can be easily generalized to allow shrinkage over K; see Fox and Dunson (2011). However we found reasonable to fix K to relatively small values and learn L with the shrinkage approach, to avoid higher computational complexity in sampling the K-variate vector $\psi(t_i)$, $i = 1, \ldots, T$.

- Finally for the variances of the error terms in vector ϵ_i, we assume the usual inverse gamma prior distribution. Specifically

$$\sigma_j^{-2} \sim \text{Ga}(a_\sigma, b_\sigma)$$

  independently for each $j = 1, \ldots, p$.

3. Posterior Computation

For a fixed truncation level L^* and a latent factor dimension K^*, the algorithm for posterior computation alternates between a simple and efficient simulation smoother step (Durbin and Koopman, 2002) to update the state space formulation of the nGP in LAF prior, and standard Gibbs sampling steps for updating the parametric component parameters from their full conditional distributions. See Bhattacharya and Dunson (2011) for a method adaptively choosing the truncation levels.

3.1 Gibbs Sampling

We outline here the main features of the algorithm for posterior computation based on observations (y_i, t_i) for $i = 1, \ldots, T$, while the complete algorithm is provided in Appendix A. Note that, since data are in practice observed at a finite number of times, the continuous time model is approximated in conducting inferences. This issue arises in analyzing data with any continuous time model.

A. Given Θ and $\{\eta_i\}_{i=1}^T$, a multivariate version of the MCMC algorithm proposed by Zhu and Dunson (2013) draws posterior samples from each dictionary element's function $\{\xi_{lk}(t_i)\}_{i=1}^T$, its first order derivative $\{\xi'_{lk}(t_i)\}_{i=1}^T$, the corresponding instantaneous mean $\{A_{lk}(t_i)\}_{i=1}^T$, the variances in the state equations $\sigma^2_{\xi_{lk}}$, $\sigma^2_{A_{lk}}$ and the variances of the error terms in the observation equation σ_j^2 with $j = 1, \ldots, p$.

B. Given Θ, $\{\sigma_j^{-2}\}_{j=1}^p$, $\{y_i\}_{i=1}^T$ and $\{\xi(t_i)\}_{i=1}^T$ we implement a block sampling of $\{\psi_k(t_i)\}_{i=1}^T$, $\{\psi'_k(t_i)\}_{i=1}^T$, $\{B_k(t_i)\}_{i=1}^T$, $\sigma^2_{\psi_k}$, $\sigma^2_{B_k}$ and ν_i following a similar approach as in step A.

C. Conditioned on $\{y_i\}_{i=1}^T$, $\{\eta_i\}_{i=1}^T$, $\{\sigma_j^{-2}\}_{j=1}^p$ and $\{\xi(t_i)\}_{i=1}^T$, and recalling the shrinkage prior for the elements of Θ in (8), we update Θ, each local shrinkage hyperparameter ϕ_{jl} and the global shrinkage hyperparameters τ_l following the standard conjugate analysis.

D. Given the posterior samples from Θ, Σ_0, $\{\xi(t_i)\}_{i=1}^T$ and $\{\psi(t_i)\}_{i=1}^T$ the realization of LAF process for $\{\mu(t_i), \Sigma(t_i), t_i \in \mathcal{T}\}$ conditioned on the data $\{y_i\}_{i=1}^T$ is

$$\begin{aligned} \mu(t_i) &= \Theta\xi(t_i)\psi(t_i), \\ \Sigma(t_i) &= \Theta\xi(t_i)\xi(t_i)^T\Theta^T + \Sigma_0. \end{aligned}$$

3.2 Hyperparameters Interpretation

We now focus our attention on the priors hyperparameters for $\sigma^2_{\xi_{lk}}$, $\sigma^2_{A_{lk}}$, $\sigma^2_{\psi_k}$ and $\sigma^2_{B_k}$. These quantities play an important role in facilitating local adaptivity and carefully tuning such values may improve mixing and convergence speed of our MCMC algorithm. Simulation studies have shown that the higher the variances in the latent state equations, the better our formulation accommodates locally adaptivity for sudden changes in Γ. A theoretical support for this data-driven consideration can be identified in the connection between the nGP and the nested smoothing splines. It has been shown by Zhu and Dunson (2013) that the posterior mean of the trajectory U with reference to the problem of nonparametric mean regression under the nGP prior can be related to the minimizer of the equation

$$\frac{1}{T}\sum_{i=1}^{T}(y_i - U(t_i))^2 + \lambda_U \int_{\mathcal{T}} (D^m U(t) - C(t))^2 dt + \lambda_C \int_{\mathcal{T}} (D^n C(t))^2 dt,$$

where C is the locally instantaneous function and $\lambda_U \in \Re^+$ and $\lambda_C \in \Re^+$ regulate the smoothness of the unknown functions U and C respectively, leading to less smoothed patterns when fixed at low values. The resulting inverse relationship between these smoothing parameters and the variances in the state equation, together with the results in the simulation studies, suggest to fix the hyperparameters in the Inverse Gamma prior for $\sigma^2_{\xi_{lk}}$, $\sigma^2_{A_{lk}}$,

$\sigma^2_{\psi_k}$ and $\sigma^2_{B_k}$ so as to allow high variances in the case in which the time series analyzed are expected to have strong changes in their covariance (or mean) dynamic. A further confirmation of the previous discussion is provided by the structure of the simulation smoother required to update the dictionary functions in our Gibbs sampling for posterior computation. More specifically, the larger the variances of $\{\omega_{i,\xi_{lk}}\}_{i=1}^T$, $\{\omega_{i,A_{lk}}\}_{i=1}^T$ and $\{\omega_{i,\psi_k}\}_{i=1}^T$, $\{\omega_{i,B_k}\}_{i=1}^T$ in the state equations, with respect to those of the vector of observations $\{y_i\}_{i=1}^T$, the higher is the weight associated to innovations in the filtering and smoothing techniques, allowing for less smoothed patterns both in the covariance and mean structures (see Durbin and Koopman, 2002).

In practical applications, it may be useful to obtain a first estimate of $\tilde{\Gamma} = \{\tilde{\mu}(t), \tilde{\Sigma}(t)\}$ to set the hyperparameters. More specifically, $\tilde{\mu}_j(t_i)$ can be the output of a standard moving average on each time series $y_j = (y_{j1}, \ldots, y_{jT})^T$, while $\tilde{\Sigma}(t_i)$ can be obtained by a simple estimator, such as the EWMA procedure. With these choices, the recursive equation

$$\tilde{\Sigma}(t_i) = (1-\lambda)\{[y_{i-1} - \tilde{\mu}(t_{i-1})][y_{i-1} - \tilde{\mu}(t_{i-1})]^T\} + \lambda\tilde{\Sigma}(t_{i-1}),$$

become easy to implement.

3.3 Online Updating

The problem of online updating represents a key point in multivariate time series with high frequency data. Referring to our formulation, we are interested in updating an approximated posterior for $\Gamma_{T+H} = \{\mu(t_{T+h}), \Sigma(t_{T+h}), h = 1, \ldots, H\}$ once a new vector of observations $\{y_i\}_{i=T+1}^{T+H}$ is available, instead of rerunning posterior computation for the whole time series.

Using the posterior estimates of the Gibbs sampler based on observations available up to time T, it is easy to implement (see Appendix B) a highly computationally tractable online updating algorithm which alternates between steps A, B and D outlined in the previous section for the new set of observations, and that can be initialized at $T+1$ using the one step ahead predictive distribution for the latent state vectors in the state space formulation. Such initialization procedure for latent state vectors in the algorithm depends on the sample moments of the posterior distribution for the latent states at T. As is known for Kalman smoothers (see, e.g., Durbin and Koopman, 2001), this could lead to computational problems in the online updating due to the larger conditional variances of the latent states at the end of the sample (i.e., at T). To overcome this problem, we replace the previous assumptions for the initial values with a data-driven initialization scheme. In particular, instead of using only the new observations for the online updating, we run the algorithm for $\{y_i\}_{i=T-k}^{T+H}$, with k small. As a result the distribution of the smoothed states at T is not anymore affected by the problem of large conditional variances leading to better online updating performance.

It is important to notice that the algorithm is not fully online in updating only the time-varying dictionary functions, while fixing the time-constant model parameters at their posterior mean. An alternative for properly propagating uncertainties while maintaining computational tractability may be to add a further step in the online updating procedure sampling the time-constant quantities conditionally on the updated dictionary functions and the quantities stored during the initial sampling. Such an approach may be reasonable if the initial time window considered is not enough large to ensure a consistent estimate

of the time-constant parameters and if the number of time series analyzed p is tractable. Since we search for a relatively fast procedure, and provided that for moderately large T the posterior for the time-stationary parameters rapidly becomes concentrated, we preferred our initially proposed algorithm in order to avoid the p draws from an L^* dimensional Gaussian in the sampling of Θ, which may slow down the online updating procedure for large p.

4. Simulation Studies

The aim of the following simulation studies is to compare the performance of our proposed LAF with respect to BCR, and to the models for multivariate stochastic volatility most widely used in practice, specifically: EWMA, PC-GARCH, GO-GARCH and DCC-GARCH. In order to assess whether and to what extent LAF can accommodate, in practice, even sharp changes in the time-varying means and covariances and to evaluate the costs of our flexible approach in settings where the mean and covariance functions do not require locally adaptive estimation techniques, we focus on two different sets of simulated data. The first is based on an underlying structure characterized by locally varying smoothness processes, while the second has means and covariances evolving in time through smooth processes. In the last subsection we also analyze the performance of the proposed online updating algorithm.

4.1 Simulated Data

A. Locally varying smoothness processes: We generate a set of 5-dimensional observations y_i for each t_i in the discrete set $\mathcal{T}_o = \{1, 2, \ldots, 100\}$, from the latent factor model in (5) with $\Lambda(t_i) = \Theta\xi(t_i)$. To allow sharp changes of means and covariances in the generating mechanism, we consider a 2×2 (i.e. $L = K = 2$) matrix $\{\xi(t_i)\}_{i=1}^{100}$ of time-varying functions adapted from Donoho and Johnstone (1994) with locally varying smoothness (more specifically we choose 'bumps' functions). The latent mean dictionary elements in $\{\psi(t_i)\}_{i=1}^{100}$ are simulated from a Gaussian process $\mathrm{GP}(0, c)$ with length scale $\kappa = 10$, while the elements in matrix Θ can be obtained from the shrinkage prior in (8) with $a_1 = a_2 = 10$. Finally the elements of the diagonal matrix Σ_0^{-1} are sampled independently from $\mathrm{Ga}(1, 0.1)$.

B. Smooth processes: We consider the same data set of 10-dimensional observations y_i with $t_i \in \mathcal{T}_o = \{1, 2, \ldots, 100\}$ investigated in Fox and Dunson (2011, Section 4.1). The settings are similar to the previous with exception of $\{\xi(t_i)\}_{i=1}^{100}$ which are 5×4 matrices of smooth GP dictionary functions with length scale $\kappa = 10$.

4.2 Estimation Performance

A. Locally varying smoothness processes:
Posterior computation for LAF is performed by using truncation levels $L^* = K^* = 2$ (at higher level settings we found that the shrinkage prior on Θ results in posterior samples of the elements in the additional columns concentrated around 0). We place a $\mathrm{Ga}(1, 0.1)$ prior on the precision parameters σ_j^{-2} and choose $a_1 = a_2 = 2$. As regards the nGP prior for each dictionary element $\xi_{lk}(t)$ with $l = 1, \ldots, L^*$ and $k = 1, \ldots, K^*$, we choose diffuse but proper priors for the initial values by setting $\sigma^2_{\mu_{lk}} = \sigma^2_{\alpha_{lk}} = 100$

and place an InvGa$(2, 10^8)$ prior on each $\sigma^2_{\xi_{lk}}$ and $\sigma^2_{A_{lk}}$ in order to allow less smooth behavior according to a previous graphical analysis of $\tilde{\Sigma}(t_i)$ estimated via EWMA. Similarly we set $\sigma^2_{\mu_k} = \sigma^2_{\alpha_k} = 100$ in the prior for the initial values of the latent state equations resulting from the nGP prior for $\psi_k(t)$, and consider $a_\psi = a_B = b_\psi = b_B = 0.005$ to balance the rough behavior induced on the nonparametric mean functions by the settings of the nGP prior on $\xi_{lk}(t)$, as suggested from previous graphical analysis. Note also that for posterior computation, we first scale the predictor space to $(0, 1]$, leading to $\delta_i = 1/100$, for $i = 1, \ldots, 100$.

For inference in BCR we consider the same previous hyperparameters setting for Θ and Σ_0 priors as well as the same truncation levels K^* and L^*, while the length scale κ in GP prior for $\xi_{lk}(t)$ and $\psi_k(t)$ has been set to 10 using the data-driven heuristic outlined in Fox and Dunson (2011). In both cases we run 50,000 Gibbs iterations discarding the first 20,000 as burn-in and thinning the chain every 5 samples.

As regards the other approaches, EWMA has been implemented by choosing the smoothing parameter λ that minimizes the mean squared error (MSE) between the estimated covariances and the true values. PC-GARCH algorithm follows the steps provided by Burns (2005) with GARCH(1,1) assumed for the conditional volatilities of each single time series and the principal components. GO-GARCH and DCC-GARCH recall the formulations provided by van der Weide (2002) and Engle (2002) respectively, assuming a GARCH(1,1) for the conditional variances of the processes analyzed, which proves to be a correct choice in many financial applications and also in our setting. Note that, differently from LAF and BCR, the previous approaches do not model explicitly the mean process $\{\mu(t_i)\}_{i=1}^{100}$ but work directly on the innovations $\{y_i - \mu(t_i)\}_{i=1}^{100}$. Therefore in these cases we first model the conditional mean via smoothing spline and in a second step we estimate the models working on the innovations. The smoothing parameter for spline estimation has been set to 0.7, which was found to be appropriate to best reproduce the true dynamic of $\{\mu(t_i)\}_{i=1}^{100}$.

B. Smooth processes:

We mainly keep the same setting of the previous simulation study with few differences. Specifically, L^* and K^* has been fixed to 5 and 4 respectively (also in this case the choice of the truncation levels proves to be appropriate, reproducing the same results provided in the simulation study of Fox and Dunson (2011) where $L^* = 10$ and $K^* = 10$). Moreover the scale parameters in the Inverse Gamma prior on each $\sigma^2_{\xi_{lk}}$ and $\sigma^2_{A_{lk}}$ has been set to 10^4 in order to allow a smoother behavior according to a previous graphical analysis of $\tilde{\Sigma}(t_i)$ estimated via EWMA, but without forcing the nGP prior to be the same as a GP prior. Following Fox and Dunson (2011) we run 10,000 Gibbs iterations which proved to be enough to reach convergence, and discarded the first 5,000 as burn-in.

In the first set of simulated data, we analyzed mixing by the Gelman-Rubin procedure (see, e.g., Gelman and Rubin, 1992), based on potential scale reduction factors computed for each chain by splitting the sampled quantities in 6 pieces of same length. The analysis shows slower mixing for BCR compared with LAF. Specifically, in LAF 95% of the chains have a potential reduction factor lower than 1.35, with a median equal to 1.11, while in

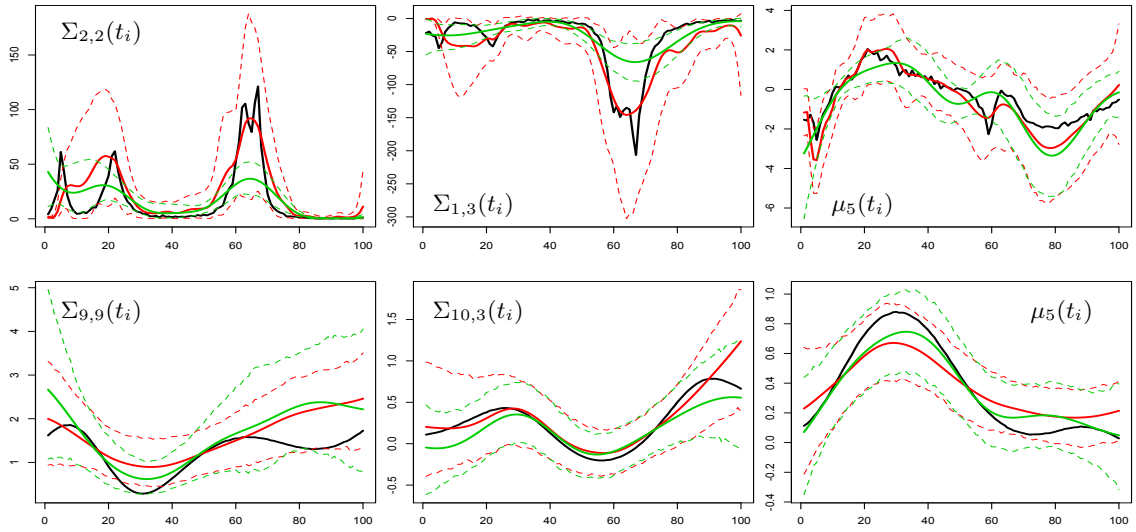


Figure 2: For locally varying smoothness simulation (top) and smooth simulation (bottom), plots of truth (black) and posterior mean respectively of LAF (solid red line) and BCR (solid green line) for selected components of the variance (left), covariance (middle), mean (right). For both approaches the dotted lines represent the 95% highest posterior density intervals.

BCR the 95% quantile is 1.44 and the median equals 1.18. Less problematic is the mixing for the second set of simulated data, with potential scale reduction factors having median equal to 1.05 for both approaches and 95% quantiles equal to 1.15 and 1.31 for LAF and BCR, respectively.

Figure 2 compares, in both simulated samples, true and posterior mean of the process $\Gamma = \{\mu(t_i), \Sigma(t_i), i = 1, \ldots, 100\}$ over the predictor space $\mathcal{T}_o$ together with the point-wise 95% highest posterior density (hpd) intervals for LAF and BCR. From the upper plots we can clearly note that our approach is able to capture conditional heteroscedasticity as well as mean patterns, also in correspondence of sharp changes in the time-varying true functions. The major differences compared to the true values can be found at the beginning and at the end of the series and are likely to be related to the structure of the simulation smoother which also causes a widening of the credibility bands at the very end of the series; for references regarding this issue see Durbin and Koopman (2001). However, even in the most problematic cases, the true values are within the bands of the 95% hpd intervals. Much more problematic is the behavior of the posterior distributions for BCR which over-smooth both covariance and mean functions leading also to many 95% hpd intervals not containing the true values. Bottom plots in Figure 2 show that the performance of our approach is very close to that of BCR, when data are simulated from a model where the covariances and means evolve smoothly across time and local adaptivity is not required. This happens even if the hyperparameters in LAF are set in order to maintain separation between nGP and GP prior, suggesting large support property for the proposed approach.

	Mean	90% Quantile	95% Quantile	Max
Covariance $\{\Sigma(t_i)\}$				
EWMA	1.37	2.28	5.49	85.86
PC-GARCH	1.75	2.49	6.48	229.50
GO-GARCH	2.40	3.66	10.32	173.41
DCC-GARCH	1.75	2.21	6.95	226.47
BCR	1.80	2.25	7.32	142.26
LAF	0.90	1.99	4.52	36.95
Mean $\{\mu(t_i)\}$				
SPLINE	0.064	0.128	0.186	2.595
BCR	0.087	0.185	0.379	2.845
LAF	0.062	0.123	0.224	2.529

Table 1: LOCALLY VARYING SMOOTHNESS PROCESSES: Summaries of the standardized squared errors between true values $\{\mu(t_i)\}_{i=1}^{100}$ and $\{\Sigma(t_i)\}_{i=1}^{100}$ and estimated quantities $\{\hat{\Sigma}(t_i)\}_{i=1}^{100}$ and $\{\hat{\mu}(t_i)\}_{i=1}^{100}$ computed with different approaches.

	Mean	90% Quantile	95% Quantile	Max
Covariance $\{\Sigma(t_i)\}$				
EWMA	0.030	0.081	0.133	1.119
PC-GARCH	0.018	0.048	0.076	0.652
GO-GARCH	0.043	0.104	0.202	1.192
DCC-GARCH	0.022	0.057	0.110	0.466
BCR	0.009	0.019	0.039	0.311
LAF	0.009	0.022	0.044	0.474
Mean $\{\mu(t_i)\}$				
SPLINE	0.007	0.019	0.027	0.077
BCR	0.005	0.015	0.024	0.038
LAF	0.005	0.017	0.026	0.050

Table 2: SMOOTH PROCESSES: Summaries of the standardized squared errors between true values $\{\mu(t_i)\}_{i=1}^{100}$ and $\{\Sigma(t_i)\}_{i=1}^{100}$ and estimated quantities $\{\hat{\Sigma}(t_i)\}_{i=1}^{100}$ and $\{\hat{\mu}(t_i)\}_{i=1}^{100}$ computed with different approaches.

The comparison of the summaries of the squared errors between true process $\Gamma = \{\mu(t_i), \Sigma(t_i), i = 1, \ldots, 100\}$ and the estimated quantities $\hat{\Gamma} = \{\hat{\mu}(t_i), \hat{\Sigma}(t_i), i = 1, \ldots, 100\}$ standardized with the range of the true processes $r_\mu = \max_{i,j}\{\mu_j(t_i)\} - \min_{i,j}\{\mu_j(t_i)\}$ and $r_\Sigma = \max_{i,j,k}\{\Sigma_{j,k}(t_i)\} - \min_{i,j,k}\{\Sigma_{j,k}(t_i)\}$ respectively, once again confirms the overall better performance of our approach relative to all the considered competitors. Table 1 shows that, when local adaptivity is required, LAF provides a superior performance having standardized residuals lower than those of the other approaches. EWMA seems to provide quite accurate estimates, but it is important to underline that we choose the optimal smoothing parameter λ in order to minimize the MSE between estimated and true parameters, which are clearly not known in practical applications. Different values of λ reduces significantly the performance of EWMA, which shows also lack of robustness. The closeness of the summaries of LAF and BCR in Table 2 confirms the flexibility of LAF even in settings where local adaptivity is not required and highlights the better performance of the two approaches with respect to the other competitors also when smooth processes are investigated.

To better understand the improvement of our approach in allowing locally varying smoothness and to evaluate the consequences of the over-smoothing induced by BCR on the

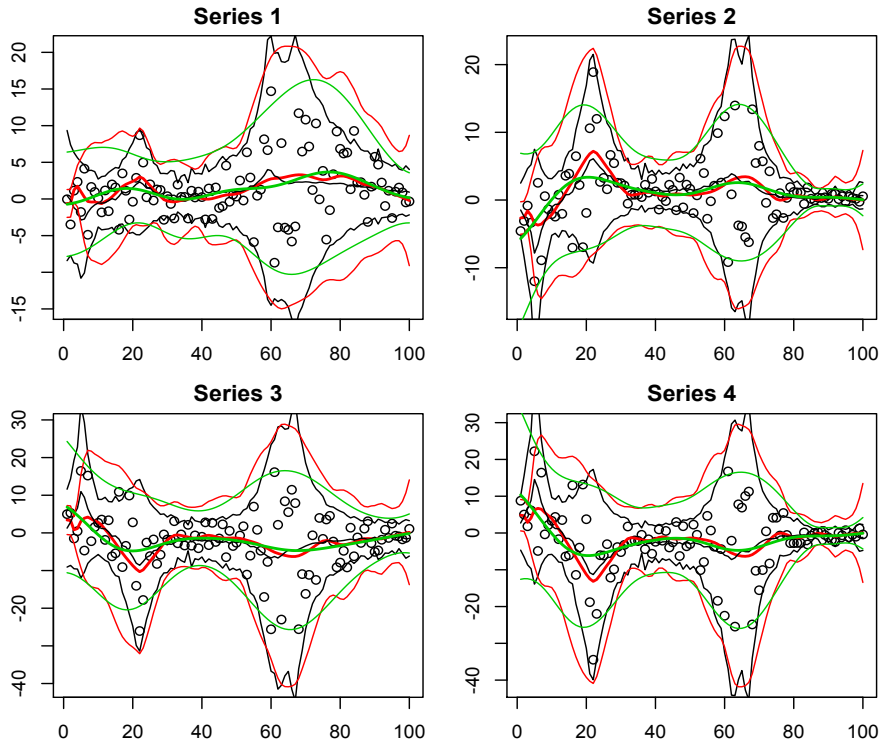


Figure 3: For 4 selected simulated series: time-varying mean $\mu_j(t_i)$ and 2.5% and 97.5% quantiles of the marginal distribution of y_{ji} with true mean and variance (black), mean and variance from posterior mean of LAF (red), mean and variance from posterior mean of BCR (green). Black points represent the simulated data.

distribution of y_i with $i = 1, \ldots, 100$ consider Figure 3 which shows, for some selected series $\{y_{ji}\}_{i=1}^{100}$ in the first simulated data set, the time-varying mean together with the point-wise 2.5% and 97.5% quantiles of the marginal distribution of y_{ji} induced respectively by the true mean and true variance, the posterior mean of $\mu_j(t_i)$ and $\Sigma_{jj}(t_i)$ from our proposed approach and the posterior mean of the same quantities from BCR. We can clearly see that the marginal distribution of y_{ji} induced by BCR is over-concentrated near the mean, leading to incorrect inferences. Note that our proposal is also able to accommodate heavy tails, a typical characteristic in financial series.

4.3 Online Updating Performance

To analyze the performance of the online updating algorithm in LAF model, we simulate 50 new observations $\{y_i\}_{i=101}^{150}$ with $t_i \in \mathcal{T}_o^* = \{101, \ldots, 150\}$, considering the same Θ and Σ_0 used in the generating mechanism for the first simulated data set and taking the 50 subsequent observations of the bumps functions for the dictionary elements $\{\xi(t_i)\}_{i=101}^{150}$; finally the additional latent mean dictionary elements $\{\psi(t_i)\}_{i=101}^{150}$ are simulated as before maintaining the continuity with the previously simulated functions $\{\psi(t_i)\}_{i=1}^{100}$. According to

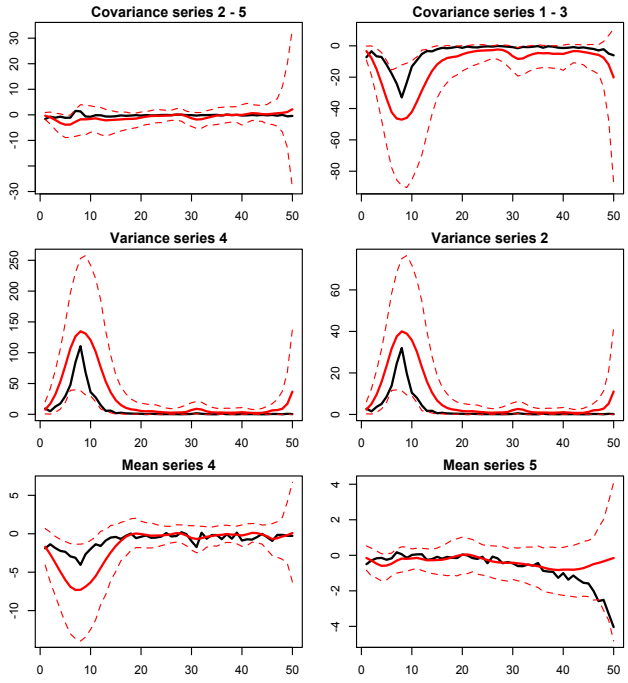


Figure 4: Plots of truth (black) and posterior mean of the online updating procedure (solid red line) for selected components of the covariance (top), variance (middle), mean (bottom). The dotted lines represent the 95% highest posterior density intervals.

the algorithm described in Subsection 3.3, we fix Θ, Σ_0, Σ_ξ, Σ_A,Σ_ψ and Σ_B at their posterior mean from the previous Gibbs sampler and consider the last three observations y_{98}, y_{99} and y_{100} (i.e. $k = 3$) to initialize the simulation smoother in $i = 101$ through the proposed data-driven initialization approach. Posterior computation shows good performance in terms of mixing, and convergence is assessed after 5,000 Gibbs iterations with a small burn-in of 500.

Figure 4 compares true mean and covariance to posterior mean of a selected set of components of $\Gamma_* = \{\mu(t_i), \Sigma(t_i), i = 101, \ldots, 150\}$ including also the 95% hpd intervals. The results clearly show that the online updating is characterized by a good performance which allows to capture the behavior of new observations conditioning on the previous estimates. Note that the posterior distribution of the approximated mean and covariance functions tends to slightly over-estimate the patterns of the functions at sharp changes, however also in these cases the true values are within the bands of the credibility intervals. Finally note that the data-driven initialization ensures a good behavior at the beginning of the series, while the results at the end have wider uncertainty bands as expected.

5. Application Study

Spurred by the recent growth of interest in the dynamic dependence structure between financial markets in different countries, and in its features during the crises that have

followed in recent years, we applied our LAF to the multivariate time series of the main National Stock Market Indices.

5.1 National Stock Market Indices, Introduction and Motivation

National Stock Market Indices represent technical tools that allow, through the synthesis of numerous data on the evolution of the various stocks, to detect underlying trends in the financial market, with reference to a specific basis of currency and time. More specifically, each Market Index can be defined as a weighted sum of the values of a set of national stocks, whose weighting factors is equal to the ratio of its market capitalization in a specific date and overall of the whole set on the same date.

In this application we focus our attention on the multivariate weekly time series of the main 33 (i.e. $p = 33$) National Stock Market Indices from 12/07/2004 to 25/06/2012. Figure 5 shows the main features in terms of stationarity, mean patterns and volatility of two selected National Stock Market Indices downloaded from `http://finance.yahoo.com/`. The non-stationary behavior, together with the different bases of currency and time, motivate the use of logarithmic returns $y_{ji} = \log(I_{ji}/I_{ji-1})$, where I_{ji} is the value of the Stock Market Index j at time t_i. Beside this, the marginal distribution of log returns shows heavy tails and irregular cyclical trends in the nonparametric estimation of the mean, while EWMA estimates highlight rapid changes of volatility during the financial crises observed in the recent years. All these results, together with large p settings and high frequency data typical in financial fields, motivate the use of our approach to obtain a better characterization of the time-varying dependence structure among financial markets.

5.2 LAF for National Stock Market Indices

We consider the heteroscedastic model $y_i \sim \mathrm{N}_{33}(\mu(t_i), \Sigma(t_i))$ for $i = 1, \ldots, 415$ and t_i in the discrete set $\mathcal{T}_o = \{1, 2, \ldots, 415\}$, where the elements of $\Gamma = \{\mu(t_i), \Sigma(t_i), i = 1, \ldots, 415\}$, defined by (7a)-(7b), are induced by the dynamic latent factor model outlined in (5)-(6).

Posterior computation is performed by first rescaling the predictor space $\mathcal{T}_o$ to $(0, 1]$ and using the same setting of the first simulation study, with the exception of the truncation levels fixed at $K^* = 4$ and $L^* = 5$ (which we found to be sufficiently large from the fact that the last few columns of the posterior samples for Θ assumed values close to 0) and the hyperparameters of the nGP prior for each $\xi_{lk}(t)$ and $\psi_k(t)$ with $l = 1, \ldots, L^*$ and $k = 1, \ldots, K^*$, set to $a_\xi = a_A = a_\psi = a_B = 2$ and $b_\xi = b_A = b_\psi = b_B = 5 \times 10^7$ to capture also rapid changes in the mean functions according to Figure 5. Missing values in our data set do not represent a limitation since the Bayesian approach allows us to update our posterior considering solely the observed data. We run 10,000 Gibbs iterations with a burn-in of 2,500. Examination of trace plots of the posterior samples for $\Gamma = \{\mu(t_i), \Sigma(t_i), i = 1, \ldots, 415\}$ showed no evidence against convergence.

Posterior distributions for the variances in Figure 6 demonstrate that we are clearly able to capture the rapid changes in the dynamics of volatility that occur during the world financial crisis of 2008, in early 2010 with the Greek debt crisis and in the summer of 2011 with the financial speculation in government bonds of European countries together with the rejection of the U.S. budget and the downgrading of the United States rating. Moreover,

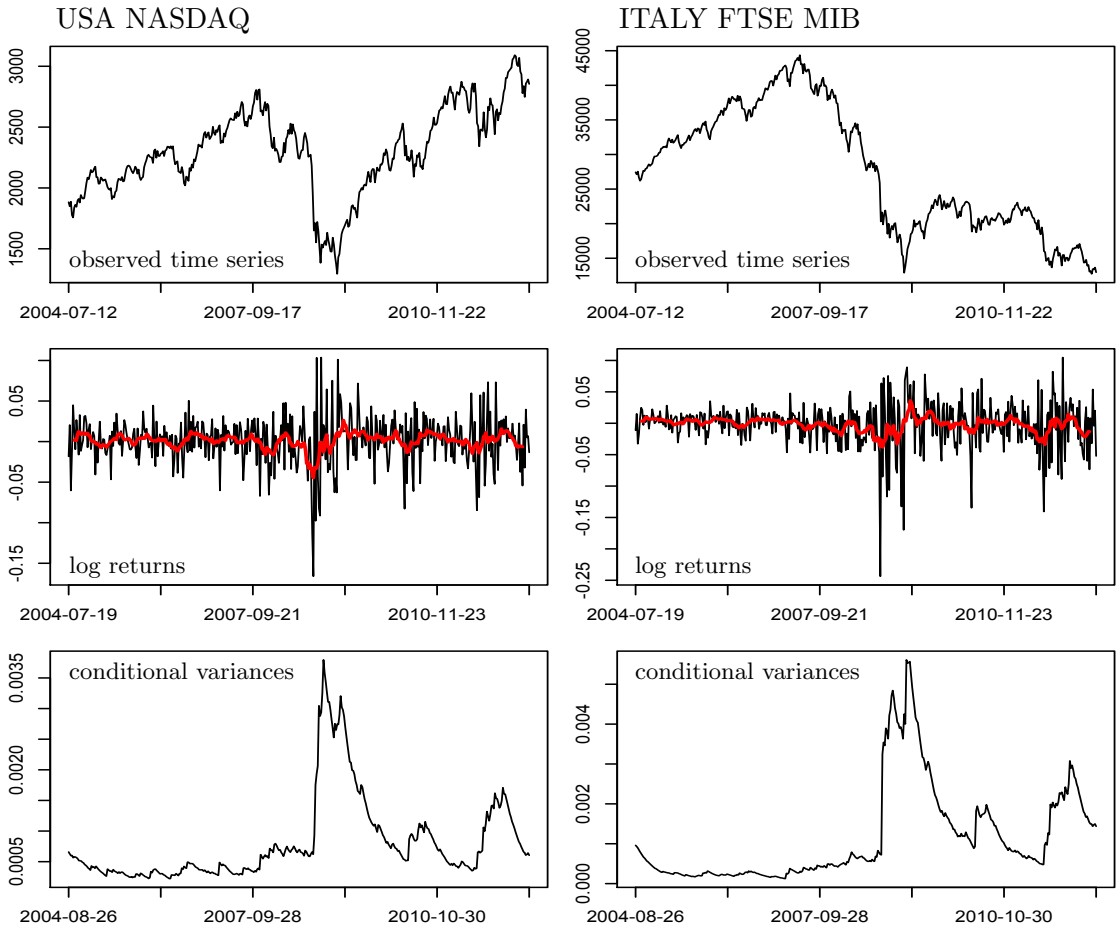


Figure 5: Plots of the main features of USA NASDAQ (left) and ITALY FTSE MIB (right). Specifically: observed time series (top), log returns series with nonparametric mean estimation via 12 week Equally Weighted Moving Average (red) in the middle, EWMA volatility estimates (bottom).

the resulting marginal distribution of the log returns induced by the posterior mean of $\mu_j(t)$ and $\Sigma_{jj}(t)$, shows that we are also able to accommodate heavy tails as well as mean patterns cycling irregularly between slow and more rapid changes.

Important information about the ability of our model to capture the evolution of world geo-economic structure during different finance scenarios is provided in Figures 7 and 8. From the correlations between NASDAQ and the other National Stock Market Indices (based on the posterior mean $\{\hat{\Sigma}(t_i)\}_{i=1}^{415}$ of the covariances function) in Figure 7, we can immediately notice the presence of a clear geo-economic structure in world financial markets (more evident in LAF than in BCR), where the dependence between the U.S. and European countries is systematically higher than that of South East Asian Nations (Economic Tigers), showing also different reactions to crises. Plots at the top of the Figure 8 confirms the above considerations showing how Western countries exhibit more connection with countries closer

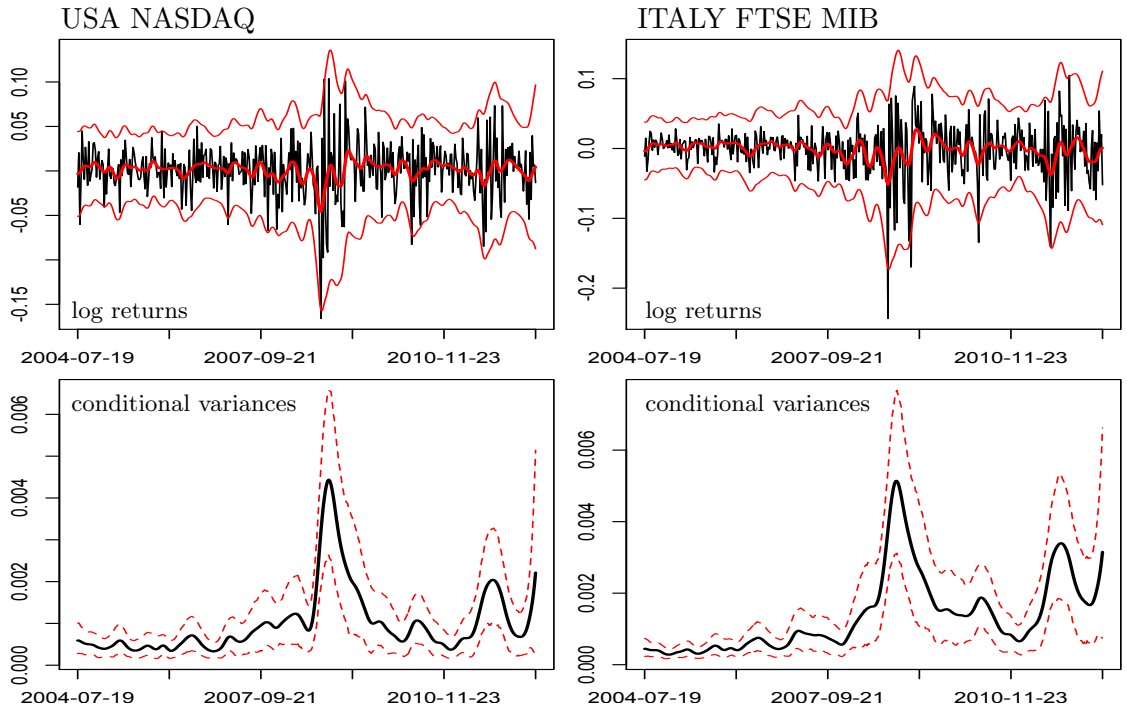


Figure 6: Top: Plot for 2 National Stock Market Indices, respectively USA NASDAQ (left) and ITALY FTSE MIB (right), of the log returns (black) and the time-varying estimated mean $\{\hat{\mu}_j(t_i)\}_{i=1}^{415}$ together with the time-varying 2.5% and 97.5% quantiles (red) of the marginal distribution of y_{ji} from LAF. Bottom: posterior mean (black) and 95% hpd (dotted red) for the variances $\{\Sigma_{jj}(t_i)\}_{i=1}^{415}$.

in terms of geographical, political and economic structure; the same holds for Eastern countries where we observe a reversal of the colored curves. As expected, Russia is placed in a middle path between the two blocks. A further element that our model captures about the structure of the markets is shown in the plots at the bottom of Figure 8. The time-varying regression coefficients obtained from the standard formulas of the conditional normal distribution based on the posterior mean of $\Gamma = \{\mu(t_i), \Sigma(t_i), i = 1, \ldots, 415\}$ highlight clearly the increasing dependence of European countries with higher crisis in sovereign debt and Germany, which plays a central role in Eurozone as expected.

The flexibility of the proposed approach and the possibility of accommodating varying smoothness in the trajectories over time, allow us to obtain a good characterization of the dynamic dependence structure according with the major theories on financial crisis. The top plot in Figure 7 shows how the change of regime in correlations occurs exactly in correspondence to the burst of the U.S. housing bubble (A), in the second half of 2006. Moreover we can immediately notice that the correlations among financial markets increase significantly during the crises, showing a clear international financial contagion effect in agreement with other theories on financial crisis (see, e.g., Baig and Goldfajn, 1999; Claessens and Forbes, 2001). As expected the persistence of high levels of correlation is evident during the global

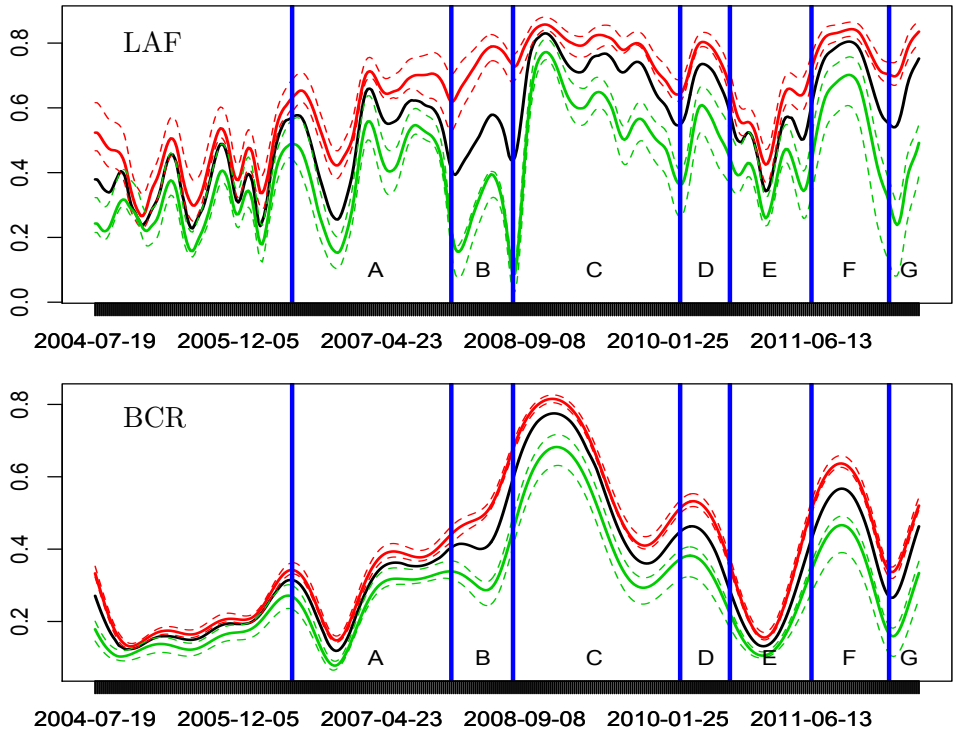


Figure 7: Black line: For USA NASDAQ median of correlations with the other 32 National Stock Market Indices based on posterior mean of $\{\Sigma(t_i)\}_{i=1}^{415}$. Red lines: 25%, 75% (dotted lines) and 50% (solid line) quantiles of correlations between USA NASDAQ and European countries (without considering Greece and Russia which present a specific pattern). Green lines: 25%, 75% (dotted lines) and 50% (solid line) quantiles of correlations between USA NASDAQ and the countries of South-east Asia (Asian Tigers and India). Timeline: (A) burst of U.S. housing bubble; (B) risk of failure of the first U.S. credit agencies (Bear Stearns, Fannie Mae and Freddie Mac); (C) world financial crisis after the Lehman Brothers' bankruptcy; (D) Greek debt crisis; (E) financial reform launched by Barack Obama and E.U. efforts to save Greece (the two peaks represent respectively Irish debt crisis and Portugal debt crisis); (F) worsening of European sovereign-debt crisis and the rejection of the U.S. budget; (G) crisis of credit institutions in Spain and the growing financial instability of the Eurozone.

financial crisis between late-2008 and end-2009 (C), at the beginning of which our approach also captures a sharp variation in the correlations between the U.S. and Economic Tigers, which lead to levels close to those of Europe. Further rapid changes are identified in correspondence of Greek crisis (D), the worsening of European sovereign-debt crisis and the rejection of the U.S. budget (F) and the recent crisis of credit institutions in Spain together with the growing financial instability Eurozone (G). Finally, even in the period of U.S. financial reform launched by Barack Obama and E.U. efforts to save Greece (E), we

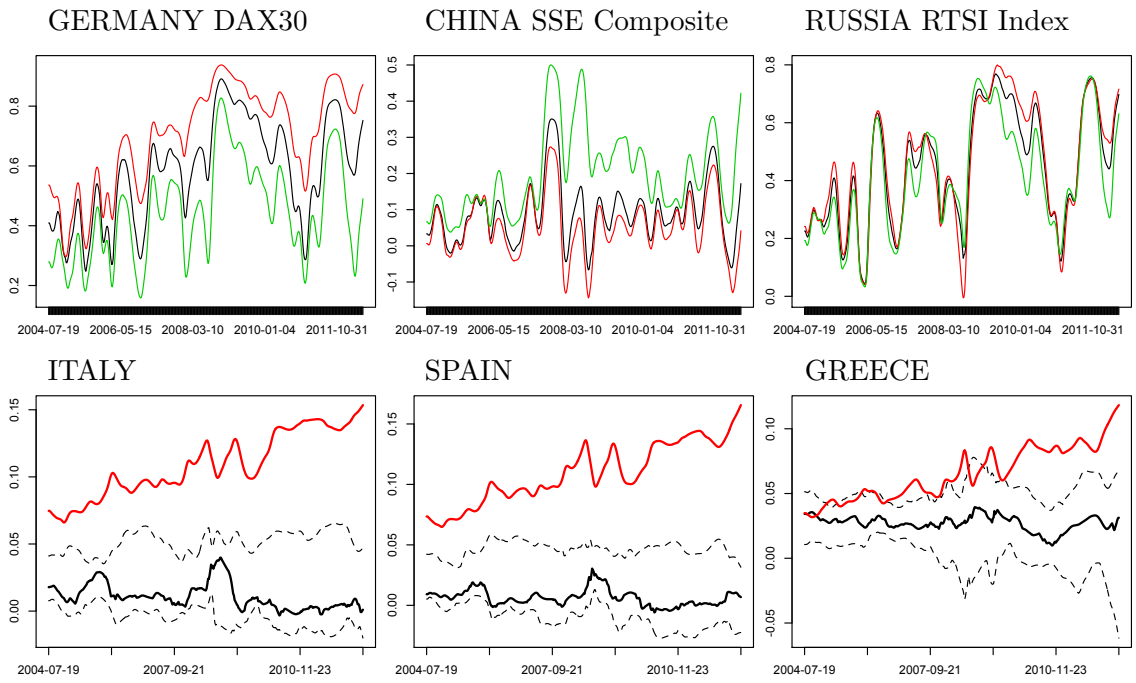


Figure 8: Top: For 3 selected National Stock Market Indices, plot of the median of the correlation based on posterior mean of $\{\Sigma(t_i)\}_{i=1}^{415}$ with the other 32 world stock indices (black), the European countries without considering Greece and Russia (red) and the Asian Tigers including India (green). Bottom: For 3 of the European countries more subject to sovereign debt crisis, plot of 25%, 50% and 75% quantiles of the time-varying regression parameters based on posterior mean $\{\hat{\Sigma}(t_i)\}_{i=1}^{415}$ with the other countries (black) and Germany (red).

can notice two peaks representing respectively Irish debt crisis and Portugal debt crisis. Note also that BCR, as expected, tends to over-smooth the dynamic dependence structure during the financial crisis, proving to be not able to model the sharp change in the correlations between USA NASDAQ and Economic Tigers during late-2008, and the two peaks representing respectively Irish and Portugal debt crisis at the beginning of 2011.

5.3 National Stock Market Indices, Updating and Predicting

The possibility to quickly update the estimates and the predictions as soon as new data arrive, represents a crucial aspect to obtain quantitative informations about the future scenarios of the crisis in financial markets. To answer this goal, we apply the online updating algorithm presented in Subsection 3.3, to the new set of weekly observations $\{y_i\}_{i=416}^{422}$ from 02/07/2012 to 13/08/2012 conditioning on posterior estimates of the Gibbs sampler based on observations $\{y_i\}_{i=1}^{415}$ available up to 25/06/2012. We initialized the simulation smoother algorithm with the last 8 observations of the previous sample.

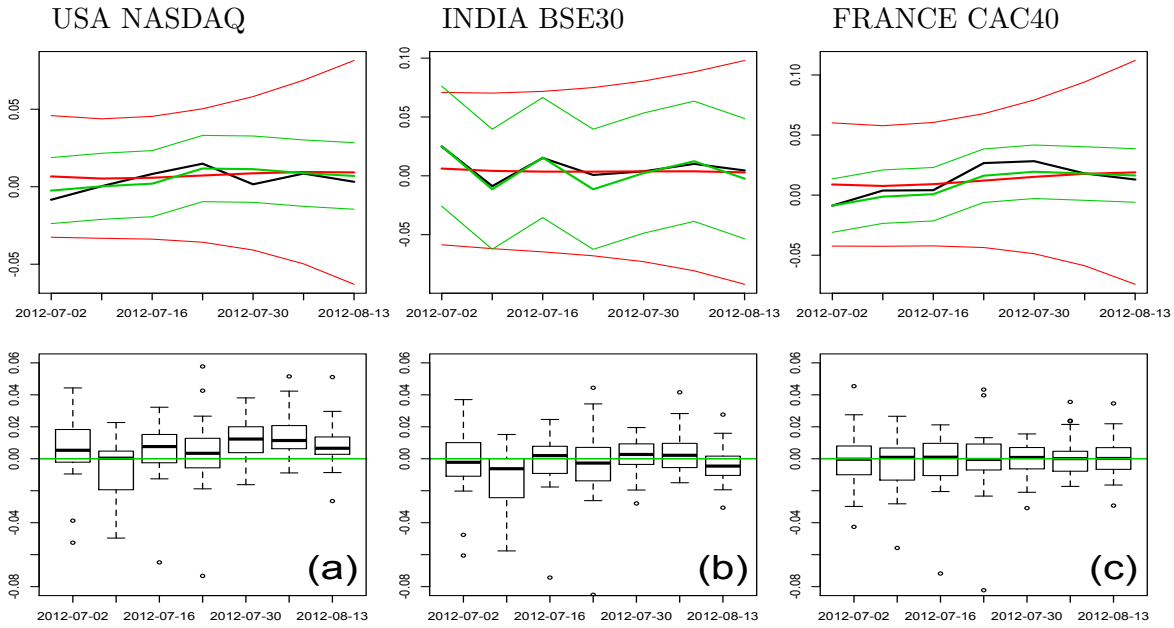


Figure 9: Top: For 3 selected National Stock Market Indices, respectively USA NASDAQ (left), INDIA BSE30 (middle) and FRANCE CAC40 (right), plot of the observed log returns (black) together with the mean and the 2.5% and 97.5% quantiles of the marginal distribution (red) and conditional distribution given the other 32 National Stock Market Indices (green) based on the posterior mean of $\Gamma_* = \{\mu(t_i), \Sigma(t_i), i = 416, \ldots, 422\}$ from the online updating procedure for the new observations from 02/07/2012 to 13/08/2012. Bottom: boxplots of the one step ahead prediction errors for the 33 National Stock Market Indices, where the predicted values are respectively: (a) unconditional mean $\{\tilde{y}_{i+1}\}_{i=415}^{421} = 0$, (b) marginal mean of the one step ahead predictive distribution, (c) conditional mean given the log returns of the other 32 NSI at $i+1$ of the one step ahead predictive distribution. Predictions for (b) and (c) are induced by the posterior mean of $\{\mu(t_{i+1|i}), \Sigma(t_{i+1|i}), i = 415, \ldots, 421\}$ of LAF.

Plots at the top of Figure 9 show, for 3 selected National Stock Market Indices, the new observed log returns $\{y_{ji}\}_{i=416}^{422}$ (black) together with the mean and the 2.5% and 97.5% quantiles of the marginal distribution (red) and conditional distribution (green) of $y_{ji}|y_i^{-j}$ with $y_i^{-j} = \{y_{qi}, q \neq j\}$. We use standard formulas of the multivariate normal distribution based on the posterior mean of the updated $\Gamma_* = \{\mu(t_i), \Sigma(t_i), i = 416, \ldots, 422\}$ after 5,000 Gibbs iterations with a burn in of 500. This is sufficient for convergence based on examining trace plots of the time-varying mean and covariance matrices. From these results, we can clearly notice the good performance of our proposed online updating algorithm in obtaining a characterization for the distribution of new observations. Also note that the multivariate approach together with a flexible model for the mean and covariance, allow for significant improvements when the conditional distribution of an index given the others is analyzed.

To obtain further informations about the predictive performance of our LAF, we can easily use our online updating algorithm to obtain h step-ahead predictions for $\Gamma_{T+H|T} = \{\mu(t_{T+h|T}), \Sigma(t_{T+h|T}), h = 1, \ldots, H\}$. In particular, referring to Durbin and Koopman (2001), we can generate posterior samples of $\Gamma_{T+H|T}$ merely by treating $\{y_i\}_{i=T+1}^{T+H}$ as missing values in the proposed online updating algorithm. Here, we consider the one step ahead prediction (i.e. $H = 1$) problem for the new observations. More specifically, for each i from 415 to 421, we update the mean and covariance functions conditioning on informations up to t_i through the online algorithm and then obtain the predicted posterior distribution for $\Sigma(t_{i+1|i})$ and $\mu(t_{i+1|i})$ by adding to the sample considered for the online updating a last column y_{i+1} of missing values.

Plots at the bottom of Figure 9, show the boxplots of the one step ahead prediction errors for the 33 National Stock Market Indices obtained as the difference between the predicted value $\tilde{y}_{j,i+1}$ and, once available, the observed log return $y_{j,i+1}$ with $i + 1 = 416, \ldots, 422$ corresponding to weeks from 02/07/2012 to 13/08/2012. In (a) we forecast the future log returns with the unconditional mean $\{\tilde{y}_{i+1}\}_{i=415}^{421} = 0$, which is what is often done in practice under the general assumption of zero mean, stationary log returns. In (b) we consider $\tilde{y}_{i+1} = \hat{\mu}(t_{i+1|i})$, the posterior mean of the one step ahead predictive distribution of $\mu(t_{i+1|i})$, obtained from the previous proposed approach after 5,000 Gibbs iteration with a burn in of 500. Finally in (c) we suppose that the log returns of all National Stock Market Indices except that of country j (i.e., $y_{j,i+1}$) become available at t_{i+1} and, considering $y_{i+1|i} \sim N_p(\hat{\mu}(t_{i+1|i}), \hat{\Sigma}(t_{i+1|i}))$ with $\hat{\mu}(t_{i+1|i})$ and $\hat{\Sigma}(t_{i+1|i})$ posterior mean of the one step ahead predictive distribution respectively for $\mu(t_{i+1|i})$ and $\Sigma(t_{i+1|i})$, we forecast $y_{j,i+1}$ with the conditional mean of $y_{j,i+1|i}$ given the other log returns at time t_{i+1}. Comparing boxplots in (a) with those in (b) we can see that our model allows to obtain improvements also in terms of prediction. Furthermore, by analyzing the boxplots in (c) we can notice how our ability to obtain a good characterization of the time-varying covariance structure can play a crucial role also in improving forecasting, since it enters into the standard formula for calculating the conditional mean in the normal distribution.

6. Discussion

In this paper, we have presented a continuous time multivariate stochastic process for time series to obtain a better characterization for mean and covariance temporal dynamics. Maintaining simple conjugate posterior updates and tractable computations in moderately large p settings, our model increases significantly the flexibility of previous approaches as it captures sharp changes both in mean and covariance dynamics while accommodating heavy tails. Beside these key advantages, the state space formulation enables development of a fast online updating algorithm particularly useful for high frequency data.

The simulation studies highlight the flexibility and the overall better performance of LAF with respect to the models for multivariate stochastic volatility most widely used in practice, both when adaptive estimation techniques are required, and also when the underlying mean and covariance structures do not show sharp changes in their dynamic.

The application to the problem of capturing temporal and geo-economic structure between the main financial markets demonstrates the utility of our approach and the improvements that can be obtained in the analysis of multivariate financial time series with

reference to (i) heavy tails, (ii) locally adaptive mean regression, (iii) sharp changes in covariance functions, (iii) high dimensional data set, (iv) online updating with high frequency data (v) missing values and (vi) predictions. Potentially further improvements are possible using a stochastic differential equation model that explicitly incorporates prior information on dynamics.

Acknowledgments

This research was partially supported by grant R01ES17240 from the National Institute of Environmental Health Sciences (NIEHS) of the National Institutes of Health (NIH) and by grant CPDA121180/12 from the University of Padua, Italy.

Appendix A. Posterior Computation

For a fixed truncation level L^* and a latent factor dimension K^* the detailed steps of the Gibbs sampler for posterior computations are:

1. Define the vector of the latent states and the error terms in the state space equation resulting from nGP prior for dictionary elements as

$$\begin{aligned} \Xi_i &= [\xi_{11}(t_i), \xi_{21}(t_i), \ldots, \xi_{L^*K^*}(t_i), \xi'_{11}(t_i), \ldots, \xi'_{L^*K^*}(t_i), A_{11}(t_i), \ldots, A_{L^*K^*}(t_i)]^T, \\ \Omega_{i,\xi} &= [\omega_{i,\xi_{11}}, \omega_{i,\xi_{21}}, \ldots, \omega_{i,\xi_{L^*K^*}}, \omega_{i,A_{11}}, \omega_{i,A_{21}}, \ldots, \omega_{i,A_{L^*K^*}}]^T. \end{aligned}$$

   Given Θ, $\{\eta_i\}_{i=1}^T$, $\{y_i\}_{i=1}^T$, Σ_0 and the variances in latent state equations $\{\sigma^2_{\xi_{lk}}\}$, $\{\sigma^2_{A_{lk}}\}$, with $l = 1, \ldots, L^*$ and $k = 1, \ldots, K^*$; update $\{\Xi_i\}_{i=1}^T$ by using the simulation smoother in the following state space model

$$y_i = [\eta_i^T \otimes \Theta, 0_{p\times(2\times K^*\times L^*)}]\Xi_i + \epsilon_i, \tag{9}$$

$$\Xi_{i+1} = T_i\Xi_i + R_i\Omega_{i,\xi}, \tag{10}$$

   where the observation equation in (9) results by applying the *vec* operator in the latent factor model $y_i = \Theta\xi(t_i)\eta_i + \epsilon_i$. More specifically recalling the property $vec(ABC) = (C^T \otimes A)vec(B)$ we obtain

$$\begin{aligned} y_i = vec(y_i) &= vec\{\Theta\xi(t_i)\eta_i + \epsilon_i\} \\ &= vec\{\Theta\xi(t_i)\eta_i\} + vec(\epsilon_i) \\ &= (\eta_i^T \otimes \Theta)vec\{\xi(t_i)\} + \epsilon_i. \end{aligned}$$

   The state equation in (10) is a joint representation of the equations resulting from the nGP prior on each $\xi_{lk}(t)$ defined in (3). As a result, the $(3\times L^*\times K^*)\times(3\times L^*\times K^*)$ matrix T_i together with the $(3\times L^*\times K^*)\times(2\times L^*\times K^*)$ matrix R_i reproduce, for each dictionary element the state equation in (3) by fixing to 0 the coefficients relating latent states with different (l, k) (from the independence between the dictionary elements). Finally, recalling the assumptions on $\omega_{i,\xi_{lk}}$ and $\omega_{i,A_{lk}}$, $\Omega_{i,\xi}$ is normally distributed with $\mathrm{E}[\Omega_{i,\xi}] = 0$ and $\mathrm{E}[\Omega_{i,\xi}\Omega^T_{i,\xi}] = \mathrm{diag}(\sigma^2_{\xi_{11}}\delta_i, \ldots, \sigma^2_{\xi_{L^*K^*}}\delta_i, \sigma^2_{A_{11}}\delta_i, \ldots, \sigma^2_{A_{L^*K^*}}\delta_i)$.

2. Given $\{\Xi_i\}_{i=1}^T$ sample each $\sigma^2_{\xi_{lk}}$ and $\sigma^2_{A_{lk}}$ respectively from

$$\begin{aligned}\sigma^2_{\xi_{lk}}|\{\Xi_i\} &\sim \text{InvGa}\left(a_\xi + \frac{T}{2}, b_\xi + \frac{1}{2}\sum_{i=1}^{T-1}\frac{(\xi'_{lk}(t_{i+1}) - \xi'_{lk}(t_i) - A_{lk}(t_i)\delta_i)^2}{\delta_i}\right),\\ \sigma^2_{A_{lk}}|\{\Xi_i\} &\sim \text{InvGa}\left(a_A + \frac{T}{2}, b_A + \frac{1}{2}\sum_{i=1}^{T-1}\frac{(A_{lk}(t_{i+1}) - A_{lk}(t_i))^2}{\delta_i}\right).\end{aligned}$$

3. Similarly to Ξ_i and $\Omega_{i,\xi}$ let

$$\begin{aligned}\Psi_i &= [\psi_1(t_i), \psi_2(t_i), \ldots, \psi_{K^*}(t_i), \psi'_1(t_i), \ldots, \psi'_{K^*}(t_i), B_1(t_i), \ldots, B_{K^*}(t_i)]^T,\\ \Omega_{i,\psi} &= [\omega_{i,\psi_1}, \omega_{i,\psi_2}, \ldots, \omega_{i,\psi_{K^*}}, \omega_{i,B_1}, \omega_{i,B_2}, \ldots, \omega_{i,B_{K^*}}]^T,\end{aligned}$$

be the vectors of the latent states and error terms in the state space equation resulting from nGP prior for $\psi(t)$. Conditional on Θ, $\{\xi(t_i)\}_{i=1}^T$, $\{y_i\}_{i=1}^T$, Σ_0, and the variances in latent state equations $\{\sigma^2_{\psi_k}\}$, $\{\sigma^2_{B_k}\}$, with $k = 1, \ldots, K^*$; sample $\{\Psi_i\}_{i=1}^T$ from the simulation smoother in the following state space model

$$\begin{aligned}y_i &= [\Theta\xi(t_i), 0_{p\times(2\times K^*)}]\Psi_i + \varpi_i, &(11)\\ \Psi_{i+1} &= G_i\Psi_i + F_i\Omega_{i,\psi}, &(12)\end{aligned}$$

$\varpi_i \sim N(0, \Theta\xi(t_i)\xi(t_i)^T\Theta^T + \Sigma_0)$. The observation equation in (11) results by marginalizing out ν_i in the latent factor model with nonparametric mean regression $y_i = \Theta\xi(t_i)\psi(t_i) + \Theta\xi(t_i)\nu_i + \epsilon_i$. Analogously to Ξ_i, the state equation in (12) is a joint representation of the state equation induced by the nGP prior on each $\psi_k(t)$ defined in (4); where the $(3\times K^*)\times(3\times K^*)$ matrix G_i and the $(3\times K^*)\times(2\times K^*)$ matrix F_i are constructed with the same goal of the matrices T_i and R_i in the state space model for Ξ_i. Finally, $\Omega_{i,\psi} \sim N_{2\times K^*}(0, \text{diag}(\sigma^2_{\psi_1}\delta_i, \sigma^2_{\psi_2}\delta_i, \ldots, \sigma^2_{\psi_{K^*}}\delta_i, \sigma^2_{B_1}\delta_i, \sigma^2_{B_2}\delta_i, \ldots, \sigma^2_{B_{K^*}}\delta_i))$.

4. Given $\{\Psi_i\}_{i=1}^T$ update each $\sigma^2_{\psi_k}$ and $\sigma^2_{B_k}$ respectively from

$$\begin{aligned}\sigma^2_{\psi_k}|\{\Psi_i\} &\sim \text{InvGa}\left(a_\psi + \frac{T}{2}, b_\psi + \frac{1}{2}\sum_{i=1}^{T-1}\frac{(\psi'_k(t_{i+1}) - \psi'_k(t_i) - B_k(t_i)\delta_i)^2}{\delta_i}\right),\\ \sigma^2_{B_k}|\{\Psi_i\} &\sim \text{InvGa}\left(a_B + \frac{T}{2}, b_B + \frac{1}{2}\sum_{i=1}^{T-1}\frac{(B_k(t_{i+1}) - B_k(t_i))^2}{\delta_i}\right).\end{aligned}$$

5. Conditioned on Θ, Σ_0, y_i, $\xi(t_i)$ and $\psi(t_i)$, and recalling $\nu_i \sim N_{K^*}(0, I_{K^*})$; the standard conjugate posterior distribution $\nu_i|\Theta, \Sigma_0, \tilde{y}_i, \xi(t_i), \psi(t_i)$ is

$$\text{N}_{K^*}\left((I + \xi(t_i)^T\Theta^T\Sigma_0^{-1}\Theta\xi(t_i))^{-1}\xi(t_i)^T\Theta^T\Sigma_0^{-1}\tilde{y}_i, (I + \xi(t_i)^T\Theta^T\Sigma_0^{-1}\Theta\xi(t_i))^{-1}\right),$$

with $\tilde{y}_i = y_i - \Theta\xi(t_i)\psi(t_i)$.

6. Conditioned on Θ, $\{\eta_i\}_{i=1}^T$, $\{y_i\}_{i=1}^T$, and $\{\xi(t_i)\}_{i=1}^T$ (obtained from Ξ_i), the standard conjugate posterior from which to update σ_j^{-2} is

$$\sigma_j^{-2}|\Theta, \{\eta_i\}, \{y_i\}, \{\xi(t_i)\} \sim \text{Ga}\left(a_\sigma + \frac{T}{2}, b_\sigma + \frac{1}{2}\sum_{i=1}^{T}(y_{ji} - \theta_{j\cdot}\xi(t_i)\eta_i)^2\right).$$

Where $\theta_{j\cdot} = [\theta_{j1}, \ldots, \theta_{jL^*}]$

7. Given $\{\eta_i\}_{i=1}^T$, $\{y_i\}_{i=1}^T$, $\{\xi(t_i)\}_{i=1}^T$ and the hyperparameters ϕ and τ the shrinkage prior on Θ combined with the likelihood for the latent factor model lead to the Gaussian posterior

$$\theta_{j\cdot}|\{\eta_i\}, \{y_i\}, \{\xi(t_i)\}, \phi, \tau \sim \text{N}_{L^*}\left(\tilde{\Sigma}_\theta \tilde{\eta}^T \sigma_j^{-2} \begin{bmatrix} y_{j1} \\ \vdots \\ y_{jT} \end{bmatrix}, \tilde{\Sigma}_\theta\right),$$

where $\tilde{\eta}^T = [\xi(t_1)\eta_1, \xi(t_2)\eta_2, \ldots, \xi(t_T)\eta_T]$ and

$$\tilde{\Sigma}_\theta^{-1} = \sigma_j^{-2}\tilde{\eta}^T\tilde{\eta} + \text{diag}(\phi_{j1}\tau_1, \ldots, \phi_{jL^*}\tau_{L^*}).$$

8. The Gamma prior on the local shrinkage hyperparameter ϕ_{jl} implies the standard conjugate posterior given θ_{jl} and τ_l

$$\phi_{jl}|\theta_{jl}, \tau_l \sim \text{Ga}\left(2, \frac{3 + \tau_l\theta_{jl}^2}{2}.\right)$$

9. Conditioned on Θ and τ, sample the global shrinkage hyperparameters from

$$\vartheta_1|\Theta, \tau^{(-1)} \sim \text{Ga}\left(a_1 + \frac{pL^*}{2}, 1 + \frac{1}{2}\sum_{l=1}^{L^*}\tau_l^{(-1)}\sum_{j=1}^{p}\phi_{jl}\theta_{jl}^2\right),$$
$$\vartheta_h|\Theta, \tau^{(-h)} \sim \text{Ga}\left(a_2 + \frac{p(L^* - h + 1)}{2}, 1 + \frac{1}{2}\sum_{l=h}^{L^*}\tau_l^{(-h)}\sum_{j=1}^{p}\phi_{jl}\theta_{jl}^2\right),$$

where $\tau_l^{(-h)} = \prod_{t=1, t\neq h}^{l}\vartheta_t$ for $h = 1, \ldots, L^*$.

10. Given the posterior samples from Θ, Σ_0, $\{\xi(t_i)\}_{i=1}^T$ and $\{\psi(t_i)\}_{i=1}^T$ the realization of the LAF process for $\{\mu(t_i), \Sigma(t_i), t_i \in \mathcal{T}\}$ conditioned on the data $\{y_i\}_{i=1}^T$ is

$$\begin{aligned} \mu(t_i) &= \Theta\xi(t_i)\psi(t_i), \\ \Sigma(t_i) &= \Theta\xi(t_i)\xi(t_i)^T\Theta^T + \Sigma_0. \end{aligned}$$

Appendix B. Online Updating Algorithm

Consider Θ, Σ_0, $\{\sigma^2_{\xi_{lk}}\}$, $\{\sigma^2_{A_{lk}}\}$, $\{\sigma^2_{\psi_k}\}$ and $\{\sigma^2_{B_k}\}$ fixed at their posterior mean $\hat{\Theta}$, $\hat{\Sigma}_0$, $\{\hat{\sigma}^2_{\xi_{lk}}\}$, $\{\hat{\sigma}^2_{A_{lk}}\}$, $\{\hat{\sigma}^2_{\psi_k}\}$, $\{\hat{\sigma}^2_{B_k}\}$ respectively, and let $\hat{\Xi}_T$, $\hat{\Sigma}_{\Xi_T}$ and $\hat{\Psi}_T$, $\hat{\Sigma}_{\Psi_T}$ be the sample mean and covariance matrix of the posterior distribution respectively for Ξ_T and Ψ_T obtained from the posterior estimates of the Gibbs sampler conditioned on $\{y_i\}_{i=1}^T$.

1. Given $\hat{\Theta}$, $\hat{\Sigma}_0$, $\{\hat{\sigma}^2_{\xi_{lk}}\}$, $\{\hat{\sigma}^2_{A_{lk}}\}$, $\{\eta_i\}_{i=T+1}^{T+H}$ and $\{y_i\}_{i=T+1}^{T+H}$ update $\{\Xi_i\}_{i=T+1}^{T+H}$ by using the simulation smoother in the following state space model

$$\begin{aligned} y_i &= [\eta_i^T \otimes \hat{\Theta}, 0_{p\times(2\times K^*\times L^*)}]\Xi_i + \epsilon_i, \\ \Xi_{i+1} &= T_i\Xi_i + R_i\Omega_{i,\xi}, \end{aligned}$$

   where Ξ_{T+1} can be initialized from the standard one step ahead predictive distribution for the state space model $\Xi_{T+1} \sim \mathrm{N}(T_T\hat{\Xi}_T, T_T\hat{\Sigma}_{\Xi_T}T_T^T + R_TE[\Omega_{T,\xi}\Omega_{T,\xi}^T]R_T^T)$.

2. Conditioned on $\hat{\Theta}$, $\hat{\Sigma}_0$, $\{\hat{\sigma}^2_{\psi_k}\}$, $\{\hat{\sigma}^2_{B_k}\}$, $\{\xi(t_i)\}_{i=T+1}^{T+H}$ and $\{y_i\}_{i=T+1}^{T+H}$ sample $\{\Psi_i\}_{i=T+1}^{T+H}$ through the simulation smoother in the state space model

$$\begin{aligned} y_i &= [\hat{\Theta}\xi(t_i), 0_{p\times(2\times K^*)}]\Psi_i + \varpi_i, \\ \Psi_{i+1} &= G_i\Psi_i + F_i\Omega_{i,\psi}. \end{aligned}$$

   Similarly to Ξ_{T+1}, $\Psi_{T+1} \sim \mathrm{N}(G_T\hat{\Psi}_T, G_T\hat{\Sigma}_{\Psi_T}G_T^T + F_TE[\Omega_{T,\psi}\Omega_{T,\psi}^T]F_T^T)$.

3. Given $\hat{\Theta}$, $\hat{\Sigma}_0$, $\{y_i\}$, $\xi(t_i)$ and $\psi(t_i)$, for $i = T+1, \ldots, T+H$, sample ν_i from the standard conjugate posterior distribution for $\nu_i|\hat{\Theta}, \hat{\Sigma}_0, \tilde{y}_i, \xi(t_i), \psi(t_i)$:

$$\mathrm{N}_{K^*}\left((I + \xi(t_i)^T\hat{\Theta}^T\hat{\Sigma}_0^{-1}\hat{\Theta}\xi(t_i))^{-1}\xi(t_i)^T\hat{\Theta}^T\hat{\Sigma}_0^{-1}\tilde{y}_i, (I + \xi(t_i)^T\hat{\Theta}^T\hat{\Sigma}_0^{-1}\hat{\Theta}\xi(t_i))^{-1}\right),$$

   with $\tilde{y}_i = y_i - \hat{\Theta}\xi(t_i)\psi(t_i)$.

4. Compute the updated covariance $\{\Sigma(t_i)\}_{i=T+1}^{T+H}$ and mean $\{\mu(t_i)\}_{i=T+1}^{T+H}$ from the usual equations

$$\begin{aligned} \Sigma(t_i) &= \hat{\Theta}\xi(t_i)\xi(t_i)^T\hat{\Theta}^T + \hat{\Sigma}_0, \\ \mu(t_i) &= \hat{\Theta}\xi(t_i)\psi(t_i). \end{aligned}$$

References

O. Aguilar and M. West. Bayesian dynamic factor models and portfolio allocation. *Journal of Business & Economic Statistics*, 18(3):338–357, 2000.

C.O. Alexander. Orthogonal GARCH. *Mastering Risk*, 2:21–38, 2001.

T. Baig and I. Goldfajn. Financial Market Contagion in the Asian Crisis. *IMF Staff Papers*, 46(2):167–195, 1999.

A. Bhattacharya and D.B. Dunson. Sparse Bayesian infinite factor models. *Biometrika*, 98(2):291–306, 2011.

T. Bollerslev, R.F. Engle and J.M. Wooldridge. A capital-asset pricing model with time-varying covariances. *Journal of Political Economy*, 96(1):116–131, 1988.

M. Bru. Wishart Processes. *Journal of Theoretical Probability*, 4(4):725–751, 1991.

P. Burns. Multivariate GARCH with Only Univariate Estimation. Available electronically at `http://www.burns-stat.com/pages/Working/multgarchuni.pdf`, 2005.

C.M. Carvalho, J.E. Lucas, Q. Wang, J. Chang, J.R. Nevins and M. West. High-dimensional sparse factor modeling - Applications in gene expression genomics. *Journal of the American Statistical Association*, 103(484):1438–1456, 2008.

S. Claessens and K. Forbes. *International Financial Contagion*. Springer, New York, 2001.

Z. Ding. *Time series analysis of speculative returns*. PhD thesis, University of California, San Diego, 1994.

D.L. Donoho and J.M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994.

D.L. Donoho and J.M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432):1200–1224, 1995.

J. Durbin and S. Koopman. *Time Series Analysis by State Space Methods*. Oxford University Press Inc., New York, 2001.

J. Durbin and S. Koopman. A simple and efficient simulation smoother for state space time series analysis. *Biometrika*, 89(3):603–615, 2002.

R.F. Engle and K.F. Kroner. Multivariate simultaneous generalized ARCH. *Econometric Theory*, 11(1):122–150, 1995.

R.F. Engle. Dynamic conditional correlation: a simple class of multivariate generalized autoregressive conditional heteroskedasticity models. *Journal of Business & Economic Statistics*, 20(3):339–350, 2002.

J. Fan and I. Gijbels. Data-driven bandwidth selection in local polynomial fitting: variable bandwidth and spatial adaptation. *Journal of the Royal Statistical Society. Series B*, 57(2):371–394, 1995.

E. Fox and D.B. Dunson. Bayesian Nonparametric Covariance Regression. Available electronically at `http://arxiv.org/pdf/1101.2017.pdf`, 2011.

J.H. Friedman. Multivariate Adaptive Regression Splines. *Annals of Statistics*, 19(1):1–67, 1991.

A. Gelman and D.B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457–472, 1992.

E.I. George and R.E. McCulloch. Variable selection via Gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889, 1993.

J. Geweke and G. Zhou. Measuring the pricing error of the arbitrage pricing theory. *Review of Financial Studies*, 9(2):557–587, 1996.

J. Ghosh and D.B. Dunson. Default Prior Distributions and Efficient Posterior Computation in Bayesian Factor Analysis. *Journal of Computational and Graphical Statistics*, 18(2):306–320, 2009.

T.J. Hastie and R.J. Tibshirani. *Generalized Additive Models*. Chapman & Hall, London, 1990.

J.Z. Huang, C.O. Wu and L. Zhou. Varying-coefficient models and basis function approximations for the analysis of repeated measurements. *Biometrika*, 89(1):111–128, 2002.

R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering* , 82(1):35–45, 1960.

H.F. Lopes and C.M. Carvalho. Factor stochastic volatility with time-varying loadings and Markov switching regimes. *Journal of Statistical Planning and Inference*, 137(10):3082–3091, 2007.

H.F. Lopes, D. Gamerman and E. Salazar. Generalized spatial dynamic factor models. *Computational Statistics & Data Analysis*, 55(3):1319–1330, 2011.

H.F. Lopes, E. Salazar and D. Gamerman. Spatial Dynamic Factor Analysis. *Bayesian Analysis*, 3(4):759–792, 2008.

H.F. Lopes and M. West. Bayesian model assessment in factor analysis. *Statistica Sinica*, 14(1):41–67, 2004.

J. Nakajima and M. West. Bayesian analysis of latent threshold dynamic models. *Journal of Business and Economic Statistics*, 31(2):151–164, 2013.

D. Pati, A. Bhattacharya, N.S. Pillai and D.B. Dunson. Bayesian high-dimensional covariance matrix estimation. Available electronically at `http://ftp.stat.duke.edu/WorkingPapers/12-05.html`, 2012.

C.E. Rasmussen and C.K.I Williams. *Gaussian processes for machine learning*. MIT Press, Boston, 2006.

B.W. Silverman. Spline Smoothing: The Equivalent Variable Kernel Method. *Annals of Statistics*, 12(3):898-916, 1984.

M. Smith and R. Kohn. Nonparametric regression using Bayesian variable selection. *Journal of Econometrics*, 75(2):317-343, 1996.

R.S. Tsay. *Analysis of Financial Time Series*. II ed., Wiley, New York, 2005.

R. van der Weide. GO-GARCH: a multivariate generalized orthogonal GARCH model. *Journal of Applied Econometrics*, 17(5):549–564, 2002.

M. West. Bayesian factor regression models in the large p, small n paradigm. In *Proceedings of the Seventh Valencia International Meeting*, pages 723–732, Tenerife, Spain, 2003.

A.G. Wilson and Z. Ghahramani. Generalised Wishart Processes. Available electronically at `http://arxiv.org/pdf/1101.0240.pdf`, 2010.

R.L. Wolpert, M.A. Clyde and C. Tu. Stochastic expansions using continuous dictionaries: Levy adaptive regression kernels. *Annals of Statistics*, 39(4):1916–1962, 2011.

C.O. Wu, C.T. Chiang and D.R. Hoover. Asymptotic confidence regions for kernel smoothing of a varying-coefficient model with longitudinal data. *Journal of the American Statistical Association*, 93(444):1388-1402, 1998.

B. Zhu and D.B. Dunson. Locally Adaptive Bayes Nonparametric Regression via Nested Gaussian Processes. *Journal of the American Statistical Association*, 108(504):1445–1456, 2013.

Journal of Machine Learning Research 15 (2014) 1523-1548 Submitted 5/13; Revised 2/14; Published 4/14

Iteration Complexity of Feasible Descent Methods for Convex Optimization

Po-Wei Wang B97058@CSIE.NTU.EDU.TW
Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
Department of Computer Science
National Taiwan University
Taipei 106, Taiwan

Editor: S. Sathiya Keerthi

Abstract

In many machine learning problems such as the dual form of SVM, the objective function to be minimized is convex but not strongly convex. This fact causes difficulties in obtaining the complexity of some commonly used optimization algorithms. In this paper, we proved the global linear convergence on a wide range of algorithms when they are applied to some non-strongly convex problems. In particular, we are the first to prove $O(\log(1/\epsilon))$ time complexity of cyclic coordinate descent methods on dual problems of support vector classification and regression.

Keywords: convergence rate, convex optimization, iteration complexity, feasible descent methods

1. Introduction

We consider the following convex optimization problem

$$\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}), \quad \text{where } f(\boldsymbol{x}) \equiv g(E\boldsymbol{x}) + \boldsymbol{b}^\top \boldsymbol{x}, \tag{1}$$

where $g(\boldsymbol{t})$ is a strongly convex function with Lipschitz continuous gradient, E is a constant matrix, and $\mathcal{X}$ is a polyhedral set. Many popular machine learning problems are of this type. For example, given training label-instance pairs $(y_i, \boldsymbol{z}_i),\ i = 1, \ldots, l$, the dual form of L1-loss linear SVM (Boser et al., 1992) is[1]

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} - \mathbf{1}^T \boldsymbol{\alpha} \\ \text{subject to} \quad & \boldsymbol{w} = E\boldsymbol{\alpha},\ 0 \le \alpha_i \le C,\ i = 1, \ldots, l, \end{aligned} \tag{2}$$

where $E = \left[y_1 \boldsymbol{z}_1, \ldots, y_l \boldsymbol{z}_l\right]$, $\mathbf{1}$ is the vector of ones, and C is a given upper bound. Although $\boldsymbol{w}^\top \boldsymbol{w}/2$ is strongly convex in $\boldsymbol{w}$, the objective function of (2) may not be strongly convex in $\boldsymbol{\alpha}$. Common optimization approaches for these machine learning problems include cyclic coordinate descent and others. Unfortunately, most existing results prove only local linear

1. Note that we omit the bias term in the SVM formulation.

convergence, so the number of total iterations cannot be calculated. One of the main difficulties is that $f(\boldsymbol{x})$ may not be strongly convex. In this work, we establish the global linear convergence for a wide range of algorithms for problem (1). In particular, we are the first to prove that the popularly used cyclic coordinate descent methods for dual SVM problems converge linearly since the beginning. Many researchers have stated the importance of such convergence-rate analysis. For example, Nesterov (2012) said that it is *"almost impossible to estimate the rate of convergence"* for general cases. Saha and Tewari (2013) also agreed that *"little is known about the non-asymptotic convergence"* for cyclic coordinate descent methods and they felt *"this gap in the literature needs to be filled urgently."*

Luo and Tseng (1992a) are among the first to establish the asymptotic linear convergence to a non-strongly convex problem related to (1). If $\mathcal{X}$ is a box (possibly unbounded) and a cyclic coordinate descent method is applied, they proved ϵ-optimality in $O(r_0 + \log(1/\epsilon))$ time, where r_0 is an unknown number. Subsequently, Luo and Tseng (1993) considered a class of feasible descent methods that broadly covers coordinate descent and gradient projection methods. For problems including (1), they proved the asymptotic linear convergence. The key concept in their analysis is a local error bound, which states how close the current solution is to the solution set compared with the norm of projected gradient $\nabla^+ f(\boldsymbol{x})$.

$$\min_{\boldsymbol{x}^* \in \mathcal{X}^*} \|\boldsymbol{x}^r - \boldsymbol{x}^*\| \leq \kappa \|\nabla^+ f(\boldsymbol{x}^r)\|, \quad \forall r \geq r_0, \tag{3}$$

where r_0 is the above-mentioned unknown iteration index, $\mathcal{X}^*$ is the solution set of problem (1), κ is a positive constant, and $\boldsymbol{x}^r$ is the solution produced after the r-th iteration. Because r_0 is unknown, we call (3) a local error bound, which only holds near the solution set. Local error bounds have been used in other works for convergence analysis such as Luo and Tseng (1992b). If $r_0 = 0$, we call (3) a global error bound from the beginning, and it may help to obtain a global convergence rate. If $f(\boldsymbol{x})$ is strongly convex and $\mathcal{X}$ is a polyhedral set, a global error bound has been established by Pang (1987, Theorem 3.1). One of the main contributions of our work is to prove a global error bound of the possibly non-strongly convex problem (1). Then we are able to establish the global linear convergence and $O(\log(1/\epsilon))$ time complexity for the feasible descent methods.

We briefly discuss some related works, which differ from ours in certain aspects. Chang et al. (2008) applied an (inexact) cyclic coordinate descent method for the primal problem of L2-loss SVM. Because the objective function is strongly convex, they are able to prove the linear convergence since the first iteration. Further, Beck and Tetruashvili (2013) established global linear convergence for block coordinate gradient descent methods on general smooth and strongly convex objective functions. Tseng and Yun (2009) applied a greedy version of block coordinate descent methods on the non-smooth separable problems covering the dual form of SVM. However, they proved only asymptotic linear convergence and $O(1/\epsilon)$ complexity. Moreover, for large-scale linear SVM (i.e., kernels are not used), cyclic rather than greedy coordinate descent methods are more commonly used in practice.[2] Wright (2012) considered the same non-smooth separable problems in Tseng and Yun (2009) and introduced a reduced-Newton acceleration that has asymptotic quadratic convergence.

2. It is now well known that greedy coordinate descent methods such as SMO (Platt, 1998) are less suitable for linear SVM; see some detailed discussion in Hsieh et al. (2008, Section 4.1).

For L1-regularized problems, Saha and Tewari (2013) proved $O(1/\epsilon)$ complexity for cyclic coordinate descent methods under a restrictive isotonic assumption.

Although this work focuses on deterministic algorithms, we briefly review past studies on stochastic (randomized) methods. An interesting fact is that there are more studies on the complexity of randomized rather than deterministic coordinate descent methods. Shalev-Shwartz and Tewari (2009) considered L1-regularized problems, and their stochastic coordinate descent method converges in $O(1/\epsilon)$ iterations in expectation. Nesterov (2012) extended the settings to general convex objective functions and improved the iteration bound to $O(1/\sqrt{\epsilon})$ by proposing an accelerated method. For strongly convex function, he proved that the randomized coordinate descent method converges linearly in expectation. Shalev-Shwartz and Zhang (2013a) provided a sub-linear convergence rate for a stochastic coordinate ascent method, but they focused on the duality gap. Their work is interesting because it bounds the primal objective values. Shalev-Shwartz and Zhang (2013b) refined the sub-linear convergence to be $O(\min(1/\epsilon, 1/\sqrt{\epsilon}))$. Richtárik and Takáč (2011) studied randomized block coordinate descent methods for non-smooth convex problems and had sub-linear convergence on non-strongly convex functions. If the objective function is strongly convex and separable, they obtained linear convergence. Tappenden et al. (2013) extended the methods to inexact settings and had similar convergence rates to those in Richtárik and Takáč (2011).

Our main contribution is a global error bound for the non-strongly convex problem (1), which ensures the global linear convergence of feasible descent methods. The main theorems are presented in Section 2, followed by examples in Section 3. The global error bound is discussed in Section 4, and the proof of global linear convergence of feasible descent methods is given in Section 5. We conclude in Section 6 while leaving properties of projected gradients in Appendix A.

2. Main Results

Consider the general convex optimization problem

$$\min_{\boldsymbol{x}\in\mathcal{X}} \quad f(\boldsymbol{x}), \tag{4}$$

where $f(\boldsymbol{x})$ is proper convex and $\mathcal{X}$ is nonempty, closed, and convex. We will prove global linear convergence for a class of optimization algorithms if problem (4) satisfies one of the following assumptions.

Assumption 1 *$f(\boldsymbol{x})$ is σ strongly convex and its gradient is ρ Lipschitz continuous. That is, there are constants $\sigma > 0$ and ρ such that*

$$\sigma\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2 \leq (\nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x}_2))^\top(\boldsymbol{x}_1 - \boldsymbol{x}_2), \quad \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$$

and

$$\|\nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x}_2)\| \leq \rho\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|, \quad \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}.$$

Assumption 2 *$\mathcal{X} = \{\boldsymbol{x} \mid A\boldsymbol{x} \leq \boldsymbol{d}\}$ is a polyhedral set, the optimal solution set $\mathcal{X}^*$ is non-empty, and*

$$f(\boldsymbol{x}) = g(E\boldsymbol{x}) + \boldsymbol{b}^\top\boldsymbol{x}, \tag{5}$$

where $g(\boldsymbol{t})$ is σ_g strongly convex and $\nabla f(\boldsymbol{x})$ is ρ Lipschitz continuous. This assumption corresponds to problem (1) *that motivates this work.*

The optimal set $\mathcal{X}^*$ under Assumption 1 is non-empty following Weierstrass extreme value theorem.[3] Subsequently, we make several definitions before presenting the main theorem.

Definition 3 (Convex Projection Operator)

$$[\boldsymbol{x}]^+_{\mathcal{X}} \equiv \arg\min_{\boldsymbol{y}\in\mathcal{X}} \|\boldsymbol{x}-\boldsymbol{y}\|.$$

From Weierstrass extreme value theorem and the strong convexity of $\|\boldsymbol{x}-\boldsymbol{y}\|^2$ to $\boldsymbol{y}$, the unique $[\boldsymbol{x}]^+_{\mathcal{X}}$ exists for any $\mathcal{X}$ that is closed, convex, and non-empty.

Definition 4 (Nearest Optimal Solution)

$$\bar{\boldsymbol{x}} \equiv [\boldsymbol{x}]^+_{\mathcal{X}^*}.$$

With this definition, $\min_{\boldsymbol{x}^*\in\mathcal{X}^*} \|\boldsymbol{x}-\boldsymbol{x}^*\|$ could be simplified to $\|\boldsymbol{x}-\bar{\boldsymbol{x}}\|$.

Definition 5 (Projected Gradient)

$$\nabla^+ f(\boldsymbol{x}) \equiv \boldsymbol{x} - [\boldsymbol{x}-\nabla f(\boldsymbol{x})]^+_{\mathcal{X}}.$$

As shown in Lemma 24, the projected gradient is zero if and only if $\boldsymbol{x}$ is an optimal solution. Therefore, it can be used to check the optimality. Further, we can employ the projected gradient to define an error bound, which measures the distance between $\boldsymbol{x}$ and the optimal set; see the following definition.

Definition 6 *An optimization problem admits a* ***global error bound*** *if there is a constant κ such that*

$$\|\boldsymbol{x}-\bar{\boldsymbol{x}}\| \le \kappa\|\nabla^+ f(\boldsymbol{x})\|, \quad \forall \boldsymbol{x}\in\mathcal{X}. \tag{6}$$

A relaxed condition called ***global error bound from the beginning*** *if the above inequality holds only for $\boldsymbol{x}$ satisfying*

$$\boldsymbol{x}\in\mathcal{X} \;\; \textit{and} \;\; f(\boldsymbol{x}) - f(\bar{\boldsymbol{x}}) \le M,$$

where M is a constant. Usually, we have

$$M \equiv f(\boldsymbol{x}^0) - f^*,$$

where $\boldsymbol{x}^0$ is the start point of an optimization algorithm and f^* is the optimal function value. Therefore, we called this as a bound "from the beginning."

3. The strong convexity in Assumption 1 implies that the sublevel set is bounded (Vial, 1983). Then Weierstrass extreme value theorem can be applied.

The global error bound is a property of the optimization problem and is independent from the algorithms. If a bound holds,[4] then using Lemmas 23, 24, and (6) we can obtain

$$\frac{1}{2+\rho}\|\nabla^+ f(\boldsymbol{x})\| \leq \|\boldsymbol{x}-\bar{\boldsymbol{x}}\| \leq \kappa\|\nabla^+ f(\boldsymbol{x})\|, \quad \forall \boldsymbol{x} \in \mathcal{X}.$$

This property indicates that $\|\nabla^+ f(\boldsymbol{x})\|$ is useful to estimate the distance to the optimum. We will show that a global error bound enables the proof of global linear convergence of some optimization algorithms. The bound under Assumption 1, which requires strong convexity, was already proved in Pang (1987) with

$$\kappa = \frac{1+\rho}{\sigma}.$$

However, for problems under Assumption 2 such as the dual form of L1-loss SVM, the objective function is not strongly convex, so a new error bound is required. We prove the bound in Section 4 with

$$\kappa = \theta^2(1+\rho)\left(\frac{1+2\|\nabla g(\boldsymbol{t}^*)\|^2}{\sigma_g} + 4M\right) + 2\theta\|\nabla f(\bar{\boldsymbol{x}})\|, \tag{7}$$

where $\boldsymbol{t}^*$ is a constant vector that equals $E\boldsymbol{x}^*$, $\forall \boldsymbol{x}^* \in \mathcal{X}^*$ and θ is the constant from Hoffman's bound (Hoffman, 1952; Li, 1994).

$$\theta \equiv \sup_{\boldsymbol{u},\boldsymbol{v}} \left\{ \left\| \begin{matrix} \boldsymbol{u} \\ \boldsymbol{v} \end{matrix} \right\| \;\middle|\; \begin{matrix} \|A^\top \boldsymbol{u} + \left(\begin{smallmatrix} E \\ \boldsymbol{b}^\top \end{smallmatrix}\right)^\top \boldsymbol{v}\| = 1,\ \boldsymbol{u} \geq 0. \\ \text{The corresponding rows of } A,\ E \text{ to } \boldsymbol{u},\ \boldsymbol{v}\text{'s} \\ \text{non-zero elements are linearly independent.} \end{matrix} \right\}.$$

Specially, when $\boldsymbol{b} = \boldsymbol{0}$ or $\mathcal{X} = \mathbb{R}^l$, the constant could be simplified to

$$\kappa = \theta^2 \frac{1+\rho}{\sigma_g}. \tag{8}$$

Now we define a class of optimization algorithms called the feasible descent methods for solving (4).

Definition 7 (Feasible Descent Methods) *A sequence $\{\boldsymbol{x}^r\}$ is generated by a feasible descent method if for every iteration index r, $\{\boldsymbol{x}^r\}$ satisfies*

$$\boldsymbol{x}^{r+1} = [\boldsymbol{x}^r - \omega_r \nabla f(\boldsymbol{x}^r) + \boldsymbol{e}^r]^+_{\mathcal{X}}, \tag{9}$$

$$\|\boldsymbol{e}^r\| \leq \beta\|\boldsymbol{x}^r - \boldsymbol{x}^{r+1}\|, \tag{10}$$

$$f(\boldsymbol{x}^r) - f(\boldsymbol{x}^{r+1}) \geq \gamma\|\boldsymbol{x}^r - \boldsymbol{x}^{r+1}\|^2, \tag{11}$$

where $\inf_r \omega_r > 0$, $\beta > 0$, *and* $\gamma > 0$.

4. Note that not all problems have a global error bound. An example is $\min_{x\in\mathbb{R}} x^4$.

The framework of feasible descent methods broadly covers many algorithms that use the first-order information. For example, the projected gradient descent, the cyclic coordinate descent, the proximal point minimization, the extragradient descent, and matrix splitting algorithms are all feasible descent methods (Luo and Tseng, 1993). With the global error bound under Assumption 1 or Assumption 2, in the following theorem we prove the global linear convergence for all algorithms that fit into the feasible descent methods.

Theorem 8 (Global Linear Convergence) *If an optimization problem satisfies Assumption 1 or 2, then any feasible descent method on it has global linear convergence. To be specific, the method converges Q-linearly with*

$$f(\boldsymbol{x}^{r+1}) - f^* \leq \frac{\phi}{\phi+\gamma}(f(\boldsymbol{x}^r) - f^*), \quad \forall r \geq 0,$$

where κ is the error bound constant in (6)*,*

$$\phi = (\rho + \frac{1+\beta}{\underline{\omega}})(1 + \kappa\frac{1+\beta}{\underline{\omega}}), \quad \text{and} \quad \underline{\omega} \equiv \min(1, \inf_r \omega_r).$$

This theorem enables global linear convergence in many machine learning problems. The proof is given in Section 5. In Section 3, we discuss examples on cyclic coordinate descent methods.

3. Examples: Cyclic Coordinate Descent Methods

Cyclic coordinate descent methods are now widely used for machine learning problems because of its efficiency and simplicity (solving a one-variable sub-problem at a time). Luo and Tseng (1992a) proved the asymptotic linear convergence if sub-problems are solved exactly, and here we further show the global linear convergence.

3.1 Exact Cyclic Coordinate Descent Methods for Dual SVM Problems

In the following algorithm, each one-variable sub-problem is exactly solved.

Definition 9 *A cyclic coordinate descent method on a box $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_l$ is defined by the update rule*

$$x_i^{r+1} = \arg\min_{x_i \in \mathcal{X}_i} f(x_1^{r+1}, \ldots, x_{i-1}^{r+1}, x_i, x_{i+1}^r, \ldots, x_l^r), \quad \textit{for } i = 1, \ldots, l, \tag{12}$$

where $\mathcal{X}_i$ is the region under box constraints for coordinate i.

The following lemma shows that coordinate descent methods are special cases of the feasible descent methods.

Lemma 10 *The cyclic coordinate descent method is a feasible descent method with*

$$\omega_r = 1, \ \forall r, \quad \beta = 1 + \rho\sqrt{l},$$

and

$$\gamma = \begin{cases} \frac{\sigma}{2} & \textit{if Assumption 1 holds,} \\ \frac{1}{2}\min_i \|E_i\|^2 & \textit{if Assumption 2 holds with } \|E_i\| > 0, \ \forall i, \end{cases}$$

where E_i is the ith column of E.

Proof This lemma can be directly obtained using Proposition 3.4 of Luo and Tseng (1993). Our assumptions correspond to cases (a) and (c) in Theorem 2.1 of Luo and Tseng (1993), which fulfill conditions needed by their Proposition 3.4. ■

For faster convergence, we may randomly permute all variables before each cycle of updating them (e.g., Hsieh et al., 2008). This setting does not affect the proof of Lemma 10.

Theorem 8 and Lemma 10 immediately imply the following corollary.

Corollary 11 *The cyclic coordinate descent methods have global linear convergence if Assumption 1 is satisfied or Assumption 2 is satisfied with* $\|E_i\| > 0,\ \forall i$.

Next, we analyze the cyclic coordinate descent method to solve dual SVM problems. The method can be traced back to Hildreth (1957) for quadratic programming problems and has recently been widely used following the work by Hsieh et al. (2008). For L1-loss SVM, we have shown in (2) that the objective function can be written in the form of (1) by a strongly convex function $g(\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{w}/2$ and $E_i = y_i \boldsymbol{z}_i$ for all label-instance pair $(y_i, \boldsymbol{z}_i)$. Hsieh et al. (2008) pointed out that $\|E_i\| = 0$ implies the optimal α_i^* is C, which can be obtained at the first iteration and is never changed. Therefore, we need not consider such variables at all. With all conditions satisfied, Corollary 11 implies that cyclic coordinate descent method for dual L1-loss SVM has global linear convergence. For dual L2-loss SVM, the objective function is

$$\frac{1}{2}\boldsymbol{\alpha}^\top Q \boldsymbol{\alpha} - \mathbf{1}^\top \boldsymbol{\alpha} + \frac{1}{2C}\boldsymbol{\alpha}^\top \boldsymbol{\alpha}, \tag{13}$$

where $Q_{t,j} = y_t y_j \boldsymbol{z}_t^\top \boldsymbol{z}_j, \forall 1 \le t, j \le l$ and $\mathbf{1}$ is the vector of ones. Eq. (13) is strongly convex and its gradient is Lipschitz continuous, so Assumption 1 and Corollary 11 imply the global linear convergence.

We move on to check the dual problems of support vector regression (SVR). Given value-instance pairs $(y_i, \boldsymbol{z}_i),\ i = 1, \ldots, l$, the dual form of L1-loss m-insensitive SVR (Vapnik, 1995) is

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^\top \begin{bmatrix} Q & -Q \\ -Q & Q \end{bmatrix} \boldsymbol{\alpha} + \begin{bmatrix} m\mathbf{1} - \boldsymbol{y} \\ m\mathbf{1} + \boldsymbol{y} \end{bmatrix}^\top \boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \le \alpha_i \le C,\ i = 1, \ldots, 2l, \end{aligned} \tag{14}$$

where $Q_{t,j} = \boldsymbol{z}_t^\top \boldsymbol{z}_j, \forall 1 \le t, j \le l$, and m and C are given parameters. Similar to the case of classification, we can also perform cyclic coordinate descent methods; see Ho and Lin (2012, Section 3.2). Note that Assumption 2 must be used here because for any Q, the Hessian in (14) is only positive semi-definite rather than positive definite. In contrast, for classification, if Q is positive definite, the objective function in (2) is strongly convex and Assumption 1 can be applied. To represent (14) in the form of (1), let

$$E_i = \boldsymbol{z}_i,\ \ i = 1, \ldots, l \ \text{ and } \ E_i = -\boldsymbol{z}_i,\ \ i = l+1, \ldots, 2l.$$

Then $g(\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{w}/2$ with $\boldsymbol{w} = E\boldsymbol{\alpha}$ is a strongly convex function to $\boldsymbol{w}$. Similar to the situation in classification, if $\|E_i\| = 0$, then the optimal α_i^* is bounded and can be obtained at the first iteration. Without considering these variables, Corollary 11 implies the global linear convergence.

3.2 Inexact Cyclic Coordinate Descent Methods for Primal SVM Problems

In some situations the sub-problems (12) of cyclic coordinate descent methods cannot be easily solved. For example, in Chang et al. (2008) to solve the primal form of L2-loss SVM,

$$\min_{\boldsymbol{w}} \ f(\boldsymbol{w}), \quad \text{where } f(\boldsymbol{w}) \equiv \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C\sum_{i=1}^{l} \max(1 - y_i \boldsymbol{w}^\top \boldsymbol{z}_i, 0)^2, \tag{15}$$

each sub-problem does not have a closed-form solution, and they approximately solve the sub-problem until a sufficient decrease condition is satisfied. They have established the global linear convergence, but we further show that their method can be included in our framework.

To see that Chang et al. (2008)'s method is a feasible descent method, it is sufficient to prove that (9)-(11) hold. First, we notice that their sufficient decrease condition for updating each variable can be accumulated. Thus, for one cycle of updating all variables, we have

$$f(\boldsymbol{w}^r) - f(\boldsymbol{w}^{r+1}) \geq \gamma \|\boldsymbol{w}^r - \boldsymbol{w}^{r+1}\|^2,$$

where $\gamma > 0$ is a constant. Next, because (15) is unconstrained, if $\boldsymbol{z}_i \in \mathbb{R}^n, \forall i$, we can make

$$\mathcal{X} = \mathbb{R}^n \text{ and } \boldsymbol{e}^r = \boldsymbol{w}^{r+1} - \boldsymbol{w}^r + \nabla f(\boldsymbol{w}^r)$$

such that

$$\boldsymbol{w}^{r+1} = [\boldsymbol{w}^r - \nabla f(\boldsymbol{w}^r) + \boldsymbol{e}^r]^+_{\mathcal{X}}.$$

Finally, from Appendix A.3 of Chang et al. (2008),

$$\|\boldsymbol{e}^r\| \leq \|\boldsymbol{w}^r - \boldsymbol{w}^{r+1}\| + \|\nabla f(\boldsymbol{w}^r)\| \leq \beta \|\boldsymbol{w}^r - \boldsymbol{w}^{r+1}\|,$$

where $\beta > 0$ is a constant. Therefore, all conditions (9)-(11) hold. Note that (15) is strongly convex because of the $\boldsymbol{w}^\top \boldsymbol{w}$ term and $\nabla f(\boldsymbol{w})$ is Lipschitz continuous from (Lin et al., 2008, Section 6.1), so Assumption 1 is satisfied. With Theorem 8, the method by Chang et al. (2008) has global linear convergence.

3.3 Gauss-Seidel Methods for Solving Linear Systems

Gauss-Seidel (Seidel, 1874) is a classic iterative method to solve a linear system

$$Q\boldsymbol{\alpha} = \boldsymbol{b}. \tag{16}$$

Gauss-Seidel iterations take the following form.

$$\alpha_i^{r+1} = \frac{b_i - \sum_{j=1}^{i-1} Q_{ij}\alpha_j^{r+1} - \sum_{j=i+1}^{l} Q_{ij}\alpha_j^r}{Q_{ii}}. \tag{17}$$

If Q is symmetric positive semi-definite and (16) has at least one solution, then the following optimization problem

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^l} \quad \frac{1}{2}\boldsymbol{\alpha}^\top Q\boldsymbol{\alpha} - \boldsymbol{b}^\top \boldsymbol{\alpha} \tag{18}$$

has the same solution set as (16). Further, α_i^{r+1} in (17) is the solution of minimizing (18) over α_i while fixing $\alpha_1^{r+1}, \ldots, \alpha_{i-1}^{r+1}, \alpha_{i+1}^r, \ldots, \alpha_l^r$. Therefore, Gauss-Seidel method is a special case of coordinate descent methods.

Clearly, we need $Q_{ii} > 0, \forall i$ so that (17) is well defined. This condition also implies that

$$Q = E^\top E, \text{ where } E \text{ has no zero column.} \tag{19}$$

Otherwise, $\|E_i\| = 0$ leads to $Q_{ii} = 0$ so the $Q_{ii} > 0$ assumption is violated. Note that the $E^\top E$ factorization exists because Q is symmetric positive semi-definite. Using (19) and Lemma 10, Gauss-Seidel method is a feasible descent method. By Assumption 2 and our main Theorem 8, we have the following convergence result.

Corollary 12 *If*

1. *Q is symmetric positive semi-definite and $Q_{ii} > 0, \forall i$, and*
2. *The linear system (16) has at least a solution,*

then the Gauss-Seidel method has global linear convergence.

This corollary covers some well-known results of the Gauss-Seidel method, which were previously proved by other ways. For example, in most numerical linear algebra textbooks (e.g., Golub and Van Loan, 1996), it is proved that if Q is strictly diagonally dominant (i.e., $Q_{ii} > \sum_{j \neq i} |Q_{ij}|, \ \forall i$), then the Gauss-Seidel method converges linearly. We show in Lemma 28 that a strictly diagonally dominant matrix is positive definite, so Corollary 12 immediately implies global linear convergence.

3.4 Quantity of the Convergence Rate

To demonstrate the relationship between problem parameters (e.g., number of instances and features) and the convergence rate constants, we analyze the constants κ and ϕ for two problems. The first example is the exact cyclic coordinate descent method for the dual problem (2) of L1-loss SVM. For simplicity, we assume $\|E_i\| = 1, \ \forall i$, where E_i denotes the ith column of E. We have

$$\sigma_g = 1 \tag{20}$$

by $g(\boldsymbol{t}) = \boldsymbol{t}^\top \boldsymbol{t}/2$. Observe the following primal formulation of L1-loss SVM.

$$\min_{\boldsymbol{w}} \quad P(\boldsymbol{w}), \ \text{ where } \ P(\boldsymbol{w}) \equiv \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C \sum_{i=1}^{l} \max(1 - y_i \boldsymbol{w}^\top \boldsymbol{z}_i, 0).$$

Let $\boldsymbol{w}^*$ and $\boldsymbol{\alpha}^*$ be any optimal solution of the primal and the dual problems, respectively. By KKT optimality condition, we have $\boldsymbol{w}^* = E\boldsymbol{\alpha}^*$. Consider $\boldsymbol{\alpha}^0 = \mathbf{0}$ as the initial feasible solution. With the duality and the strictly decreasing property of $\{f(\boldsymbol{\alpha}^r)\}$,

$$f(\boldsymbol{\alpha}^r) - f(\boldsymbol{\alpha}^*) \leq f(\mathbf{0}) - f(\boldsymbol{\alpha}^*) = f(\mathbf{0}) + P(\boldsymbol{w}^*) \leq f(\mathbf{0}) + P(\mathbf{0}) \leq 0 + Cl \equiv M. \tag{21}$$

Besides,

$$\frac{1}{2}\boldsymbol{w}^{*\top}\boldsymbol{w}^* \leq P(\boldsymbol{w}^*) \leq P(\mathbf{0}) \leq Cl \ \text{ implies } \ \|\boldsymbol{w}^*\| = \|E\boldsymbol{\alpha}^*\| \leq \sqrt{2Cl}. \tag{22}$$

From (22),

$$\|\nabla f(\bar{\boldsymbol{\alpha}})\| \le \|E\|\|E\boldsymbol{\alpha}^*\| + \|\mathbf{1}\| \le \sqrt{\Sigma_i \|E_i\|^2}\|E\boldsymbol{\alpha}^*\| + \|\mathbf{1}\| \le \sqrt{2Cl} + \sqrt{l}. \tag{23}$$

To conclude, by (7), (20), (21), (22), (23), and $\nabla g(\boldsymbol{w}^*) = \boldsymbol{w}^*$,

$$\begin{aligned}\kappa &= \theta^2(1+\rho)\left(\frac{1+2\|\nabla g(\boldsymbol{w}^*)\|^2}{\sigma_g} + 4M\right) + 2\theta\|\nabla f(\bar{\boldsymbol{\alpha}})\| \\ &\le \theta^2(1+\rho)((1+4Cl)+4Cl) + 2\theta(\sqrt{2Cl}+\sqrt{l}) \\ &= O(\rho\theta^2 Cl).\end{aligned}$$

Now we examine the rate ϕ for linear convergence. From Theorem 8, we have

$$\begin{aligned}\phi &= \left(\rho + \frac{1+\beta}{\underline{\omega}}\right)\left(1 + \kappa\frac{1+\beta}{\underline{\omega}}\right) \\ &= (\rho + 2 + \rho\sqrt{l})(1 + \kappa(2+\rho\sqrt{l})) \\ &= O(\rho^3\theta^2 Cl^2),\end{aligned}$$

where

$$\underline{\omega} = 1, \;\; \beta = 1 + \rho\sqrt{l}, \;\; \gamma = \frac{1}{2} \tag{24}$$

are from Lemma 10 and the assumption that $\|E_i\| = 1, \; \forall i$. To conclude, we have $\kappa = O(\rho\theta^2 Cl)$ and $\phi = O(\rho^3\theta^2 Cl^2)$ for the exact cyclic coordinate descent method for the dual problem of L1-loss SVM.

Next we consider the Gauss-Seidel method for solving linear systems in Section 3.3 by assuming $\|Q\| = 1$ and $Q_{ii} > 0, \; \forall i$, where $\|Q\|$ denotes the spectral norm of Q. Similar to (20), we have $\sigma_g = 1$ by $g(\boldsymbol{t}) = \boldsymbol{t}^\top\boldsymbol{t}/2$. Further, $\rho = 1$ from

$$\|\nabla f(\boldsymbol{\alpha}_1) - \nabla f(\boldsymbol{\alpha}_2)\| \le \|Q\|\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\| = \|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|.$$

Because the optimization problem is unconstrained, by (8) we have

$$\kappa = \theta^2\frac{1+\rho}{\sigma_g} = 2\theta^2, \tag{25}$$

where θ is defined as

$$\theta \equiv \sup_{\boldsymbol{v}} \left\{ \|\boldsymbol{v}\| \;\middle|\; \begin{array}{l} \|E^\top\boldsymbol{v}\| = 1. \\ \text{The corresponding rows of } E \text{ to } \boldsymbol{v}\text{'s} \\ \text{non-zero elements are linearly independent.} \end{array} \right\}, \tag{26}$$

and E is from the factorization of Q in (19). Let $\boldsymbol{\nu}$ be the normalized eigen-vector of Q with the smallest non-zero eigen-value $\sigma_{\text{min-nnz}}$. We can observe that

$$\text{The solution } \boldsymbol{v} \text{ in (26) is } \frac{\boldsymbol{\nu}}{\sqrt{\sigma_{\text{min-nnz}}}} \text{ and } \theta^2 = \frac{1}{\sigma_{\text{min-nnz}}}. \tag{27}$$

From Lemma 10, $\underline{\omega}$, β, and γ of the Gauss-Seidel method are the same as (24). Thus, Theorem 8, (24), (25), and (27) give the convergence rate constant

$$\phi = (3+\sqrt{l})(1+\kappa(2+\sqrt{l})) = (3+\sqrt{l})(1+\frac{4+2\sqrt{l}}{\sigma_{\min\text{-nnz}}}). \tag{28}$$

With (24), (28), and Theorem 8, the Gauss-Seidel method on solving linear systems has linear convergence with

$$f(\boldsymbol{\alpha}^{r+1}) - f^* \leq (1 - \frac{\sigma_{\min\text{-nnz}}}{4(6+5\sqrt{l}+l)+(7+2\sqrt{l})\sigma_{\min\text{-nnz}}})(f(\boldsymbol{\alpha}^r) - f^*), \quad \forall r \geq 0.$$

We discuss some related results. A similar rate of linear convergence appears in Beck and Tetruashvili (2013). They assumed f is $\sigma_{\min}$ strongly convex and the optimization problem is unconstrained. By considering a block coordinate descent method with a conservative rule of selecting the step size, they showed

$$f(\boldsymbol{\alpha}^{r+1}) - f^* \leq (1 - \frac{\sigma_{\min}}{2(1+l)})(f(\boldsymbol{\alpha}^r) - f^*), \quad \forall r \geq 0.$$

Our obtained rate is comparable, but is more general to cover singular Q.

4. Proofs of Global Error Bounds

In this section, we prove the global error bound (6) under Assumptions 1 or 2. The following theorem proves the global error bound under Assumption 1.

Theorem 13 (Pang 1987, Theorem 3.1) *Under Assumption 1,*

$$\|\boldsymbol{x} - \bar{\boldsymbol{x}}\| \leq \kappa \|\nabla^+ f(\boldsymbol{x})\|, \quad \forall \boldsymbol{x} \in \mathcal{X},$$

where $\kappa = (1+\rho)/\sigma$.

Proof Because $f(\boldsymbol{x})$ is strongly convex, $\mathcal{X}^*$ has only one element $\bar{\boldsymbol{x}}$. From Lemmas 22 and 24, the result holds immediately. ■

The rest of this section focuses on proving a global error bound under Assumption 2. We start by sketching the proof. First, observe that the optimal set is a polyhedron by Lemma 14. Then $\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|$ is identical to the distance of $\boldsymbol{x}$ to the polyhedron. A known technique to bound the distance between $\boldsymbol{x}$ and a polyhedron is Hoffman's bound (Hoffman, 1952). Because the original work uses L1-norm, we provide in Lemma 15 a special version of Li (1994) that uses L2-norm. With the feasibility of $\boldsymbol{x}$, there is

$$\|\boldsymbol{x} - \bar{\boldsymbol{x}}\| \leq \theta\left(A, \begin{pmatrix} E \\ \boldsymbol{b}^\top \end{pmatrix}\right) \left\| \begin{matrix} E(\boldsymbol{x} - \bar{\boldsymbol{x}}) \\ \boldsymbol{b}^\top(\boldsymbol{x} - \bar{\boldsymbol{x}}) \end{matrix} \right\|,$$

where $\theta\left(A, \begin{pmatrix} E \\ \boldsymbol{b}^\top \end{pmatrix}\right)$ is a constant related to A, E, and $\boldsymbol{b}$. Subsequently, we bound $\|E(\boldsymbol{x}-\bar{\boldsymbol{x}})\|^2$ and $(\boldsymbol{b}^\top(\boldsymbol{x} - \bar{\boldsymbol{x}}))^2$ in Lemmas 16 and 17 by values consisting of $\|\nabla^+ f(\boldsymbol{x})\|$ and $\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|$. Such bounds are obtained using properties of the optimization problem such as the strong convexity of $g(\cdot)$. Finally, we obtain a quadratic inequality involving $\|\nabla^+ f(\boldsymbol{x})\|$ and $\|\boldsymbol{x}-\bar{\boldsymbol{x}}\|$, which eventually leads to a global error bound under Assumption 2.

We begin the formal proof by expressing the optimal set as a polyhedron.

Lemma 14 (Optimal Condition) *Under Assumption 2, there are unique* $\boldsymbol{t}^*$ *and* s^* *such that* $\forall \boldsymbol{x}^* \in \mathcal{X}^*$,

$$E\boldsymbol{x}^* = \boldsymbol{t}^*, \;\; \boldsymbol{b}^\top \boldsymbol{x}^* = s^*, \;\; \text{and} \;\; A\boldsymbol{x}^* \leq \boldsymbol{d}. \tag{29}$$

Note that A *and* $\boldsymbol{d}$ *are the constants for generating the feasible set* $\mathcal{X} = \{\boldsymbol{x} \mid A\boldsymbol{x} \leq \boldsymbol{d}\}$. *Further,*

$$\boldsymbol{x}^* \text{ satisfies (29)} \Leftrightarrow \boldsymbol{x}^* \in \mathcal{X}^*. \tag{30}$$

Specially, when $\boldsymbol{b} = \boldsymbol{0}$ *or* $\mathcal{X} = \mathbb{R}^l$,[5]

$$E\boldsymbol{x}^* = \boldsymbol{t}^*, \; A\boldsymbol{x}^* \leq \boldsymbol{d} \Leftrightarrow \boldsymbol{x}^* \in \mathcal{X}^*. \tag{31}$$

Proof First, we prove (29). The proof is similar to Lemma 3.1 in Luo and Tseng (1992a). For any $\boldsymbol{x}_1^*, \boldsymbol{x}_2^* \in \mathcal{X}^*$, from the convexity of $f(\boldsymbol{x})$,

$$f((\boldsymbol{x}_1^* + \boldsymbol{x}_2^*)/2) = (f(\boldsymbol{x}_1^*) + f(\boldsymbol{x}_2^*))/2.$$

By the definition of $f(\boldsymbol{x})$ in Assumption 2, we have

$$g((E\boldsymbol{x}_1^* + E\boldsymbol{x}_2^*)/2) + \boldsymbol{b}^\top(\boldsymbol{x}_1^* + \boldsymbol{x}_2^*)/2 = (g(E\boldsymbol{x}_1^*) + g(E\boldsymbol{x}_2^*) + \boldsymbol{b}^\top(\boldsymbol{x}_1^* + \boldsymbol{x}_2^*))/2.$$

Cancel $\boldsymbol{b}^\top(\boldsymbol{x}_1^* + \boldsymbol{x}_2^*)/2$ from both sides. By the strong convexity of $g(\boldsymbol{t})$, we have $E\boldsymbol{x}_1^* = E\boldsymbol{x}_2^*$. Thus, $\boldsymbol{t}^* \equiv E\boldsymbol{x}^*$ is unique. Similarly, because $f(\boldsymbol{x}_1^*) = f(\boldsymbol{x}_2^*)$,

$$g(\boldsymbol{t}^*) + \boldsymbol{b}^\top \boldsymbol{x}_1^* = g(\boldsymbol{t}^*) + \boldsymbol{b}^\top \boldsymbol{x}_2^*.$$

Therefore, $s^* \equiv \boldsymbol{b}^\top \boldsymbol{x}^*$ is unique, and $A\boldsymbol{x}^* \leq \boldsymbol{d}$, $\forall \boldsymbol{x}^* \in \mathcal{X}^*$ holds naturally by $\mathcal{X}^* \subseteq \mathcal{X}$. Further,

$$f(\boldsymbol{x}^*) = g(\boldsymbol{t}^*) + s^*, \quad \forall \boldsymbol{x}^* \in \mathcal{X}^*. \tag{32}$$

The result in (29) immediately implies the ($\Leftarrow$) direction of (30). For the ($\Rightarrow$) direction, for any $\boldsymbol{x}^*$ satisfying

$$E\boldsymbol{x}^* = \boldsymbol{t}^*, \;\; \boldsymbol{b}^\top \boldsymbol{x}^* = s^*, \;\; A\boldsymbol{x}^* \leq \boldsymbol{d},$$

we have $f(\boldsymbol{x}^*) = g(\boldsymbol{t}^*) + s^*$. From (32), $\boldsymbol{x}^*$ is an optimal solution.

Now we examine the special cases. If $\boldsymbol{b} = \boldsymbol{0}$, we have $\boldsymbol{b}^\top \boldsymbol{x} = 0$, $\forall \boldsymbol{x} \in \mathcal{X}$. Therefore, (30) is reduced to (31). On the other hand, if $\mathcal{X} = \mathbb{R}^l$, the optimization problem is unconstrained. Thus,

$$\boldsymbol{x}^* \text{ is optimal} \Leftrightarrow \nabla f(\boldsymbol{x}^*) = \boldsymbol{0} = E^\top \nabla g(\boldsymbol{t}^*) + \boldsymbol{b}.$$

As a result, $E\boldsymbol{x}^* = \boldsymbol{t}^*$ is a necessary and sufficient optimality condition. ■

Because the optimal set is a polyhedron, we will apply the following Hoffman's bound in Lemma 18 to upper-bound the distance to the optimal set by the violation of the polyhedron's linear inequalities.

5. When $\mathcal{X} = \mathbb{R}^l$, we can take zero A and $\boldsymbol{d}$ for a trivial linear inequality.

Lemma 15 (Hoffman's Bound) *Let P be the non-negative orthant and consider a non-empty polyhedron*

$$\{\boldsymbol{x}^* \mid A\boldsymbol{x}^* \leq \boldsymbol{d},\ \ E\boldsymbol{x}^* = \boldsymbol{t}\}.$$

For any $\boldsymbol{x}$, there is a feasible point $\boldsymbol{x}^*$ such that

$$\|\boldsymbol{x} - \boldsymbol{x}^*\| \leq \theta(A, E) \left\| \begin{bmatrix} [A\boldsymbol{x} - \boldsymbol{d}]_P^+ \\ E\boldsymbol{x} - \boldsymbol{t} \end{bmatrix} \right\|, \tag{33}$$

where

$$\theta(A, E) \equiv \sup_{\boldsymbol{u},\boldsymbol{v}} \left\{ \left\| \begin{matrix} \boldsymbol{u} \\ \boldsymbol{v} \end{matrix} \right\| \ \middle| \ \begin{array}{l} \|A^\top \boldsymbol{u} + E^\top \boldsymbol{v}\| = 1,\ \boldsymbol{u} \geq \boldsymbol{0}. \\ \textit{The corresponding rows of } A,\ E \textit{ to } \boldsymbol{u},\ \boldsymbol{v}\textit{'s} \\ \textit{non-zero elements are linearly independent.} \end{array} \right\}. \tag{34}$$

Note that $\theta(A, E)$ is independent of $\boldsymbol{x}$.

The proof of the lemma is given in Appendix B. Before applying Hoffman's bound, we need some technical lemmas to bound $\|E\boldsymbol{x} - \boldsymbol{t}^*\|^2$ and $(\boldsymbol{b}^\top \boldsymbol{x} - s^*)^2$, which will appear on the right-hand side of Hoffman's bound for the polyhedron of the optimal set.

Lemma 16 *Under Assumption 2, we have constants ρ and σ_g such that*

$$\|E\boldsymbol{x} - \boldsymbol{t}^*\|^2 \leq \frac{1+\rho}{\sigma_g} \|\nabla^+ f(\boldsymbol{x})\| \|\boldsymbol{x} - \bar{\boldsymbol{x}}\|, \quad \forall \boldsymbol{x} \in \mathcal{X}.$$

Proof By $E\bar{\boldsymbol{x}} = \boldsymbol{t}^*$ from Lemma 14, the strong convexity of $g(\boldsymbol{t})$, and the definition of $f(\boldsymbol{x})$ in (5), there exists σ_g such that

$$\sigma_g \|E\boldsymbol{x} - \boldsymbol{t}^*\|^2 \leq (\nabla g(E\boldsymbol{x}) - \nabla g(E\bar{\boldsymbol{x}}))^\top (E\boldsymbol{x} - E\bar{\boldsymbol{x}}) = (\nabla f(\boldsymbol{x}) - \nabla f(\bar{\boldsymbol{x}}))^\top (\boldsymbol{x} - \bar{\boldsymbol{x}}).$$

By Lemma 21, the above inequality becomes

$$\sigma_g \|E\boldsymbol{x} - \boldsymbol{t}^*\|^2 \leq (1+\rho) \|\nabla^+ f(\boldsymbol{x}) - \nabla^+ f(\bar{\boldsymbol{x}})\| \|\boldsymbol{x} - \bar{\boldsymbol{x}}\|,$$

where ρ is the constant for the Lipschitz continuity of ∇f. Because $\bar{\boldsymbol{x}}$ is an optimal solution, $\nabla^+ f(\bar{\boldsymbol{x}}) = \boldsymbol{0}$ by Lemma 24. Thus, the result holds. ■

Next we bound $(\boldsymbol{b}^\top \boldsymbol{x} - s^*)^2$.

Lemma 17 *Under Assumption 2 and the condition*

$$f(\boldsymbol{x}) - f(\bar{\boldsymbol{x}}) \leq M, \tag{35}$$

there exists a constant $\rho > 0$ such that

$$\begin{aligned} &(\boldsymbol{b}^\top \boldsymbol{x} - s^*)^2 \\ &\leq 4(1+\rho) M \|\nabla^+ f(\boldsymbol{x})\| \|\boldsymbol{x} - \bar{\boldsymbol{x}}\| + 4\|\nabla f(\bar{\boldsymbol{x}})\|^2 \|\nabla^+ f(\boldsymbol{x})\|^2 + 2\|\nabla g(\boldsymbol{t}^*)\|^2 \|E\boldsymbol{x} - \boldsymbol{t}^*\|^2. \end{aligned}$$

Proof By $\boldsymbol{b}^\top \bar{\boldsymbol{x}} = s^*$ and $E\bar{\boldsymbol{x}} = \boldsymbol{t}^*$ from Lemma 14 and the definition of $f(\boldsymbol{x})$, we have

$$\boldsymbol{b}^\top \boldsymbol{x} - s^* = \nabla f(\bar{\boldsymbol{x}})^\top (\boldsymbol{x} - \bar{\boldsymbol{x}}) - \nabla g(\boldsymbol{t}^*)^\top (E\boldsymbol{x} - \boldsymbol{t}^*).$$

Square both sides of the equality. Then by $(a-b)^2 \leq 2a^2 + 2b^2$,

$$(\boldsymbol{b}^\top \boldsymbol{x} - s^*)^2 \leq 2(\nabla f(\bar{\boldsymbol{x}})^\top (\boldsymbol{x} - \bar{\boldsymbol{x}}))^2 + 2(\nabla g(\boldsymbol{t}^*)^\top (E\boldsymbol{x} - \boldsymbol{t}^*))^2. \tag{36}$$

Consider the right-hand side in (36). The second term can be bounded by $2\|\nabla g(\boldsymbol{t}^*)\|^2 \|E\boldsymbol{x} - \boldsymbol{t}^*\|^2$, and the first term is bounded using the inequalities

$$\begin{aligned}
\nabla f(\bar{\boldsymbol{x}})^\top (\boldsymbol{x} - \bar{\boldsymbol{x}}) &\leq \nabla f(\boldsymbol{x})^\top (\boldsymbol{x} - \bar{\boldsymbol{x}}) \\
&\leq \nabla^+ f(\boldsymbol{x})^\top (\boldsymbol{x} - \bar{\boldsymbol{x}} + \nabla f(\boldsymbol{x}) - \nabla^+ f(\boldsymbol{x})) \\
&\leq \nabla^+ f(\boldsymbol{x})^\top (\boldsymbol{x} - \bar{\boldsymbol{x}} + \nabla f(\boldsymbol{x}) - \nabla f(\bar{\boldsymbol{x}}) + \nabla f(\bar{\boldsymbol{x}})) \\
&\leq (1+\rho)\|\nabla^+ f(\boldsymbol{x})\|\|\boldsymbol{x} - \bar{\boldsymbol{x}}\| + \nabla^+ f(\boldsymbol{x})^\top \nabla f(\bar{\boldsymbol{x}}).
\end{aligned} \tag{37}$$

The first inequality is by convexity, the second is by Lemma 19,[6] the third is by $\|\nabla^+ f(\boldsymbol{x})\|^2 \geq 0$, and the last is by the Lipschitz continuity of ∇f. By the optimality of $\bar{\boldsymbol{x}}$,

$$\nabla f(\bar{\boldsymbol{x}})^\top ([\boldsymbol{x} - \nabla f(\boldsymbol{x})]^+_{\mathcal{X}} - \boldsymbol{x} + \boldsymbol{x} - \bar{\boldsymbol{x}}) \geq 0. \tag{38}$$

Thus, (38), the convexity of $f(\cdot)$, and (35) imply that

$$\nabla f(\bar{\boldsymbol{x}})^\top \nabla^+ f(\boldsymbol{x}) \leq \nabla f(\bar{\boldsymbol{x}})^\top (\boldsymbol{x} - \bar{\boldsymbol{x}}) \leq f(\boldsymbol{x}) - f(\bar{\boldsymbol{x}}) \leq M. \tag{39}$$

Let

$$a \equiv \nabla f(\bar{\boldsymbol{x}})^\top (\boldsymbol{x} - \bar{\boldsymbol{x}}),\ u \equiv (1+\rho)\|\nabla^+ f(\boldsymbol{x})\|\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|,\ v \equiv \nabla f(\bar{\boldsymbol{x}})^\top \nabla^+ f(\boldsymbol{x}).$$

Then we have

$$a \leq u + v \text{ from (37)},\ a \geq v \geq 0 \text{ from (39)}, \text{ and } u \geq 0.$$

Therefore, $a^2 \leq au + av \leq au + v(u+v) \leq 2au + 2v^2$, and

$$\begin{aligned}
&(\nabla f(\bar{\boldsymbol{x}})^\top (\boldsymbol{x} - \bar{\boldsymbol{x}}))^2 \\
&\leq 2(\nabla f(\bar{\boldsymbol{x}})^\top (\boldsymbol{x} - \bar{\boldsymbol{x}}))(1+\rho)\|\nabla^+ f(\boldsymbol{x})\|\|\boldsymbol{x} - \bar{\boldsymbol{x}}\| + 2(\nabla f(\bar{\boldsymbol{x}})^\top \nabla^+ f(\boldsymbol{x}))^2 \\
&\leq 2(1+\rho)M\|\nabla^+ f(\boldsymbol{x})\|\|\boldsymbol{x} - \bar{\boldsymbol{x}}\| + 2\|\nabla f(\bar{\boldsymbol{x}})\|^2 \|\nabla^+ f(\boldsymbol{x})\|^2.
\end{aligned}$$

The last inequality is from (39) and Cauchy's inequality. Together with (36) the result immediately holds. ■

Combining the previous two lemmas, we are now ready to prove the global error bound.

Theorem 18 (Error Bound) *Under Assumption 2 and any $M > 0$, we have*

$$\|\boldsymbol{x} - \bar{\boldsymbol{x}}\| \leq \kappa \|\nabla^+ f(\boldsymbol{x})\|, \quad \forall \boldsymbol{x} \text{ with } \boldsymbol{x} \in \mathcal{X} \text{ and } f(\boldsymbol{x}) - f^* \leq M,$$

6. Note that we use $([\boldsymbol{x} - \nabla f(\boldsymbol{x})]^+_{\mathcal{X}} - \boldsymbol{x} + \nabla f(\boldsymbol{x}))^\top ([\boldsymbol{x} - \nabla f(\boldsymbol{x})]^+_{\mathcal{X}} - \boldsymbol{x} + \boldsymbol{x} - \bar{\boldsymbol{x}}) \leq 0$ and $\nabla^+ f(\boldsymbol{x}) = \boldsymbol{x} - [\boldsymbol{x} - \nabla f(\boldsymbol{x})]^+_{\mathcal{X}}$.

where

$$\kappa = \theta^2(1+\rho)(\frac{1+2\|\nabla g(\boldsymbol{t}^*)\|^2}{\sigma_g} + 4M) + 2\theta\|\nabla f(\bar{\boldsymbol{x}})\|,$$

and $\theta \equiv \theta\left(A, \left(\begin{smallmatrix} E \\ \boldsymbol{b}^\top \end{smallmatrix}\right)\right)$ *is defined in Lemma 15. Specially, when* $\boldsymbol{b} = \boldsymbol{0}$ *or* $\mathcal{X} = \mathbb{R}^l$,

$$\kappa = \theta(A, E)^2 \frac{1+\rho}{\sigma_g}.$$

Proof Consider the following polyhedron of the optimal solutions,

$$\mathcal{X}^* = \{\boldsymbol{x}^* \mid E\boldsymbol{x}^* = \boldsymbol{t}^*,\ \boldsymbol{b}^\top \boldsymbol{x}^* = s^*,\ A\boldsymbol{x}^* \le \boldsymbol{d}\},$$

where $\boldsymbol{t}^*$ and s^* are values described in Lemma 14. We can then apply Lemma 15 to have for any $\boldsymbol{x}$, there exists $\boldsymbol{x}^* \in \mathcal{X}^*$ such that

$$\|\boldsymbol{x} - \boldsymbol{x}^*\| \le \theta\left(A, \left(\begin{smallmatrix} E \\ \boldsymbol{b}^\top \end{smallmatrix}\right)\right) \left\| \begin{bmatrix} [A\boldsymbol{x} - \boldsymbol{d}]_P^+ \\ E\boldsymbol{x} - \boldsymbol{t}^* \\ \boldsymbol{b}^\top \boldsymbol{x} - s^* \end{bmatrix} \right\|, \tag{40}$$

where $\theta\left(A, \left(\begin{smallmatrix} E \\ \boldsymbol{b}^\top \end{smallmatrix}\right)\right)$, independent of $\boldsymbol{x}$, is defined in Lemma 15. Denote $\theta\left(A, \left(\begin{smallmatrix} E \\ \boldsymbol{b}^\top \end{smallmatrix}\right)\right)$ as θ for simplicity. By considering only feasible $\boldsymbol{x}$ and using the definition of $\bar{\boldsymbol{x}}$, (40) implies

$$\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|^2 \le \|\boldsymbol{x} - \boldsymbol{x}^*\|^2 \le \theta^2(\|E\boldsymbol{x} - \boldsymbol{t}^*\|^2 + (\boldsymbol{b}^\top \boldsymbol{x} - s^*)^2), \quad \forall \boldsymbol{x} \in \mathcal{X}.$$

With Lemmas 16 and 17, if $f(\boldsymbol{x}) - f^* \le M$, we can bound $\|E\boldsymbol{x} - \boldsymbol{t}^*\|^2$ and $(\boldsymbol{b}^\top \boldsymbol{x} - s^*)^2$ to obtain

$$\begin{aligned} &\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|^2 \\ &\le \theta^2(1+\rho)(\frac{1+2\|\nabla g(\boldsymbol{t}^*)\|^2}{\sigma_g} + 4M)\|\nabla^+ f(\boldsymbol{x})\|\|\boldsymbol{x} - \bar{\boldsymbol{x}}\| + 4\theta^2\|\nabla f(\bar{\boldsymbol{x}})\|^2\|\nabla^+ f(\boldsymbol{x})\|^2. \end{aligned} \tag{41}$$

Let

$$\begin{aligned} &a \equiv \|\boldsymbol{x} - \bar{\boldsymbol{x}}\|, \quad c \equiv 2\theta\|\nabla f(\bar{\boldsymbol{x}})\|\|\nabla^+ f(\boldsymbol{x})\|, \quad \text{and} \\ &b \equiv \theta^2(1+\rho)(\frac{1+2\|\nabla g(\boldsymbol{t}^*)\|^2}{\sigma_g} + 4M)\|\nabla^+ f(\boldsymbol{x})\|. \end{aligned} \tag{42}$$

Then we can rewrite (41) as

$$a^2 \le ba + c^2 \quad \text{with} \quad a \ge 0,\ b \ge 0,\ c \ge 0. \tag{43}$$

We claim that

$$a \le b + c. \tag{44}$$

Otherwise, $a > b + c$ implies that

$$a^2 > a(b+c) > ba + c^2,$$

a violation to (43). By (42) and (44), the proof is complete.

Now we examine the special case of $\boldsymbol{b} = \mathbf{0}$ or $\mathcal{X} = \mathbb{R}^l$. From (31) in Lemma 14, we can apply Lemma 15 to have the existence of $\theta(A, E)$ such that $\forall \boldsymbol{x} \in \mathcal{X}$, there is $\boldsymbol{x}^* \in \mathcal{X}^*$ so that

$$\|\boldsymbol{x} - \bar{\boldsymbol{x}}\| \leq \|\boldsymbol{x} - \boldsymbol{x}^*\| \leq \theta(A, E)\|E\boldsymbol{x} - \boldsymbol{t}^*\|.$$

With Lemma 16, we have

$$\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|^2 \leq \theta(A, E)^2 \frac{1+\rho}{\sigma_g} \|\nabla^+ f(\boldsymbol{x})\| \|\boldsymbol{x} - \bar{\boldsymbol{x}}\|.$$

After canceling $\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|$ from both sides, the proof is complete. ■

5. Proof of Theorem 8

The proof is modified from Theorem 3.1 of Luo and Tseng (1993). They applied a local error bound to obtain asymptotic local linear convergence, while ours applies a global error bound to have linear convergence from the first iteration.

By (9) and Lemma 20, we have

$$\begin{aligned}
&\|\boldsymbol{x}^r - [\boldsymbol{x}^r - \omega_r \nabla f(\boldsymbol{x}^r)]^+_{\mathcal{X}}\| \\
&\leq \|\boldsymbol{x}^r - \boldsymbol{x}^{r+1}\| + \|\boldsymbol{x}^{r+1} - [\boldsymbol{x}^r - \omega_r \nabla f(\boldsymbol{x}^r)]^+_{\mathcal{X}}\| \\
&= \|\boldsymbol{x}^r - \boldsymbol{x}^{r+1}\| + \|[\boldsymbol{x}^r - \omega_r \nabla f(\boldsymbol{x}^r) + \boldsymbol{e}^r]^+_{\mathcal{X}} - [\boldsymbol{x}^r - \omega_r \nabla f(\boldsymbol{x}^r)]^+_{\mathcal{X}}\| \\
&\leq \|\boldsymbol{x}^r - \boldsymbol{x}^{r+1}\| + \|\boldsymbol{e}^r\|. \qquad (45)
\end{aligned}$$

By Lemma 26, the left-hand side of above inequality could be bounded below by

$$\underline{\omega}\|\boldsymbol{x}^r - [\boldsymbol{x}^r - \nabla f(\boldsymbol{x}^r)]^+_{\mathcal{X}}\| \leq \|\boldsymbol{x}^r - [\boldsymbol{x}^r - \omega_r \nabla f(\boldsymbol{x}^r)]^+_{\mathcal{X}}\|,$$

where $\underline{\omega} = \min(1, \inf_r \omega_r)$. With Theorems 13 or 18, (45), and (10), we have

$$\|\boldsymbol{x}^r - \bar{\boldsymbol{x}}^r\| \leq \kappa \|\nabla^+ f(\boldsymbol{x}^r)\| \leq \kappa \frac{\|\boldsymbol{x}^r - [\boldsymbol{x}^r - \omega_r \nabla f(\boldsymbol{x}^r)]^+_{\mathcal{X}}\|}{\underline{\omega}} \leq \kappa \frac{1+\beta}{\underline{\omega}} \|\boldsymbol{x}^r - \boldsymbol{x}^{r+1}\|, \qquad (46)$$

where $\bar{\boldsymbol{x}}^r$ is the projection of $\boldsymbol{x}^r$ to the optimal set.

$$\bar{\boldsymbol{x}}^r \equiv [\boldsymbol{x}^r]^+_{\mathcal{X}^*}.$$

Next, we bound $f(\boldsymbol{x}^{r+1}) - f(\bar{\boldsymbol{x}}^r)$. Lemma 19 and the definition of $\boldsymbol{x}^{r+1}$ imply that

$$\left(\boldsymbol{x}^r - \boldsymbol{x}^{r+1} + \boldsymbol{e}^r\right)^\top \left(\boldsymbol{x}^{r+1} - \bar{\boldsymbol{x}}^r\right) \geq \omega_r \nabla f(\boldsymbol{x}^r)^\top (\boldsymbol{x}^{r+1} - \bar{\boldsymbol{x}}^r). \qquad (47)$$

From the convexity of $f(\boldsymbol{x})$,

$$\begin{aligned}
&f(\boldsymbol{x}^{r+1}) - f(\bar{\boldsymbol{x}}^r) \leq \nabla f(\boldsymbol{x}^{r+1})^\top (\boldsymbol{x}^{r+1} - \bar{\boldsymbol{x}}^r) \\
&= (\nabla f(\boldsymbol{x}^{r+1}) - \nabla f(\boldsymbol{x}^r))^\top (\boldsymbol{x}^{r+1} - \bar{\boldsymbol{x}}^r) + \nabla f(\boldsymbol{x}^r)^\top (\boldsymbol{x}^{r+1} - \bar{\boldsymbol{x}}^r) \\
&\leq \|\nabla f(\boldsymbol{x}^{r+1}) - \nabla f(\boldsymbol{x}^r)\| \|\boldsymbol{x}^{r+1} - \bar{\boldsymbol{x}}^r\| + \frac{1}{\omega_r} (\boldsymbol{x}^r - \boldsymbol{x}^{r+1} + \boldsymbol{e}^r)^\top (\boldsymbol{x}^{r+1} - \bar{\boldsymbol{x}}^r) \qquad (48) \\
&\leq \left(\rho \|\boldsymbol{x}^{r+1} - \boldsymbol{x}^r\| + \frac{1}{\underline{\alpha}} \|\boldsymbol{x}^r - \boldsymbol{x}^{r+1}\| + \frac{1}{\underline{\alpha}} \|\boldsymbol{e}^r\|\right) \|\boldsymbol{x}^{r+1} - \bar{\boldsymbol{x}}^r\|. \qquad (49)
\end{aligned}$$

Inequality (48) is from (47), and (49) follows from the Lipschitz continuity of $\nabla f(\boldsymbol{x})$. In addition,

$$\|\boldsymbol{x}^{r+1} - \bar{\boldsymbol{x}}^r\| \leq \|\boldsymbol{x}^{r+1} - \boldsymbol{x}^r\| + \|\boldsymbol{x}^r - \bar{\boldsymbol{x}}^r\|. \tag{50}$$

From (46), (10), and (50), each term in (49) is bounded by $\|\boldsymbol{x}^r - \boldsymbol{x}^{r+1}\|$. Therefore,

$$f(\boldsymbol{x}^{r+1}) - f(\bar{\boldsymbol{x}}^r) \leq \phi\|\boldsymbol{x}^r - \boldsymbol{x}^{r+1}\|^2, \quad \text{where } \phi = (\rho + \frac{1+\beta}{\underline{\omega}})(1 + \kappa\frac{1+\beta}{\underline{\omega}}).$$

From (11) and the above inequality,

$$f(\boldsymbol{x}^{r+1}) - f(\bar{\boldsymbol{x}}^r) \leq \frac{\phi}{\phi+\gamma}\left(f(\boldsymbol{x}^r) - f(\bar{\boldsymbol{x}}^r)\right), \quad \forall r.$$

Because $f(\boldsymbol{x})$ is convex, $f(\bar{\boldsymbol{x}}^r)$, $\forall r$ correspond to the same unique optimal function value. Thus the global linear convergence is established.

6. Discussions and Conclusions

For future research, we plan to extend the analysis to other types of algorithms and problems (e.g., L1-regularized problems). Further, the global error bound will be useful in analyzing stopping criteria and the effect of parameter changes on the running time of machine learning problems (for example, the change of parameter C in SVM).

In conclusion, by focusing on a convex but non-strongly convex problem (1), we established a global error bound. We then proved the global linear convergence on a wide range of deterministic algorithms, including cyclic coordinate descent methods for dual SVM and SVR. Consequently, the time complexity of these algorithms is $O(\log(1/\epsilon))$.

Acknowledgments

This work was supported in part by the National Science Council of Taiwan via the grant 101-2221-E-002-199-MY3. The authors thank associate editor and anonymous reviewers for valuable comments.

Appendix A. Properties of Projected Gradient

We present some properties of projected gradient used in the proofs. Most of them are known in the literature, but we list them here for completeness. Throughout this section, we assume $\mathcal{X}$ is a non-empty, closed, and convex set.

First, we present a fundamental result used in the paper: the projection theorem to a non-empty closed convex set $\mathcal{X}$. The convex projection in Definition 3 is equivalent to the following inequality on the right-hand side of (51). That is, if the inequality holds for any $\boldsymbol{z}$, this $\boldsymbol{z}$ will be the result of the convex projection and vise versa.

Lemma 19 (Projection Theorem)

$$\boldsymbol{z} = [\boldsymbol{x}]^+_{\mathcal{X}} \Leftrightarrow (\boldsymbol{z} - \boldsymbol{x})^\top(\boldsymbol{z} - \boldsymbol{y}) \leq 0, \ \forall \boldsymbol{y} \in \mathcal{X}. \tag{51}$$

Proof The proof is modified from Hiriart-Urruty and Lemaréchal (2001, Theorem 3.1.1).

From the convexity of $\mathcal{X}$,

$$\alpha \boldsymbol{y} + (1-\alpha)\boldsymbol{z} \in \mathcal{X}, \quad \forall \boldsymbol{y} \in \mathcal{X},\ \forall \alpha \in [0,1].$$

By Definition 3,

$$\|\boldsymbol{x} - \boldsymbol{z}\|^2 \leq \|\boldsymbol{x} - (\alpha \boldsymbol{y} + (1-\alpha)\boldsymbol{z})\|^2, \quad \forall \boldsymbol{y} \in \mathcal{X},\ \forall \alpha \in [0,1].$$

The inequality can be written as

$$0 \leq \alpha(\boldsymbol{z} - \boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{z}) + \frac{1}{2}\alpha^2 \|\boldsymbol{y} - \boldsymbol{z}\|^2.$$

Divide α from both sides, and let $\alpha \downarrow 0$. Then we have ($\Rightarrow$).

For ($\Leftarrow$), if $\boldsymbol{z} = \boldsymbol{x}$, then $0 = \|\boldsymbol{z} - \boldsymbol{x}\| \leq \|\boldsymbol{y} - \boldsymbol{x}\|$ holds for all $\boldsymbol{y} \in \mathcal{X}$. Thus, $\boldsymbol{z} = [\boldsymbol{x}]^+_{\mathcal{X}}$. If $\boldsymbol{z} \neq \boldsymbol{x}$, then for any $\boldsymbol{y} \in \mathcal{X}$,

$$\begin{aligned} 0 \geq (\boldsymbol{z} - \boldsymbol{x})^\top (\boldsymbol{z} - \boldsymbol{y}) &= \|\boldsymbol{x} - \boldsymbol{z}\|^2 + (\boldsymbol{y} - \boldsymbol{x})^\top (\boldsymbol{x} - \boldsymbol{z}) \\ &\geq \|\boldsymbol{x} - \boldsymbol{z}\|^2 - \|\boldsymbol{x} - \boldsymbol{y}\|\|\boldsymbol{x} - \boldsymbol{z}\|. \end{aligned}$$

Divide $\|\boldsymbol{x} - \boldsymbol{z}\| > 0$ from both sides. Because the inequality is valid for all $\boldsymbol{y}$, ($\Leftarrow$) holds. ■

The following lemma shows that the projection operator is Lipschitz continuous.

Lemma 20 (Lipschitz Continuity of Convex Projection)

$$\|[\boldsymbol{x}]^+_{\mathcal{X}} - [\boldsymbol{y}]^+_{\mathcal{X}}\| \leq \|\boldsymbol{x} - \boldsymbol{y}\|, \quad \forall \boldsymbol{x}, \boldsymbol{y}.$$

Proof The proof is modified from Hiriart-Urruty and Lemaréchal (2001) Proposition 3.1.3.

Let $\boldsymbol{u} = [\boldsymbol{x}]^+_{\mathcal{X}}$ and $\boldsymbol{v} = [\boldsymbol{y}]^+_{\mathcal{X}}$. If $\boldsymbol{u} = \boldsymbol{v}$, then the result holds immediately. If not, with Lemma 19 we have

$$(\boldsymbol{u} - \boldsymbol{x})^\top (\boldsymbol{u} - \boldsymbol{v}) \leq 0, \tag{52}$$

$$(\boldsymbol{v} - \boldsymbol{y})^\top (\boldsymbol{v} - \boldsymbol{u}) \leq 0. \tag{53}$$

Summing (52) and (53), we have

$$(\boldsymbol{u} - \boldsymbol{v})^\top (\boldsymbol{u} - \boldsymbol{x} - \boldsymbol{v} + \boldsymbol{y}) \leq 0.$$

We could rewrite it as

$$\|\boldsymbol{u} - \boldsymbol{v}\|^2 \leq (\boldsymbol{u} - \boldsymbol{v})^\top (\boldsymbol{x} - \boldsymbol{y}) \leq \|\boldsymbol{u} - \boldsymbol{v}\|\|\boldsymbol{x} - \boldsymbol{y}\|.$$

Cancel $\|\boldsymbol{u} - \boldsymbol{v}\| > 0$ at both sides. Then the result holds. ■

Lemma 21 *Assume $\nabla f(\boldsymbol{x})$ is ρ Lipschitz continuous. Then $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$,*

$$(\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y}))^\top (\boldsymbol{x} - \boldsymbol{y}) \leq (1+\rho)\|\nabla^+ f(\boldsymbol{x}) - \nabla^+ f(\boldsymbol{y})\|\|\boldsymbol{x} - \boldsymbol{y}\|.$$

Proof For simplification, we will use $\nabla_{\boldsymbol{x}} \equiv \nabla f(\boldsymbol{x})$ and $\nabla^+_{\boldsymbol{x}} \equiv \nabla^+ f(\boldsymbol{x})$ in this proof.
From Lemma 19,

$$([\boldsymbol{x} - \nabla_{\boldsymbol{x}}]^+_{\mathcal{X}} - \boldsymbol{x} + \nabla_{\boldsymbol{x}})^\top ([\boldsymbol{x} - \nabla_{\boldsymbol{x}}]^+_{\mathcal{X}} - [\boldsymbol{y} - \nabla_{\boldsymbol{y}}]^+_{\mathcal{X}}) \leq 0.$$

With the definition of $\nabla^+ f(\boldsymbol{x})$, this inequality can be rewritten as

$$(\nabla_{\boldsymbol{x}} - \nabla^+_{\boldsymbol{x}})^\top (\boldsymbol{x} - \nabla^+_{\boldsymbol{x}} - \boldsymbol{y} + \nabla^+_{\boldsymbol{y}}) \leq 0.$$

Further, we have

$$\nabla_{\boldsymbol{x}}{}^\top (\boldsymbol{x} - \boldsymbol{y}) \leq \nabla^+_{\boldsymbol{x}}{}^\top (\boldsymbol{x} - \boldsymbol{y}) + \nabla_{\boldsymbol{x}}{}^\top (\nabla^+_{\boldsymbol{x}} - \nabla^+_{\boldsymbol{y}}) - \nabla^+_{\boldsymbol{x}}{}^\top (\nabla^+_{\boldsymbol{x}} - \nabla^+_{\boldsymbol{y}}). \tag{54}$$

Similarly,

$$\nabla_{\boldsymbol{y}}{}^\top (\boldsymbol{y} - \boldsymbol{x}) \leq \nabla^+_{\boldsymbol{y}}{}^\top (\boldsymbol{y} - \boldsymbol{x}) + \nabla_{\boldsymbol{y}}{}^\top (\nabla^+_{\boldsymbol{y}} - \nabla^+_{\boldsymbol{x}}) - \nabla^+_{\boldsymbol{y}}{}^\top (\nabla^+_{\boldsymbol{y}} - \nabla^+_{\boldsymbol{x}}). \tag{55}$$

Summing (54) and (55) leads to

$$\begin{aligned}
&(\nabla_{\boldsymbol{x}} - \nabla_{\boldsymbol{y}})^\top (\boldsymbol{x} - \boldsymbol{y}) \\
&\leq (\nabla^+_{\boldsymbol{x}} - \nabla^+_{\boldsymbol{y}})^\top (\boldsymbol{x} - \boldsymbol{y}) + (\nabla_{\boldsymbol{x}} - \nabla_{\boldsymbol{y}})^\top (\nabla^+_{\boldsymbol{x}} - \nabla^+_{\boldsymbol{y}}) - \|\nabla^+_{\boldsymbol{x}} - \nabla^+_{\boldsymbol{y}}\|^2 \\
&\leq (\nabla^+_{\boldsymbol{x}} - \nabla^+_{\boldsymbol{y}})^\top (\boldsymbol{x} - \boldsymbol{y}) + (\nabla_{\boldsymbol{x}} - \nabla_{\boldsymbol{y}})^\top (\nabla^+_{\boldsymbol{x}} - \nabla^+_{\boldsymbol{y}}).
\end{aligned}$$

With $\nabla f(\boldsymbol{x})$ being ρ Lipschitz continuous, we have

$$\begin{aligned}
(\nabla_{\boldsymbol{x}} - \nabla_{\boldsymbol{y}})^\top (\boldsymbol{x} - \boldsymbol{y}) &\leq \|\nabla^+_{\boldsymbol{x}} - \nabla^+_{\boldsymbol{y}}\| (\|\boldsymbol{x} - \boldsymbol{y}\| + \|\nabla_{\boldsymbol{x}} - \nabla_{\boldsymbol{y}}\|) \\
&\leq (1 + \rho) \|\nabla^+_{\boldsymbol{x}} - \nabla^+_{\boldsymbol{y}}\| \|\boldsymbol{x} - \boldsymbol{y}\|.
\end{aligned}$$

■

The next two lemmas correspond to the strong convexity and Lipschitz continuity of projected gradient.

Lemma 22 *If $f(\boldsymbol{x})$ is σ strongly convex and $\nabla f(\boldsymbol{x})$ is ρ Lipschitz continuous,*

$$\frac{\sigma}{1 + \rho} \|\boldsymbol{x} - \boldsymbol{y}\| \leq \|\nabla^+ f(\boldsymbol{x}) - \nabla^+ f(\boldsymbol{y})\|, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}.$$

Proof With the strong convexity and Lemma 21,

$$\sigma \|\boldsymbol{x} - \boldsymbol{y}\|^2 \leq (\nabla_{\boldsymbol{x}} - \nabla_{\boldsymbol{y}})^\top (\boldsymbol{x} - \boldsymbol{y}) \leq (1 + \rho) \|\nabla^+_{\boldsymbol{x}} - \nabla^+_{\boldsymbol{y}}\| \|\boldsymbol{x} - \boldsymbol{y}\|.$$

If $\boldsymbol{x} \neq \boldsymbol{y}$, we have the result after canceling $\|\boldsymbol{x} - \boldsymbol{y}\|$ from both sides. For the situation of $\boldsymbol{x} = \boldsymbol{y}$, the result obviously holds. ■

Lemma 23 (Lipschitz Continuity of Projected Gradient) *If $\nabla f(\boldsymbol{x})$ is ρ Lipschitz continuous, then*

$$\|\nabla^+ f(\boldsymbol{x}) - \nabla^+ f(\boldsymbol{y})\| \leq (2 + \rho) \|\boldsymbol{x} - \boldsymbol{y}\|, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}.$$

Proof By the definition of projected gradient and Lemma 20,

$$\begin{aligned}\|\nabla^+ f(\boldsymbol{x}) - \nabla^+ f(\boldsymbol{y})\| &\leq \|\boldsymbol{x} - \boldsymbol{y}\| + \|[\boldsymbol{x} - \nabla f(\boldsymbol{x})]^+_{\mathcal{X}} - [\boldsymbol{y} - \nabla f(\boldsymbol{y})]^+_{\mathcal{X}}\| \\ &\leq \|\boldsymbol{x} - \boldsymbol{y}\| + \|\boldsymbol{x} - \boldsymbol{y}\| + \|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \\ &\leq (2 + \rho)\|\boldsymbol{x} - \boldsymbol{y}\|.\end{aligned}$$

The last inequality follows from the ρ Lipschitz continuity of $\nabla f(\boldsymbol{x})$. ■

A useful property of projected gradient is to test whether a solution is optimal; see the following lemma.

Lemma 24 *For any $\boldsymbol{x} \in \mathcal{X}$,*

$$\boldsymbol{x} \text{ is optimal for problem } (4) \Leftrightarrow \nabla^+ f(\boldsymbol{x}) = \boldsymbol{0}.$$

Proof From Lemma 19 and the definition of $\nabla^+ f(\boldsymbol{x})$,

$$\begin{aligned}&\nabla^+ f(\boldsymbol{x}) = \boldsymbol{0} \Leftrightarrow \boldsymbol{x} = [\boldsymbol{x} - \nabla f(\boldsymbol{x})]^+_{\mathcal{X}} \\ \Leftrightarrow\ &(\boldsymbol{x} - (\boldsymbol{x} - \nabla f(\boldsymbol{x})))^\top (\boldsymbol{x} - \boldsymbol{y}) \leq 0, \quad \forall \boldsymbol{y} \in \mathcal{X} \\ \Leftrightarrow\ &\nabla f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}) \geq 0, \quad \forall \boldsymbol{y} \in \mathcal{X} \\ \Leftrightarrow\ &\boldsymbol{x} \text{ is optimal.}\end{aligned}$$

The last relation follows from the optimality condition of convex programming problems. ■

The next two lemmas discuss properties of projected gradient defined with different scalars on the negative gradient direction.

Lemma 25 *$\forall \boldsymbol{x} \in \mathcal{X}$,*

$$\|\boldsymbol{x} - [\boldsymbol{x} - \alpha \nabla f(\boldsymbol{x})]^+_{\mathcal{X}}\| \text{ is monotonically increasing for all } \alpha > 0.^7$$

Proof Let

$$\boldsymbol{u} = \boldsymbol{x} - \alpha_1 \nabla f(\boldsymbol{x}), \tag{56}$$
$$\boldsymbol{v} = \boldsymbol{x} - \alpha_2 \nabla f(\boldsymbol{x}), \tag{57}$$

where $0 < \alpha_1 < \alpha_2$. By Lemma 19, we have

$$([\boldsymbol{u}]^+_{\mathcal{X}} - \boldsymbol{u})^\top ([\boldsymbol{u}]^+_{\mathcal{X}} - [\boldsymbol{v}]^+_{\mathcal{X}}) \leq 0, \tag{58}$$
$$([\boldsymbol{v}]^+_{\mathcal{X}} - \boldsymbol{v})^\top ([\boldsymbol{v}]^+_{\mathcal{X}} - [\boldsymbol{u}]^+_{\mathcal{X}}) \leq 0. \tag{59}$$

Let $\boldsymbol{z} = [\boldsymbol{u}]^+_{\mathcal{X}} - [\boldsymbol{v}]^+_{\mathcal{X}}$. Expanding the definition of $\boldsymbol{u}$ and $\boldsymbol{v}$ leads to

$$\alpha_1 \nabla f(\boldsymbol{x})^\top \boldsymbol{z} \leq (\boldsymbol{x} - [\boldsymbol{u}]^+_{\mathcal{X}})^\top \boldsymbol{z} \leq (\boldsymbol{x} - [\boldsymbol{v}]^+_{\mathcal{X}})^\top \boldsymbol{z} \leq \alpha_2 \nabla f(\boldsymbol{x})^\top \boldsymbol{z}, \tag{60}$$

7. The proof is modified from `http://math.stackexchange.com/questions/201168/projection-onto-closed-convex-set` .

where the first and the last inequalities are from (58) and (59), respectively, and the second inequality is from $([\boldsymbol{u}]^+_{\mathcal{X}}-[\boldsymbol{v}]^+_{\mathcal{X}})^\top \boldsymbol{z} = \boldsymbol{z}^\top \boldsymbol{z} \geq 0$. With $0 < \alpha_1 < \alpha_2$, (60) implies $\nabla f(\boldsymbol{x})^\top \boldsymbol{z} \geq 0$ and

$$(\boldsymbol{x} - [\boldsymbol{u}]^+_{\mathcal{X}})^\top \boldsymbol{z} \geq 0.$$

Using this inequality,

$$\|\boldsymbol{x} - [\boldsymbol{v}]^+_{\mathcal{X}}\|^2 = \|\boldsymbol{x} - [\boldsymbol{u}]^+_{\mathcal{X}} + \boldsymbol{z}\|^2 = \|\boldsymbol{x} - [\boldsymbol{u}]^+_{\mathcal{X}}\|^2 + 2(\boldsymbol{x} - [\boldsymbol{u}]^+_{\mathcal{X}})^\top \boldsymbol{z} + \|\boldsymbol{z}\|^2 \geq \|\boldsymbol{x} - [\boldsymbol{u}]^+_{\mathcal{X}}\|^2.$$

Therefore, from (56)-(57),

$$\|\boldsymbol{x} - [\boldsymbol{x} - \alpha_2 \nabla f(\boldsymbol{x})]^+_{\mathcal{X}}\| \geq \|\boldsymbol{x} - [\boldsymbol{x} - \alpha_1 \nabla f(\boldsymbol{x})]^+_{\mathcal{X}}\|.$$

With $0 < \alpha_1 < \alpha_2$, the proof is complete. ■

Lemma 26 *$\forall \boldsymbol{x} \in \mathcal{X}$ and $\alpha > 0$, if*

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{x} - [\boldsymbol{x} - \nabla f(\boldsymbol{x})]^+_{\mathcal{X}}, \\ \boldsymbol{v} &= \boldsymbol{x} - [\boldsymbol{x} - \alpha \nabla f(\boldsymbol{x})]^+_{\mathcal{X}}, \end{aligned}$$

then

$$\min(1, \alpha)\|\boldsymbol{u}\| \leq \|\boldsymbol{v}\| \leq \max(1, \alpha)\|\boldsymbol{u}\|.$$

Proof From Lemma 1 in Gafni and Bertsekas (1984), $\|\boldsymbol{x} - [\boldsymbol{x} - \alpha \nabla f(\boldsymbol{x})]^+_{\mathcal{X}}\|/\alpha$ is monotonically decreasing for all $\alpha > 0$. Thus,

$$\alpha\|\boldsymbol{x} - [\boldsymbol{x} - \nabla f(\boldsymbol{x})]^+_{\mathcal{X}}\| \leq \|\boldsymbol{x} - [\boldsymbol{x} - \alpha \nabla f(\boldsymbol{x})]^+_{\mathcal{X}}\|, \quad \forall \alpha \leq 1.$$

From Lemma 25, we have

$$\|\boldsymbol{x} - [\boldsymbol{x} - \nabla f(\boldsymbol{x})]^+_{\mathcal{X}}\| \leq \|\boldsymbol{x} - [\boldsymbol{x} - \alpha \nabla f(\boldsymbol{x})]^+_{\mathcal{X}}\|, \quad \forall \alpha \geq 1.$$

Therefore, $\min(1, \alpha)\|\boldsymbol{u}\| \leq \|\boldsymbol{v}\|$. A similar proof applies to $\|\boldsymbol{v}\| \leq \max(1, \alpha)\|\boldsymbol{u}\|$. ■

Appendix B. Proof of Hoffman's Bound (Lemma 15)

The following proof is a special case of Mangasarian and Shiau (1987) and Li (1994), which bounds the distance of a point to the polyhedron by the violation of inequalities. We begin with an elementary theorem in convex analysis.

Lemma 27 (Carathèodory's Theorem) *For a non-empty polyhedron*

$$A^\top \boldsymbol{u} + E^\top \boldsymbol{v} = \boldsymbol{y}, \;\; \boldsymbol{u} \geq \boldsymbol{0}, \tag{61}$$

there is a feasible point $(\boldsymbol{u}, \boldsymbol{v})$ such that

The corresponding rows of A, E to $\boldsymbol{u}$, $\boldsymbol{v}$'s non-zero elements are linearly independent. (62)

Proof Let $(\boldsymbol{u}, \boldsymbol{v})$ be a point in the polyhedron, and therefore $E^\top \boldsymbol{v} = \boldsymbol{y} - A^\top \boldsymbol{u}$. If the corresponding rows of E to non-zero elements of $\boldsymbol{v}$ are not linearly independent, we can modify $\boldsymbol{v}$ so that $E^\top \boldsymbol{v}$ remains the same and E's rows corresponding to $\boldsymbol{v}$'s non-zero elements are linearly independent. Thus, without loss of generality, we assume that E is full row-rank. Denote $\boldsymbol{a}_i^\top$ as the ith row of A and $\boldsymbol{e}_j^\top$ as the jth row of E. If the corresponding rows of A, E to non-zero elements of $\boldsymbol{u}, \boldsymbol{v}$ are not linearly independent, there exists $(\boldsymbol{\lambda}, \boldsymbol{\xi})$ such that

1. $(\boldsymbol{\lambda}, \boldsymbol{\xi}) \neq \mathbf{0}$.
2. $(\boldsymbol{\lambda}, \boldsymbol{\xi})$'s non-zero elements correspond to the non-zero elements of $(\boldsymbol{u}, \boldsymbol{v})$. That is, $\lambda_i = 0$ if $u_i = 0$, $\forall i$, and $\xi_j = 0$ if $v_j = 0$, $\forall j$.
3. $(\boldsymbol{\lambda}, \boldsymbol{\xi})$ satisfies
$$\sum_{i:\, u_i>0,\, \lambda_i \neq 0} \lambda_i \boldsymbol{a}_i + \sum_{j:\, v_j \neq 0,\, \xi_j \neq 0} \xi_j \boldsymbol{e}_j = \mathbf{0}.$$

Besides, the set $\{i \mid u_i > 0,\ \lambda_i \neq 0\}$ is not empty because the rows of E are linearly independent. Otherwise, a contradiction occurs from $\boldsymbol{\lambda} = \mathbf{0}$, $\boldsymbol{\xi} \neq \mathbf{0}$, and

$$\sum_{j:\, v_j \neq 0,\, \xi_j \neq 0} \xi_i \boldsymbol{e}_j = \mathbf{0}.$$

By choosing

$$s = \min_{i:\, u_i>0,\, \lambda_i \neq 0} \frac{u_i}{\lambda_i} > 0,$$

we have

$$A^\top(\boldsymbol{u} - s\boldsymbol{\lambda}) + E^\top(\boldsymbol{v} - s\boldsymbol{\xi}) = A^\top \boldsymbol{u} + E^\top \boldsymbol{v} = \boldsymbol{y} \quad \text{and} \quad \boldsymbol{u} - s\boldsymbol{\lambda} \geq \mathbf{0}.$$

This means that $(\boldsymbol{u} - s\boldsymbol{\lambda}, \boldsymbol{v} - s\boldsymbol{\xi})$ is also a member of the polyhedron (61) and has less non-zero elements than $(\boldsymbol{u}, \boldsymbol{v})$. The process could be repeatedly applied until there is a point satisfying the linearly independent condition (62). Thus, if the polyhedron is not empty, we can always find a $(\boldsymbol{u}, \boldsymbol{v})$ such that its non-zero elements correspond to linearly independent rows in (A, E). ■

Now we prove Hoffman's bound (Lemma 15) by Carathèodory's theorem and the KKT optimality condition of a convex projection problem.

Proof If $\boldsymbol{x}$ is in the polyhedron, we can take $\boldsymbol{x}^* = \boldsymbol{x}$ and the inequality (33) holds naturally for every positive θ. Now if $\boldsymbol{x}$ does not belong to the polyhedron, consider the following convex projection problem

$$\min_{\boldsymbol{p}} \ \|\boldsymbol{p} - \boldsymbol{x}\|, \quad \text{subject to } A\boldsymbol{p} \leq \boldsymbol{d},\ E\boldsymbol{p} = \boldsymbol{t}. \tag{63}$$

The polyhedron is assumed to be non-empty, so a unique optimal solution $\boldsymbol{x}^*$ of this problem exists. Because $\boldsymbol{x}$ is not in the polyhedron, we have $\boldsymbol{x}^* \neq \boldsymbol{x}$. Then by the KKT optimality

condition, a unique optimal $\boldsymbol{x}^*$ for (63) happens only if there are $\boldsymbol{u}^*$ and $\boldsymbol{v}^*$ such that

$$\begin{aligned}&\frac{\boldsymbol{x}^*-\boldsymbol{x}}{\|\boldsymbol{x}^*-\boldsymbol{x}\|}=-A^\top\boldsymbol{u}^*-E^\top\boldsymbol{v}^*,\quad \boldsymbol{u}^*\geq\boldsymbol{0},\\&A\boldsymbol{x}^*\leq\boldsymbol{d},\quad E\boldsymbol{x}^*=\boldsymbol{t},\quad u_i^*(A\boldsymbol{x}^*-\boldsymbol{d})_i=0,\ \forall i=1,\ldots,l.\end{aligned}$$

Denote

$$I=\{i\mid(A\boldsymbol{x}^*-\boldsymbol{d})_i=0\}.$$

Because $u_i^*=0,\forall i\notin I$, $(\boldsymbol{u}_I^*,\boldsymbol{v}^*)$ is a feasible point of the following polyhedron.

$$-A_I^\top\boldsymbol{u}_I-E^\top\boldsymbol{v}=\frac{\boldsymbol{x}^*-\boldsymbol{x}}{\|\boldsymbol{x}^*-\boldsymbol{x}\|},\quad \boldsymbol{u}_I\geq\boldsymbol{0},\tag{64}$$

where A_I is a sub-matrix of A's rows corresponding to I. Then the polyhedron in (64) is non-empty. From Lemma 27, there exists a feasible $(\hat{\boldsymbol{u}}_I,\hat{\boldsymbol{v}})$ such that

$$\text{The corresponding rows of } A_I,E \text{ to non-zero } \hat{\boldsymbol{u}}_I,\hat{\boldsymbol{v}} \text{ are linearly independent.}\tag{65}$$

Expand $\hat{\boldsymbol{u}}_I$ to a vector $\hat{\boldsymbol{u}}$ so that

$$\hat{u}_i=0,\ \forall i\notin I.\tag{66}$$

Then (65) becomes

$$\text{The corresponding rows of } A,E \text{ to non-zero } \hat{\boldsymbol{u}},\hat{\boldsymbol{v}} \text{ are linearly independent.}\tag{67}$$

By multiplying $(\boldsymbol{x}^*-\boldsymbol{x})^\top$ on the first equation of (64), we have

$$\|\boldsymbol{x}^*-\boldsymbol{x}\|=\hat{\boldsymbol{u}}^\top A(\boldsymbol{x}-\boldsymbol{x}^*)+\hat{\boldsymbol{v}}^\top E(\boldsymbol{x}-\boldsymbol{x}^*)=\hat{\boldsymbol{u}}^\top(A\boldsymbol{x}-\boldsymbol{d})+\hat{\boldsymbol{v}}^\top(E\boldsymbol{x}-\boldsymbol{t}).\tag{68}$$

The last equality is from $E\boldsymbol{x}^*=\boldsymbol{t}$ and (66). Further, by the non-negativity of $\hat{\boldsymbol{u}}$,

$$\hat{\boldsymbol{u}}^\top(A\boldsymbol{x}-\boldsymbol{d})\leq\hat{\boldsymbol{u}}^\top[A\boldsymbol{x}-\boldsymbol{d}]_P^+.\tag{69}$$

From (68) and (69),

$$\|\boldsymbol{x}^*-\boldsymbol{x}\|\leq\hat{\boldsymbol{u}}^\top[A\boldsymbol{x}-\boldsymbol{d}]_P^++\hat{\boldsymbol{v}}^\top(E\boldsymbol{x}-\boldsymbol{t})\leq\left\|\begin{matrix}\hat{\boldsymbol{u}}\\\hat{\boldsymbol{v}}\end{matrix}\right\|\left\|\begin{matrix}[A\boldsymbol{x}-\boldsymbol{d}]_P^+\\E\boldsymbol{x}-\boldsymbol{t}\end{matrix}\right\|.\tag{70}$$

Next we bound $\left\|\begin{smallmatrix}\hat{\boldsymbol{u}}\\\hat{\boldsymbol{v}}\end{smallmatrix}\right\|$. With (64) and (67), we have

$$\|A^\top\hat{\boldsymbol{u}}+E^\top\hat{\boldsymbol{v}}\|=1\text{ and }\left\|\begin{smallmatrix}\hat{\boldsymbol{u}}\\\hat{\boldsymbol{v}}\end{smallmatrix}\right\|\leq\theta(A,E),$$

where $\theta(A,E)$ is defined in (34). Together with (70), the proof is complete. ■

Note that this version of Hoffman's bound is not the sharpest one. For a more complex but tighter bound, please refer to Li (1994).

Appendix C. Strictly Diagonally Dominance and Positive Definiteness

Lemma 28 *If a symmetric matrix Q is strictly diagonally dominant*

$$Q_{ii} > \sum_{j \neq i} |Q_{ij}|, \quad \forall i, \tag{71}$$

then it is positive definite. The reverse is not true.

Proof The result is modified from Rennie (2005). Because Q is symmetric,

$$Q = RDR^\top, \tag{72}$$

where R is an orthogonal matrix containing Q's eigen-vectors as its columns and D is a real-valued diagonal matrix containing Q's eigen-values. Let $\boldsymbol{u}$ be any eigen-vector of Q. We have $\boldsymbol{u} \neq \mathbf{0}$; otherwise, from (72), the corresponding $Q_{ii} = 0$ and (71) is violated. Let λ be the eigen-value such that $\lambda \boldsymbol{u} = Q\boldsymbol{u}$. Choose $i = \arg\max_j |u_j|$. Because $\boldsymbol{u} \neq \mathbf{0}$, we have either $u_i > 0$ or $u_i < 0$. If $u_i > 0$,

$$Q_{ij} u_j \geq -|Q_{ij}| u_i, \forall j \text{ and } \lambda u_i = \sum_j Q_{ij} u_j \geq (Q_{ii} - \sum_{j \neq i} |Q_{ij}|) u_i. \tag{73}$$

If $u_i < 0$,

$$Q_{ij} u_j \leq -|Q_{ij}| u_i, \forall j \text{ and } \lambda u_i = \sum_j Q_{ij} u_j \leq (Q_{ii} - \sum_{j \neq i} |Q_{ij}|) u_i. \tag{74}$$

By (73) and (74), we have $\lambda \geq Q_{ii} - \sum_{j \neq i} |Q_{ij}| > 0$. Therefore, Q is positive definite.

On the other hand, the following matrix

$$Q = \begin{pmatrix} 2 & 3 \\ 3 & 10 \end{pmatrix}$$

is positive definite but not diagonally dominant. Thus, the reverse is not true. ■

References

Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, 23(4):2037–2060, 2013.

Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. Coordinate descent method for large-scale L2-loss linear SVM. *Journal of Machine Learning Research*, 9:1369–1398, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/cdl2.pdf`.

Eli M. Gafni and Dimitri P. Bertsekas. Two-metric projection methods for constrained optimization. *SIAM Journal on Control and Optimization*, 22:936–964, 1984.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

Clifford Hildreth. A quadratic programming procedure. *Naval Research Logistics Quarterly*, 4:79–85, 1957.

Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Fundamentals of Convex Analysis*. Springer Verlag, 2001.

Chia-Hua Ho and Chih-Jen Lin. Large-scale linear support vector regression. *Journal of Machine Learning Research*, 13:3323–3348, 2012. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/linear-svr.pdf`.

Alan J Hoffman. On approximate solutions of systems of linear inequalities. *Journal of Research of the National Bureau of Standards*, 49(4):263–265, 1952.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/cddual.pdf`.

Wu Li. Sharp Lipschitz constants for basic optimal solutions and basic feasible solutions of linear programs. *SIAM Journal on Control and Optimization*, 32(1):140–153, 1994.

Chih-Jen Lin, Ruby C. Weng, and S. Sathiya Keerthi. Trust region Newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9:627–650, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/logistic.pdf`.

Zhi-Quan Luo and Paul Tseng. On the convergence of coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72 (1):7–35, 1992a.

Zhi-Quan Luo and Paul Tseng. On the linear convergence of descent methods for convex essentially smooth minimization. *SIAM Journal on Control and Optimization*, 30(2): 408–425, 1992b.

Zhi-Quan Luo and Paul Tseng. Error bounds and convergence analysis of feasible descent methods: a general approach. *Annals of Operations Research*, 46:157–178, 1993.

Olvi L. Mangasarian and Tzong-Huei Shiau. Lipschitz continuity of solutions of linear inequalities, programs and complementarity problems. *SIAM Journal on Control and Optimization*, 25(3):583–595, 1987.

Yurii E. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Jong-Shi Pang. A posteriori error bounds for the linearly-constrained variational inequality problem. *Mathematics of Operations Research*, 12(3):474–484, 1987.

John C. Platt. Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.

Jason D. M. Rennie. Regularized logistic regression is strictly convex. Technical report, MIT, 2005. URL `http://qwone.com/~jason/writing/convexLR.pdf`.

Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. 2011. URL `http://link.springer.com/article/10.1007/s10107-012-0614-z`. To appear in Mathematical Programming.

Ankan Saha and Ambuj Tewari. On the nonasymptotic convergence of cyclic coordinate descent methods. *SIAM Journal on Optimization*, 23(1):576–601, 2013.

Ludwig Seidel. Ueber ein verfahren, die gleichungen, auf welche die methode der kleinsten quadrate führt, sowie lineäre gleichungen überhaupt, durch successive annäherung aufzulösen. *Abhandlungen der Bayerischen Akademie der Wissenschaften. Mathematisch-Naturwissenschaftliche Abteilung*, 11(3):81–108, 1874.

Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l1 regularized loss minimization. In *Proceedings of the Twenty Sixth International Conference on Machine Learning (ICML)*, 2009.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14:567–599, 2013a.

Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization, 2013b. arXiv preprint arXiv:1309.2375.

Rachael Tappenden, Peter Richtárik, and Jacek Gondzio. Inexact coordinate descent: complexity and preconditioning, 2013. arXiv preprint arXiv:1304.5530.

Paul Tseng and Sangwoon Yun. Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. *Journal of Optimization Theory and Applications*, 140:513–535, 2009.

Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, NY, 1995.

Jean-Philippe Vial. Strong and weak convexity of sets and functions. *Mathematics of Operations Research*, 8(2):231–259, 1983.

Stephen J Wright. Accelerated block-coordinate relaxation for regularized optimization. *SIAM Journal on Optimization*, 22(1):159–186, 2012.

Journal of Machine Learning Research 15 (2014) 1549-1591 Submitted 11/12; Revised 12/13; Published 4/14

High-Dimensional Covariance Decomposition into Sparse Markov and Independence Models

Majid Janzamin MJANZAMI@UCI.EDU
Animashree Anandkumar A.ANANDKUMAR@UCI.EDU
Department of Electrical Engineering and Computer Science
University of California
Irvine, CA 92697, USA

Editor: Bin Yu

Abstract

Fitting high-dimensional data involves a delicate tradeoff between faithful representation and the use of sparse models. Too often, sparsity assumptions on the fitted model are too restrictive to provide a faithful representation of the observed data. In this paper, we present a novel framework incorporating sparsity in different domains. We decompose the observed covariance matrix into a sparse Gaussian Markov model (with a sparse precision matrix) and a sparse independence model (with a sparse covariance matrix). Our framework incorporates sparse covariance and sparse precision estimation as special cases and thus introduces a richer class of high-dimensional models. We characterize sufficient conditions for identifiability of the two models, viz., Markov and independence models. We propose an efficient decomposition method based on a modification of the popular ℓ_1-penalized maximum-likelihood estimator (ℓ_1-MLE). We establish that our estimator is consistent in both the domains, i.e., it successfully recovers the supports of both Markov and independence models, when the number of samples n scales as $n = \Omega(d^2 \log p)$, where p is the number of variables and d is the maximum node degree in the Markov model. Our experiments validate these results and also demonstrate that our models have better inference accuracy under simple algorithms such as loopy belief propagation.

Keywords: high-dimensional covariance estimation, sparse graphical model selection, sparse covariance models, sparsistency, convex optimization

1. Introduction

Covariance estimation is a classical problem in multi-variate statistics. The idea that second-order statistics capture important and relevant relationships between a given set of variables is natural. Finding the sample covariance matrix based on observed data is straightforward and widely used (Anderson, 1984). However, the sample covariance matrix is ill-behaved in high-dimensions, where the number of dimensions p is typically much larger than the number of available samples n ($p \gg n$). Here, the problem of covariance estimation is ill-posed since the number of unknown parameters is larger than the number of available samples, and the sample covariance matrix becomes singular in this regime.

Various solutions have been proposed for high-dimensional covariance estimation. Intuitively, by restricting the class of covariance models to those with a limited number of free parameters, we can successfully estimate the models in high dimensions. A natural mech-

anism to achieve this is to impose a sparsity constraint on the covariance matrix. In other words, it is presumed that there are only a few (off-diagonal) non-zero entries in the covariance matrix, which implies that the variables under consideration approximately satisfy *marginal independence*, corresponding to the zero pattern of the covariance matrix (Kauermann, 1996) (and we refer to such models as independence models). Many works have studied this setting and have provided guarantees for high-dimensional estimation through simple thresholding of the sample covariance matrix and other related schemes. See Section 1.2. In many settings, however, marginal independence is too restrictive and does not hold. For instance, consider the dependence between the monthly stock returns of various companies listed on the S&P 100 index. It is quite possible that a wide range of complex (and unobserved) factors such as the economic climate, interest rates etc., affect the returns of all the companies. Thus, it is not realistic to model the stock returns of various companies through a sparse covariance model.

A popular alternative sparse model, based on *conditional independence* relationships, has gained widespread acceptance in recent years (Lauritzen, 1996). In this case, sparsity is imposed *not* on the covariance matrix, but on the inverse covariance or the *precision* matrix. It can be shown that the zero pattern of the precision matrix corresponds to a set of conditional-independence relationships and such models are referred to as graphical or Markov models. Going back to the stock market example, a first-order approximation is to model the companies in different divisions[1] as conditionally independent given the S&P 100 index variable, which captures the overall trends of the stock returns, and thus removes much of the dependence between the companies in different divisions. High-dimensional estimation in models with sparse precision matrices has been widely studied, and guarantees for estimation have been provided under a set of sufficient conditions. See Section 1.2 for related works. However, sparse Markov models may not be always sufficient to capture all the statistical relationships among variables. Going back to the stock market example, the approximation of using the S&P index node to capture the dependence between companies of different divisions may not be enough. For instance, there can still be a large *residual* dependence between the companies in manufacturing and mining divisions, which cannot be accounted by the S&P index node.

In this paper, we consider decomposition of the observed data into two domains, viz., Markov and independence domains. We posit that the observed data results in a sparse graphical model under structured perturbations in the form of an independence model, see Figure 1. This framework encapsulates Markov and independence models, and incorporates a richer class of models which can faithfully capture complex relationships, such as in the stock market example above, and yet retain parsimonious representation. The idea that a combination of Markov and independence models can provide good model-fitting is not by itself new and perhaps the work which is closest to ours is the work by Choi et al. (2010), where multi-resolution models with a known hierarchy of variables is considered. Their model consists of a combination of a sparse precision matrix, which captures the conditional independence across scales, and a sparse covariance matrix, which captures the residual in-scale correlations. Heuristics for learning and inference are provided in Choi et al. (2010). However, the approach in Choi et al. (2010) has several deficiencies, including

1. See `http://www.osha.gov/pls/imis/sic_manual.html` for classifications of the companies.

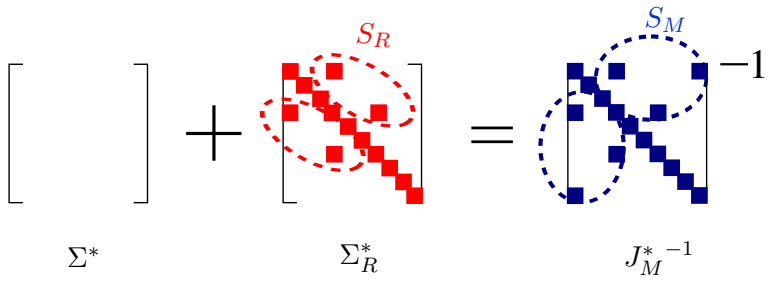


Figure 1: Representation of the covariance decomposition problem, where perturbing the observed covariance matrix with a structured noise model results in a sparse graphical model. The case where the noise model has sparse marginal dependencies is considered.

lack of theoretical guarantees, assumption of a known sparsity support for the Markov model, use of expectation maximization (EM) which has no guarantees of reaching the global optimum, non-identifiability due to the presence of both latent variables and residual correlations, and so on. In contrast, we develop efficient convex optimization methods for decomposition, which are easily implementable and also provide theoretical guarantees for successful recovery. In summary, in this paper, we provide an in-depth study of efficient methods and guarantees for joint estimation of a combination of Markov and independence models.

Our model reduces to sparse covariance and sparse inverse covariance estimation for certain choices of tuning parameter. Therefore, we incorporate a range of models from sparse covariance to sparse inverse covariance.

1.1 Summary of Contributions

We consider joint estimation of Markov and independence models, given observed data in a high dimensional setting. Our contributions in this paper are three fold. First, we derive a set of sufficient restrictions, under which there is a unique decomposition into the two domains, viz., the Markov and the independence domains, thereby leading to an *identifiable* model. Second, we propose novel and efficient estimators for obtaining the decomposition, under both exact and sample statistics. Third, we provide strong theoretical guarantees for high-dimensional learning, both in terms of norm guarantees and *sparsistency* in each domain, viz., the Markov and the independence domain.

Our learning method is based on convex optimization. We adapt the popular ℓ_1-penalized maximum likelihood estimator (MLE), proposed originally for sparse Markov model selection and has efficient implementation in the form of graphical lasso (Friedman et al., 2007). This method involves an ℓ_1 penalty on the precision matrix, which is a convex relaxation of the ℓ_0 penalty, in order to encourage sparsity in the precision matrix. The Lagrangian dual of this program is a *maximum entropy* solution which approximately fits the given sample covariance matrix. We modify this program to our setting as follows: we incorporate an additional ℓ_1 penalty term involving the residual covariance matrix (corresponding to the independence model) in the max-entropy program. This term can be viewed as encouraging sparsity in the independence domain, while fitting a maximum en-

tropy Markov model to the rest of the sample correlations. We characterize the optimal solution of the above program, and also provide intuitions on the class of Markov and independence model combinations which can be incorporated under this framework. As a byproduct of this analysis, we obtain a set of conditions for identifiability of the two model components.

We provide strong theoretical guarantees for our proposed method under a set of sufficient conditions. We establish that it is possible to obtain *sparsistency* and norm guarantees in both the Markov and the independence domains. We establish that the number of samples n is required to scale as $n = \Omega(d^2 \log p)$ for consistency, where p is the number of variables, and d is the maximum degree in the Markov graph. The set of sufficient conditions for successful recovery are based on the so-called notion of *mutual incoherence*, which controls the dependence between different sets of variables (Ravikumar et al., 2011). In Section 7, the synthetic experiments are run on a model which does not necessarily satisfy sufficient mutual incoherence conditions; But we observe that our method has good numerical estimation performance even when the above incoherence conditions are not fully satisfied.

We establish that our estimation reduces to sparse covariance and sparse inverse covariance estimation for certain choices of tuning parameter. On one end, it reduces to the ℓ_1 penalized MLE for sparse precision estimation (Ravikumar et al., 2011). On the other extreme, it reduces to (soft) threshold estimator for sparse covariance estimator, on lines of Bickel and Levina (2008). Moreover, our conditions for successful recovery are similar to those previously characterized for consistent estimation of sparse covariance/precision matrix.

Our experiments validate our theoretical results on the sample complexity and demonstrate that our method is able to learn a richer class of models, compared to sparse graphical model selection, while requiring similar number of samples. In particular, our method is able to provide better estimates for the overall precision matrix, which is dense in general, while the performance of ℓ_1-based optimization is worse since it attempts to approximate the dense matrix via a sparse estimate. Additionally, we demonstrate that our estimated models have better accuracy under simple distributed inference algorithms such as loopy belief propagation (LBP). This is because the Markov components of the estimated models tend to be more *walk summable* (Malioutov et al., 2006), since some of the correlations can be "transferred" to the residual matrix. Thus, in addition to learning a richer model class, incorporating sparsity in both covariance and precision domains, we also learn models amenable to efficient inference. We also apply our method to real data sets. We see the resulting models are fairly interpretable for the real data sets. For instance, for stock returns data set, we observe in both Markov and residual graphs that there exist edges among companies in the same division or industry, e.g., in the residual graph, nodes "HD", "WMT", "TGT" and "MCD", all belonging to division Retail Trade form a partition. Also for foreign exchange rate data set, we observe that the statistical dependencies of foreign exchange rates are correlated with the geographical locations of countries, e.g., it is observed in the learned model that the exchange rates of Asian countries are more correlated.

1.2 Related Works

There have been numerous works on high-dimensional covariance selection and estimation, and we describe them below. In all the settings below based on sparsity of the covariance matrix in some basis, the notion of consistent estimation of the sparse support is known as *sparsistency*.

Sparse Graphical Models: Estimation of covariance matrices by exploiting the sparsity pattern in the inverse covariance or the precision matrix has a long history. The sparsity pattern of the precision matrix corresponds to a Markov graph of a graphical model which characterizes the set of conditional independence relationships between the variables. Chow and Liu established that the maximum likelihood estimate (MLE) for tree graphical models reduces to a maximum weighted spanning tree algorithm where the edge weights correspond to empirical mutual information. The seminal work by Dempster (1972) on covariance selection over chordal graphs analyzed the convex program corresponding to the Gaussian MLE and its dual, when the graph structure is known.

In the high-dimensional regime, penalized likelihood methods have been used in a number of works to achieve parsimony in covariance selection. Penalized MLE based on ℓ_1 penalty has been used in Huang et al. (2006); Meinshausen and Bühlmann (2006); d'Aspremont et al. (2008); Banerjee et al. (2008); Rothman et al. (2008); Ravikumar et al. (2011), among numerous other works, where sparsistency and norm guarantees for recovery in high dimensions are provided. Graphical lasso (Friedman et al., 2007) is an efficient and popular implementation for the ℓ_1-MLE. There have also been recent extensions to group sparsity structures(Yuan and Lin, 2006; Zhao et al., 2009), scenarios with missing samples (Loh and Wainwright, 2011) , semi-parametric settings based on non-paranormals (Liu et al., 2009), and to the non-parametric setting (Kolar et al., 2010). In addition to the convex methods, there have also been a number of non-convex methods for Gaussian graphical model selection (Spirtes and Meek, 1995; Kalisch and Bühlmann, 2007; Zhang, 2009; Anandkumar et al., 2011; Zhang, 2008). While we base much of our consistency analysis on Ravikumar et al. (2011), we also need to develop novel techniques to handle the delicate issue of errors in the two domains, viz., Markov and independence domains.

Sparse Covariance Matrices: In contrast to the above formulation, alternatively we can impose sparsity on the covariance matrix. Note that the zero pattern in the covariance matrix corresponds to marginal independence relationships (Cox and Wermuth, 1993; Kauermann, 1996; Banerjee and Richardson, 2003). High-dimensional estimation of sparse covariance models has been extensively studied in El Karoui (2008); Bickel and Levina (2008); Cai et al. (2010), among others. Wagaman and Levina (2009) consider block-diagonal and banded covariance matrices and propose an Isomap method for discovering meaningful orderings of variables. The work in Lam and Fan (2009) provides unified results for sparsistency under different sparsity assumptions, viz., sparsity in precision matrices, covariance matrices and models with sparse Cholesky decomposition.

The above works provide strong guarantees for covariance selection and estimation under various sparsity assumptions. However, they cannot handle matrices which are combinations of different sparse representations, but are otherwise dense when restricted to any single representation.

Decomposable Regularizers: Recent works have considered model decomposition based on observed samples into desired parts through convex relaxation approaches. Typically, each part is represented as an *algebraic variety*, which are based on *semi-algebraic* sets, and conditions for recovery of each component are characterized. For instance, decomposition of the inverse covariance matrix into sparse and low-rank varieties is considered in Chandrasekaran et al. (2009, 2010a); Candès et al. (2009) and is relevant for latent Gaussian graphical model. The work in Silva et al. (2011) considers finding a sparse-approximation using a small number of positive semi-definite (PSD) matrices, where the "basis" or the set of PSD matrices is specified a priori. In Negahban et al. (2010), a unified framework is provided for high-dimensional analysis of the so-called M-estimators, which optimize the sum of a convex loss function with decomposable regularizers. A general framework for decomposition into a specified set of algebraic varieties was studied in Chandrasekaran et al. (2010b).

The above formulations, however, cannot incorporate our scenario, which consists of a combination of sparse Markov and independence graphs. This is because, although the constraints on the inverse covariance matrix (Markov graph) and the covariance matrix (independence graph) can each be specified in a straightforward manner, their combined constraints on the resulting covariance matrix is not easy to incorporate into a learning method. In particular, we do not have a decomposable regularizer for this setting.

Multi-Resolution Models: Perhaps the work which is closest to ours is the work by Choi et al. (2010), where multi-resolution models with a known hierarchy of variables is considered. The model consists of a combination of a sparse precision matrix, which captures the conditional independence across scales, and a sparse covariance matrix, which captures the residual in-scale correlations. Heuristics for learning and inference are provided. However, the work has three main deficiencies: the sparsity support is assumed to be known, the proposed heuristics have no theoretical guarantees for success and the models considered are in general not identifiable, due to the presence of both latent variables and residual correlations.

2. Preliminaries and Problem Statement

Notation: For any vector $v \in \mathbb{R}^p$ and a real number $a \in [1, \infty)$, the notation $\|v\|_a$ refers to the ℓ_a norm of vector v given by $\|v\|_a := \left(\sum_{i=1}^p |v_i|^a\right)^{\frac{1}{a}}$. For any matrix $U \in \mathbb{R}^{p \times p}$, the induced or the operator norm is given by $\|\!|U|\!\|_{a,b} := \max_{\|z\|_a=1} \|Uz\|_b$ for parameters $a, b \in [1, \infty)$. Specifically, we use the ℓ_∞ operator norm which is equivalent to $\|\!|U|\!\|_\infty = \max_{i=1,\ldots,p} \sum_{j=1}^p |U_{ij}|$. We also have $\|\!|U|\!\|_1 = \|\!|U^T|\!\|_\infty$. Another induced norm is the spectral norm $\|\!|U|\!\|_2$ (or $\|\!|U|\!\|$) which is equivalent to the maximum singular value of U. We also use the ℓ_∞ element-wise norm notation $\|U\|_\infty$ to refer to the maximum absolute value of the entries of U. Note that it is not a matrix norm but a norm on the vectorized form of the matrix. The trace inner product of two matrices is denoted by $\langle U, V \rangle := \mathrm{Tr}(U^T V) = \sum_{i,j} U_{ij} V_{ij}$. Finally, we use the usual notation for asymptotics: $f(n) = \Omega(g(n))$ if $f(n) \geq cg(n)$ for some constant $c > 0$ and $f(n) = O(g(n))$ if $f(n) \leq c' g(n)$ for some constant $c' < \infty$.

2.1 Gaussian Graphical Models

A Gaussian graphical model is a family of jointly Gaussian distributions which factor in accordance to a given graph. Given a graph $G = (V, E)$, with $V = \{1, \ldots, p\}$, consider a vector of Gaussian random variables $\mathbf{X} = [X_1, X_2, \ldots, X_p]$, where each node $i \in V$ is associated with a scalar Gaussian random variable X_i. A Gaussian graphical model Markov on G has a probability density function (pdf) that may be parameterized as

$$f_{\mathbf{X}}(\mathbf{x}) \propto \exp\left[-\frac{1}{2}\mathbf{x}^T J \mathbf{x} + \mathbf{h}^T \mathbf{x}\right], \tag{1}$$

where J is a positive-definite symmetric matrix whose sparsity pattern corresponds to that of the graph G. More precisely,

$$J(i,j) = 0 \iff (i,j) \notin G.$$

The matrix J is known as the potential or concentration matrix, the non-zero entries $J(i,j)$ as the edge potentials, and the vector $\mathbf{h}$ as the potential vector. The form of parameterization in (1) is known as the information form and is related to the standard mean-covariance parameterization of the Gaussian distribution as

$$\boldsymbol{\mu} = J^{-1}\mathbf{h}, \quad \Sigma = J^{-1},$$

where $\boldsymbol{\mu} := \mathbb{E}[\mathbf{X}]$ is the mean vector and $\Sigma := \mathbb{E}[(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^T]$ is the covariance matrix.

We say that a jointly Gaussian random vector $\mathbf{X}$ with joint pdf $f(\mathbf{x})$ satisfies local Markov property with respect to a graph G if

$$f(x_i | \mathbf{x}_{\mathcal{N}(i)}) = f(x_i | \mathbf{x}_{V \setminus i})$$

holds for all nodes $i \in V$, where $\mathcal{N}(i)$ denotes the set of neighbors of node $i \in V$ and, $V \setminus i$ denotes the set of all nodes excluding i. More generally, we say that $\mathbf{X}$ satisfies the global Markov property, if for all disjoint sets $A, B \subset V$, we have

$$f(\mathbf{x}_A, \mathbf{x}_B | \mathbf{x}_S) = f(\mathbf{x}_A | \mathbf{x}_S) f(\mathbf{x}_B | \mathbf{x}_S).$$

where set S is a *separator*[2] of A and B. The local and global Markov properties are equivalent for non-degenerate Gaussian distributions (Lauritzen, 1996).

On lines of the above description of graphical models, consider the class of Gaussian models[3] $\mathcal{N}(\mu, \Sigma_{G_c})$, where the covariance matrix is supported on a graph G_c (henceforth referred to as the conjugate graph), i.e.,

$$\Sigma_{G_c}(i,j) = 0 \equiv (i,j) \notin G_c.$$

Recall that uncorrelated Gaussian variables are independent, and thus,

$$X_i \perp\!\!\!\perp X_j \equiv (i,j) \notin G_c.$$

2. A set $S \subset V$ is a separator for sets A and B if the removal of nodes in S partitions A and B into distinct components.
3. In the sequel, we denote the Markov graph, corresponding the support of the information matrix, as G and the conjugate graph, corresponding to the support of the covariance matrix, as G_c.

Equivalence between pairwise independence and global Markov properties were studied in Cox and Wermuth (1993); Kauermann (1996); Banerjee and Richardson (2003).

In this paper, we posit that the observed model results in a sparse graphical model under structure perturbations in the form of an independence model:

$$\Sigma^* + \Sigma_R^* = {J_M^*}^{-1}, \quad \text{Supp}(J_M^*) = G_M, \text{Supp}(\Sigma_R^*) = G_R, \tag{2}$$

where $\text{Supp}(\cdot)$ denotes the set of non-zero (off-diagonal) entries, G_M denotes the Markov graph and G_R, the independence graph.

2.2 Problem Statement

We now give a detailed description of our problem statement, which consists of the covariance decomposition problem (given exact statistics) and covariance estimation problem (given a set of samples).

2.2.1 Covariance Decomposition Problem

A fundamental question to be addressed is the identifiability of the model parameters.

Definition 1 (Identifiability) *A parametric model $\{P_\theta : \theta \in \Theta\}$ is identifiable with respect to a measure μ if there do not exist two distinct parameters $\theta_1 \neq \theta_2$ such that $P_{\theta_1} = P_{\theta_2}$ almost everywhere with respect to μ.*

Thus, if a model is not identifiable, there is no hope of estimating the model parameters from observed data. A Gaussian graphical model (with no hidden variables) belongs to the family of standard exponential distributions (Wainwright and Jordan, 2008, Ch. 3). Under non-degeneracy conditions, it is also in the minimal form, and as such is identifiable (Brown, 1986). In our setting in (2), however, identifiability is not straightforward to address, and forms an important component of the covariance decomposition problem, described below.

Decomposition Problem: Given the covariance matrix $\Sigma^* = {J_M^*}^{-1} - \Sigma_R^*$ as in (2), where J_M^* is an unknown concentration matrix and Σ_R^* is an unknown residual covariance matrix, how and under what conditions can we uniquely recover J_M^* and Σ_R^* from Σ^*?

In other words, we want to address whether the matrices J_M^* and Σ_R^* are *identifiable*, given Σ^*, and if so, how can we design efficient methods to recover them. If we do not impose any additional restrictions, there exists an *equivalence class* of models which form solutions to the decomposition problem. For instance, we can model Σ^* entirely through an independence model ($\Sigma^* = \Sigma_R^*$), or through a Markov model ($\Sigma^* = {J_M^*}^{-1}$). However, in most scenarios, these extreme cases are not desirable, since they result in dense models, while we are interested in sparse representations with a parsimonious use of edges in both the graphs, viz., the Markov and the independence graphs. In Section 3.1, we provide a sufficient set of structural and parametric conditions to guarantee identifiability of the Markov and the independence components, and in Section 3.2, we propose an optimization program to obtain them.

2.2.2 Covariance Estimation Problem

In the above decomposition problem, we assume that the exact covariance matrix Σ^* is known. However, in practice, we only have access to samples, and we describe this setting below.

Denote $\widehat{\Sigma}^n$ as the sample covariance matrix[4]

$$\widehat{\Sigma}^n := \frac{1}{n}\sum_{k=1}^{n} x_{(k)}x_{(k)}^T, \tag{3}$$

where $x_{(k)}, k = 1, ..., n$ are n i.i.d. observations of a zero mean Gaussian random vector $X \sim \mathcal{N}(0, \Sigma^*)$, where $X := (X_1, ..., X_p)$. Now the estimation problem is described below.

Estimation Problem: Assume that there exists a unique decomposition $\Sigma^* = {J_M^*}^{-1} - \Sigma_R^*$ where J_M^* is an unknown concentration matrix with bounded entries and Σ_R^* is an unknown sparse residual covariance matrix given a set of constraints. Given the sample covariance matrix $\widehat{\Sigma}^n$, our goal is to find estimates of J_M^* and Σ_R^* with provable guarantees.

In the sequel, we relate the exact and the sample versions of the decomposition problem. In Section 4, we propose a modified optimization program to obtain efficient estimates of the Markov and independence components. Under a set of sufficient conditions, we provide guarantees in terms of *sparsistency*, *sign consistency*, and *norm* guarantees, defined below.

Definition 2 (Estimation Guarantees) *We say that an estimate $(\widehat{J}_M, \widehat{\Sigma}_R)$ to the decomposition problem in (2), given a sample covariance matrix $\widehat{\Sigma}^n$, is sparsistent or model consistent, if the supports of $\widehat{J}_M$ and $\widehat{\Sigma}_R$ coincide with the supports of J_M^* and Σ_R^* respectively. It is said to be sign consistent, if additionally, the respective signs coincide. The norm guarantees on the estimates is in terms of bounds on $\|\widehat{J}_M - J_M^*\|$ and $\|\widehat{\Sigma}_R - \Sigma_R^*\|$, under some norm $\|\cdot\|$.*

3. Analysis under Exact Statistics

In this section, we provide the results under exact statistics.

3.1 Conditions for Unique Decomposition

We first provide a set of sufficient conditions under which we can guarantee that the decomposition of Σ^* in (2) into concentration matrix J_M^* and residual matrix Σ_R^* is unique.[5] We impose the following set of constraints on the two matrices:

(A.0) Σ^* and J_M^* are positive definite matrices, i.e., $\Sigma^* \succ 0, J_M^* \succ 0$.

(A.1) Off-diagonal entries of J_M^* are bounded from above, i.e., $\|J_M^*\|_{\infty,\text{off}} \leq \lambda^*$, for some $\lambda^* > 0$.

4. Without loss of generality, we limit our analysis to zero-mean Gaussian models. The results can be easily generalized to models with non-zero means.

5. We drop the positive definite constraint on the residual matrix Σ_R^* thereby allowing for a richer class of covariance decomposition. In Section 5.3, we modify the conditions and the learning method to incorporate positive definite residual matrices Σ_R^*.

(A.2) Diagonal entries of Σ_R^* are zero: $\big(\Sigma_R^*\big)_{ii} = 0$, and the support of its off-diagonal entries satisfies

$$\big(\Sigma_R^*\big)_{ij} \neq 0 \iff |\big(J_M^*\big)_{ij}| = \lambda^*, \quad \forall\, i \neq j.$$

(A.3) For any i, j, we have $\text{sign}\big(\big(\Sigma_R^*\big)_{ij}\big).\text{sign}\big(\big(J_M^*\big)_{ij}\big) \geq 0$, i.e, the signs are the same.

Indeed, the above constraints restrict the class of models for which we can provide guarantees. However, in many scenarios, the above assumptions may be reasonable, and we now provide some justifications. (A.0) is a natural assumption to impose since we are interested in valid Σ^* and J_M^* matrices. Condition (A.1) corresponds to bounded off-diagonal entries of J_M^*. Intuitively, this limits the extent of "dependence" between the variables in the Markov model, and can lead to models where inference can be performed with good accuracy using simple algorithms such as belief propagation. Condition (A.2) limits the support of the residual matrix Σ_R^*: the residual covariances are captured at those locations (edges) where the concentration entries $(J_M^*)_{i,j}$ are "clipped" (i.e., the bound λ^* is achieved). Intuitively, the Markov matrix J_M^* is unable to capture all the correlations between the node pairs due to clipping, and the residual matrix Σ_R^* captures the remaining correlations at the clipped locations. Condition (A.3) additionally characterizes the signs of the entries of Σ_R^*. For the special case, when the Markov model is attractive, i.e., $(J_M^*)_{i,j} \leq 0$ for $i \neq j$, the residual entries $(\Sigma_R^*)_{i,j}$ are also all negative. This implies that the model corresponding to Σ^* is also attractive, since it only consists of positive correlations. By default, we set the diagonal entries of the residual matrix to zero in (A.2) and thus, assume that the Markov matrix captures all the variances in the model. In Section 4.2.1, we provide a simple example of a Markov chain and a residual covariance model satisfying the above conditions.

It is also worth mentioning that the number of model parameters satisfying above conditions is equivalent to the number of parameters in the special case of sparse inverse covariance estimation when $\lambda \to \infty$ (Ravikumar et al., 2011). It is assumed in assumption (A.2) that the residual matrix Σ_R^* takes nonzero value when the corresponding entry in the Markov matrix J_M^* takes its maximum absolute value λ^*. This assumption in conjunction with the sign assumption in (A.3), exactly determines the Markov entry $\big(J_M\big)_{ij}$ when the corresponding residual entry $\big(\Sigma_R\big)_{ij} \neq 0$. So, for each (i, j) pair, only one of the entries $\big(J_M\big)_{ij}$ and $\big(\Sigma_R\big)_{ij}$ are unknown which results that the proposed model in this paper does not introduce additional parameters comparing to the sparse inverse covariance estimation, which is interesting.

According to the above discussion, we observe that the overall covariance and inverse covariance matrices Σ^* and $J^* = \Sigma^{*-1}$ are dense, but represented with small number of parameters. It is interesting that we are able to represent models with dense patterns, but it is important to notice that the sparse representation leads to some restrictions on the model.

In the sequel, we propose an efficient method to recover the respective matrices J_M^* and Σ_R^* under conditions (A.0)-(A.3) and then establish the uniqueness of the decomposition. Finally, note that we do not impose any sparsity constraints on the concentration matrix J_M^*, and in fact, our method and guarantees allow for dense matrices J_M^*, when the exact

covariance matrix Σ^* is available. However, when only samples are available, we limit ourselves to sparse J_M^* and provide learning guarantees in the high-dimensional regime, where the number of samples can be much smaller than the number of variables.

3.2 Formulation of the Optimization Program

We now propose a method based on convex optimization for obtaining (J_M^*, Σ_R^*) given the covariance matrix Σ^* in (2). Consider the following program

$$\begin{aligned}\left(\widehat{\Sigma}_M, \widehat{\Sigma}_R\right) := \underset{\Sigma_M \succ 0, \Sigma_R}{\arg\max} \log\det \Sigma_M - \lambda \|\Sigma_R\|_{1,\text{off}} \\ \text{s.t.} \quad \Sigma_M - \Sigma_R = \Sigma^*,\ (\Sigma_R)_d = 0,\end{aligned} \tag{4}$$

where $\|\cdot\|_{1,\text{off}}$ denotes the ℓ_1 norm of the off-diagonal entries, which is the sum of the absolute values of the off-diagonal entries, and $(\cdot)_d$ denotes the diagonal entries. Intuitively, the parameter λ imposes a penalty on large residual covariances, and under favorable conditions, can encourage sparsity in the residual matrix. The program in (4) can be recast

$$\begin{aligned}\left(\widehat{\Sigma}_M, \widehat{\Sigma}_R\right) := \underset{\Sigma_M \succ 0, \Sigma_R}{\arg\max} \log\det \Sigma_M \\ \text{s.t.} \quad \Sigma_M - \Sigma_R = \Sigma^*,\ (\Sigma_R)_d = 0, \|\Sigma_R\|_{1,\text{off}} \leq C(\lambda),\end{aligned} \tag{5}$$

for some constant $C(\lambda)$ depending on λ. The objective function in the above program corresponds to the entropy of the Markov model (modulo a scaling and a shift factor) (Cover and Thomas, 2006), and thus, intuitively, the above program looks for the optimal Markov model with maximum entropy subject to an ℓ_1 constraint on the residual matrix.

We declare the optimal solution $\widehat{\Sigma}_R$ in (4) as the estimate of the residual matrix Σ_R^*, and $\widehat{J}_M := \widehat{\Sigma}_M^{-1}$ as the estimate of the Markov concentration matrix J_M^*. The justification behind these estimates is based on the fact that the Lagrangian dual of the program in (4) is (see Appendix A)

$$\begin{aligned}\widehat{J}_M := \underset{J_M \succ 0}{\arg\min} \langle \Sigma^*, J_M \rangle - \log\det J_M \\ \text{s.t.} \ \|J_M\|_{\infty,\text{off}} \leq \lambda,\end{aligned} \tag{6}$$

where $\|\cdot\|_{\infty,\text{off}}$ denotes the ℓ_∞ element-wise norm of the off-diagonal entries, which is the maximum absolute value of the off-diagonal entries. Further, we show in Appendix A that the following relations exist between the optimal primal[6] solution $\widehat{J}_M$ and the optimal dual solution $\left(\widehat{\Sigma}_M, \widehat{\Sigma}_R\right)$: $\widehat{J}_M = \widehat{\Sigma}_M^{-1}$, and thus, $\widehat{J}_M^{-1} - \widehat{\Sigma}_R = \Sigma^*$ is a valid decomposition of the covariance matrix Σ^*.

Remark 3 *Notice that when the ℓ_∞ constraint is removed in the primal program in (6), which is equivalent to letting $\lambda \to \infty$, the program corresponds to the maximum likelihood estimate, and the optimal solution in this case is $\widehat{J}_M = {\Sigma^*}^{-1}$. Similarly, in the dual program in (4), when $\lambda \to \infty$, the optimal solution corresponds to $\widehat{\Sigma}_M = \Sigma^*$ and $\widehat{\Sigma}_R = 0$. At the*

6. Henceforth, we refer to the program in (6) as the primal program and the program in (4) as the dual program.

other extreme, when $\lambda \to 0$, $\widehat{J}_M$ is a diagonal matrix, and the residual matrix $\widehat{\Sigma}_R$ is in general, a full matrix (except for the diagonal entries). Thus, the parameter λ allows us to carefully tune the contributions of the Markov and residual components, and we notice in our experiments in Section 7 that λ plays a crucial role in obtaining efficient decomposition into Markov and residual components.

3.3 Guarantees and Main Results

We now establish that the optimal solutions of the proposed optimization programs in (4) and (6) lead to a unique decomposition of the given covariance matrix Σ^* under conditions (A.0)–(A.3) given in Section 3.1.

Theorem 4 (Uniqueness of Decomposition) *Under (A.0)–(A.3), given a covariance matrix Σ^*, if we set the parameter $\lambda = \|J_M^*\|_{\infty,\text{off}}$ in the optimization program in* (4)*, then the optimal solutions of primal-dual optimization programs* (6) *and* (4) *are given by $\big(\widehat{J}_M, \widehat{\Sigma}_R\big) = \big(J_M^*, \Sigma_R^*\big)$, and the decomposition is unique.*

See the proof in Appendix C.

Thus, we establish that the proposed optimization programs in (4) and (6) *uniquely* recover the Markov concentration matrix J_M^* and the residual covariance matrix Σ_R^* given Σ^* under conditions (A.0)–(A.3).

4. Sample Analysis of the Algorithm

In this section, we provide the results under sample statistics where some i.i.d. samples of random variables are only available.

4.1 Optimization Program

We have so far provided guarantees on unique decomposition given the exact covariance matrix Σ^*. We now consider the case, when n i.i.d. samples are available from $\mathcal{N}(0, \Sigma^*)$, which allows us to estimate the sample covariance matrix $\widehat{\Sigma}^n$, as in (3).

We now modify the dual program in (4), considered in the previous section, to incorporate the sample covariance matrix $\widehat{\Sigma}^n$ as follows

$$\begin{aligned} \big(\widehat{\Sigma}_M, \widehat{\Sigma}_R\big) := \underset{\Sigma_M, \Sigma_R}{\arg\max}\ & \log\det \Sigma_M - \lambda \|\Sigma_R\|_{1,\text{off}} \\ \text{s.t. } & \|\widehat{\Sigma}^n - \Sigma_M + \Sigma_R\|_{\infty,\text{off}} \leq \gamma, \\ & \big(\Sigma_M\big)_d = \big(\widehat{\Sigma}^n\big)_d,\ \big(\Sigma_R\big)_d = 0, \\ & \Sigma_M \succ 0, \Sigma_M - \Sigma_R \succ 0. \end{aligned} \tag{7}$$

Note that, in addition to substituting Σ^* by $\widehat{\Sigma}^n$, there are two more modifications in the above program comparing to the exact case in (4). First, the positive-definiteness constraint on the overall covariance matrix $\Sigma = \Sigma_M - \Sigma_R$ is added to make sure that the overall covariance matrix estimation is valid. This constraint is not required in the exact case since we have the constraint $\Sigma = \Sigma^*$ in that case which ensures the positive-definiteness of

overall covariance matrix according to assumption (A.0) that $\Sigma^* \succ 0$. Second, the equality constraint $\Sigma_M - \Sigma_R = \Sigma^*$ is relaxed on the off-diagonal entries by introducing the new parameter γ which allows some deviation. More discussion including the Lagrangian primal form of the above optimization program and the effect of new parameter γ is provided in section 6.

4.2 Assumptions under Sample Statistics

We now provide conditions under which we can provide guarantees for estimating the Markov model J_M^* and the residual model Σ_R^*, given the sample covariance $\widehat{\Sigma}^n$ in high dimensions. These are conditions in addition to conditions (A.0)–(A.3) in Section 3.1.

The additional assumptions for successful recovery in high dimensions are based on the Hessian of the objective function in the optimization program in (19), with respect to the variable J_M, evaluated at the true Markov model J_M^*. The Hessian of this function is given by Boyd and Vandenberghe (2004)

$$\Gamma^* = {J_M^*}^{-1} \otimes {J_M^*}^{-1} = \Sigma_M^* \otimes \Sigma_M^*, \tag{8}$$

where $\otimes$ denotes the Kronecker matrix product (Horn and Johnson, 1985). Thus Γ^* is a $p^2 \times p^2$ matrix indexed by the node pairs. Based on the results for exponential families (Brown, 1986), $\Gamma^*_{(i,j),(k,l)} = \mathrm{Cov}\{X_i X_j, X_k X_l\}$, and hence it can be interpreted as an edge-based alternative to the usual covariance matrix Σ_M^*. Define K_M as the ℓ_∞ operator norm of the covariance matrix of the Markov model

$$K_M := |\!|\!| \Sigma_M^* |\!|\!|_\infty.$$

We now denote the supports of the Markov and residual models. Denote $E_M := \{(i,j) \in V \times V | i \neq j, \left(J_M^*\right)_{ij} \neq 0\}$ as the edge set of Markov matrix J_M^*. Define

$$S_M := E_M \cup \{(i,i) | i = 1, ..., p\}, \tag{9}$$

$$S_R := \{(i,j) \in V \times V | \left(\Sigma_R^*\right)_{ij} \neq 0\}. \tag{10}$$

Thus, the set S_M includes diagonal entries and also all edges of Markov graph corresponding to J_M^*. Also, recall from (A.2) in Section 3.1 that the diagonal entries of Σ_R^* are set to zero, and that the support set S_R is contained in S_M, i.e., $S_R \subset S_M$. Let S_M^c and S_R^c denote the respective complement sets. Define

$$S := S_M \cap S_R^c, \tag{11}$$

so that $\{S_R, S, S_M^c\}$ forms a partition of $\{(1, ..., p) \times (1, ..., p)\}$. This partitioning plays a crucial role in being able to provide learning guarantees. Define the maximum node degree for Markov model J_M^* as

$$d := \max_{j=1,...,p} |\{i : (i,j) \in S_M\}|.$$

Finally, for any two subsets T and T' of $V \times V$, $\Gamma^*_{TT'}$ denotes the submatrix of Γ^* indexed by T as rows and T' as columns. We now impose various constraints on the submatrices of the Hessian in (8), limited to each of the sets $\{S_R, S, S_M^c\}$.

(A.4) **Mutual Incoherence**: These conditions impose mutual incoherence among three partitions of Γ^* indexed by S_R, S_M^c and S. For some $\alpha \in (0,1]$, we have

$$\max\{\|\Gamma^*_{S_M^c S}\big(\Gamma^*_{SS}\big)^{-1}\Gamma^*_{SS_R} - \Gamma^*_{S_M^c S_R}\|_\infty, \|\Gamma^*_{S_M^c S}\big(\Gamma^*_{SS}\big)^{-1}\|_\infty\} \leq (1-\alpha), \tag{12}$$

$$K_{SS_R} := \|\big(\Gamma^*_{SS}\big)^{-1}\Gamma^*_{SS_R}\|_\infty < \frac{1}{4}. \tag{13}$$

(A.5) **Covariance Control**: For the same α specified above, we have the bound:

$$K_{SS} := \|\big(\Gamma^*_{SS}\big)^{-1}\|_\infty \leq \frac{(m-4)\alpha}{4(m-(m-1)\alpha)} \text{ for some } m > 4. \tag{14}$$

(A.6) **Eigenvalue Control**: The minimum eigenvalue of overall covariance matrix Σ^* satisfies the lower bound

$$\lambda_{\min}(\Sigma^*) \geq C_6 d\sqrt{\frac{\log(4p^\tau)}{n}} + C_7 d^2 \frac{\log(4p^\tau)}{n} \text{ for some } C_6, C_7 > 0 \text{ and } \tau > 2.$$

In (A.4), the condition in (12) bounds the effect of the non-edges of the Markov model, indexed by S_M^c, to its edges, indexed by S_R and S. Note that we distinguish between the common edges of the Markov model with the residual model (S_R) and the remaining edges of the Markov model (S). The second condition in (13) controls the influence of the edge-based terms which are shared with the residual matrix, indexed by S_R, to other edges of the Markov model, indexed by $S = S_M \cap S_R^c$. Condition (A.5) imposes ℓ_∞ bounds on the rows of $(\Gamma^*_{SS})^{-1}$. Note that for sufficiently large m, the bound in (14) tends to $\frac{\alpha}{4(1-\alpha)}$. Also note that the conditions (A.4) and (A.5) are only imposed on the Markov model J^*_M and there are no additional constraints on the residual matrix Σ^*_R (other than the conditions previously introduced in Section 3.1). In condition (A.6), it is assumed that the minimum eigenvalue of overall covariance matrix Σ^* is sufficiently far from zero to make sure that its estimation $\widehat{\Sigma}$ is positive definite and therefore a valid covariance matrix.

4.2.1 Example of a Markov Chain + Residual Covariance Model

In this section, we propose a simple model satisfying assumptions (A.0)–(A.5). Consider a Markov chain with concentration matrix J^*_M over 4 nodes, as shown in Figure 2. The diagonal entries in the corresponding covariance matrix $\Sigma^*_M = {J^*_M}^{-1}$ are set to unity, and the correlations between the neighbors in J^*_M are set uniformly to some value $\rho \in (-1,1)$, i.e., $\big(\Sigma^*_M\big)_{ij} = \rho$ for $(i,j) \in E_M$. Due to the Markov property, the correlations between other node pairs are given by $\big(\Sigma^*_M\big)_{13} = \big(\Sigma^*_M\big)_{24} = \rho^2$ and $\big(\Sigma^*_M\big)_{14} = \rho^3$. For the residual covariance matrix Σ^*_R, we consider one edge between nodes 1 and 2, i.e., $S_R = \{(1,2),(2,1)\}$. It is easy to see that conditions (A.0)–(A.2) are satisfied. Recall that $S_M^c = \{(i,j) : (i,j) \notin E_M\}$ and the remaining node pairs belongs to set $S := S_M \setminus S_R$. Through some straightforward calculations, we can show that for any $|\rho| < 0.07$, the mutual incoherence conditions in (A.4) and (A.5) are satisfied for $\alpha = 0.855$ and $m \geq 83$. Note that the value of nonzero entries of Σ^*_R are not involved or restricted by these assumptions. However, they do need to satisfy the sign condition in (A.3). Thus, we have non-trivial models

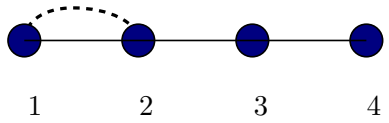


Figure 2: Example of a Markov chain and a residual covariance matrix, where a residual edge is present between nodes 1 and 2.

satisfying the set of sufficient conditions for successful high-dimensional estimation.[7] In Section 7, the synthetic experiments are run on a model which does not necessarily satisfy mutual incoherence conditions (A.4) and (A.5); But we observe that our method has good numerical estimation performance even when the above incoherence conditions are not fully satisfied.

4.3 Guarantees and Main Results

We are now ready to provide the main result of this paper.

Theorem 5 *Consider a Gaussian distribution with covariance matrix* $\Sigma^* = {J_M^*}^{-1} - \Sigma_R^*$ *satisfying conditions (A.0)-(A.6). Given a sample covariance matrix* $\widehat{\Sigma}^n$ *using* n *i.i.d. samples from the Gaussian model, let* $\big(\widehat{J}_M, \widehat{\Sigma}_R\big)$ *denote the optimal solutions of the primal-dual pair* (19) *and* (7)*, with parameters* $\gamma = C_1\sqrt{\log p/n}$ *and* $\lambda = \lambda^* + C_2\sqrt{\log p/n}$ *for some constants* $C_1, C_2 > 0$*, where* $\lambda^* := \|J_M^*\|_{\infty,\text{off}}$*. Suppose that* $\big(\Sigma_R^*\big)_{\min} := \min_{(i,j)\in S_R} \big|\big(\Sigma_R^*\big)_{ij}\big|$ *scales as* $\big(\Sigma_R^*\big)_{\min} = \Omega\big(\sqrt{\log p/n}\big)$ *and the sample size* n *is lower bounded as*

$$n = \Omega\big(d^2 \log p\big), \tag{15}$$

then with probability greater than $1 - 1/p^c \to 1$ *(for some* $c > 0$*), we have:*

a) The estimates $\widehat{J}_M \succ 0$ *and* $\widehat{\Sigma}_R$ *satisfy* ℓ_∞ *bounds*

$$\|\widehat{J}_M - J_M^*\|_\infty = O\bigg(\sqrt{\frac{\log p}{n}}\bigg),$$
$$\|\widehat{\Sigma}_R - \Sigma_R^*\|_\infty = O\bigg(\sqrt{\frac{\log p}{n}}\bigg).$$

b) The estimate $\widehat{\Sigma}_R$ *is sparsistent and sign consistent with* Σ_R^**.*

c) If in addition, $\big(J_M^*\big)_{\min} := \min_{(i,j)\in S_M} \big|\big(J_M^*\big)_{ij}\big|$ *scales as* $\big(J_M^*\big)_{\min} = \Omega\big(\sqrt{\log p/n}\big)$*, then the estimate* $\widehat{J}_M$ *is sparsistent and sign consistent with* J_M^**.*

7. Similarly, for the case when the correlations corresponding to Markov edges are distinct as $\big(\Sigma_M^*\big)_{12} = \rho_1, \big(\Sigma_M^*\big)_{23} = \rho_2$, and $\big(\Sigma_M^*\big)_{34} = \rho_3$, we can argue the same conditions. For compatibility with Figure 2, assume that ρ_1 is the maximum among these three parameters, and therefore, the residual edge is between nodes 1 and 2. This is because the maximum of off-diagonal entries of J_M^* also happens in entry $(1,2)$. Then, the same condition $|\rho_1| < 0.07$ is sufficient for satisfying conditions (A.0)–(A.5).

Proof See Appendix D. ■

Remark 6 *Here, we provide a few more observations and extensions as follows.*

1. **Non-asymptotic sample complexity and error bounds:** *In the above theorem, we establish that the number of samples is required to scale as* $n = \Omega(d^2 \log p)$. *In fact, our results are non-asymptotic, and the exact constants are provided in inequality* (31). *The non-asymptotic form of error bounds are also provided in* (34) *and* (40).

2. **Extension to sub-Gaussian and other distributions:** *In the above theorem, we considered Gaussian distribution. Similar to high dimensional covariance estimation in Ravikumar et al. (2011), the result in the theorem can be easily extended to sub-Gaussian and other distributions with known tail conditions.*

3. **Comparison between direct estimation of Σ^* and the above decomposition:** *The overall matrix* Σ^* *(and* J^**) is a full matrix in general. Thus, if we want to estimate it directly, we need* $n = \Omega\big(p^2 \log p\big)$ *samples since the maximum node degree is* $\Theta(p)$. *Therefore, we can not estimate it directly in high dimensional regime and it demonstrates the importance of such sparse covariance + inverse covariance models for estimation.*

We discussed Remark 3 that the parameter λ allows us to carefully tune the contributions of the Markov and residual components. When $\lambda \to \infty$, the program corresponds to ℓ_1-penalized maximum likelihood estimator which is well-studied in Ravikumar et al. (2011); Rothman et al. (2008). In this case, $\widehat{\Sigma}_R = 0$ and all the dependencies among random variables are captured by the sparse graphical model represented by $\widehat{J}_M$. On the other extreme, when $\lambda^* = 0$ and thus $\lambda = C_2\sqrt{\log p/n} \to 0$, with increasing the number of samples n, the off-diagonal entries in $\widehat{J}_M$ are bounded too tight by λ (refer to the primal program in (19)) and therefore the residual covariance matrix $\widehat{\Sigma}_R$ captures most of the dependencies among random variables. In this case, we have the covariance estimation $\widehat{\Sigma} = \widehat{\Sigma}_M - \widehat{\Sigma}_R$, where the diagonal entries are included in $\widehat{\Sigma}_M$ and the off-diagonal entries are mostly included in $-\widehat{\Sigma}_R$. In order to explain the results for these cases in a more concrete way, we explicitly mention the results for both sparse inverse covariance estimation ($\lambda \to \infty$) and sparse covariance estimation ($\lambda \approx 0$) methods in the following subsections. Note that both of these are special cases of the general result expressed in Theorem 5. Thus, in Theorem 5, we generalize these extreme cases to models with a linear combination of sparse covariance and sparse inverse covariance matrices.

5. Discussions and Extension

In this section, we first provide a detailed discussion of special cases sparse covariance and sparse inverse covariance estimation. Then, the extension of results to the structured noise model is mentioned.

5.1 Sparse Inverse Covariance Estimation

In this section, we mention the result for sparse inverse covariance estimation in high dimensional regime. This result is provided by Ravikumar et al. (2011) and is a special case of Theorem 5 when the parameter λ goes to infinity. Before proposing the explicit result in Corollary 7, we state how the required conditions in Theorem 5 reduces to the conditions in Ravikumar et al. (2011).
Since the support of residual matrix Σ_R^* is a zero matrix in this special case, the mutual incoherence conditions in (A.4) reduce exactly to the same mutual incoherence condition in Ravikumar et al. (2011) as

$$\|\Gamma^*_{S^cS}\big(\Gamma^*_{SS}\big)^{-1}\|_\infty \leq (1-\alpha) \text{ for some } \alpha \in (0,1], \tag{16}$$

where $S = S_M$ is the support of Markov matrix $J^* = J^*_M$ as defined in (9). Also note that the covariance control condition (A.5) is not required any more.
Furthermore, the sample complexity and convergence rate of J^*_M estimation in Theorem 5 exactly reduce to the results in Ravikumar et al. (2011) as (for $q = 8, l = 3$)

$$n > \overline{n}_f\left(p^\tau; 1/\max\left\{v_*, 2ld\Big(1+\frac{q}{\alpha}\Big)K_{SS}K_M \max\Big\{1, \frac{2}{l-1}\Big(1+\frac{q}{\alpha}\Big)K_{SS}K_M^2\Big\}\right\}\right), \tag{17}$$

$$\|\widehat{J} - J^*\|_\infty \leq 2K_{SS}\Big(1+\frac{q}{\alpha}\Big)\overline{\delta}_f(p^\tau; n), \tag{18}$$

where the result is valid for any $q \geq 8$ and $l > 1$.

Corollary 7 (Sparse Inverse Covariance Estimation (Ravikumar et al., 2011)) *Consider a Gaussian distribution with covariance matrix $\Sigma^* = J^{*-1}$ satisfying mutual incoherence condition* (16)*. Given a sample covariance matrix $\widehat{\Sigma}^n$ using n i.i.d. samples from the Gaussian model, let $\widehat{J}$ denote the optimal solution of the primal-dual pair* (19) *and* (7)*, with parameters $\gamma = C_1\sqrt{\log p/n}$ and $\lambda \to \infty$ (removing ℓ_∞ constraints in the primal program* (19)*) for some constant $C_1 > 0$. Suppose that the sample size n is lower bounded as*

$$n = \Omega\big(d^2 \log p\big),$$

then with probability greater than $1 - 1/p^c \to 1$ (for some $c > 0$), we have:

a) The estimate $\widehat{J} \succ 0$ satisfies ℓ_∞ bound

$$\|\widehat{J} - J^*\|_\infty = O\Big(\sqrt{\frac{\log p}{n}}\Big).$$

b) If in addition $\big(J^*\big)_{\min} := \min_{(i,j)\in S_M} |\big(J^*\big)_{ij}|$ scales as $\big(J^*\big)_{\min} = \Omega\big(\sqrt{\log p/n}\big)$, the estimate $\widehat{J}$ is sparsistent and sign consistent with J^*.

Remark 8 (Comparison of general result in Theorem 5 and sparse inverse covariance estimation in Corollary 7) *Considering the results in Theorem 5, sample complexity and convergence rate of estimated models are exactly the same as results in Ravikumar et al. (2011) with only some minor differences in coefficients. Compare* (31) *with* (17)

for sample complexity and (34) *with* (18) *for convergence rate of estimated Markov matrix* $\widehat{J}_M$*. But regarding the mutual incoherence conditions, we observe that the conditions for the special case sparse inverse covariance estimation in* (16) *are less restrictive than the conditions for the general case in* (12)-(13)*. Since the sparse inverse covariance estimation (Ravikumar et al., 2011) is a special case of the general model in this paper, this additional limitation on models is inevitable, i.e., it is natural that we need some more incoherence conditions in order to be able to recover both the Markov and residual models in the general case.*

5.2 Sparse Covariance Estimation

High-dimensional estimation of sparse covariance models has been studied in Bickel and Levina (2008). They propose an estimation of a class of sparse covariance matrices by "hard thresholding". They also prove spectral norm guarantees on the error between the estimated and exact covariance matrices. We also recover similar results in the other extreme case of proposed program (7) when $\lambda \approx 0$. The program reduces to the sparse covariance estimator as discussed earlier. In order to see that again, let us investigate the dual program restated as follows

$$\begin{aligned}\big(\widehat{\Sigma}_M, \widehat{\Sigma}_R\big) := \underset{\Sigma_M, \Sigma_R}{\arg\max}\ & \log\det \Sigma_M - \lambda \|\Sigma_R\|_{1,\text{off}} \\ \text{s.t. } & \|\widehat{\Sigma}^n - \Sigma_M + \Sigma_R\|_{\infty,\text{off}} \leq \gamma, \\ & \big(\Sigma_M\big)_d = \big(\widehat{\Sigma}^n\big)_d,\ \big(\Sigma_R\big)_d = 0, \\ & \Sigma_M \succ 0, \Sigma_M - \Sigma_R \succ 0.\end{aligned}$$

When the parameter $\lambda \approx 0$, the variable Σ_R is very slightly penalized in the objective function. Therefore, most of the statistical dependencies are captured by Σ_R and thus, off-diagonal entries of Σ_M take very small values. Furthermore, according to the property of optimization program that the support of Σ_R is contained within the support of J_M, sparsity on Σ_R is encouraged by the effect of parameter γ.
It is also observed that we are approximately performing "soft thresholding" in program (7) (when $\lambda \approx 0$) comparing to "hard thresholding" in Bickel and Levina (2008). Consider the case $\lambda = 0$, where the Markov part Σ_M is a diagonal matrix. Therefore, the $\|\widehat{\Sigma}^n - \Sigma_M + \Sigma_R\|_{\infty,\text{off}} \leq \gamma$ constraint in the dual program (7) reduces to $\|\widehat{\Sigma}^n + \Sigma_R\|_{\infty,\text{off}} \leq \gamma$ where it is seen that the negative soft thresholding is performed on matrix $\widehat{\Sigma}^n$ with threshold parameter γ, given by

$$S_\gamma(x) = \text{sign}(-x)(|x| - \gamma)_+.$$

Notice that we need to have $\lambda \approx 0$ for recovering the sparse covariance matrix given empirical covariances and in this case, we can view the estimator as approximately performing soft thresholding.

Finally, we propose the corollary for this special case. Before that, we need some additional definitions for a general covariance matrix Σ^*. Similar to definition (10), the support of a covariance matrix Σ^* is defined as

$$S_\Sigma := \{(i,j) \in V \times V | \Sigma^*_{ij} \neq 0\}.$$

The maximum node degree for a covariance matrix Σ^* is also defined as

$$d_\Sigma := \max_{j=1,\ldots,p} |\{i : (i,j) \in S_\Sigma\}|.$$

Corollary 9 (Sparse Covariance Estimation) *Consider a Gaussian distribution with covariance matrix Σ^* satisfying eigenvalue control condition (A.6). Given a sample covariance matrix $\widehat{\Sigma}^n$ using n i.i.d. samples from the Gaussian model, let $\big(\widehat{\Sigma}_M, \widehat{\Sigma}_R\big)$ denote the optimal solutions of the primal-dual pair (19) and (7), with parameters $\gamma = C_1\sqrt{\log p/n}$ and $\lambda = C_2\sqrt{\log p/n}$ for some constants $C_1, C_2 > 0$. The estimated covariance matrix $\widehat{\Sigma}$ is defined as $\widehat{\Sigma}_{\text{off}} := -\widehat{\Sigma}_R$ and $\widehat{\Sigma}_d := \big(\widehat{\Sigma}_M\big)_d$. Suppose that $\big(\Sigma^*_{\text{off}}\big)_{\min} := \min_{(i,j)\in S_\Sigma, i\neq j} |\big(\Sigma^*\big)_{ij}|$ scales as $\big(\Sigma^*_{\text{off}}\big)_{\min} = \Omega\big(\sqrt{\log p/n}\big)$ and the sample size n is lower bounded as*

$$n = \Omega\big(d_\Sigma^2 \log p\big),$$

then with probability greater than $1 - 1/p^c \to 1$ (for some $c > 0$), we have:

a) The estimate $\widehat{\Sigma}$ satisfies ℓ_∞ bound

$$\|\widehat{\Sigma} - \Sigma^*\|_{\infty,\text{off}} = O\bigg(\sqrt{\frac{\log p}{n}}\bigg).$$

b) The estimate $\widehat{\Sigma}_{\text{off}}$ is sparsistent and sign consistent with Σ^*_{off}.

Proof See Appendix F. ■

5.3 Structured Noise Model

In the discussion up to now, we considered general residual matrices Σ^*_R, not necessarily positive definite, thereby allowing for a rich class of covariance decomposition models. In this section, we modify the conditions and the learning method to incorporate positive-definite residual matrices Σ^*_R.

We regularize the diagonal entries in an appropriate way to ensure that both J^*_M and Σ^*_R are positive definite. Thus, the identifiability assumptions (A.0)-(A.3) are modified as follows:

(A.0') Σ^*, Σ^*_R and J^*_M are positive definite matrices, i.e., $\Sigma^* \succ 0, \Sigma^*_R \succ 0, J^*_M \succ 0$.

(A.1') J^*_M is normalized such that $\big(J^*_M\big)_d = \lambda^*_1$ for some $\lambda^*_1 > 0$ and off-diagonal entries of J^*_M are bounded from above, i.e., $\|J^*_M\|_{\infty,\text{off}} \leq \lambda^*_2$, for some $\lambda^*_2 > 0$.

(A.2') The off-diagonal entries of Σ^*_R satisfy

$$\big(\Sigma^*_R\big)_{ij} \neq 0 \iff |\big(J^*_M\big)_{ij}| = \lambda^*_2, \quad \forall\, i \neq j.$$

(A.3') For any i, j, we have $\text{sign}\big(\big(\Sigma^*_R\big)_{ij}\big).\text{sign}\big(\big(J^*_M\big)_{ij}\big) \geq 0$, i.e, the signs are the same.

It is seen in (A.1') that we put additional restrictions on diagonal entries of the Markov matrix J_M^* in order to have nonzero diagonal entries for the residual matrix Σ_R^*.
Similar to the general form of dual program introduced in (23), we propose the following optimization program to estimate the Markov and residual components in the structured noise model:

$$\begin{aligned}\big(\widehat{\Sigma}_M, \widehat{\Sigma}_R\big) := \underset{\Sigma_M, \Sigma_R \succ 0}{\arg\max} \log\det \Sigma_M - \lambda_1 \|\Sigma_R\|_{1,\text{on}} - \lambda_2 \|\Sigma_R\|_{1,\text{off}} \\ \text{s.t.} \quad \|\widehat{\Sigma}^n + \Sigma_R - \Sigma_M\|_{\infty,\text{off}} \leq \gamma, \\ \big(\widehat{\Sigma}^n\big)_d + \big(\Sigma_R\big)_d = \big(\Sigma_M\big)_d.\end{aligned}$$

The decomposition result under exact statistics can be similarly proven by setting parameter $\gamma = 0$ when the identifiability assumptions (A.0')-(A.3') are satisfied. Furthermore, under additional estimation assumptions (A.4)-(A.6), the sample statistics guarantees in Theorem 5 can be also extended to the solutions of above program.

6. Proof Outline

In this section, the Lagrangian primal form for the proposed dual program (7) is provided first and then the proof outlne is presented. For now, we drop the positive-definiteness constraint $\Sigma_M - \Sigma_R \succ 0$ in the proposed dual program (7). We finally show that this constraint is satisfied for the proposed estimation under specified conditions and thus this constraint can be dropped. In the subsequent discussion, we drop this constraint. It is shown in Appendix A that the primal form for this reduced dual program is

$$\begin{aligned}\widehat{J}_M := \underset{J_M \succ 0}{\arg\min} \ \langle \widehat{\Sigma}^n, J_M \rangle - \log\det J_M + \gamma \|J_M\|_{1,\text{off}} \\ \text{s.t.} \ \|J_M\|_{\infty,\text{off}} \leq \lambda,\end{aligned} \tag{19}$$

We further establish that $\widehat{\Sigma}_M = \widehat{J}_M^{-1}$ is valid between the dual variable Σ_M and primal variable J_M and thus,

$$\|\widehat{\Sigma}^n - \widehat{J}_M^{-1} + \widehat{\Sigma}_R\|_{\infty,\text{off}} \leq \gamma. \tag{20}$$

Comparing the above with the exact decomposition $\Sigma^* = {J_M^*}^{-1} - \Sigma_R^*$ in (2), we note that for the sample version, we do not exactly fit the Markov and the residual models with the sample covariance matrix $\widehat{\Sigma}^n$, but allow for some divergence, depending on γ. Similarly, the primal program (19) has an additional ℓ_1 penalty term on $\widehat{J}_M$, which is absent in (6). Having a non-zero γ in the primal program enables us to impose a sparsity constraint on $\widehat{J}_M$, which in turn, enables us to estimate the matrices in the high dimensional regime ($p \gg n$), under a set of conditions of sufficient conditions given in section 4.2.

We now provide a high-level description of the proof for Theorem 5. The detailed proof is given in Appendix D. The proof is based on the primal-dual witness method, which has been previously employed in Ravikumar et al. (2011) and other works. However, we require significant modifications of this approach in order to handle the more complex setting of covariance decomposition.

In the primal-dual witness method, we define a modified version of the original optimization program (19). Note that the key idea in constructing the modified version is to be

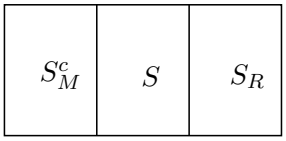


Figure 3: The sets S_R, S and S_M^c form a partition of $\{(1,...,p) \times (1,...,p)\}$, where p is the number of nodes, S_R is the support of the residual covariance matrix Σ_R^* and S_M is the support of the precision matrix J_M^* of the Markov model and S_M^c is its complement.

able to analyze it and prove guarantees for it in a less complicated way comparing to the original version. Let us denote the solutions of the modified program by $\big(\widetilde{J}_M, \widetilde{\Sigma}_R\big)$ pair. In general, the optimal solutions of the two programs, original and modified one, are different. However, under conditions (A.0)–(A.5), we establish that their optimal solutions coincide. See Appendix D for details. Through this equivalence, we thus establish that the optimal solution $\big(\widehat{J}_M, \widehat{\Sigma}_R\big)$ of the original program in (19) inherits all the properties of the optimal solution $\big(\widetilde{J}_M, \widetilde{\Sigma}_R\big)$ of the modified program, i.e., the solutions of the modified program act as witness for the original program. In the following, we define the modified optimization program and its properties. The primal-dual witness method steps which guarantee the equivalence between solutions of the original and the modified program are mentioned in Appendix D.

We modify the sample version of our optimization program in (19) as follows:

$$\widetilde{J}_M := \underset{J_M \succ 0}{\arg\min} \, \langle \widehat{\Sigma}^n, J_M \rangle - \log\det J_M + \gamma \|J_M\|_{1,\text{off}} \qquad (21)$$
$$\text{s.t. } \big(J_M\big)_{S_M^c} = 0, \ \big(J_M\big)_{S_R} = \lambda \operatorname{sign}\Big(\big(J_M^*\big)_{S_R}\Big).$$

Note that since we do not a priori know the supports of the original matrices J_M^* and Σ_R^*, the above program cannot be implemented in practice, but is only a device useful for proving consistency results. We observe that the objective function in the modified program above is the same as the original program in (19), and only the constraints on the precision matrix are different in the two programs. In the above program in (21), constraints on the entries of the precision matrix when limited to sets S_R and S_M^c are more restrictive, while those in set $S := S_M \setminus S_R$ are more relaxed (i.e., the ℓ_∞ constraints present in (19) are removed above), compared to the original program in (19). Recall that S_M denotes the support of the Markov model, while $S_R \subseteq S_M$ denotes the support of the residual or the independence model. See Figure 3.

We now discuss the properties of the optimal solution $\big(\widetilde{J}_M, \widetilde{\Sigma}_R\big)$ of the modified program in (21). Since the precision matrix entries on S_M^c are set to zero in (21), we have that $\text{Supp}(\widetilde{J}_M) \subseteq \text{Supp}(J_M^*)$. Denoting $\widetilde{\Sigma}_R$ as the residual covariance matrix corresponding to the modified program (21), we can similarly characterize it in the following form derived from duality:

$$\big(\widetilde{\Sigma}_R\big)_{ij} = \begin{cases} 0 & \text{for} \quad (i,j) \in S \\ \widetilde{\beta}_{ij} & \text{for} \quad (i,j) \in S_R, S_M^c, \end{cases} \qquad (22)$$

where $\widetilde{\beta}_{ij}$ are the Lagrangian multipliers corresponding to the equality constraints in the modified program (21).

Define estimation errors $\widetilde{\Delta}_J := \widetilde{J}_M - J_M^*$ and $\widetilde{\Delta}_R := \widetilde{\Sigma}_R - \Sigma_R^*$ for the modified program in (21). It is easy to see that $\big(\widetilde{\Delta}_J\big)_{S_R} = \lambda_\delta$, $\big(\widetilde{\Delta}_J\big)_{S_M^c} = 0$, $\big(\widetilde{\Delta}_R\big)_S = 0$, where $\lambda_\delta := \lambda - \lambda^* > 0$. This implies that in any of the three sets S, S_R or S_M^c, only one of the two estimation errors $\widetilde{\Delta}_J$ or $\widetilde{\Delta}_R$ can be non-zero (or is at most λ_δ). This property is crucial to be able to decouple the perturbations in the Markov and the independence domains, and thereby gives bounds on the individual perturbations. It is not clear if there is an alternative partitioning of the variables (here the partition is S, S_R and S_M^c) which allows us to decouple the estimation errors for $\widetilde{J}_M$ and $\widetilde{\Sigma}_R$. Through this decoupling, we are able to provide bounds on estimation errors $\widetilde{\Delta}_J$ and $\widetilde{\Delta}_R$ and thus, Theorem 5 is established.

7. Experiments

In this section, we provide synthetic and real experimental results for the proposed algorithm. We term our proposed optimization program as $\ell_1 + \ell_\infty$ method and compare it with the well-known ℓ_1 method which is a special case of the proposed algorithm when $\lambda = \infty$. The primal optimization program (19) is implemented via the ADMM (Alternating Direction Method of Multipliers) technique proposed in Mohan (2013). We also compare the performance of belief propagation on the proposed model.

7.1 Synthetic Data

We build a Markov + residual synthetic model in the following way. We choose 0.2 fraction of Markov edges randomly to introduce residual edges. The underlying graph for the Markov part is a $q \times q$ 2-D grid structure (4-nearest neighbor grid). Therefore, the number of nodes is $p = q^2$. Because of assumption (A.2), we randomly set 0.2 fraction of nonzero Markov off-diagonal entries to $\{-0.2, 0.2\}$, and the rest of nonzero off-diagonal entries in J_M^* (corresponding to the grid edges) are randomly chosen from set $\pm[0.15, 0.2]$, i.e., $\big(J_M^*\big)_{ij} \in [-0.2, -0.15] \cup [0.15, 0.2]$, for all $(i,j) \in E_M$. Note that 0.2 fraction of edges take the maximum absolute value which is needed by assumption (A.2). Then we ensure that J_M^* is positive definite by adding some uniform diagonal weighting. The nonzero entries of Σ_R^* are chosen from $\pm[0.15, 0.2]$ such that the sign of residual entry is the same as the sign of overlapping Markov entry (assumption (A.3)). We also generate a random mean in the interval $[0, 1]$ for each variable. Note that this generated synthetic model does not necessarily satisfy mutual incoherence conditions (A.4) and (A.5); But we observe in the following that our method has good numerical estimation performance even when the incoherence conditions are not fully satisfied.

Before we provide experiment results, it is worth mentioning that the realization of above model is an example that both Markov and residual matrices J_M^* and Σ_R^* are sparse, while the overall covariance matrix $\Sigma^* = {J_M^*}^{-1} - \Sigma_R^*$ and concentration matrix $J^* = {\Sigma^*}^{-1}$ are both dense matrices.

Size(p)	c_γ	λ
25	2.23	0.2
64	2.08	0.2
100	2.01	0.2
400	1.85	0.2
900	1.83	0.2

Table 1: Regularization parameters used for grid-structured Markov graph simulations in Figure 4. Note that $\gamma = c_\gamma \sqrt{\log p / n}$.

7.1.1 Effect of graph size p

We apply our method ($\ell_1 + \ell_\infty$ method) to random realizations of the above described model $\Sigma^* = {J_M^*}^{-1} - \Sigma_R^*$ with different sizes $p \in \{25, 64, 100, 400, 900\}$. Normalized Dist $\left(\widehat{J}_M, J_M^*\right)$, the edit distance between the estimated and exact Markov components $\widehat{J}_M$ and J_M^*, and normalized Dist $\left(\widehat{\Sigma}_R, \Sigma_R^*\right)$, the edit distance between the estimated and exact residual components $\widehat{\Sigma}_R$ and Σ_R^* as a function of number of samples are plotted in Figure 4 for different sizes p.

In Figure 4.a, normalized Dist $\left(\widehat{J}_M, J_M^*\right)$ is plotted and in Figure 4.b, the same is plotted with rescaled horizontal axis $n/\log p$. We observe that by increasing the number of samples, the edit distance decreases, and by increasing the size of problem, it becomes harder to recover the components which are intuitive. More importantly, we observe in the rescaled graph that the plots for different sizes p make a lineup which is consistent with the theoretical results saying that[8] $n = O(d^2 \log p)$ is sufficient for correct recovery.

Similarly, in Figure 4.c, normalized[9] Dist $\left(\widehat{\Sigma}_R, \Sigma_R^*\right)$ is plotted and in Figure 4.d, the same is plotted with rescaled horizontal axis $n/\log p$. We similarly have the initial observations that by increasing the number of samples, the edit distance decreases, and by increasing the size of problem, it becomes harder to recover the components. The theoretical sample complexity $n = O(d^2 \log p)$ is also validated in Figure 4.d.

The value of regularization parameters used for this simulation are provided in Table 1. Since in the synthetic experiments, we know the value of $\lambda^* := \|J_M^*\|_{\infty,\text{off}}$, parameter λ is set to $\lambda^* = 0.2$. It is observed that the recovery of sparsity pattern of the Markov component J_M^* is fairly robust to the choice of this parameter. For choosing parameter γ, the experiment is run for several values of γ to see which one gives the best recovery result. The effect of parameter γ is discussed in detail in the next subsection.

7.1.2 Effect of regularization parameter γ

We apply our method ($\ell_1 + \ell_\infty$ method) to random realizations of the above described grid-structured synthetic model $\Sigma^* = {J_M^*}^{-1} - \Sigma_R^*$ with fixed size $p = 64$. Here, we fix the

8. Note that in the grid graph, $d = 4$ is fixed for different sizes p.
9. The normalized distance for recovering residual component is greater than 1 for small n. Since we normalize the distance with the number of edges in the exact model, this may happen.

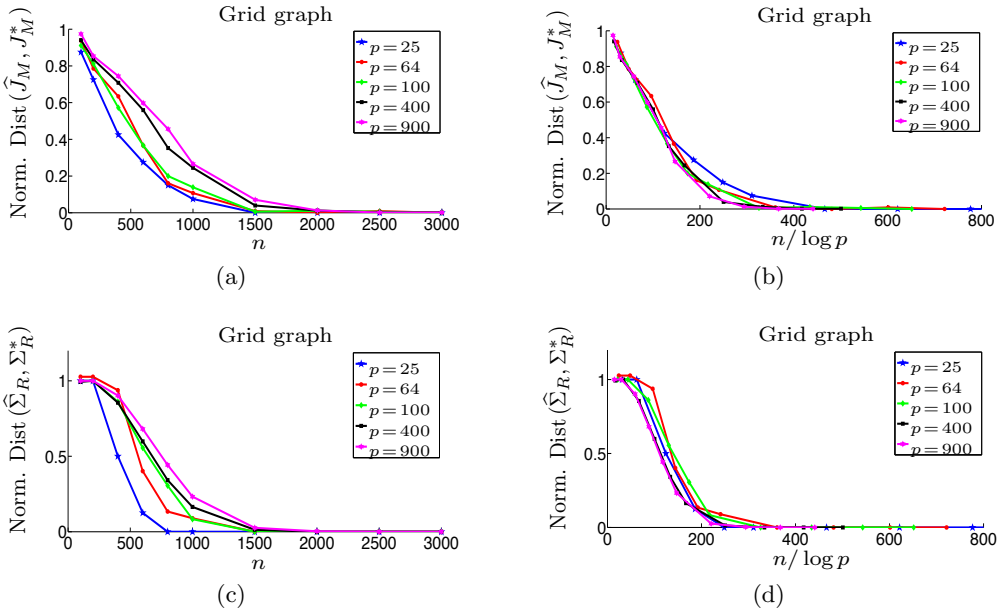


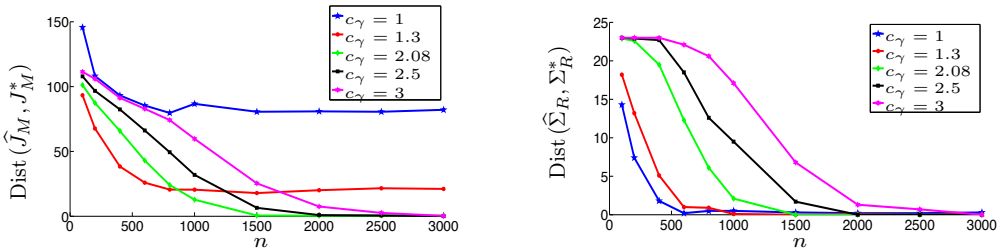
Figure 4: Simulation results for grid-structured Markov graph with different size p. (a-b) Normalized edit distance between the estimated Markov component $\widehat{J}_M$ and the exact Markov component J^*_M. In panel (b), the horizontal axis is rescaled as $n/\log p$. (c-d) Normalized edit distance between the estimated residual component $\widehat{\Sigma}_R$ and the exact residual component Σ^*_R. In panel (d), the horizontal axis is rescaled as $n/\log p$. Each point in the figures is derived from averaging 10 trials.

regularization parameter[10] $\lambda = 0.2$ and change the regularization parameter $\gamma = c_\gamma \sqrt{\log p/n}$ where $c_\gamma \in \{1, 1.3, 2.08, 2.5, 3\}$. The edit distance between the estimated and exact Markov components $\widehat{J}_M$ and J^*_M, and the edit distance between the estimated and exact residual components $\widehat{\Sigma}_R$ and Σ^*_R are plotted in Figure 5. We observe the pattern that for c_γ less than some optimal value c^*_γ, the Markov component is not recovered, and for values greater than the optimal value, the components are recovered with different statistical efficiency, where by increasing c_γ, the statistical rate of Markov component recovery becomes worse. For the simulations of previous subsection provided in Figure 4, we choose some regularization parameter close to c^*_γ. For example, we choose $c_\gamma = 2.08$ for $p = 64$ as suggested by Figure 5.

7.1.3 Comparing $\ell_1 + \ell_\infty$ and ℓ_1 methods

We apply $\ell_1 + \ell_\infty$ and ℓ_1 methods to a random realization of the above described grid-structured synthetic model[11] $\Sigma^* = {J^*_M}^{-1} - \Sigma^*_R$ with size $p = 64$. The edit distance between the estimated and exact Markov components $\widehat{J}_M$ and J^*_M is plotted in Figure 6.a. We observe that the behaviour of $\ell_1 + \ell_\infty$ method is very close to ℓ_1 method which suggests that

10. λ is set to the maximum absolute value of off-diagonal entries of Markov matrix J^*_M.
11. Here, we choose the nonzero off-diagonal entries of J^*_M randomly from $\{-0.2, 0.2\}$.



(a) (b)

Figure 5: Simulation results for grid graph with fixed size $p = 64$ and regularization parameters $\lambda = 0.2$ and varying $c_\gamma \in \{1, 1.3, 2.08, 2.5, 3\}$ where $\gamma = c_\gamma\sqrt{\log p/n}$. (a) Edit distance between the estimated Markov component $\widehat{J}_M$ and the exact Markov component J^*_M. (b) Edit distance between the estimated residual component $\widehat{\Sigma}_R$ and the exact residual component Σ^*_R. Each point in the figures is derived from averaging 10 trials.

sparsity pattern of J^*_M can be estimated efficiently under either methods. The edit distance between the estimated and exact residual components $\widehat{\Sigma}_R$ and Σ^*_R is plotted in Figure 6.b. Since there is not any off-diagonal ℓ_∞ constraints in ℓ_1 method, it can not recover the residual matrix Σ^*_R. Finally the ℓ_∞-elementwise norm of error between the estimated precision matrix $\widehat{J}$ and the exact precision matrix J^* is sketched for both methods in Figure 6.c. We observe the advantage of proposed $\ell_1 + \ell_\infty$ method in estimating the overall model precision matrix $J^* = \Sigma^{*-1}$. Note that the same regularization parameters provided in Table 1 are used for the simulations of this subsection, except for ℓ_1 method that we have $\lambda = \infty$.

7.1.4 Benefit of applying LBP (Loopy Belief Propagation) to the proposed model

We compare the result of applying LBP to J^* and J^*_M components of a random realization of the above described grid-structured synthetic model.[12] The log of average mean and variance errors over all nodes are sketched in Figure 7 throughout the iterations. We observe that LBP does not converge for J^* model. It is shown in Malioutov et al. (2006) that if a model is walk-summable, then the mean estimates under LBP converge and are correct. The spectral norms of the partial correlation matrices are $\|\overline{R}_M\| = 0.8613$ and $\|\overline{R}\| = 3.2446$ for J^*_M and J^* models respectively. Thus, the matrix J^* is not walk-summable and therefore its convergence under LBP is not guaranteed and this is seen in Figure 7. On the other hand, LBP is accurate for J^*_M matrix. Thus, our method learns models which are better suited for inference under loopy belief propagation.

7.2 Real Data

The proposed algorithm is also applied to foreign exchange rate and monthly stock returns data sets to learn a Markov plus residual model introduced in the paper. It is important to

12. Here, we choose 0.5 fraction of Markov edges randomly to introduce residual edges.

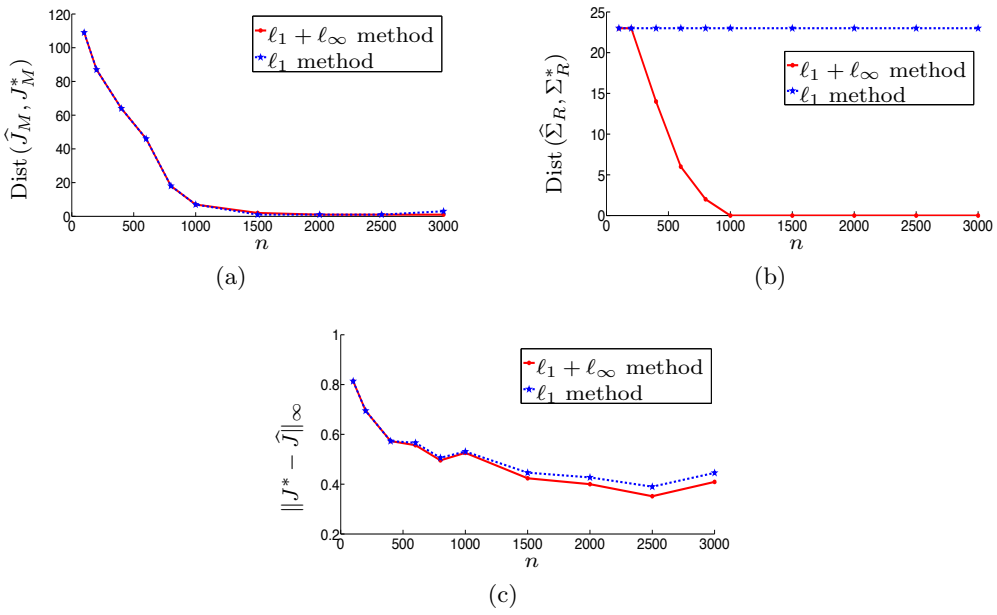


Figure 6: Simulation results for grid graph with size $p = 64$. (a) Edit distance between the estimated Markov component $\widehat{J}_M$ and the exact Markov component J^*_M. (b) Edit distance between the estimated residual component $\widehat{\Sigma}_R$ and the exact residual component Σ^*_R. (c) Precision matrix estimation error $\|J^* - \widehat{J}\|_\infty$, where $\widehat{J} = \widehat{J}_M$ for ℓ_1 method and $\widehat{J} = \left(\widehat{J}_M^{-1} - \widehat{\Sigma}_R\right)^{-1}$ for $\ell_1 + \ell_\infty$ method.

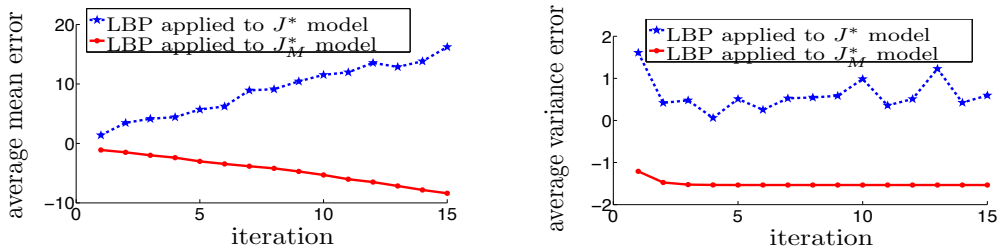


Figure 7: Performance under loopy belief propagation for the overall model (J^*) and the Markov component (J^*_M).

note that the real data sets can be modeled by different models not necessarily satisfying the conditions proposed in this paper. But, here we observe that the resulting Markov plus residual models are fairly interpretable for the corresponding real data sets. The interpretations are discussed in detail in the following sections.

7.2.1 Foreign exchange rate data

In this section, we apply the proposed algorithm to the foreign exchange rate data set.[13] The data set includes monthly exchange rates of 19 countries currency with respect to US dollars from October 1983 to January 2012. Thus, the data set has 340 samples of 19 variables. We apply the optimization program (7) with a slight modification. Since the underlying model for this data set does not necessarily satisfy the proposed eigenvalue condition (A.6), we need to make sure that the overall covariance matrix estimation $\widehat{\Sigma}$ is positive definite and thus a valid covariance matrix. We add an additional constraint to the optimization program (7), imposing a lower bound on the minimum eigenvalue of overall covariance matrix $\lambda_{min}(\Sigma)$, i.e., $\lambda_{min}(\Sigma) \geq \sigma_{min}$. The parameter σ_{min} is set to 0.001 in this experiment.

The resulting edges of Markov and residual matrices for some moderate choice of regularization parameters $\gamma = 20$ and $\lambda = 0.004$ are plotted in Figure 8. The choice of regularization parameters are further discussed at the end of this subsection. We observe sparsity on both Markov and residual structures. There are two main observations in the learned model in Figure 8. First, it is seen that the statistical dependencies of foreign exchange rates are correlated with the geographical locations of countries, e.g., it is observed in the learned model that the exchange rates of Asian countries are more correlated. We can refer to Asian countries "South Korea", "Japan","China","Sri Lanka", "Taiwan", "Thailand" and "India" in the Markov model where several edges exist between them while other nodes in the graph have much lower degrees. We observe similar patterns in the residual matrix, e.g., there is an edge between "India" and "Sri Lanka" in the residual model. We also see the interesting phenomena in the Markov graph that there exist some high degree nodes such as "South Korea" and "Japan". The presence of high degree nodes suggests that incorporating hidden variables can further lead to sparser representations, and this has been observed before in other works, e.g., Choi et al. (2010); Chandrasekaran et al. (2010a); Choi et al. (2011).

The regularization parameters are chosen such that the resulting Markov and residual graphs are reasonably sparse, while still being informative. Increasing the parameter γ makes both Markov and residual components sparser, and increasing parameter λ makes the residual component sparser. In addition, it is worth discussing the fact that we chose parameter γ relatively large compared to parameter λ in this simulation. In Theorem 4, we have $\gamma = C_1\sqrt{\log p/n}$ and $\lambda = \lambda^* + C_2\sqrt{\log p/n}$. Now, if C_1 is large compared to C_2 and furthermore λ^* is small, γ can be larger than λ. Hence, we have an agreement between theory and practice.

7.2.2 Monthly stock returns data

In this section we apply the algorithm to monthly stock returns of a number of companies in the S&P 100 stock index. We pick 17 companies in divisions "E.Trans, Comm, Elec&Gas" and "G.Retail Trade" and apply the optimization program (19) to their stock returns data to learn the model. The resulting edges for Markov and residual matrices are plotted in Figure 9 for regularization parameters $\gamma = 2.2e-03$ and $\lambda = 1e-04$. There is sparsity on both Markov and residual structure. The isolated nodes in the Markov graph are not

13. Data set available at `http://research.stlouisfed.org/fred2/categories/15/downloaddata`.

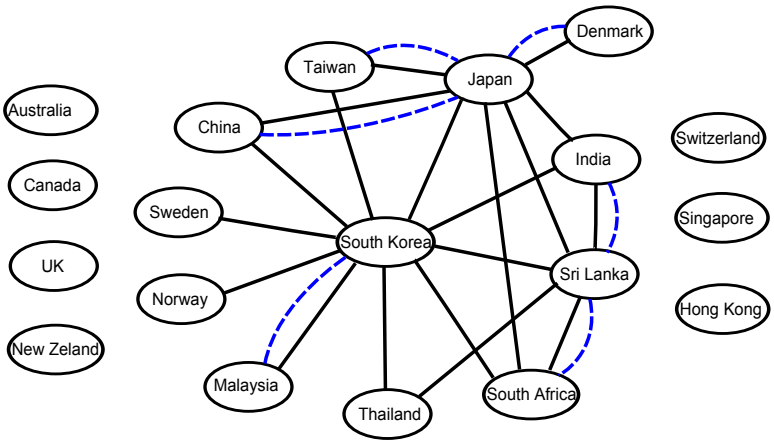


Figure 8: Markov and independence graph structures for the foreign exchange rate data set with regularization parameters $\gamma = 20$ and $\lambda = 0.004$. Solid edges indicate Markov model and dotted edges indicate independence model.

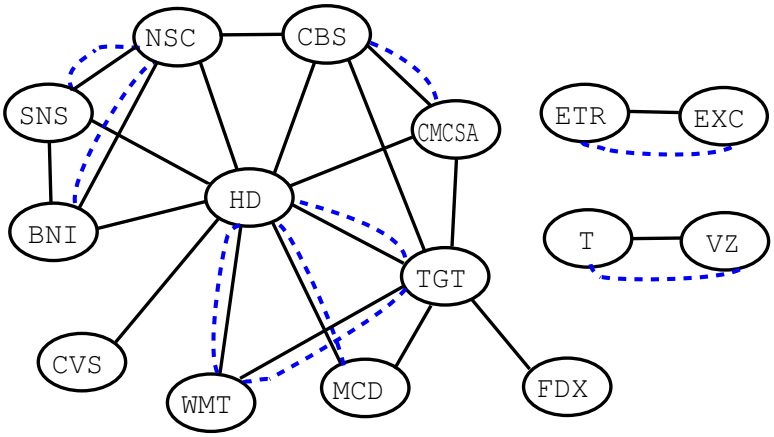


Figure 9: Markov and independence graph structures for the monthly stock returns data set with regularization parameters $\gamma = 2.2e-03$ and $\lambda = 1e-04$. Solid edges indicate Markov model and dotted edges indicate independence model.

presented in the figure. We see in both Markov and residual graphs that there exist higher correlations among stock returns of companies in the same division or industry. There are 5 connected partitions in the residual graph. e.g., nodes "HD", "WMT", "TGT" and "MCD", all belonging to division Retail Trade form a partition. This is also observed for the telecommunication industries (companies "T" and "VZ") and energy industries (companies "ETR" and "EXC"). We see a similar pattern in the Markov graph but with more edges. Similar to exchange rate data set results, we also observe high degree nodes in the Markov graph such as "HD" and "TGT" which suggest incorporating hidden nodes.

8. Conclusion

In this paper, we provided an in-depth study of convex optimization methods and guarantees for high-dimensional covariance matrix decomposition. Our methods unify the existing results for sparse covariance/precision estimation and introduce a richer class of models with sparsity in multiple domains. We provide consistency guarantees for estimation in both the Markov and the residual domains, and establish efficient sample complexity results for our method. These findings open up many future directions to explore. One important aspect is to relax the sparsity constraints imposed in the two domains, and to develop new methods to enable decomposition of such models. Other considerations include extension to discrete models and other models for the residual covariance matrix (e.g., low rank matrices). Such findings will push the envelope of efficient models for high-dimensional estimation. It is worth mentioning while in many scenarios it is important to incorporate latent variables, in our framework it is challenging to incorporate both latent variables as well as marginal independencies, and provide learning guarantees, and we defer it to future work.

Acknowledgements

We thank Karthik Mohan for helpful discussions on running experiments. We also acknowledge useful discussions with Max Welling, Babak Hassibi and Martin Wainwright. We also thank Bin Yu and the JMLR reviewers for valuable comments that have significantly improved the manuscript. M. Janzamin is supported by NSF Award CCF-1219234 and ARO Award W911NF-12-1-0404. A. Anandkumar is supported in part by Microsoft Faculty Fellowship, NSF Career award CCF-1254106, NSF Award CCF-1219234, AFOSR Award FA9550-10-1-0310, and ARO Award W911NF-12-1-0404.

Appendix A. Duality Between Programs

In this section we prove duality between programs (19) and (7) (when the positive-definiteness constraint $\Sigma_M - \Sigma_R \succ 0$ is dropped). By doing this, the duality between programs (6) and (4) is also proved since they are special cases of (19) and (7) when γ is set to zero and $\widehat{\Sigma}^n$ is substituted with Σ^*.

Before we prove duality, we introduce the concept of *subdifferential* or *subgradient* for a convex function not necessarily differentiable. Subgradient (subdifferential) generalizes the gradient (derivative) concept to nondifferentiable functions. Supposing convex function $f : \mathbb{R}^n \to \mathbb{R}$, the subgradient at a point x_0 which is usually denoted by $\partial f(x_0)$ consists of all vectors c such that

$$f(x) \geq f(x_0) + \langle c, x - x_0 \rangle, \quad \forall x \in \mathrm{Dom}\, f.$$

In order to prove duality, we start from program (7) (when the positive-definiteness constraint $\Sigma_M - \Sigma_R \succ 0$ is dropped) and derive the primal form (19). Program (7) can be written in the following equivalent form where λ_1 goes to infinity and λ_2 is used instead of λ.

$$\big(\widehat{\Sigma}_M, \widehat{\Sigma}_R\big) := \underset{\Sigma_M \succ 0, \Sigma_R}{\arg\max} \log\det \Sigma_M - \lambda_1 \|\Sigma_R\|_{1,\mathrm{on}} - \lambda_2 \|\Sigma_R\|_{1,\mathrm{off}} \tag{23}$$

$$\text{s.t.} \quad \|\widehat{\Sigma}^n - \Sigma_M + \Sigma_R\|_{\infty,\text{off}} \leq \gamma,$$
$$\big(\Sigma_M\big)_d - \big(\Sigma_R\big)_d = \big(\widehat{\Sigma}^n\big)_d.$$

By introducing the dual variable J_M for above program, we have:

$$\min_{\substack{\|J_M\|_{\infty,\text{on}}\leq\lambda_1 \\ \|J_M\|_{\infty,\text{off}}\leq\lambda_2}} -\langle J_M, \Sigma_R\rangle = -\lambda_1\|\Sigma_R\|_{1,\text{on}} - \lambda_2\|\Sigma_R\|_{1,\text{off}},$$

where $(\widehat{J}_M)_{\text{on}} \in \lambda_1\partial\|\widehat{\Sigma}_R\|_{1,\text{on}}$, $(\widehat{J}_M)_{\text{off}} \in \lambda_2\partial\|\widehat{\Sigma}_R\|_{1,\text{off}}$ minimizes the above program. Thus, we have the following equivalent form for program (23):

$$\min_{\substack{\|J_M\|_{\infty,\text{on}}\leq\lambda_1 \\ \|J_M\|_{\infty,\text{off}}\leq\lambda_2}} \max_{\substack{\Sigma_M\succ 0,\Sigma_R \\ \|\widehat{\Sigma}^n-\Sigma_M+\Sigma_R\|_{\infty,\text{off}}\leq\gamma \\ (\Sigma_M)_d-(\Sigma_R)_d=\left(\widehat{\Sigma}^n\right)_d}} \log\det\Sigma_M - \langle J_M, \Sigma_R\rangle,$$

where the order of programs is exchanged. If we define the new variable $\Sigma = \Sigma_M - \Sigma_R$, and use Σ as the new variable in the program instead of Σ_R, the inner max program becomes

$$\max_{\substack{\Sigma_M\succ 0,\Sigma \\ \|\widehat{\Sigma}^n-\Sigma\|_{\infty,\text{off}}\leq\gamma,\Sigma_d=\left(\widehat{\Sigma}^n\right)_d}} \log\det\Sigma_M - \langle J_M, \Sigma_M\rangle + \langle J_M, \Sigma\rangle.$$

Since the objective function and constraints are disjoint functions of variables Σ and Σ_M, we can do optimization individually for two variables. The optimizers are $\widehat{\Sigma}_M = J_M^{-1}$ and $\widehat{\Sigma} = \widehat{\Sigma}^n + \gamma Z_\gamma$, where Z_γ is a member of the subgradient of $\|\cdot\|_{1,\text{off}}$ evaluated at point J_M, i.e.,

$$(Z_\gamma)_{ij} = \begin{cases} 0 & \text{for} \quad i = j \\ \in [-1,1] & \text{for} \quad i \neq j, \big(J_M\big)_{ij} = 0 \\ \text{sign}\big(\big(J_M\big)_{ij}\big) & \text{for} \quad i \neq j, \big(J_M\big)_{ij} \neq 0. \end{cases}$$

Also note that since Σ_M should be positive definite, the variable J_M should be also positive definite. Therefore, it adds another constraint $J_M \succ 0$. If we substitute these optimizers, we get the dual program

$$\min_{\substack{J_M\succ 0 \\ \|J_M\|_{\infty,\text{on}}\leq\lambda_1 \\ \|J_M\|_{\infty,\text{off}}\leq\lambda_2}} \langle\widehat{\Sigma}^n, J_M\rangle - \log\det J_M + \gamma\|J_M\|_{1,\text{off}},$$

which is equivalent to (19) when λ_1 goes to infinity and therefore the result is proved.

Appendix B. Characterization of the Proposed Optimization Programs

We proposed programs (6) and (19) to do decomposition and estimation respectively. Former is used to decompose exact statistics to its Markov and residual covariance components and the latter is used to estimate decomposition components given sample covariance matrix. In this appendix we characterize optimal solutions of these optimization programs. Both programs are convex and therefore the optimal solutions can be characterized using standard convex optimization theory. Note that the proof of following lemmas is mentioned after the remarks.

Lemma 1 *For any $\lambda > 0$, primal problem (6) has a unique solution $\widehat{J}_M \succ 0$ which is characterized by the following equation:*

$$\Sigma^* - \widehat{J}_M^{-1} + \widehat{Z} = 0, \tag{24}$$

where $\widehat{Z}$ has the following form

$$\widehat{Z}_{ij} = \begin{cases} 0 & \text{for} \quad i = j \\ 0 & \text{for} \quad i \neq j, |\big(\widehat{J}_M\big)_{ij}| < \lambda \\ \widehat{\alpha}_{ij} \operatorname{sign}\big(\big(\widehat{J}_M\big)_{ij}\big) & \text{for} \quad i \neq j, |\big(\widehat{J}_M\big)_{ij}| = \lambda, \end{cases} \tag{25}$$

in which $\widehat{\alpha}_{ij}$ can only take nonnegative values, i.e., we have $\widehat{\alpha}_{ij} \geq 0$.

Remark 10 *Comparing Lagrangian optimality condition in (24) with relation $\Sigma^* = \widehat{J}_M^{-1} - \widehat{\Sigma}_R$ between solutions of primal-dual optimization programs (derived in Appendix A) implies the equality $\widehat{\Sigma}_R = \widehat{Z}$. Thus, $\widehat{\Sigma}_R$ entries are determined by Lagrangian multipliers of primal program. More specifically, we have*

$$(\widehat{\Sigma}_R)_{ij} = \begin{cases} 0 & \text{for} \quad i = j \\ 0 & \text{for} \quad i \neq j, |\big(\widehat{J}_M\big)_{ij}| < \lambda \\ \widehat{\alpha}_{ij} \operatorname{sign}\big(\big(\widehat{J}_M\big)_{ij}\big) & \text{for} \quad i \neq j, |\big(\widehat{J}_M\big)_{ij}| = \lambda, \end{cases} \tag{26}$$

where $\widehat{\alpha}_{ij} \geq 0$ are the Lagrangian multipliers of primal program (6).

Lemma 2 *For any $\lambda > 0$, $\gamma \geq 0$ and sample covariance matrix $\widehat{\Sigma}^n$ with strictly positive diagonal entries, primal problem (19) has a unique solution $\widehat{J}_M \succ 0$ which is characterized by the equation*

$$\widehat{\Sigma}^n - \widehat{J}_M^{-1} + \widehat{Z} = 0, \tag{27}$$

where $\widehat{Z} = \widehat{Z}_\alpha + \gamma \widehat{Z}_\gamma$. Matrix $\widehat{Z}_\gamma \in \partial \|\widehat{J}_M\|_{1,\text{off}}$ and $\widehat{Z}_\alpha$ is represented as in (25) for some Lagrangian multipliers $\widehat{\alpha}_{ij} \geq 0$.

Remark 11 *Comparing Lagrangian optimality condition in (27) with relation $\widehat{\Sigma}^n = \widehat{J}_M^{-1} - \widehat{\Sigma}_R - \gamma \widehat{Z}_\gamma$ between solutions of primal-dual optimization programs (derived in Appendix A) implies the equality $\widehat{\Sigma}_R = \widehat{Z}_\alpha$. Thus, $\widehat{\Sigma}_R$ entries are determined by the Lagrangian multipliers of primal program. More specifically, we have*

$$(\widehat{\Sigma}_R)_{ij} = \begin{cases} 0 & \text{for} \quad i = j \\ 0 & \text{for} \quad i \neq j, |\big(\widehat{J}_M\big)_{ij}| < \lambda \\ \widehat{\alpha}_{ij} \operatorname{sign}\big(\big(\widehat{J}_M\big)_{ij}\big) & \text{for} \quad i \neq j, |\big(\widehat{J}_M\big)_{ij}| = \lambda, \end{cases} \tag{28}$$

where $\widehat{\alpha}_{ij} \geq 0$ are the Lagrangian multipliers of primal program (19).

Proof We prove Lemma 2 here and Lemma 1 is a special case of that when γ is set to zero and $\widehat{\Sigma}^n$ is substituted with Σ^*.

For any $\lambda > 0$ and $\gamma \geq 0$, the optimization problem (19) is a convex programming where the objective function is strictly convex. Therefore, if the minimum is achieved it

is unique. Since off-diagonal entries of J_M are bounded according to constraints, the only issue for minimum achievement may arises for unbounded diagonal entries. It is shown in Ravikumar et al. (2011) that if diagonal entries of $\widehat{\Sigma}^n$ are strictly positive, the function is coercive with respect to diagonal entries and therefore here is no issue regarding unbounded diagonal entries. Thus, the minimum is attained in $J_M \succeq 0$. But since when J_M approaches the boundary of positive definite cone, the objective function goes to infinity, the solution is attained in the interior of the cone $J_M \succ 0$. After showing that the unique minimum is achieved, let us characterize the minimum.

Considering α_{ij} as Lagrangian multipliers of inequality constraints of program (19), the Lagrangian function is

$$\mathcal{L}(J_M, \alpha) = \langle \widehat{\Sigma}^n, J_M \rangle - \log\det J_M + \gamma \|J_M\|_{1,\text{off}} + \sum_{i \neq j} \alpha_{ij} \big[\big| \big(J_M\big)_{ij} \big| - \lambda \big].$$

We skipped positive definiteness constraint in writing Lagrangian function since it is inactive. Based on standard convex optimization theory, the matrix $\widehat{J}_M \succ 0$ is the optimal solution if and only if it satisfies KKT conditions. It should minimize the Lagrangian which happens if and only if 0 belongs to the subdifferential of Lagrangian or equivalently there exists a matrix $\widehat{Z}$ such that

$$\widehat{\Sigma}^n - \widehat{J}_M^{-1} + \widehat{Z} = 0,$$

where $\widehat{Z} = \widehat{Z}_\alpha + \gamma \widehat{Z}_\gamma$. Matrix $\widehat{Z}_\gamma \in \partial \|\widehat{J}_M\|_{1,\text{off}}$ and $\widehat{Z}_\alpha$ is

$$(\widehat{Z}_\alpha)_{ij} = \begin{cases} 0 & \text{for} \quad i = j \\ \in \widehat{\alpha}_{ij}.[-1,1] & \text{for} \quad i \neq j, \big(\widehat{J}_M\big)_{ij} = 0 \\ \widehat{\alpha}_{ij} \operatorname{sign}\big(\big(\widehat{J}_M\big)_{ij}\big) & \text{for} \quad i \neq j, \big(\widehat{J}_M\big)_{ij} \neq 0, \end{cases}$$

for some Lagrangian multipliers $\widehat{\alpha}_{ij} \geq 0$. The solution should also satisfy complementary slackness conditions $\widehat{\alpha}_{ij}.\big[\big|\big(\widehat{J}_M\big)_{ij}\big| - \lambda\big] = 0$ for $i \neq j$. Applying this condition to above $\widehat{Z}_\alpha$ representation, results to (25) form proposed in the lemma. ■

Appendix C. Proof of Theorem 4

First note that as mentioned in Remark 3, the pair $\big(\widehat{J}_M, \widehat{\Sigma}_R\big)$ given by optimization program gives a decomposition $\Sigma^* = \widehat{J}_M^{-1} - \widehat{\Sigma}_R$ which is desired.

Next, in order to prove the equivalence, we show that there is a one to one correspondence between the specified conditions (A.0)-(A.3) for valid decomposition and the characterization of optimal solution of optimization program given in lemma 1. We go through each of these conditions one by one in the following lines.

Condition (A.0) is considered in optimization program as positive definiteness of Markov matrix J_M.

Condition (A.1) is exactly the primal constraint $\|J_M^*\|_{\infty,\text{off}} \leq \lambda$.

Condition (A.2) is exactly the same as relation (26) where diagonal entries of residual covariance matrix are zero and its off-diagonal entries can be nonzero only if the absolute

value of corresponding entry in Markov matrix takes the maximum value λ.
Condition (A.3) is exactly the same as inequality $\widehat{\alpha}_{ij} \geq 0$.
In the above lines, we covered one by one correspondence for conditions (A.0)-(A.3). But note that we also covered all the equalities and inequalities that characterize unique optimal solution of optimization program. In other words by above correspondence we proved that both of the following derivations are true where second one is the reverse of first one. On one hand, any optimal solution of optimization program gives a valid decomposition under desired conditions. On the other hand, any valid decomposition under desired conditions is a solution of proposed optimization program. Thus, we can infer that these two are exactly equivalent and the result is proved. Since the solution of optimization program is unique and according to the equivalence between this solution and decomposition under those conditions, uniqueness is also established. ■

Appendix D. Proof of Theorem 5

In this appendix, we first mention an outline of the primal-dual witness method and then provide the detailed proof of the theorem.

D.1 Primal-Dual Witness Method

First, continuing the proof outline presented in section 6, we provide an outline of the primal-dual witness method steps in order to establish equivalence between optimal solutions of the original (19) and the modified (21) optimization programs.

1. The primal witness matrix $\widetilde{J}_M$ is defined as in (21).
2. The dual witness matrix is set as $\widetilde{Z} = -\widehat{\Sigma}^n + \widetilde{J}_M^{-1}$. It is defined in this way to satisfy original program optimal solution characterization mentioned in appendix B.
3. We need to check the following feasibility conditions under which the modified program solution is equivalent to the solution of original one:
   (a) $\|\widetilde{J}_M\|_{\infty,\text{off},S} \leq \lambda$: Since we relaxed the ℓ_∞ bounds on off-diagonal entries in set S, we need to make sure that the modified solution satisfies this bound in order to have equivalence between modified and original programs solutions.
   (b) Set $\big(\widetilde{Z}_\alpha\big)_{S_R} = \big(-\widehat{\Sigma}^n + \widetilde{J}_M^{-1} - \gamma\big(\widetilde{Z}_\gamma\big)\big)_{S_R}$ where $\widetilde{Z}_\gamma \in \partial\|\widetilde{J}_M\|_{1,\text{off}}$. Note that since $|\big(\widetilde{J}_M\big)_{ij}| = \lambda \neq 0$ for any $(i,j) \in S_R$, then $\widetilde{Z}_\gamma$ and therefore above equation is well-defined. Now we need to check: $\big(\widetilde{Z}_\alpha\big)_{ij}\big(\widetilde{J}_M\big)_{ij} \geq 0$ for all $(i,j) \in S_R$. This means that they have the same sign or one of them is zero. We need this condition for equivalence between solutions because Lagrangian multipliers in original program (19) corresponding to inequality constraints should be nonnegative.
   (c) $\|\widetilde{Z}\|_{\infty,S_M^c} < \gamma$: According to the $\big(J_M\big)_{S_M^c} = 0$ constraint in the modified program, all the inequality constraints become inactive in the original one when desired $\widehat{J}_M = \widetilde{J}_M$ equality is satisfied. Then, complementary slackness condition enforce all the Lagrangian multipliers corresponding to set S_M^c to be zero. These can be satisfied by the above strict dual feasibility. Also note that having

zero Lagrangian multipliers results in zero residual entries, i.e., $\big(\widetilde{\Sigma}_R\big)_{S_M^c} = 0$ and therefore $\|\widetilde{\Delta}_R\|_{\infty,S_M^c} = 0$ when this feasibility condition is satisfied.

Also note that we dropped the positive-definiteness constraint $\Sigma_M - \Sigma_R \succ 0$ in the proof outline. Thus, in addition to above conditions, we also need to show that $\widetilde{\Sigma} = \widetilde{\Sigma}_M - \widetilde{\Sigma}_R \succ 0$ in the modified program.

Before we state the detailed proof for the theorem, we introduce a pair of definitions which are used in the analysis. Let us define matrix E as difference between sample covariance matrix and the exact covariance matrix

$$E := \widehat{\Sigma}^n - \Sigma^*. \tag{29}$$

We also define $R\big(\widetilde{\Delta}_J\big)$ as the difference between $\widetilde{J}_M^{-1}$ and its first order Taylor expansion around J_M^*. Recall that $\widetilde{\Delta}_J$ was defined as $\widetilde{\Delta}_J := \widetilde{J}_M - J_M^*$. According to results for first order derivative of inverse function J_M^{-1} (Boyd and Vandenberghe, 2004), the remainder is

$$R\big(\widetilde{\Delta}_J\big) = \widetilde{J}_M^{-1} - {J_M^*}^{-1} + {J_M^*}^{-1}\widetilde{\Delta}_J {J_M^*}^{-1}. \tag{30}$$

D.2 Proof of the Theorem

Exploiting lemmata mentioned in Appendix E, Theorem 5 is proved as follows:

Proof According to the sample error bound mentioned in Lemma 4, we have $\|E\|_\infty \leq \overline{\delta}_f(p^\tau; n)$ for some $\tau > 2$ with probability greater than or equal to $1 - 1/p^{\tau-2}$. In the discussion after this, it is assumed that the above bound for $\|E\|_\infty$ is satisfied and therefore the following results are valid with probability greater than or equal to $1 - 1/p^{\tau-2}$.

By choosing $\gamma = \frac{m}{\alpha}\overline{\delta}_f(p^\tau; n)$, we have $\|E\|_\infty \leq \frac{\alpha}{m}\gamma$ as desired for Lemma 5. Choosing λ_δ as in (36) (compatible with what mentioned in the theorem), we only need to show that the other bound on $\|R\|_\infty$ is also satisfied to be able to apply Lemma 5. As stated in the remark after Theorem 5, the bound on sample complexity is not asymptotic and we assume the following lower bound on the number of samples which is compatible with the asymptotic form mentioned in the theorem:

$$n > \overline{n}_f\left(p^\tau; 1/\max\left\{v_*, 4ld\Big(1+\frac{m}{\alpha}\Big)K_{SS}K_M \max\Big\{1, \frac{4}{l-1}\Big(1+\frac{m}{\alpha}\Big)K_{SS}K_M^2\Big\}\right\}\right), \tag{31}$$

for some $l > 1$. Because of monotonic behaviour of the tail function, for any n satisfying above bound, we have:

$$\overline{\delta}_f\big(p^\tau; n\big) \leq \min\left\{\frac{1}{v_*}, \frac{1}{4ld(1+\frac{m}{\alpha})K_{SS}K_M}, \frac{l-1}{16ld(1+\frac{m}{\alpha})^2K_{SS}^2K_M^3}\right\}, \tag{32}$$

According to the selection for regularization parameters λ_δ and γ and the bound on sample error $\|E\|_\infty$, we have:

$$r := 2K_{SS_R}\lambda_\delta + 2K_{SS}\big(\|E\|_\infty + \gamma\big) \leq \left[\frac{4K_{SS_R}K_{SS}}{1-2K_{SS_R}}\Big(1+\frac{\alpha}{m}\Big)\frac{m}{\alpha} + 2K_{SS}\Big(1+\frac{m}{\alpha}\Big)\right]\overline{\delta}_f(p^\tau; n)$$

$$
\begin{aligned}
&= 2K_{SS}\Big(1+\frac{m}{\alpha}\Big)\overline{\delta}_f(p^\tau;n)\frac{1}{1-2K_{SS_R}} \quad (=:\lambda_\delta)\\
&< 4K_{SS}\Big(1+\frac{m}{\alpha}\Big)\overline{\delta}_f(p^\tau;n),
\end{aligned}
$$

where in the last inequality, we used the second condition is assumption (A.4) that $K_{SS_R} < 1/4$. Note that second line is equal to λ_δ since we assigned the same value in (36). Applying the bound (32) on above inequality, we have

$$
\begin{aligned}
2K_{SS_R}\lambda_\delta + 2K_{SS}\big(\|E\|_\infty+\gamma\big) &< \min\left\{\frac{1}{ldK_M}, \frac{l-1}{4ld(1+\frac{m}{\alpha})K_{SS}K_M^3}\right\}\\
&\le \min\left\{\frac{1}{ldK_M}, \frac{l-1}{2ldK_{SS}K_M^3}\right\}.
\end{aligned}
$$

Thus, the conditions for Lemma 7 are satisfied and we have

$$
\|\widetilde{\Delta}_J\|_{\infty,S} \le 2K_{SS_R}\lambda_\delta + 2K_{SS}\big(\|E\|_\infty+\gamma\big) \le \lambda_\delta < 4K_{SS}\Big(1+\frac{m}{\alpha}\Big)\overline{\delta}_f(p^\tau;n). \tag{33}
$$

Above inequalities tell us multiple things. First, since the error $\|\widetilde{\Delta}_J\|_{\infty,S}$ is bounded by λ_δ, the $\widetilde{J}_M$ entries in set S can not deviate from exact one J_M^* more than λ_δ. We also assumed that the off-diagonal entries in J_M^* are bounded by λ^*. Therefore according to the definition of $\lambda_\delta := \lambda - \lambda^*$, the entries in $\big(\widetilde{J}_M\big)_{\text{off},S}$ are bounded by λ and therefore the condition (a) for feasibility of primal-dual witness method is satisfied, i.e., we have $\|\widetilde{J}_M\|_{\infty,\text{off},S} \le \lambda$. Second, since $\|\widetilde{\Delta}_J\|_{\infty,S_R} = \lambda_\delta$, we have $\|\widetilde{\Delta}_J\|_{\infty,S} \le \|\widetilde{\Delta}_J\|_{\infty,S_R}$ and therefore $\|\widetilde{\Delta}_J\|_\infty = \|\widetilde{\Delta}_J\|_{\infty,S_R} = \lambda_\delta$ which results the following error bound

$$
\|\widetilde{\Delta}_J\|_\infty := \|\widetilde{J}_M - J_M^*\|_\infty \le 4K_{SS}\Big(1+\frac{m}{\alpha}\Big)\overline{\delta}_f(p^\tau;n). \tag{34}
$$

Furthermore, $\|\widetilde{\Delta}_J\|_\infty < \frac{1}{ldK_M}$ bound can be concluded from above inequality by substituting $\overline{\delta}_f(p^\tau;n)$ from (32). Thus, the condition for Lemma 6 is satisfied and we have the following bound on the remainder term

$$
\begin{aligned}
\|R(\widetilde{\Delta}_J)\|_\infty &\le \frac{l}{l-1}d\|\widetilde{\Delta}_J\|_\infty^2 K_M^3\\
&\le \frac{16l}{l-1}dK_M^3K_{SS}^2\Big(1+\frac{m}{\alpha}\Big)^2\big[\overline{\delta}_f(p^\tau;n)\big]^2\\
&= \left[\frac{16l}{l-1}dK_M^3K_{SS}^2\Big(1+\frac{m}{\alpha}\Big)^2\overline{\delta}_f(p^\tau;n)\right]\overline{\delta}_f(p^\tau;n)\\
&\le \overline{\delta}_f(p^\tau;n) = \frac{\alpha}{m}\gamma,
\end{aligned}
$$

where in the second inequality, we used error bound in (34) and the last inequality is concluded from bound (32).

Now the conditions for Lemma 5 are satisfied and therefore we have the upper bound on $\|\widetilde{\Delta}_R\|_{\infty,S_R} < C_3\gamma$ and the strict dual feasibility on S_M^c. Second result satisfies condition (c) of the primal-dual witness method feasibility conditions. The upper bound on $\|\widetilde{\Delta}_R\|_{\infty,S_R}$ in

conjunction with the lower bound on $\left(\Sigma_R^*\right)_{\min} > C_3\gamma$ (mentioned in the theorem), ensures that the sign of Σ_R^* and $\widetilde{\Sigma}_R$ are the same which results that the condition (b) of the feasibility conditions for primal-dual witness method is satisfied. Since all three conditions (a)-(c) are satisfied, we have equivalence between the modified program and the original one under conditions specified in the theorem. It gives us both results (a) and (b) in the theorem. Then by assuming lower bound on minimum nonzero value of J_M^*, the result in part (c) is also proved.

As mentioned before, we need to show that the dropped constraint $\Sigma = \Sigma_M - \Sigma_R \succ 0$ is also satisfied. Since the conditions for Corollary 13 in Appendix E.5 are satisfied, we have the spectral norm error bound (41) on overall covariance matrix Σ. Applying the inverse tail function for Gaussian distribution in (35) to assumption (A.6) results that the minimum eigenvalue of exact covariance matrix Σ^* satisfies lower bound $\lambda_{\min}(\Sigma^*) \geq \left(C_4 + \frac{m}{\alpha}C_3\right)d\overline{\delta}_f(p^\tau; n) + C_5 d^2\left[\overline{\delta}_f(p^\tau; n)\right]^2$ where $C_6 := \left(C_4 + \frac{m}{\alpha}C_3\right)\sqrt{2q^2}$ and $C_7 := 2q^2C_5$. Then by exploiting Weyl's theorem (Theorem 4.3.1 in Horn and Johnson (1985)), the estimated covariance matrix $\widehat{\Sigma}$ is positive definite and thus valid. Therefore, the result is proved. ■

Appendix E. Auxiliary Lemmata

First, the tail condition for a probability distribution is defined as follows.

Definition 12 (Tail Condition) *The random vector X satisfies tail condition with parameters f and v_* if there exists a constant $v_* \in (0, \infty)$ and function $f : \mathbb{N} \times (0, \infty) \to (0, \infty)$ such that for any $(i, j) \in V \times V$:*

$$\mathbb{P}[|\widehat{\Sigma}^n - \Sigma_{ij}^*| \geq \delta] \leq \frac{1}{f(n, \delta)} \text{ for all } \delta \in (0, \frac{1}{v_*}].$$

Note that since the function $f(n, \delta)$ is an increasing function of both variables n and δ, we define the inverse functions $\overline{n}_f(r; \delta)$ and $\overline{\delta}_f(r; n)$ with respect to variables n and δ respectively (when the other argument is fixed), where $f(n, \delta) = r$.

E.1 Concentration Bounds

From Lemma 1 in Ravikumar et al. (2011), we have the following concentration bound for the empirical covariance matrix of Gaussian random variables.

Lemma 3 (Ravikumar et al. 2011) *Consider a set of Gaussian random variables with covariance matrix Σ^*. Given n i.i.d. samples, the sample covariance matrix $\widehat{\Sigma}^n$ satisfies*

$$\mathbb{P}[|\widehat{\Sigma}_{ij}^n - \Sigma_{ij}^*| > \delta] \leq 4\exp\left\{-\frac{n\delta^2}{2q^2}\right\} \text{ for all } \delta \in (0, q),$$

for some constant $q > 0$.

Thus the tail function for Gaussian random vector takes the exponential form with the following corresponding inverse functions:

$$\overline{n}_f(r;\delta) = \frac{2q^2\log(4r)}{\delta^2}, \quad \overline{\delta}_f(r;n) = \sqrt{\frac{2q^2\log(4r)}{n}} \tag{35}$$

Applying above Lemma, we get the following bound for sampling error.

Lemma 4 (Ravikumar et al. 2011) *For any $\tau > 2$ and sample size n such that $\overline{\delta}_f(p^\tau;n) < 1/v_*$, we have*

$$\mathbb{P}\big[\|E\|_\infty \geq \overline{\delta}_f(p^\tau;n)\big] \leq \frac{1}{p^{\tau-2}} \to 0.$$

E.2 Feasibility Conditions

In the following lemma, we propose some conditions to bound the residual error $\|\widetilde{\Delta}_R\|_{\infty,S_R}$ and also satisfy the condition (c) of feasibility conditions required for equivalence between the witness solution and the original one.

Lemma 5 *Suppose that*

$$\begin{aligned} \max\{\|R\|_\infty, \|E\|_\infty\} &\leq \frac{\alpha}{m}\gamma, \\ \lambda_\delta &= \frac{2K_{SS}}{1-2K_{SS_R}}\Big(1+\frac{\alpha}{m}\Big)\gamma, \end{aligned} \tag{36}$$

then

a) $\|\widetilde{\Delta}_R\|_{\infty,S_R} \leq C_3\gamma$ *for some* $C_3 > 0$.

b) $\|\widetilde{Z}\|_{\infty,S_M^c} < \gamma$.

Proof Applying definitions (29) and (30) to optimality condition considered in second step of primal-dual witness method construction, gives the following equivalent equation

$$J_M^{*-1}\widetilde{\Delta}_J J_M^{*-1} - \Sigma_R^* - R\big(\widetilde{\Delta}_J\big) + E + \widetilde{Z} = 0. \tag{37}$$

Above equation is a $p \times p$ matrix equation. We can rewrite it as a linear equation with size p^2 if we use the vectorized form of matrices. Vectorized form of a matrix $D \in \mathbb{R}^{p\times p}$ is a column vector $\overline{D} \in \mathbb{R}^{p^2}$ which is composed by concatenating the rows of matrix D in a single column vector. In the vectorized form, we have

$$\mathrm{vec}\big(J_M^{*-1}\widetilde{\Delta}_J J_M^{*-1}\big) = \big(J_M^{*-1}\otimes J_M^{*-1}\big)\overline{\widetilde{\Delta}}_J = \Gamma^*\overline{\widetilde{\Delta}}_J.$$

Decomposing the vectorized form of (37) into three disjoint partitions S, S_R and S_M^c gives the following decomposed form

$$\begin{bmatrix} \Gamma^*_{SS} & \Gamma^*_{SS_R} & \Gamma^*_{SS_M^c} \\ \Gamma^*_{S_RS} & \Gamma^*_{S_RS_R} & \Gamma^*_{S_RS_M^c} \\ \Gamma^*_{S_M^cS} & \Gamma^*_{S_M^cS_R} & \Gamma^*_{S_M^cS_M^c} \end{bmatrix} \begin{bmatrix} \big(\overline{\widetilde{\Delta}}_J\big)_S \\ \overrightarrow{\lambda_\delta} \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ \big(\overline{\Sigma}^*_R\big)_{S_R} \\ 0 \end{bmatrix} + \begin{bmatrix} \big(-\overline{R}+\overline{E}+\overline{\widetilde{Z}}\big)_S \\ \big(-\overline{R}+\overline{E}+\overline{\widetilde{Z}}\big)_{S_R} \\ \big(-\overline{R}+\overline{E}+\overline{\widetilde{Z}}\big)_{S_M^c} \end{bmatrix} = 0, \tag{38}$$

where we used the equalities $\big(\overline{\widetilde{\Delta}}_J\big)_{S_R} = \overrightarrow{\lambda_\delta}$ and $\big(\overline{\widetilde{\Delta}}_J\big)_{S_M^c} = 0$. Note that vector $\overrightarrow{\lambda_\delta}$ only includes $\pm\lambda_\delta$ entries according to the constraints in the modified program. Also note that Σ^*_R is zero in sets S and S_M^c. We also dropped the argument $\widetilde{\Delta}_J$ from remainder function $R\big(\widetilde{\Delta}_J\big)$ to simplify the notation.

Similar to the original program, the matrix $\widetilde{Z}$ is composed of two parts, $\widetilde{Z}_\beta$ and $\widetilde{Z}_\gamma$, i.e., $\widetilde{Z} = \widetilde{Z}_\beta + \gamma\widetilde{Z}_\gamma$. Matrix $\widetilde{Z}_\beta = \widetilde{\Sigma}_R$ from equation (22), includes Lagrangian multipliers and $\widetilde{Z}_\gamma \in \partial\|\widetilde{J}_M\|_{1,\text{off}}$. For set S, $\big(\widetilde{Z}_\beta\big)_S = 0$, since we don't have any constraint in the program and therefore the Lagrangian multipliers are zero. Applying this to the first row of equation (38) and since Γ^*_{SS} is invertible, we have the following for error $\overline{\widetilde{\Delta}}_J$ in set S

$$\big(\overline{\widetilde{\Delta}}_J\big)_S = {\Gamma^*_{SS}}^{-1}\Big[-\Gamma^*_{SS_R}\overrightarrow{\lambda_\delta} + \overline{R}_S - \overline{E}_S - \gamma\big(\overline{\widetilde{Z}}_\gamma\big)_S\Big], \tag{39}$$

In set S_R, $\overline{\widetilde{Z}}_{S_R} = \big(\overline{\widetilde{\Sigma}}_R\big)_{S_R} + \gamma\big(\overline{\widetilde{Z}}_\gamma\big)_{S_R}$. Applying this to the second row of equation (38) results

$$\Gamma^*_{S_RS}\big(\overline{\widetilde{\Delta}}_J\big)_S + \Gamma^*_{S_RS_R}\overrightarrow{\lambda_\delta} + \big(\overline{\widetilde{\Delta}}_R\big)_{S_R} + \gamma\big(\overline{\widetilde{Z}}_\gamma\big)_{S_R} - \overline{R}_{S_R} + \overline{E}_{S_R} = 0,$$

Recall that we defined $\widetilde{\Delta}_R := \widetilde{\Sigma}_R - \Sigma^*_R$. Substituting (39) in above equation results the following for error $\overline{\widetilde{\Delta}}_R$ in set S_R

$$\begin{aligned}\big(\overline{\widetilde{\Delta}}_R\big)_{S_R} =& -\Gamma^*_{S_RS}{\Gamma^*_{SS}}^{-1}\Big[-\Gamma^*_{SS_R}\overrightarrow{\lambda_\delta} + \overline{R}_S - \overline{E}_S - \gamma\big(\overline{\widetilde{Z}}_\gamma\big)_S\Big]\\ &- \Gamma^*_{S_RS_R}\overrightarrow{\lambda_\delta} - \gamma\big(\overline{\widetilde{Z}}_\gamma\big)_{S_R} + \overline{R}_{S_R} - \overline{E}_{S_R}.\end{aligned}$$

Taking ℓ_∞ element-wise norm from above equation and using inequality $\|Ax\|_\infty \le |\!|\!|A|\!|\!|_\infty\|x\|_\infty$ for any matrix $A \in \mathbb{R}^{r\times s}$ and vector $x \in \mathbb{R}^s$, results the bound

$$\begin{aligned}\|\widetilde{\Delta}_R\|_{\infty,S_R} \le |\!|\!|-\Gamma^*_{S_RS}{\Gamma^*_{SS}}^{-1}\Gamma^*_{SS_R} + \Gamma^*_{S_RS_R}|\!|\!|_\infty\lambda_\delta + |\!|\!|\Gamma^*_{S_RS}{\Gamma^*_{SS}}^{-1}|\!|\!|_\infty\big[\|\overline{R}_S\|_\infty + \|\overline{E}_S\|_\infty + \gamma\big]\\ + \big(\|\overline{R}_{S_R}\|_\infty + \|\overline{E}_{S_R}\|_\infty + \gamma\big),\end{aligned}$$

where we used the fact that $\|\overrightarrow{\lambda_\delta}\|_\infty = \lambda_\delta$ and $\|\widetilde{Z}_\gamma\|_\infty = 1$. Now if we apply the assumptions mentioned in the lemma,

$$\begin{aligned}\|\widetilde{\Delta}_R\|_{\infty,S_R} \le \Bigg[\frac{2K_{SS}(m+\alpha)}{m(1-2K_{SS_R})}|\!|\!|-\Gamma^*_{S_RS}{\Gamma^*_{SS}}^{-1}\Gamma^*_{SS_R} + \Gamma^*_{S_RS_R}|\!|\!|_\infty\\ + \big(1+\frac{2\alpha}{m}\big)\big(1+|\!|\!|\Gamma^*_{S_RS}{\Gamma^*_{SS}}^{-1}|\!|\!|_\infty\big)\Bigg]\gamma = C_3\gamma,\end{aligned} \tag{40}$$

which proves part (a) of the Lemma.

Now if we substitute (39) in the equation from third row of (38), we have

$$\overline{\widetilde{Z}}_{S_M^c} = -\Gamma^*_{S_M^cS}{\Gamma^*_{SS}}^{-1}\Big[-\Gamma^*_{SS_R}\overrightarrow{\lambda_\delta} + \overline{R}_S - \overline{E}_S - \gamma\big(\overline{\widetilde{Z}}_\gamma\big)_S\Big] - \Gamma^*_{S_M^cS_R}\overrightarrow{\lambda_\delta} + \overline{R}_{S_M^c} - \overline{E}_{S_M^c}.$$

Taking ℓ_∞ element-wise norm from above equation gives the following bound

$$\|\widetilde{Z}\|_{\infty,S_M^c} \le |\!|\!|\Gamma^*_{S_M^cS}{\Gamma^*_{SS}}^{-1}\Gamma^*_{SS_R} - \Gamma^*_{S_M^cS_R}|\!|\!|_\infty\lambda_\delta + |\!|\!|\Gamma^*_{S_M^cS}{\Gamma^*_{SS}}^{-1}|\!|\!|_\infty\big[\|\overline{R}_S\|_\infty + \|\overline{E}_S\|_\infty + \gamma\big]$$

$$+ \|\overline{R}_{S_M^c}\|_\infty + \|\overline{E}_{S_M^c}\|_\infty,$$

where we used the fact that $\|\widetilde{Z}_\gamma\|_\infty = 1$. Applying assumption (A.4) to above bound results

$$\|\widetilde{Z}\|_{\infty,S_M^c} \leq (1-\alpha)\lambda_\delta + (2-\alpha)\left[\|R\|_\infty + \|E\|_\infty\right] + (1-\alpha)\gamma.$$

Using assumptions stated in the Lemma, we have

$$\begin{aligned}
\|\widetilde{Z}\|_{\infty,S_M^c} &\leq \left[\frac{2K_{SS}}{1-2K_{SS_R}}\Big(1+\frac{\alpha}{m}\Big)(1-\alpha) + (2-\alpha)\frac{2\alpha}{m} + (1-\alpha)\right]\gamma \\
&< \left[4K_{SS}\Big(1+\frac{\alpha}{m}\Big)(1-\alpha) + (2-\alpha)\frac{2\alpha}{m} + (1-\alpha)\right]\gamma \\
&< \left[4K_{SS}\frac{m-(m-1)\alpha}{m} + \frac{4\alpha}{m} + (1-\alpha)\right]\gamma \leq \gamma,
\end{aligned}$$

where we used the bound on K_{SS_R} in assumption (A.4) in the second inequality and the fact that $\alpha > 0$ in the third inequality. Final inequality is derived from assumption (A.5) which finishes the proof of part (b). ■

E.3 Control of Remainder

In the following Lemma which is stated and proved in lemma 5 in Ravikumar et al. (2011), the argument $\widetilde{\Delta}_J$ controls the remainder function behavior.

Lemma 6 *Suppose that the element-wise ℓ_∞ bound $\|\widetilde{\Delta}_J\|_\infty \leq \frac{1}{lK_M d}$ for some $l > 1$ holds. Then*

$$R\big(\widetilde{\Delta}_J\big) = \big({J_M^*}^{-1}\widetilde{\Delta}_J\big)^2 Q {J_M^*}^{-1},$$

where $Q := \sum_{k=0}^{\infty}(-1)^k\big({J_M^*}^{-1}\widetilde{\Delta}_J\big)^k$ with bound $\|\!|Q^T|\!\|_\infty \leq \frac{l}{l-1}$. Also, in terms of element-wise ℓ_∞ norm, we have

$$\|R\big(\widetilde{\Delta}_J\big)\|_\infty \leq \frac{l}{l-1} d\|\widetilde{\Delta}_J\|_\infty^2 K_M^3.$$

E.4 Control of $\widetilde{\Delta}_J$

According to the primal-dual witness solutions construction, we have the error bounds on $\widetilde{\Delta}_J$ within the sets S_R and S_M^c such that $\|\widetilde{\Delta}_J\|_{\infty,S_R} = \lambda_\delta$ and $\|\widetilde{\Delta}_J\|_{\infty,S_M^c} = 0$. In the following lemma, we propose some conditions to control the error $\|\widetilde{\Delta}_J\|_{\infty,S}$.

Lemma 7 *Suppose that*

$$r := 2K_{SS_R}\lambda_\delta + 2K_{SS}\big(\|E\|_\infty + \gamma\big) \leq \min\left\{\frac{1}{ldK_M}, \frac{l-1}{2ldK_{SS}K_M^3}\right\},$$

then we have the following element-wise ℓ_∞ bound for $\big(\widetilde{\Delta}_J\big)_S$,

$$\|\widetilde{\Delta}_J\|_{\infty,S} \leq r.$$

The proof is within the same lines of Lemma 6 proof in Ravikumar et al. (2011) but with some modifications since the error $\|\widetilde{\Delta}_J\|_{\infty,S_R}$ is not zero and therefore the nonzero value λ_δ arises in the final result. Since the modified optimization program (21) is different with the modified program in Ravikumar et al. (2011), it is worth discussing about existing a unique solution for the modified optimization program (21). This uniqueness can be shown with similar discussion presented in Appendix B for uniqueness of the solution of original program (19). We only need to show that there is no problem in uniqueness by removing the off-diagonal constaraints for set S in the modified program. By Lagrangian duality, the ℓ_1 penalty term $\gamma\|J_M\|_{1,\text{off}}$ can be moved to constraints as $\|J_M\|_{1,\text{off}} \leq C(\gamma)$ for some bounded $C(\gamma)$. Therefore, the off-diagonal entries in set S where the corresponding constraints were relaxed in the modified program are still bounded because of this ℓ_1 constraint. Hence, the modified program (21) has a unique solution.

E.5 Spectral Norm Error Bound on Overall Covariance Matrix $\Sigma = J_M^{-1} - \Sigma_R$

Corollary 13 *Under the same assumptions (excluding (A.6)) as Theorem 5, with probability greater than $1 - 1/p^c$, the overall covariance matrix estimate $\widehat{\Sigma} = \widehat{\Sigma}_M - \widehat{\Sigma}_R$ satisfies spectral norm error bound*

$$\|\!|\widehat{\Sigma} - \Sigma^*|\!\| \leq \Big(C_4 + \frac{m}{\alpha}C_3\Big)d\overline{\delta}_f(p^\tau;n) + C_5 d^2\big[\overline{\delta}_f(p^\tau;n)\big]^2. \tag{41}$$

Proof We first bound the spectral norm errors for the Markov and residual covariance matrices $\widehat{\Sigma}_M$ and $\widehat{\Sigma}_R$. Along the same lines as Corollary 4 proof in Ravikumar et al. (2011), the spectral norm error $\|\!|\widehat{\Sigma}_M - \Sigma_M^*|\!\|$ can be bounded as

$$\|\!|\widehat{\Sigma}_M - \Sigma_M^*|\!\| \leq C_4 d\overline{\delta}_f(p^\tau;n) + C_5 d^2\big[\overline{\delta}_f(p^\tau;n)\big]^2,$$

where $C_4 = 4\big(1+\frac{m}{\alpha}\big)K_{SS}K_M^2$ and $C_5 = \frac{16l}{l-1}\big(1+\frac{m}{\alpha}\big)^2 K_{SS}^2 K_M^3$.
The spectral norm error $\|\!|\widehat{\Sigma}_R - \Sigma_R^*|\!\|$ can be also bounded as

$$\|\!|\widehat{\Sigma}_R - \Sigma_R^*|\!\| \leq \|\!|\widehat{\Sigma}_R - \Sigma_R^*|\!\|_\infty \leq d\|\widehat{\Sigma}_R - \Sigma_R^*\|_\infty \leq \frac{m}{\alpha}C_3 d\overline{\delta}_f(p^\tau;n),$$

where the first inequality is the property of spectral norm which is bounded by ℓ_∞-operator norm, second inequality is a result of the fact that $\widehat{\Sigma}_R$ and Σ_R^* has at most d nonzero entries in each row (since $S_R \subset S_M$) and the last inequality is concluded from the upper bound on ℓ_∞ element-wise norm error on residual matrix estimation stated in part (a) of Theorem 5. Applying the above bounds to the overall covariance matrix estimation $\widehat{\Sigma} = \widehat{\Sigma}_M - \widehat{\Sigma}_R$ and using the triangular inequality for norms, the bound in (41) is proven. ■

Appendix F. Proof of Corollary 9

Proof The result in this corollary is a special case of general result in Theorem 5 when $\lambda^* = 0$ and some minor modifications are considered in problem formulation. Note that, it is expressed in assumption (A.1) that the off-diagonal entries of exact Markov matrix J_M^* are upper bounded by some positive λ^*. In order to extend the proof to the case of $\lambda^* = 0$ (The

case in this corollary), we need some minor modifications. First, the identifiability assumptions (A.0)-(A.3) can be ignored and instead it is assumed that the Markov part J_M^* (or equivalently Σ_M^*) is diagonal and the residual part Σ_R^* has only nonzero off-diagonal entries. Since the diagonal Markov matrix and off-diagonal residual matrix do not have any nonzero overlapping entries, it is natural that we do not require any more identifiability assumptions. Then, with these new assumptions, the set S_M is defined as $S_M := S_R \cup \{(i,i)|i = 1, ..., p\}$ where S_R is defined the same as (10) and also set S is defined the same as (11) which results that set S includes only diagonal entries. Thus, the off-diagonal entries belongs to sets S_R and S_M^c. Since Σ_M^* is a diagonal matrix, all submatrices of Γ^* which are indexed by sets S_R or S_M^c are complete zero matrices. The result is that the terms which are bounded in the mutual incoherence condition (A.4) are already zero and thus there is no need to consider those additional assumptions in the corollary.
By making these changes in the problem formulation, the result in Corollary 9 can be proven within the same lines of general result proof in Theorem 5. It is only required to change the constraint on set S_R in the modified optimization program to $\big(J_M\big)_{S_R} = \lambda \operatorname{sign}\Big(\big(\Sigma_R^*\big)_{S_R}\Big)$. ■

References

A. Anandkumar, V. Y. F. Tan, and A. S. Willsky. High-dimensional Gaussian graphical model selection: Tractable graph families. *Preprint, ArXiv 1107.1270*, June 2011.

T.W. Anderson. *An Introduction to Multivariate Statistical Analysis.* John Wiley & Sons, Inc., New York, NY, 1984.

M. Banerjee and T. Richardson. On a dualization of graphical Gaussian models: A correction note. *Scandinavian Journal of Statistics*, 30(4):817–820, 2003.

O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *J. of Machine Learning Research*, 9:485–516, March 2008.

P.J. Bickel and E. Levina. Covariance regularization by thresholding. *The Annals of Statistics*, 36(6):2577–2604, 2008.

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

L.D. Brown. Fundamentals of statistical exponential families: with applications in statistical decision theory. *Lecture Notes-Monograph Series, Institute of Mathematical Statistics*, 9, 1986.

T.T. Cai, C.H. Zhang, and H.H. Zhou. Optimal rates of convergence for covariance matrix estimation. *The Annals of Statistics*, 38(4):2118–2144, 2010.

E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *CoRR*, abs/0912.3599, 2009.

V. Chandrasekaran, S. Sanghavi, P.A. Parrilo, and A.S. Willsky. Rank-sparsity incoherence for matrix decomposition. *Arxiv preprint arXiv:0906.2220*, 2009.

V. Chandrasekaran, P.A. Parrilo, and A.S. Willsky. Latent variable graphical model selection via convex optimization. *Arxiv preprint arXiv:1008.1290*, 2010a.

V. Chandrasekaran, B. Recht, P.A. Parrilo, and A.S. Willsky. The convex geometry of linear inverse problems. *Arxiv preprint arXiv:1012.0621*, 2010b.

M.J. Choi, V. Chandrasekaran, and A.S. Willsky. Gaussian multiresolution models: Exploiting sparse Markov and covariance structure. *Signal Processing, IEEE Transactions on*, 58(3):1012–1024, 2010.

M.J. Choi, V.Y.F. Tan, A. Anandkumar, and A. Willsky. Learning latent tree graphical models. *J. of Machine Learning Research*, 12:1771–1812, May 2011.

T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 2006.

D.R. Cox and N. Wermuth. Linear dependencies represented by chain graphs. *Statistical Science*, 8(3):204–218, 1993.

A. d'Aspremont, O. Banerjee, and L. El Ghaoui. First-order methods for sparse covariance selection. *SIAM. J. Matrix Anal. & Appl.*, 30(56), 2008.

A.P. Dempster. Covariance selection. *Biometrics*, 28(1):157–175, 1972.

N. El Karoui. Operator norm consistent estimation of large-dimensional sparse covariance matrices. *The Annals of Statistics*, 36(6):2717–2756, 2008.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 2007.

R.A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, New York, NY, 1985.

J.Z. Huang, N. Liu, M. Pourahmadi, and L. Liu. Covariance matrix selection and estimation via penalised normal likelihood. *Biometrika*, 93(1), 2006.

M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *J. of Machine Learning Research*, 8:613–636, 2007.

G. Kauermann. On a dualization of graphical Gaussian models. *Scandinavian journal of statistics*, pages 105–116, 1996.

M. Kolar, A.P. Parikh, and E.P. Xing. On sparse nonparametric conditional covariance selection. In *International Conference on Machine Learning*, 2010.

C. Lam and J. Fan. Sparsistency and rates of convergence in large covariance matrix estimation. *Annals of statistics*, 37(6B):4254, 2009.

S.L. Lauritzen. *Graphical Models*. Clarendon Press, 1996.

H. Liu, J. Lafferty, and L. Wasserman. The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *J. of Machine Learning Research*, 10:2295–2328, 2009.

P.L. Loh and M.J. Wainwright. High-dimensional regression with noisy and missing data: Provable guarantees with non-convexity. In *Neural Information Processing*, 2011.

D.M. Malioutov, J.K. Johnson, and A.S. Willsky. Walk-sums and belief propagation in Gaussian graphical models. *J. of Machine Learning Research*, 7:2031–2064, 2006.

N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.

K. Mohan. ADMM algorithm for graphical lasso with an ℓ_∞ element-wise norm constraint. *arXiv:1311.7198*, Nov. 2013.

S. Negahban, P. Ravikumar, M.J. Wainwright, and B. Yu. A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. *Arxiv preprint arXiv:1010.2731*, 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*, (4):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.

M.K. Silva, N.J.A. Harvey, and C.M. Sato. Sparse sums of positive semidefinite matrices. *Arxiv preprint arXiv:1107.0088*, 2011.

P. Spirtes and C. Meek. Learning Bayesian networks with discrete variables from data. In *Proc. of Intl. Conf. on Knowledge Discovery and Data Mining*, pages 294–299, 1995.

AS Wagaman and E. Levina. Discovering sparse covariance structures with the isomap. *J. of Computational and Graphical Statistics*, 18(3):551–572, 2009.

M.J. Wainwright and M.I. Jordan. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

T. Zhang. Adaptive forward-backward greedy algorithm for sparse learning with linear models. In *Proc. of NIPS*, 2008.

T. Zhang. On the consistency of feature selection using greedy least squares regression. *J. of Machine Learning Research*, 10:555–568, 2009.

P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, 37(6A):3468–3497, 2009.

Journal of Machine Learning Research 15 (2014) 1593-1623 Submitted 11/11; Revised 10/13; Published 4/14

The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo

Matthew D. Hoffman MATHOFFM@ADOBE.COM
Adobe Research
601 Townsend St.
San Francisco, CA 94110, USA

Andrew Gelman GELMAN@STAT.COLUMBIA.EDU
Departments of Statistics and Political Science
Columbia University
New York, NY 10027, USA

Editor: Anthanasios Kottas

Abstract

Hamiltonian Monte Carlo (HMC) is a Markov chain Monte Carlo (MCMC) algorithm that avoids the random walk behavior and sensitivity to correlated parameters that plague many MCMC methods by taking a series of steps informed by first-order gradient information. These features allow it to converge to high-dimensional target distributions much more quickly than simpler methods such as random walk Metropolis or Gibbs sampling. However, HMC's performance is highly sensitive to two user-specified parameters: a step size ϵ and a desired number of steps L. In particular, if L is too small then the algorithm exhibits undesirable random walk behavior, while if L is too large the algorithm wastes computation. We introduce the No-U-Turn Sampler (NUTS), an extension to HMC that eliminates the need to set a number of steps L. NUTS uses a recursive algorithm to build a set of likely candidate points that spans a wide swath of the target distribution, stopping automatically when it starts to double back and retrace its steps. Empirically, NUTS performs at least as efficiently as (and sometimes more efficiently than) a well tuned standard HMC method, without requiring user intervention or costly tuning runs. We also derive a method for adapting the step size parameter ϵ on the fly based on primal-dual averaging. NUTS can thus be used with no hand-tuning at all, making it suitable for applications such as BUGS-style automatic inference engines that require efficient "turnkey" samplers.

Keywords: Markov chain Monte Carlo, Hamiltonian Monte Carlo, Bayesian inference, adaptive Monte Carlo, dual averaging

1. Introduction

Hierarchical Bayesian models are a mainstay of the machine learning and statistics communities. Exact posterior inference in such models is rarely tractable, however, and so researchers and practitioners must usually resort to approximate statistical inference methods. Deterministic approximate inference algorithms (for example, those reviewed by Wainwright and Jordan 2008) can be efficient, but introduce bias and can be difficult to apply to some models. Rather than computing a deterministic approximation to a target posterior (or other) distribution, Markov chain Monte Carlo (MCMC) methods offer schemes for

drawing a series of correlated samples that will converge in distribution to the target distribution (Neal, 1993). MCMC methods are sometimes less efficient than their deterministic counterparts, but are more generally applicable and are asymptotically unbiased.

Not all MCMC algorithms are created equal. For complicated models with many parameters, simple methods such as random-walk Metropolis (Metropolis et al., 1953) and Gibbs sampling (Geman and Geman, 1984) may require an unacceptably long time to converge to the target distribution. This is in large part due to the tendency of these methods to explore parameter space via inefficient random walks (Neal, 1993). When model parameters are continuous rather than discrete, Hamiltonian Monte Carlo (HMC), also known as hybrid Monte Carlo, is able to suppress such random walk behavior by means of a clever auxiliary variable scheme that transforms the problem of sampling from a target distribution into the problem of simulating Hamiltonian dynamics (Neal, 2011). The cost of HMC per independent sample from a target distribution of dimension D is roughly $O(D^{5/4})$, which stands in sharp contrast with the $O(D^2)$ cost of random-walk Metropolis (Creutz, 1988).

HMC's increased efficiency comes at a price. First, HMC requires the gradient of the log-posterior. Computing the gradient for a complex model is at best tedious and at worst impossible, but this requirement can be made less onerous by using automatic differentiation (Griewank and Walther, 2008). Second, HMC requires that the user specify at least two parameters: a step size ϵ and a number of steps L for which to run a simulated Hamiltonian system. A poor choice of either of these parameters will result in a dramatic drop in HMC's efficiency. Methods from the adaptive MCMC literature (see Andrieu and Thoms 2008 for a review) can be used to tune ϵ on the fly, but setting L typically requires one or more costly tuning runs, as well as the expertise to interpret the results of those tuning runs. This hurdle limits the more widespread use of HMC, and makes it challenging to incorporate HMC into a general-purpose inference engine such as BUGS (Gilks and Spiegelhalter, 1992), JAGS (`http://mcmc-jags.sourceforge.net`), Infer.NET (Minka et al.), HBC (Daume III, 2007), or PyMC (Patil et al., 2010).

The main contribution of this paper is the No-U-Turn Sampler (NUTS), an MCMC algorithm that closely resembles HMC, but eliminates the need to choose the problematic number-of-steps parameter L. We also provide a new dual averaging (Nesterov, 2009) scheme for automatically tuning the step size parameter ϵ in both HMC and NUTS, making it possible to run NUTS with no hand-tuning at all. We will show that the tuning-free version of NUTS samples as efficiently as (and sometimes more efficiently than) HMC, even ignoring the cost of finding optimal tuning parameters for HMC. Thus, NUTS brings the efficiency of HMC to users (and generic inference systems) that are unable or disinclined to spend time tweaking an MCMC algorithm.

Our algorithm has been implemented in C++ as part of the new open-source Bayesian inference package, Stan (Stan Development Team, 2013). Matlab code implementing the algorithms, along with Stan code for models used in our simulation study, are also available at `http://www.cs.princeton.edu/~mdhoffma/`.

2. Hamiltonian Monte Carlo

In Hamiltonian Monte Carlo (HMC) (Neal, 2011, 1993; Duane et al., 1987), we introduce an auxiliary momentum variable r_d for each model variable θ_d. In the usual implementation,

Algorithm 1 Hamiltonian Monte Carlo

Given θ^0, ϵ, L, $\mathcal{L}$, M:
for $m = 1$ to M **do**
  Sample $r^0 \sim \mathcal{N}(0, I)$.
  Set $\theta^m \leftarrow \theta^{m-1}, \tilde{\theta} \leftarrow \theta^{m-1}, \tilde{r} \leftarrow r^0$.
  for $i = 1$ to L **do**
    Set $\tilde{\theta}, \tilde{r} \leftarrow \text{Leapfrog}(\tilde{\theta}, \tilde{r}, \epsilon)$.
  end for
  With probability $\alpha = \min\left\{1, \frac{\exp\{\mathcal{L}(\tilde{\theta}) - \frac{1}{2}\tilde{r}\cdot\tilde{r}\}}{\exp\{\mathcal{L}(\theta^{m-1}) - \frac{1}{2}r^0\cdot r^0\}}\right\}$, set $\theta^m \leftarrow \tilde{\theta}$, $r^m \leftarrow -\tilde{r}$.
end for

function Leapfrog(θ, r, ϵ)
Set $\tilde{r} \leftarrow r + (\epsilon/2)\nabla_\theta \mathcal{L}(\theta)$.
Set $\tilde{\theta} \leftarrow \theta + \epsilon\tilde{r}$.
Set $\tilde{r} \leftarrow \tilde{r} + (\epsilon/2)\nabla_\theta \mathcal{L}(\tilde{\theta})$.
return $\tilde{\theta}, \tilde{r}$.

these momentum variables are drawn independently from the standard normal distribution, yielding the (unnormalized) joint density

$$p(\theta, r) \propto \exp\{\mathcal{L}(\theta) - \tfrac{1}{2} r \cdot r\},$$

where $\mathcal{L}$ is the logarithm of the joint density of the variables of interest θ (up to a normalizing constant) and $x \cdot y$ denotes the inner product of the vectors x and y. We can interpret this augmented model in physical terms as a fictitious Hamiltonian system where θ denotes a particle's position in D-dimensional space, r_d denotes the momentum of that particle in the dth dimension, $\mathcal{L}$ is a position-dependent negative potential energy function, $\frac{1}{2} r \cdot r$ is the kinetic energy of the particle, and $\log p(\theta, r)$ is the negative energy of the particle. We can simulate the evolution over time of the Hamiltonian dynamics of this system via the Störmer-Verlet ("leapfrog") integrator, which proceeds according to the updates

$$r^{t+\epsilon/2} = r^t + (\epsilon/2)\nabla_\theta \mathcal{L}(\theta^t); \quad \theta^{t+\epsilon} = \theta^t + \epsilon r^{t+\epsilon/2}; \quad r^{t+\epsilon} = r^{t+\epsilon/2} + (\epsilon/2)\nabla_\theta \mathcal{L}(\theta^{t+\epsilon}),$$

where r^t and θ^t denote the values of the momentum and position variables r and θ at time t and ∇_θ denotes the gradient with respect to θ. Since the update for each coordinate depends only on the other coordinates, the leapfrog updates are volume-preserving—that is, the volume of a region remains unchanged after mapping each point in that region to a new point via the leapfrog integrator.

A standard procedure for drawing M samples via Hamiltonian Monte Carlo is described in Algorithm 1. I denotes the identity matrix and $\mathcal{N}(\mu, \Sigma)$ denotes a multivariate normal distribution with mean μ and covariance matrix Σ. For each sample m, we first resample the momentum variables from a standard multivariate normal, which can be interpreted as a Gibbs sampling update. We then apply L leapfrog updates to the position and momentum variables θ and r, generating a proposal position-momentum pair $\tilde{\theta}, \tilde{r}$. We propose setting $\theta^m = \tilde{\theta}$ and $r^m = -\tilde{r}$, and accept or reject this proposal according to the Metropolis

algorithm (Metropolis et al., 1953). This is a valid Metropolis proposal because it is time-reversible and the leapfrog integrator is volume-preserving; using an algorithm for simulating Hamiltonian dynamics that did not preserve volume complicates the computation of the Metropolis acceptance probability (Lan et al., 2012). The negation of $\tilde{r}$ in the proposal is theoretically necessary to produce time-reversibility, but can be omitted in practice if one is only interested in sampling from $p(\theta)$.

The term $\log \frac{p(\tilde{\theta},\tilde{r})}{p(\theta,r)}$, on which the acceptance probability α depends, is the negative change in energy of the simulated Hamiltonian system from time 0 to time ϵL. If we could simulate the Hamiltonian dynamics exactly, then α would always be 1, since energy is conserved in Hamiltonian systems. The error introduced by using a discrete-time simulation depends on the step size parameter ϵ—specifically, the change in energy $|\log \frac{p(\tilde{\theta},\tilde{r})}{p(\theta,r)}|$ is proportional to ϵ^2 for large L, or ϵ^3 if $L = 1$ (Leimkuhler and Reich, 2004). In principle the error can grow without bound as a function of L, but it typically does not due to the symplecticness of the leapfrog discretization. This allows us to run HMC with many leapfrog steps, generating proposals for θ that have high probability of acceptance even though they are distant from the previous sample.

The performance of HMC depends strongly on choosing suitable values for ϵ and L. If ϵ is too large, then the simulation will be inaccurate and yield low acceptance rates. If ϵ is too small, then computation will be wasted taking many small steps. If L is too small, then successive samples will be close to one another, resulting in undesirable random walk behavior and slow mixing. If L is too large, then HMC will generate trajectories that loop back and retrace their steps. This is doubly wasteful, since work is being done to bring the proposal $\tilde{\theta}$ *closer* to the initial position θ^{m-1}. Worse, if L is chosen so that the parameters jump from one side of the space to the other each iteration, then the Markov chain may not even be ergodic (Neal, 2011). More realistically, an unfortunate choice of L may result in a chain that is ergodic but slow to move between regions of low and high density.

3. Eliminating the Need to Hand-Tune HMC

HMC is a powerful algorithm, but its usefulness is limited by the need to tune the step size parameter ϵ and number of steps L. Tuning these parameters for any particular problem requires some expertise, and usually one or more preliminary runs. Selecting L is particularly problematic; it is difficult to find a simple metric for when a trajectory is too short, too long, or "just right," and so practitioners commonly rely on heuristics based on autocorrelation statistics from preliminary runs (Neal, 2011).

Below, we present the No-U-Turn Sampler (NUTS), an extension of HMC that eliminates the need to specify a fixed value of L. In Section 3.2 we present schemes for setting ϵ based on the dual averaging algorithm of Nesterov (2009).

3.1 No-U-Turn Hamiltonian Monte Carlo

Our first goal is to devise an MCMC sampler that retains HMC's ability to suppress random walk behavior without the need to set the number L of leapfrog steps that the algorithm takes to generate a proposal. We need some criterion to tell us when we have simulated the dynamics for "long enough," that is, when running the simulation for more steps would

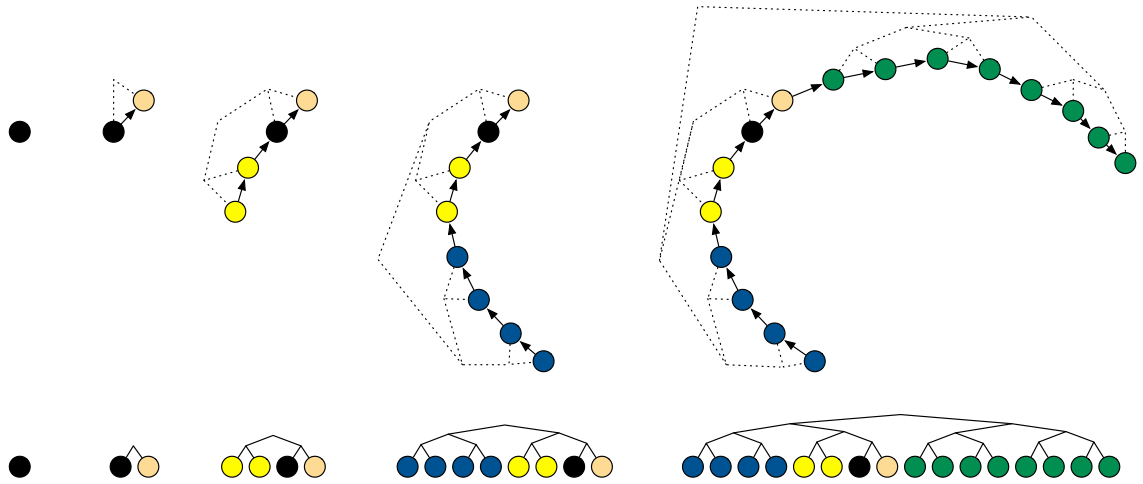


Figure 1: Example of building a binary tree via repeated doubling. Each doubling proceeds by choosing a direction (forwards or backwards in time) uniformly at random, then simulating Hamiltonian dynamics for 2^j leapfrog steps in that direction, where j is the number of previous doublings (and the height of the binary tree). The figures at top show a trajectory in two dimensions (with corresponding binary tree in dashed lines) as it evolves over four doublings, and the figures below show the evolution of the binary tree. In this example, the directions chosen were forward (light orange node), backward (yellow nodes), backward (blue nodes), and forward (green nodes).

no longer increase the distance between the proposal $\tilde{\theta}$ and the initial value of θ. We use a convenient criterion based on the dot product between $\tilde{r}$ (the current momentum) and $\tilde{\theta} - \theta$ (the vector from our initial position to our current position), which is the derivative with respect to time (in the Hamiltonian system) of half the squared distance between the initial position θ and the current position $\tilde{\theta}$:

$$\frac{d}{dt}\frac{(\tilde{\theta}-\theta)\cdot(\tilde{\theta}-\theta)}{2} = (\tilde{\theta}-\theta)\cdot\frac{d}{dt}(\tilde{\theta}-\theta) = (\tilde{\theta}-\theta)\cdot\tilde{r}. \tag{1}$$

In other words, if we were to run the simulation for an infinitesimal amount of additional time, then this quantity is proportional to the progress we would make away from our starting point θ.

This suggests an algorithm in which one runs leapfrog steps until the quantity in Equation 1 becomes less than 0; such an approach would simulate the system's dynamics until the proposal location $\tilde{\theta}$ started to move back towards θ. Unfortunately this algorithm does not guarantee time reversibility, and is therefore not guaranteed to converge to the correct distribution. NUTS overcomes this issue by means of a recursive algorithm that preserves reversibility by running the Hamiltonian simulation both forward and backward in time.

NUTS begins by introducing a slice variable u with conditional distribution $p(u|\theta, r) = \text{Uniform}(u; [0, \exp\{\mathcal{L}(\theta) - \frac{1}{2}r \cdot r\}])$, which renders the conditional distribution $p(\theta, r|u) =$

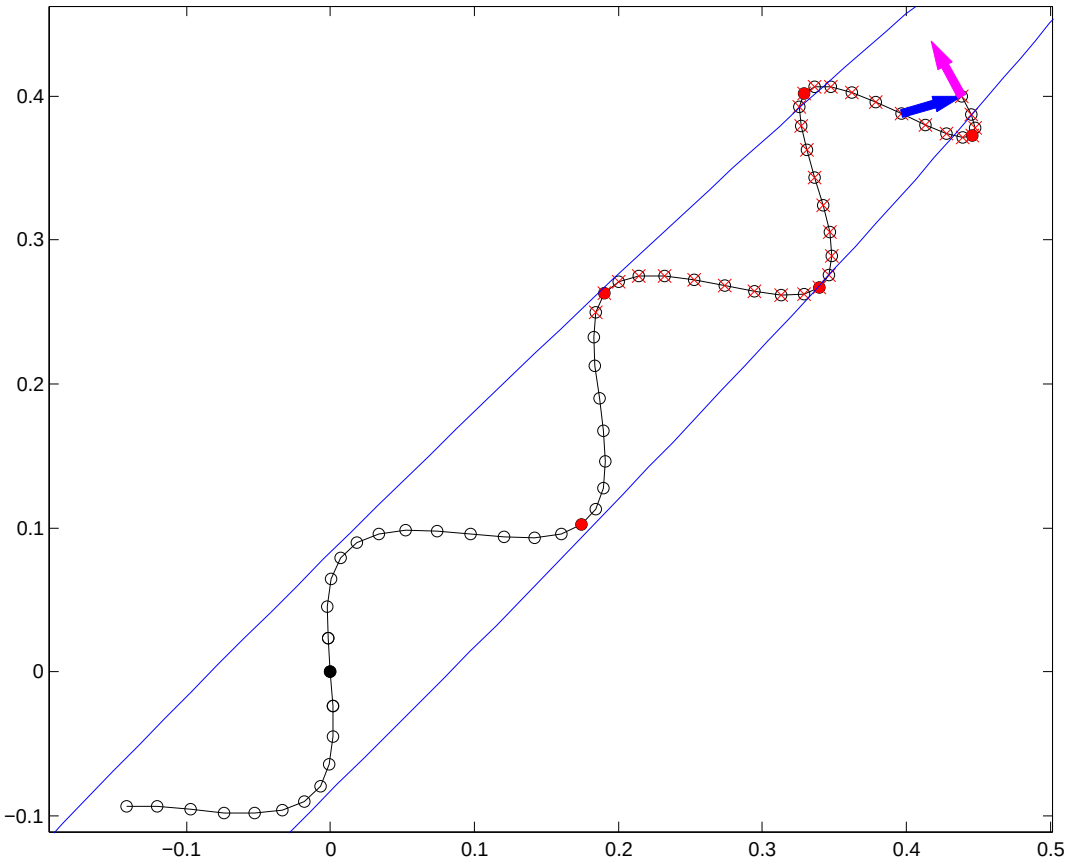


Figure 2: Example of a trajectory generated during one iteration of NUTS. The blue ellipse is a contour of the target distribution, the black open circles are the positions θ traced out by the leapfrog integrator and associated with elements of the set of visited states $\mathcal{B}$, the black solid circle is the starting position, the red solid circles are positions associated with states that must be excluded from the set $\mathcal{C}$ of possible next samples because their joint probability is below the slice variable u, and the positions with a red "x" through them correspond to states that must be excluded from $\mathcal{C}$ to satisfy detailed balance. The blue arrow is the vector from the positions associated with the leftmost to the rightmost leaf nodes in the rightmost height-3 subtree, and the magenta arrow is the (normalized) momentum vector at the final state in the trajectory. The doubling process stops here, since the blue and magenta arrows make an angle of more than 90 degrees. The crossed-out nodes with a red "x" are in the right half-tree, and must be ignored when choosing the next sample.

$\text{Uniform}(\theta, r; \{\theta', r' | \exp\{\mathcal{L}(\theta) - \frac{1}{2} r \cdot r\} \geq u\})$. This slice sampling step is not strictly necessary, but it simplifies both the derivation and the implementation of NUTS.

At a high level, after resampling $u|\theta, r$, NUTS uses the leapfrog integrator to trace out a path forwards and backwards in fictitious time, first running forwards or backwards 1 step, then forwards or backwards 2 steps, then forwards or backwards 4 steps, etc. This doubling process implicitly builds a balanced binary tree whose leaf nodes correspond to position-momentum states, as illustrated in Figure 1. The doubling is halted when the subtrajectory from the leftmost to the rightmost nodes of any balanced subtree of the overall binary tree starts to double back on itself (i.e., the fictional particle starts to make a "U-turn"). At this point NUTS stops the simulation and samples from among the set of points computed during the simulation, taking care to preserve detailed balance. Figure 2 illustrates an example of a trajectory computed during an iteration of NUTS.

Pseudocode implementing a efficient version of NUTS is provided in Algorithm 3. A detailed derivation follows below, along with a simplified version of the algorithm that motivates and builds intuition about Algorithm 3 (but uses much more memory and makes smaller jumps).

3.1.1 Derivation of Simplified NUTS Algorithm

NUTS further augments the model $p(\theta, r) \propto \exp\{\mathcal{L}(\theta) - \frac{1}{2} r \cdot r\}$ with a slice variable u (Neal, 2003). The joint probability of θ, r, and u is

$$p(\theta, r, u) \propto \mathbb{I}[u \in [0, \exp\{\mathcal{L}(\theta) - \tfrac{1}{2} r \cdot r\}]],$$

where $\mathbb{I}[\cdot]$ is 1 if the expression in brackets is true and 0 if it is false. The (unnormalized) marginal probability of θ and r (integrating over u) is

$$p(\theta, r) \propto \exp\{\mathcal{L}(\theta) - \tfrac{1}{2} r \cdot r\},$$

as in standard HMC. The conditional probabilities $p(u|\theta, r)$ and $p(\theta, r|u)$ are each uniform, so long as the condition $u \leq \exp\{\mathcal{L}(\theta) - \frac{1}{2} r \cdot r\}$ is satisfied.

We also add a finite set $\mathcal{C}$ of candidate position-momentum states and another finite set $\mathcal{B} \supseteq \mathcal{C}$ to the model. $\mathcal{B}$ will be the set of all position-momentum states that the leapfrog integrator traces out during a given NUTS iteration, and $\mathcal{C}$ will be the subset of those states to which we can transition without violating detailed balance. $\mathcal{B}$ will be built up by randomly taking forward and backward leapfrog steps, and $\mathcal{C}$ will selected deterministically from $\mathcal{B}$. The random procedure for building $\mathcal{B}$ and $\mathcal{C}$ given θ, r, u, and ϵ will define a conditional distribution $p(\mathcal{B}, \mathcal{C}|\theta, r, u, \epsilon)$, upon which we place the following conditions:

C.1: All elements of $\mathcal{C}$ must be chosen in a way that preserves volume. That is, any deterministic transformations of θ, r used to add a state θ', r' to $\mathcal{C}$ must have a Jacobian with unit determinant.

C.2: $p((\theta, r) \in \mathcal{C}|\theta, r, u, \epsilon) = 1$.

C.3: $p(u \leq \exp\{\mathcal{L}(\theta') - \frac{1}{2} r' \cdot r'\}|(\theta', r') \in \mathcal{C}) = 1$.

C.4: If $(\theta, r) \in \mathcal{C}$ and $(\theta', r') \in \mathcal{C}$ then for any $\mathcal{B}$, $p(\mathcal{B}, \mathcal{C}|\theta, r, u, \epsilon) = p(\mathcal{B}, \mathcal{C}|\theta', r', u, \epsilon)$.

C.1 ensures that $p(\theta, r|(\theta, r) \in \mathcal{C}) \propto p(\theta, r)$, that is, if we restrict our attention to the elements of $\mathcal{C}$ then we can treat the unnormalized probability density of a particular element

of $\mathcal{C}$ as an unnormalized probability mass. C.2 says that the current state θ, r must be included in $\mathcal{C}$. C.3 requires that any state in $\mathcal{C}$ be in the slice defined by u, that is, that any state $(\theta', r') \in \mathcal{C}$ must have equal (and positive) conditional probability density $p(\theta', r'|u)$. C.4 states that $\mathcal{B}$ and $\mathcal{C}$ must have equal probability of being selected regardless of the current state θ, r as long as $(\theta, r) \in \mathcal{C}$ (which it must be by C.2).

Deferring for the moment the question of how to construct and sample from a distribution $p(\mathcal{B}, \mathcal{C}|\theta, r, u, \epsilon)$ that satisfies these conditions, we will now show that the the following procedure leaves the joint distribution $p(\theta, r, u, \mathcal{B}, \mathcal{C}|\epsilon)$ invariant:

1. sample $r \sim \mathcal{N}(0, I)$,
2. sample $u \sim \text{Uniform}([0, \exp\{\mathcal{L}(\theta^t) - \frac{1}{2} r \cdot r\}])$,
3. sample $\mathcal{B}, \mathcal{C}$ from their conditional distribution $p(\mathcal{B}, \mathcal{C}|\theta^t, r, u, \epsilon)$,
4. sample $\theta^{t+1}, r \sim T(\theta^t, r, \mathcal{C})$,

where $T(\theta', r'|\theta, r, \mathcal{C})$ is a transition kernel that leaves the uniform distribution over $\mathcal{C}$ invariant, that is, T must satisfy

$$\frac{1}{|\mathcal{C}|} \sum_{(\theta,r)\in\mathcal{C}} T(\theta', r'|\theta, r, \mathcal{C}) = \frac{\mathbb{I}[(\theta', r') \in \mathcal{C}]}{|\mathcal{C}|}$$

for any θ', r'. The notation $\theta^{t+1}, r \sim T(\theta^t, r, \mathcal{C})$ denotes that we are resampling r in a way that depends on its current value.

Steps 1, 2, and 3 resample r, u, $\mathcal{B}$, and $\mathcal{C}$ from their conditional joint distribution given θ^t, and therefore together constitute a valid Gibbs sampling update. Step 4 is valid because the joint distribution of θ and r given $u, \mathcal{B}, \mathcal{C}$, and ϵ is uniform on the elements of $\mathcal{C}$:

$$\begin{aligned} p(\theta, r|u, \mathcal{B}, \mathcal{C}, \epsilon) &\propto p(\mathcal{B}, \mathcal{C}|\theta, r, u, \epsilon) p(\theta, r|u) \\ &\propto p(\mathcal{B}, \mathcal{C}|\theta, r, u, \epsilon) \mathbb{I}[u \leq \exp\{\mathcal{L}(\theta) - \tfrac{1}{2} r \cdot r\}] \\ &\propto \mathbb{I}[(\theta, r) \in \mathcal{C}]. \end{aligned} \tag{2}$$

Condition C.1 allows us to treat the unnormalized conditional density $p(\theta, r|u) \propto \mathbb{I}[u \leq \exp\{\mathcal{L}(\theta) - \frac{1}{2} r \cdot r\}]$ as an unnormalized conditional probability mass function. Conditions C.2 and C.4 ensure that $p(\mathcal{B}, \mathcal{C}|\theta, r, u, \epsilon) \propto \mathbb{I}[(\theta, r) \in \mathcal{C}]$ because by C.2 (θ, r) must be in $\mathcal{C}$, and by C.4 for any $\mathcal{B}, \mathcal{C}$ pair $p(\mathcal{B}, \mathcal{C}|\theta, r, u, \epsilon)$ is constant as a function of θ and r as long as $(\theta, r) \in \mathcal{C}$. Condition C.3 ensures that $(\theta, r) \in \mathcal{C} \Rightarrow u \leq \exp\{\mathcal{L}(\theta) - \frac{1}{2} r \cdot r\}$ (so the $p(\theta, r|u, \epsilon)$ term is redundant). Thus, Equation 2 implies that the joint distribution of θ and r given u and $\mathcal{C}$ is uniform on the elements of $\mathcal{C}$, and we are free to choose a new θ^{t+1}, r^{t+1} from any transition kernel that leaves this uniform distribution on $\mathcal{C}$ invariant.

We now turn our attention to the specific form for $p(\mathcal{B}, \mathcal{C}|\theta, r, u, \epsilon)$ used by NUTS. Conceptually, the generative process for building $\mathcal{B}$ proceeds by repeatedly doubling the size of a binary tree whose leaves correspond to position-momentum states. These states will constitute the elements of $\mathcal{B}$. The initial tree has a single node corresponding to the initial state. Doubling proceeds by choosing a random direction $v_j \sim \text{Uniform}(\{-1, 1\})$ and taking 2^j leapfrog steps of size $v_j \epsilon$ (i.e., forwards in fictional time if $v_j = 1$ and backwards in

fictional time if $v_j = -1$), where j is the current height of the tree. (The initial single-node tree is defined to have height 0.) For example, if $v_j = 1$, the left half of the new tree is the old tree and the right half of the new tree is a balanced binary tree of height j whose leaf nodes correspond to the 2^j position-momentum states visited by the new leapfrog trajectory. This doubling process is illustrated in Figure 1. Given the initial state θ, r and the step size ϵ, there are 2^j possible trees of height j that can be built according to this procedure, each of which is equally likely. Conversely, the probability of reconstructing a particular tree of height j starting from any leaf node of that tree is 2^{-j} regardless of which leaf node we start from.[1]

We cannot keep expanding the tree forever, of course. We want to continue expanding $\mathcal{B}$ until one end of the trajectory we are simulating makes a "U-turn" and begins to loop back towards another position on the trajectory. At that point continuing the simulation is likely to be wasteful, since the trajectory will retrace its steps and visit locations in parameter space close to those we have already visited. We also want to stop expanding $\mathcal{B}$ if the error in the simulation becomes extremely large, indicating that any states discovered by continuing the simulation longer are likely to have astronomically low probability. (This may happen if we use a step size ϵ that is too large, or if the target distribution includes hard constraints that make the log-density $\mathcal{L}$ go to $-\infty$ in some regions.)

The second rule is easy to formalize—we simply stop doubling if the tree includes a leaf node whose state θ, r satisfies

$$\mathcal{L}(\theta) - \frac{1}{2} r \cdot r - \log u < -\Delta_{\max} \tag{3}$$

for some nonnegative $\Delta_{\max}$. We recommend setting $\Delta_{\max}$ to a large value like 1000 so that it does not interfere with the algorithm so long as the simulation is even moderately accurate.

We must be careful when defining the first rule so that we can build a sampler that neither violates detailed balance nor introduces excessive computational overhead. To determine whether to stop doubling the tree at height j, NUTS considers the $2^j - 1$ balanced binary subtrees of the height-j tree that have height greater than 0. NUTS stops the doubling process when for one of these subtrees the states θ^-, r^- and θ^+, r^+ associated with the leftmost and rightmost leaves of that subtree satisfies

$$(\theta^+ - \theta^-) \cdot r^- < 0 \quad \text{or} \quad (\theta^+ - \theta^-) \cdot r^+ < 0. \tag{4}$$

That is, we stop if continuing the simulation an infinitesimal amount either forward or backward in time would reduce the distance between the position vectors θ^- and θ^+. Evaluating the condition in Equation 4 for each balanced subtree of a tree of height j requires $2^{j+1} - 2$ inner products, which is comparable to the number of inner products required by the $2^j - 1$ leapfrog steps needed to compute the trajectory. Except for very simple models with very little data, the cost of these inner products should be negligible compared to the cost of computing gradients.

1. This procedure resembles the doubling procedure devised by Neal (2003) to update scalar variables in a way that leaves their conditional distribution invariant. The doubling procedure finds a set of candidate points by repeatedly doubling the size of a segment of the real line containing the initial point. NUTS, by contrast, repeatedly doubles the size of a finite candidate set of vectors that contains the initial state.

This doubling process defines a distribution $p(\mathcal{B}|\theta, r, u, \epsilon)$. We now define a deterministic process for deciding which elements of $\mathcal{B}$ go in the candidate set $\mathcal{C}$, taking care to satisfy conditions C.1–C.4 on $p(\mathcal{B}, \mathcal{C}|\theta, r, u, \epsilon)$ laid out above. C.1 is automatically satisfied, since leapfrog steps are volume preserving and any element of $\mathcal{C}$ must be within some number of leapfrog steps of every other element of $\mathcal{C}$. C.2 is satisfied as long as we include the initial state θ, r in $\mathcal{C}$, and C.3 is satisfied if we exclude any element θ', r' of $\mathcal{B}$ for which $\exp\{\mathcal{L}(\theta') - \frac{1}{2} r' \cdot r'\} < u$. To satisfy condition C.4, we must ensure that $p(\mathcal{B}, \mathcal{C}|\theta, r, u, \epsilon) = p(\mathcal{B}, \mathcal{C}|\theta', r', u, \epsilon)$ for any $(\theta', r') \in \mathcal{C}$. For any start state $(\theta', r') \in \mathcal{B}$, there is at most one series of directions $\{v_0, \ldots, v_j\}$ for which the doubling process will reproduce $\mathcal{B}$, so as long as we choose $\mathcal{C}$ deterministically given $\mathcal{B}$ either $p(\mathcal{B}, \mathcal{C}|\theta', r', u, \epsilon) = 2^{-j} = p(\mathcal{B}, \mathcal{C}|\theta, r, u, \epsilon)$ or $p(\mathcal{B}, \mathcal{C}|\theta', r', u, \epsilon) = 0$. Thus, condition C.4 will be satisfied as long as we exclude from $\mathcal{C}$ any state θ', r' that could not have generated $\mathcal{B}$. The only way such a state can arise is if starting from θ', r' results in the stopping conditions in Equations 3 or 4 being satisfied before the entire tree has been built, causing the doubling process to stop too early. There are two cases to consider:

1. The doubling procedure was stopped because either equation 3 or Equation 4 was satisfied by a state or subtree added during the final doubling iteration. In this case we must exclude from $\mathcal{C}$ any element of $\mathcal{B}$ that was added during this final doubling iteration, since starting the doubling process from one of these would lead to a stopping condition being satisfied before the full tree corresponding to $\mathcal{B}$ has been built.

2. The doubling procedure was stopped because equation 4 was satisfied for the leftmost and rightmost leaves of the full tree corresponding to $\mathcal{B}$. In this case no stopping condition was met by any state or subtree until $\mathcal{B}$ had been completed, and condition C.4 is automatically satisfied.

Algorithm 2 shows how to construct $\mathcal{C}$ incrementally while building $\mathcal{B}$. After resampling the initial momentum and slice variables, it uses a recursive procedure resembling a depth-first search that eliminates the need to explicitly store the tree used by the doubling procedure. The BuildTree() function takes as input an initial position θ and momentum r, a slice variable u, a direction $v \in \{-1, 1\}$, a depth j, and a step size ϵ. It takes 2^j leapfrog steps of size $v\epsilon$ (i.e., forwards in time if $v = 1$ and backwards in time if $v = -1$), and returns

1. the backwardmost and forwardmost position-momentum states θ^-, r^- and θ^+, r^+ among the 2^j new states visited;

2. a set $\mathcal{C}'$ of position-momentum states containing each newly visited state θ', r' for which $\exp\{\mathcal{L}(\theta') - \frac{1}{2} r' \cdot r'\} > u$; and

3. an indicator variable s; $s = 0$ indicates that a stopping criterion was met by some state or subtree of the subtree corresponding to the 2^j new states visited by BuildTree().

At the top level, NUTS repeatedly calls BuildTree() to double the number of points that have been considered until either BuildTree() returns $s = 0$ (in which case doubling stops and the new set $\mathcal{C}'$ that was just returned must be ignored) or Equation 4 is satisfied for the new backwardmost and forwardmost position-momentum states θ^-, r^- and θ^+, r^+ yet considered (in which case doubling stops but we can use the new set $\mathcal{C}'$). Finally, we select

Algorithm 2 Naive No-U-Turn Sampler

Given θ^0, ϵ, $\mathcal{L}$, M:
for $m = 1$ to M **do**
  Resample $r^0 \sim \mathcal{N}(0, I)$.
  Resample $u \sim \text{Uniform}([0, \exp\{\mathcal{L}(\theta^{m-1} - \frac{1}{2} r^0 \cdot r^0\}])$
  Initialize $\theta^- = \theta^{m-1}$, $\theta^+ = \theta^{m-1}$, $r^- = r^0$, $r^+ = r^0$, $j = 0$, $\mathcal{C} = \{(\theta^{m-1}, r^0)\}$, $s = 1$.
  while $s = 1$ **do**
    Choose a direction $v_j \sim \text{Uniform}(\{-1, 1\})$.
    if $v_j = -1$ **then**
      $\theta^-, r^-, -, -, \mathcal{C}', s' \leftarrow \text{BuildTree}(\theta^-, r^-, u, v_j, j, \epsilon)$.
    else
      $-, -, \theta^+, r^+, \mathcal{C}', s' \leftarrow \text{BuildTree}(\theta^+, r^+, u, v_j, j, \epsilon)$.
    end if
    if $s' = 1$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}'$.
    end if
    $s \leftarrow s' \mathbb{I}[(\theta^+ - \theta^-) \cdot r^- \geq 0] \mathbb{I}[(\theta^+ - \theta^-) \cdot r^+ \geq 0]$.
    $j \leftarrow j + 1$.
  end while
  Sample θ^m, r uniformly at random from $\mathcal{C}$.
end for

function BuildTree$(\theta, r, u, v, j, \epsilon)$
if $j = 0$ **then**
  Base case—take one leapfrog step in the direction v.
  $\theta', r' \leftarrow \text{Leapfrog}(\theta, r, v\epsilon)$.
  $\mathcal{C}' \leftarrow \begin{cases} \{(\theta', r')\} & \text{if } u \leq \exp\{\mathcal{L}(\theta') - \frac{1}{2} r' \cdot r'\} \\ \emptyset & \text{else} \end{cases}$
  $s' \leftarrow \mathbb{I}[\mathcal{L}(\theta') - \frac{1}{2} r' \cdot r' > \log u - \Delta_{\max}]$.
  return $\theta', r', \theta', r', \mathcal{C}', s'$.
else
  Recursion—build the left and right subtrees.
  $\theta^-, r^-, \theta^+, r^+, \mathcal{C}', s' \leftarrow \text{BuildTree}(\theta, r, u, v, j - 1, \epsilon)$.
  if $v = -1$ **then**
    $\theta^-, r^-, -, -, \mathcal{C}'', s'' \leftarrow \text{BuildTree}(\theta^-, r^-, u, v, j - 1, \epsilon)$.
  else
    $-, -, \theta^+, r^+, \mathcal{C}'', s'' \leftarrow \text{BuildTree}(\theta^+, r^+, u, v, j - 1, \epsilon)$.
  end if
  $s' \leftarrow s' s'' \mathbb{I}[(\theta^+ - \theta^-) \cdot r^- \geq 0] \mathbb{I}[(\theta^+ - \theta^-) \cdot r^+ \geq 0]$.
  $\mathcal{C}' \leftarrow \mathcal{C}' \cup \mathcal{C}''$.
  return $\theta^-, r^-, \theta^+, r^+, \mathcal{C}', s'$.
end if

the next position and momentum θ^m, r uniformly at random from $\mathcal{C}$, the union of all of the valid sets $\mathcal{C}'$ that have been returned, which clearly leaves the uniform distribution over $\mathcal{C}$ invariant.

To summarize, Algorithm 2 defines a transition kernel that leaves $p(\theta, r, u, \mathcal{B}, \mathcal{C}|\epsilon)$ invariant, and therefore leaves the target distribution $p(\theta) \propto \exp\{\mathcal{L}(\theta)\}$ invariant. It does so by resampling the momentum and slice variables r and u, simulating a Hamiltonian trajectory

forwards and backwards in time until that trajectory either begins retracing its steps or encounters a state with very low probability, carefully selecting a subset $\mathcal{C}$ of the states encountered on that trajectory that lie within the slice defined by the slice variable u, and finally choosing the next position and momentum variables θ^m and r uniformly at random from $\mathcal{C}$. Figure 2 shows an example of a trajectory generated by an iteration of NUTS where Equation 4 is satisfied by the height-3 subtree at the end of the trajectory. Below, we will introduce some improvements to algorithm 2 that boost the algorithm's memory efficiency and allow it to make larger jumps on average.

3.1.2 Efficient NUTS

Algorithm 2 requires $2^j - 1$ evaluations of $\mathcal{L}(\theta)$ and its gradient (where j is the number of times BuildTree() is called), and $O(2^j)$ additional operations to determine when to stop doubling. In practice, for all but the smallest problems the cost of computing $\mathcal{L}$ and its gradient still dominates the overhead costs, so the computational cost of algorithm 2 per leapfrog step is comparable to that of a standard HMC algorithm. However, Algorithm 2 also requires that we store 2^j position and momentum vectors, which may require an unacceptably large amount of memory. Furthermore, there are alternative transition kernels that satisfy detailed balance with respect to the uniform distribution on $\mathcal{C}$ that produce larger jumps on average than simple uniform sampling. Finally, if a stopping criterion is satisfied in the middle of the final doubling iteration then there is no point in wasting computation to build up a set $\mathcal{C}'$ that will never be used.

The third issue is easily addressed—if we break out of the recursion as soon as we encounter a zero value for the stop indicator s then the correctness of the algorithm is unaffected and we save some computation. We can address the second issue by using a more sophisticated transition kernel to move from one state $(\theta, r) \in \mathcal{C}$ to another state $(\theta', r') \in \mathcal{C}$ while leaving the uniform distribution over $\mathcal{C}$ invariant. This kernel admits a memory-efficient implementation that only requires that we store $O(j)$ position and momentum vectors, rather than $O(2^j)$.

Consider the transition kernel

$$T(w'|w, \mathcal{C}) = \begin{cases} \frac{\mathbb{I}[w' \in \mathcal{C}^{\text{new}}]}{|\mathcal{C}^{\text{new}}|} & \text{if } |\mathcal{C}^{\text{new}}| > |\mathcal{C}^{\text{old}}|, \\ \frac{|\mathcal{C}^{\text{new}}|}{|\mathcal{C}^{\text{old}}|} \frac{\mathbb{I}[w' \in \mathcal{C}^{\text{new}}]}{|\mathcal{C}^{\text{new}}|} + \left(1 - \frac{|\mathcal{C}^{\text{new}}|}{|\mathcal{C}^{\text{old}}|}\right) \mathbb{I}[w' = w] & \text{if } |\mathcal{C}^{\text{new}}| \leq |\mathcal{C}^{\text{old}}| \end{cases},$$

where w and w' are shorthands for position-momentum states (θ, r), $\mathcal{C}^{\text{new}}$ and $\mathcal{C}^{\text{old}}$ are disjoint subsets of $\mathcal{C}$ such that $\mathcal{C}^{\text{new}} \cup \mathcal{C}^{\text{old}} = \mathcal{C}$, and $w \in \mathcal{C}^{\text{old}}$. In English, T proposes a move from $\mathcal{C}^{\text{old}}$ to a random state in $\mathcal{C}^{\text{new}}$ and accepts the move with probability $\frac{|\mathcal{C}^{\text{new}}|}{|\mathcal{C}^{\text{old}}|}$. This is equivalent to a Metropolis-Hastings kernel with proposal distribution $q(w', \mathcal{C}^{\text{old}\prime}, \mathcal{C}^{\text{new}\prime} | w, \mathcal{C}^{\text{old}}, \mathcal{C}^{\text{new}}) \propto \mathbb{I}[w' \in \mathcal{C}^{\text{new}}] \mathbb{I}[\mathcal{C}^{\text{old}\prime} = \mathcal{C}^{\text{new}}] \mathbb{I}[\mathcal{C}^{\text{new}\prime} = \mathcal{C}^{\text{old}}]$, and it is straightforward to show that it satisfies detailed balance with respect to the uniform distribution on $\mathcal{C}$, that is,

$$p(w|\mathcal{C}) T(w'|w, \mathcal{C}) = p(w'|\mathcal{C}) T(w|w', \mathcal{C}),$$

and that T therefore leaves the uniform distribution over $\mathcal{C}$ invariant. If we let $\mathcal{C}^{\text{new}}$ be the (possibly empty) set of elements added to $\mathcal{C}$ during the final iteration of the doubling (i.e., those returned by the final call to BuildTree() and $\mathcal{C}^{\text{old}}$ be the older elements of $\mathcal{C}$,

then we can replace the uniform sampling of $\mathcal{C}$ at the end of Algorithm 2 with a draw from $T(\theta^t, r^t, \mathcal{C})$ and leave the uniform distribution on $\mathcal{C}$ invariant. In fact, we can apply T after *every* doubling, proposing a move to each new half-tree in turn. Doing so leaves the uniform distribution on each partially built $\mathcal{C}$ invariant, and therefore does no harm to the invariance of the uniform distribution on the fully built set $\mathcal{C}$. Repeatedly applying T in this way increases the probability that we will jump to a state θ^{t+1} far from the initial state θ^t; considering the process in reverse, it is as though we first tried to jump to the other side of $\mathcal{C}$, then if that failed tried to make a more modest jump, and so on. This transition kernel is thus akin to delayed-rejection MCMC methods (Tierney and Mira, 1999), but in this setting we can avoid the usual costs associated with evaluating new proposals.

The transition kernel above still requires that we be able to sample uniformly from the set $\mathcal{C}'$ returned by BuildTree(), which may contain as many as 2^{j-1} elements. In fact, we can sample from $\mathcal{C}'$ without maintaining the full set $\mathcal{C}'$ in memory by exploiting the binary tree structure in Figure 1. Consider a subtree of the tree explored in a call to BuildTree(), and let $\mathcal{C}_{\text{subtree}}$ denote the set of its leaf states that are in $\mathcal{C}'$: we can factorize the probability that a state $(\theta, r) \in \mathcal{C}_{\text{subtree}}$ will be chosen uniformly at random from $\mathcal{C}'$ as

$$\begin{aligned} p(\theta, r|\mathcal{C}') &= \frac{1}{|\mathcal{C}'|} = \frac{|\mathcal{C}_{\text{subtree}}|}{|\mathcal{C}'|}\frac{1}{|\mathcal{C}_{\text{subtree}}|} \\ &= p((\theta, r) \in \mathcal{C}_{\text{subtree}}|\mathcal{C})p(\theta, r|(\theta, r) \in \mathcal{C}_{\text{subtree}}, \mathcal{C}). \end{aligned}$$

That is, $p(\theta, r|\mathcal{C}')$ is the product of the probability of choosing some node from the subtree multiplied by the probability of choosing θ, r uniformly at random from $\mathcal{C}_{\text{subtree}}$. We use this observation to sample from $\mathcal{C}'$ incrementally as we build up the tree. Each subtree above the bottom layer is built of two smaller subtrees. For each of these smaller subtrees, we sample a θ, r pair from $p(\theta, r|(\theta, r) \in \mathcal{C}_{\text{subtree}})$ to represent that subtree. We then choose between these two pairs, giving the pair representing each subtree weight proportional to how many elements of $\mathcal{C}'$ are in that subtree. This continues until we have completed the subtree associated with $\mathcal{C}'$ and we have returned a sample θ' from $\mathcal{C}'$ and an integer weight n' encoding the size of $\mathcal{C}'$, which is all we need to apply T. This procedure only requires that we store $O(j)$ position and momentum vectors in memory, rather than $O(2^j)$, and requires that we generate $O(2^j)$ extra random numbers (a cost that again is usually very small compared with the $2^j - 1$ gradient computations needed to run the leapfrog algorithm).

Algorithm 3 implements all of the above improvements in pseudocode.

3.2 Adaptively Tuning ϵ

Having addressed the issue of how to choose the number of steps L, we now turn our attention to the step size parameter ϵ. To set ϵ for both NUTS and HMC, we propose using stochastic optimization with vanishing adaptation (Andrieu and Thoms, 2008), specifically an adaptation of the primal-dual algorithm of Nesterov (2009).

Perhaps the most commonly used vanishing adaptation algorithm in MCMC is the stochastic approximation method of Robbins and Monro (1951). Suppose we have a statistic H_t that describes some aspect of the behavior of an MCMC algorithm at iteration $t \geq 1$,

Algorithm 3 Efficient No-U-Turn Sampler

Given θ^0, ϵ, $\mathcal{L}$, M:
for $m = 1$ to M **do**
  Resample $r^0 \sim \mathcal{N}(0, I)$.
  Resample $u \sim \text{Uniform}([0, \exp\{\mathcal{L}(\theta^{m-1} - \frac{1}{2} r^0 \cdot r^0\}])$
  Initialize $\theta^- = \theta^{m-1}$, $\theta^+ = \theta^{m-1}$, $r^- = r^0, r^+ = r^0, j = 0, \theta^m = \theta^{m-1}, n = 1, s = 1$.
  while $s = 1$ **do**
    Choose a direction $v_j \sim \text{Uniform}(\{-1, 1\})$.
    if $v_j = -1$ **then**
      $\theta^-, r^-, -, -, \theta', n', s' \leftarrow \text{BuildTree}(\theta^-, r^-, u, v_j, j, \epsilon)$.
    else
      $-, -, \theta^+, r^+, \theta', n', s' \leftarrow \text{BuildTree}(\theta^+, r^+, u, v_j, j, \epsilon)$.
    end if
    if $s' = 1$ **then**
      With probability $\min\{1, \frac{n'}{n}\}$, set $\theta^m \leftarrow \theta'$.
    end if
    $n \leftarrow n + n'$.
    $s \leftarrow s' \mathbb{I}[(\theta^+ - \theta^-) \cdot r^- \geq 0] \mathbb{I}[(\theta^+ - \theta^-) \cdot r^+ \geq 0]$.
    $j \leftarrow j + 1$.
  end while
end for

function BuildTree$(\theta, r, u, v, j, \epsilon)$
if $j = 0$ **then**
  Base case—take one leapfrog step in the direction v.
  $\theta', r' \leftarrow \text{Leapfrog}(\theta, r, v\epsilon)$.
  $n' \leftarrow \mathbb{I}[u \leq \exp\{\mathcal{L}(\theta') - \frac{1}{2} r' \cdot r'\}]$.
  $s' \leftarrow \mathbb{I}[\mathcal{L}(\theta') - \frac{1}{2} r' \cdot r' > \log u - \Delta_{\max}]$
  return $\theta', r', \theta', r', \theta', n', s'$.
else
  Recursion—implicitly build the left and right subtrees.
  $\theta^-, r^-, \theta^+, r^+, \theta', n', s' \leftarrow \text{BuildTree}(\theta, r, u, v, j - 1, \epsilon)$.
  if $s' = 1$ **then**
    if $v = -1$ **then**
      $\theta^-, r^-, -, -, \theta'', n'', s'' \leftarrow \text{BuildTree}(\theta^-, r^-, u, v, j - 1, \epsilon)$.
    else
      $-, -, \theta^+, r^+, \theta'', n'', s'' \leftarrow \text{BuildTree}(\theta^+, r^+, u, v, j - 1, \epsilon)$.
    end if
    With probability $\frac{n''}{n' + n''}$, set $\theta' \leftarrow \theta''$.
    $s' \leftarrow s'' \mathbb{I}[(\theta^+ - \theta^-) \cdot r^- \geq 0] \mathbb{I}[(\theta^+ - \theta^-) \cdot r^+ \geq 0]$
    $n' \leftarrow n' + n''$
  end if
  return $\theta^-, r^-, \theta^+, r^+, \theta', n', s'$.
end if

and define its expectation $h(x)$ as

$$h(x) \equiv \mathbb{E}_t[H_t|x] \equiv \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[H_t|x],$$

where $x \in \mathbb{R}$ is a tunable parameter to the MCMC algorithm. For example, if α_t is the Metropolis acceptance probability for iteration t, we might define $H_t = \delta - \alpha_t$, where δ is the desired average acceptance probability. If h is a nondecreasing function of x and a few other conditions such as boundedness of the iterates x_t are met (see Andrieu and Thoms 2008 for details), the update

$$x_{t+1} \leftarrow x_t - \eta_t H_t$$

is guaranteed to cause $h(x_t)$ to converge to 0 as long as the step size schedule defined by η_t satisfies the conditions

$$\sum_t \eta_t = \infty; \quad \sum_t \eta_t^2 < \infty. \tag{5}$$

These conditions are satisfied by schedules of the form $\eta_t \equiv t^{-\kappa}$ for $\kappa \in (0.5, 1]$. As long as the per-iteration impact of the adaptation goes to 0 (as it will if $\eta_t \equiv t^{-\kappa}$ and $\kappa > 0$) the asymptotic behavior of the sampler is unchanged. That said, in practice x often gets "close enough" to an optimal value well before the step size η has gotten close enough to 0 to avoid disturbing the Markov chain's stationary distribution. A common practice, which we follow here, is to adapt any tunable MCMC parameters during the warmup phase, and freeze the tunable parameters afterwards (e.g., Gelman et al., 2004). For the present paper, the step size ϵ is the only tuning parameter x in the algorithm. More advanced implementations could have more options, though, so we consider the tuning problem more generally.

3.2.1 Dual Averaging

The optimal values of the parameters to an MCMC algorithm during the warmup phase and the stationary phase are often quite different. Ideally those parameters would therefore adapt quickly as we shift from the sampler's initial, transient regime to its stationary regime. However, the diminishing step sizes of Robbins-Monro give disproportionate weight to the *early* iterations, which is the opposite of what we want.

Similar issues motivate the dual averaging scheme of Nesterov (2009), an algorithm for nonsmooth and stochastic convex optimization. Since solving an unconstrained convex optimization problem is equivalent to finding a zero of a nondecreasing function (the (sub)gradient of the cost function), it is straightforward to adapt dual averaging to the problem of MCMC adaptation by replacing stochastic gradients with the statistics H_t. Again assuming that we want to find a setting of a parameter $x \in \mathbb{R}$ such that $h(x) \equiv \mathbb{E}_t[H_t|x] = 0$, we can apply the updates

$$x_{t+1} \leftarrow \mu - \frac{\sqrt{t}}{\gamma} \frac{1}{t + t_0} \sum_{i=1}^{t} H_i; \quad \bar{x}_{t+1} \leftarrow \eta_t x_{t+1} + (1 - \eta_t)\bar{x}_t, \tag{6}$$

where μ is a freely chosen point that the iterates x_t are shrunk towards, $\gamma > 0$ is a free parameter that controls the amount of shrinkage towards μ, $t_0 \geq 0$ is a free parameter that stabilizes the initial iterations of the algorithm, $\eta_t \equiv t^{-\kappa}$ is a step size schedule obeying the conditions in Equation 5, and we define $\bar{x}_1 = x_1$. As in Robbins-Monro, the per-iteration impact of these updates on x goes to 0 as t goes to infinity. Specifically, for large t we have

$$x_{t+1} - x_t = O(-H_t t^{-0.5}),$$

which clearly goes to 0 as long as the statistic H_t is bounded. The sequence of averaged iterates $\bar{x}_t$ is guaranteed to converge to a value such that $h(\bar{x}_t)$ converges to 0.

The update scheme in Equation 6 is slightly more elaborate than the update scheme of Nesterov (2009), which implicitly has $t_0 \equiv 0$ and $\kappa \equiv 1$. Introducing these parameters addresses issues that are more important in MCMC adaptation than in more conventional stochastic convex optimization settings. Setting $t_0 > 0$ improves the stability of the algorithm in early iterations, which prevents us from wasting computation by trying out extreme values. This is particularly important for NUTS, and for HMC when simulation lengths are specified in terms of the overall simulation length ϵL instead of a fixed number of steps L. In both of these cases, lower values of ϵ result in more work being done per sample, so we want to avoid casually trying out extremely low values of ϵ. Setting the parameter $\kappa < 1$ allows us to give higher weight to more recent iterates and more quickly forget the iterates produced during the early warmup stages. The benefits of introducing these parameters are less apparent in the settings originally considered by Nesterov, where the cost of a stochastic gradient computation is assumed to be constant and the stochastic gradients are assumed to be drawn i.i.d. given the parameter x.

Allowing $t_0 > 0$ and $\kappa \in (0.5, 1]$ does not affect the asymptotic convergence of the dual averaging algorithm. For any $\kappa \in (0.5, 1]$, $\bar{x}_t$ will eventually converge to the same value $\frac{1}{t}\sum_{i=1}^{t} x_t$. We can rewrite the term $\frac{\sqrt{t}}{\gamma}\frac{1}{t+t_0}$ as $\frac{t\sqrt{t}}{\gamma(t+t_0)}\frac{1}{t}$; $\frac{t\sqrt{t}}{\gamma(t+t_0)}$ is still $O(\sqrt{t})$, which is the only feature needed to guarantee convergence.

We used the values $\gamma = 0.05, t_0 = 10$, and $\kappa = 0.75$ for all our experiments. We arrived at these values by trying a few settings for each parameter by hand with NUTS and HMC (with simulation lengths specified in terms of ϵL) on the stochastic volatility model described below and choosing a value for each parameter that seemed to produce reasonable behavior. Better results might be obtained with further tweaking, but these default parameters seem to work consistently well for both NUTS and HMC for all of the models that we tested. It is entirely possible that these parameter settings may not work as well for other sampling algorithms or for H statistics other than the ones described below.

3.2.2 Finding a Good Initial Value of ϵ

The dual averaging scheme outlined above should work for any initial value ϵ_1 and any setting of the shrinkage target μ. However, convergence will be faster if we start from a reasonable setting of these parameters. We recommend choosing an initial value ϵ_1 according to the simple heuristic described in Algorithm 4. In English, this heuristic repeatedly doubles or halves the value of ϵ_1 until the acceptance probability of the Langevin proposal with step size ϵ_1 crosses 0.5. The resulting value of ϵ_1 will typically be small enough to produce reasonably accurate simulations but large enough to avoid wasting large amounts of computation. We recommend setting $\mu = \log(10\epsilon_1)$, since this gives the dual averaging algorithm a preference for testing values of ϵ that are larger than the initial value ϵ_1. Large values of ϵ cost less to evaluate than small values of ϵ, and so erring on the side of trying large values can save computation.

3.2.3 Setting ϵ in HMC

In HMC we want to find a value for the step size ϵ that is neither too small (which would waste computation by taking needlessly tiny steps) nor too large (which would waste computation by causing high rejection rates). A standard approach is to tune ϵ so that HMC's average Metropolis acceptance probability is equal to some value δ. Indeed, it has been shown that (under fairly strong assumptions) the optimal value of ϵ for a given simulation length ϵL is the one that produces an average Metropolis acceptance probability of approximately 0.65 (Beskos et al., 2010; Neal, 2011). For HMC, we define a criterion $h^{\text{HMC}}(\epsilon)$ so that

$$H_t^{\text{HMC}} \equiv \min\left\{1, \frac{p(\tilde{\theta}^t, \tilde{r}^t)}{p(\theta^{t-1}, r^{t,0})}\right\}; \quad h^{\text{HMC}}(\epsilon) \equiv \mathbb{E}_t[H_t^{\text{HMC}}|\epsilon],$$

where $\tilde{\theta}^t$ and $\tilde{r}^t$ are the proposed position and momentum at the tth iteration of the Markov chain, θ^{t-1} and $r^{t,0}$ are the initial position and (resampled) momentum for the tth iteration of the Markov chain, H_t^{HMC} is the acceptance probability of this tth HMC proposal and h^{HMC} is the expected average acceptance probability of the chain in equilibrium for a fixed ϵ. Assuming that h^{HMC} is nonincreasing as a function of ϵ, we can apply the updates in Equation 6 with $H_t \equiv \delta - H_t^{\text{HMC}}$ and $x \equiv \log \epsilon$ to coerce $h^{\text{HMC}} = \delta$ for any $\delta \in (0, 1)$.

3.2.4 Setting ϵ in NUTS

Since there is no single accept/reject step in NUTS we must define an alternative statistic to Metropolis acceptance probability. For each iteration we define the statistic H_t^{NUTS} and its expectation when the chain has reached equilibrium as

$$H_t^{\text{NUTS}} \equiv \frac{1}{|\mathcal{B}_t^{\text{final}}|} \sum_{\theta, r \in \mathcal{B}_t^{\text{final}}} \min\left\{1, \frac{p(\theta, r)}{p(\theta^{t-1}, r^{t,0})}\right\}; \quad h^{\text{NUTS}} \equiv \mathbb{E}_t[H_t^{\text{NUTS}}],$$

where $\mathcal{B}_t^{\text{final}}$ is the set of all states explored during the final doubling of iteration t of the Markov chain and θ^{t-1} and $r^{t,0}$ are the initial position and (resampled) momentum for the tth iteration of the Markov chain. H^{NUTS} can be understood as the average acceptance probability that HMC would give to the position-momentum states explored during the final doubling iteration. As above, assuming that H^{NUTS} is nonincreasing in ϵ, we can apply the updates in Equation 6 with $H_t \equiv \delta - H^{\text{NUTS}}$ and $x \equiv \log \epsilon$ to coerce $h^{\text{NUTS}} = \delta$ for any $\delta \in (0, 1)$.

Algorithms 5 and 6 show how to implement HMC (with simulation length specified in terms of ϵL rather than L) and NUTS while incorporating the dual averaging algorithm derived in this section, with the above initialization scheme. Algorithm 5 requires as input a target simulation length $\lambda \approx \epsilon L$, a target mean acceptance probability δ, and a number of iterations M^{adapt} after which to stop the adaptation. Algorithm 6 requires only a target mean acceptance probability δ and a number of iterations M^{adapt}.

Algorithm 4 Heuristic for choosing an initial value of ϵ

function FindReasonableEpsilon(θ)
Initialize $\epsilon = 1$, $r \sim \mathcal{N}(0, I)$. (I denotes the identity matrix.)
Set $\theta', r' \leftarrow \text{Leapfrog}(\theta, r, \epsilon)$.
$a \leftarrow 2\mathbb{I}\left[\frac{p(\theta', r')}{p(\theta, r)} > 0.5\right] - 1$.
while $\left(\frac{p(\theta', r')}{p(\theta, r)}\right)^a > 2^{-a}$ **do**
  $\epsilon \leftarrow 2^a \epsilon$.
  Set $\theta', r' \leftarrow \text{Leapfrog}(\theta, r, \epsilon)$.
end while
return ϵ.

Algorithm 5 Hamiltonian Monte Carlo with Dual Averaging

Given θ^0, δ, λ, $\mathcal{L}, M, M^{\text{adapt}}$:
Set $\epsilon_0 = \text{FindReasonableEpsilon}(\theta), \mu = \log(10\epsilon_0), \bar{\epsilon}_0 = 1, \bar{H}_0 = 0, \gamma = 0.05, t_0 = 10, \kappa = 0.75$.
for $m = 1$ to M **do**
  Reample $r^0 \sim \mathcal{N}(0, I)$.
  Set $\theta^m \leftarrow \theta^{m-1}, \tilde{\theta} \leftarrow \theta^{m-1}, \tilde{r} \leftarrow r^0, L_m = \max\{1, \text{Round}(\lambda/\epsilon_{m-1})\}$.
  for $i = 1$ to L_m **do**
    Set $\tilde{\theta}, \tilde{r} \leftarrow \text{Leapfrog}(\tilde{\theta}, \tilde{r}, \epsilon_{m-1})$.
  end for
  With probability $\alpha = \min\left\{1, \frac{\exp\{\mathcal{L}(\tilde{\theta}) - \frac{1}{2}\tilde{r}\cdot\tilde{r}\}}{\exp\{\mathcal{L}(\theta^{m-1}) - \frac{1}{2}r^0\cdot r^0\}}\right\}$, set $\theta^m \leftarrow \tilde{\theta}, r^m \leftarrow -\tilde{r}$.
  if $m \leq M^{\text{adapt}}$ **then**
    Set $\bar{H}_m = \left(1 - \frac{1}{m+t_0}\right)\bar{H}_{m-1} + \frac{1}{m+t_0}(\delta - \alpha)$.
    Set $\log \epsilon_m = \mu - \frac{\sqrt{m}}{\gamma}\bar{H}_m, \log \bar{\epsilon}_m = m^{-\kappa}\log\epsilon_m + (1 - m^{-\kappa})\log\bar{\epsilon}_{m-1}$.
  else
    Set $\epsilon_m = \bar{\epsilon}_{M^{\text{adapt}}}$.
  end if
end for

4. Empirical Evaluation

In this section we examine the effectiveness of the dual averaging algorithm outlined in Section 3.2, examine what values of the target δ in the dual averaging algorithm yield efficient samplers, and compare the efficiency of NUTS and HMC.

For each target distribution, we ran HMC (as implemented in algorithm 5) and NUTS (as implemented in algorithm 6) with four target distributions for 2000 iterations, allowing the step size ϵ to adapt via the dual averaging updates described in Section 3.2 for the first 1000 iterations. In all experiments the dual averaging parameters were set to $\gamma = 0.05, t_0 = 10$, and $\kappa = 0.75$. We evaluated HMC with 10 logarithmically spaced target simulation lengths λ per target distribution. For each target distribution the largest value of λ that we tested was 40 times the smallest value of λ that we tested, meaning that each successive λ is $40^{1/9} \approx 1.5$ times larger than the previous λ. We tried 15 evenly spaced values of the dual averaging target δ between 0.25 and 0.95 for NUTS and 8 evenly spaced values of the dual averaging target δ between 0.25 and 0.95 for HMC. For each sampler-simulation length-δ-target distribution combination we ran 10 iterations with different random seeds.

Algorithm 6 No-U-Turn Sampler with Dual Averaging

Given θ^0, δ, $\mathcal{L}, M, M^{\text{adapt}}$:
Set $\epsilon_0 = \text{FindReasonableEpsilon}(\theta), \mu = \log(10\epsilon_0), \bar{\epsilon}_0 = 1, \bar{H}_0 = 0, \gamma = 0.05, t_0 = 10, \kappa = 0.75$.
for $m = 1$ to M **do**
  Sample $r^0 \sim \mathcal{N}(0, I)$.
  Resample $u \sim \text{Uniform}([0, \exp\{\mathcal{L}(\theta^{m-1} - \frac{1}{2} r^0 \cdot r^0\}])$
  Initialize $\theta^- = \theta^{m-1}$, $\theta^+ = \theta^{m-1}$, $r^- = r^0, r^+ = r^0, j = 0, \theta^m = \theta^{m-1}, n = 1, s = 1$.
  while $s = 1$ **do**
    Choose a direction $v_j \sim \text{Uniform}(\{-1, 1\})$.
    if $v_j = -1$ **then**
      $\theta^-, r^-, -, -, \theta', n', s', \alpha, n_\alpha \leftarrow \text{BuildTree}(\theta^-, r^-, u, v_j, j, \epsilon_{m-1}\theta^{m-1}, r^0)$.
    else
      $-, -, \theta^+, r^+, \theta', n', s', \alpha, n_\alpha \leftarrow \text{BuildTree}(\theta^+, r^+, u, v_j, j, \epsilon_{m-1}, \theta^{m-1}, r^0)$.
    end if
    if $s' = 1$ **then**
      With probability $\min\{1, \frac{n'}{n}\}$, set $\theta^m \leftarrow \theta'$.
    end if
    $n \leftarrow n + n'$.
    $s \leftarrow s' \mathbb{I}[(\theta^+ - \theta^-) \cdot r^- \geq 0] \mathbb{I}[(\theta^+ - \theta^-) \cdot r^+ \geq 0]$.
    $j \leftarrow j + 1$.
  end while
  if $m \leq M^{\text{adapt}}$ **then**
    Set $\bar{H}_m = \left(1 - \frac{1}{m + t_0}\right) \bar{H}_{m-1} + \frac{1}{m + t_0}(\delta - \frac{\alpha}{n_\alpha})$.
    Set $\log \epsilon_m = \mu - \frac{\sqrt{m}}{\gamma} \bar{H}_m, \log \bar{\epsilon}_m = m^{-\kappa} \log \epsilon_m + (1 - m^{-\kappa}) \log \bar{\epsilon}_{m-1}$.
  else
    Set $\epsilon_m = \bar{\epsilon}_{M^{\text{adapt}}}$.
  end if
end for

function $\text{BuildTree}(\theta, r, u, v, j, \epsilon, \theta^0, r^0)$
if $j = 0$ **then**
  Base case—take one leapfrog step in the direction v.
  $\theta', r' \leftarrow \text{Leapfrog}(\theta, r, v\epsilon)$.
  $n' \leftarrow \mathbb{I}[u \leq \exp\{\mathcal{L}(\theta') - \frac{1}{2} r' \cdot r'\}]$.
  $s' \leftarrow \mathbb{I}[u < \exp\{\Delta_{\max} + \mathcal{L}(\theta') - \frac{1}{2} r' \cdot r'\}]$.
  return $\theta', r', \theta', r', \theta', n', s', \min\{1, \exp\{\mathcal{L}(\theta') - \frac{1}{2} r' \cdot r' - \mathcal{L}(\theta^0) + \frac{1}{2} r^0 \cdot r^0\}\}, 1$.
else
  Recursion—implicitly build the left and right subtrees.
  $\theta^-, r^-, \theta^+, r^+, \theta', n', s', \alpha', n'_\alpha \leftarrow \text{BuildTree}(\theta, r, u, v, j - 1, \epsilon, \theta^0, r^0)$.
  if $s' = 1$ **then**
    if $v = -1$ **then**
      $\theta^-, r^-, -, -, \theta'', n'', s'', \alpha'', n''_\alpha \leftarrow \text{BuildTree}(\theta^-, r^-, u, v, j - 1, \epsilon, \theta^0, r^0)$.
    else
      $-, -, \theta^+, r^+, \theta'', n'', s'', \alpha'', n''_\alpha \leftarrow \text{BuildTree}(\theta^+, r^+, u, v, j - 1, \epsilon, \theta^0, r^0)$.
    end if
    With probability $\frac{n''}{n' + n''}$, set $\theta' \leftarrow \theta''$.
    Set $\alpha' \leftarrow \alpha' + \alpha''$, $n'_\alpha \leftarrow n'_\alpha + n''_\alpha$.
    $s' \leftarrow s'' \mathbb{I}[(\theta^+ - \theta^-) \cdot r^- \geq 0] \mathbb{I}[(\theta^+ - \theta^-) \cdot r^+ \geq 0]$
    $n' \leftarrow n' + n''$
  end if
  return $\theta^-, r^-, \theta^+, r^+, \theta', n', s', \alpha', n'_\alpha$.
end if

In total, we ran 3,200 experiments with HMC and 600 experiments with NUTS. Traditional HMC can sometimes exhibit pathological behavior when using a fixed step size and number of steps per iteration (Neal, 2011), so after warmup we jitter HMC's step size, sampling it uniformly at random each iteration from the range $[0.9\bar{\epsilon}_{M^{\text{adapt}}}, 1.1\bar{\epsilon}_{M^{\text{adapt}}}]$ so that the trajectory length may vary by $\pm 10\%$ each iteration.

We measure the efficiency of each algorithm in terms of effective sample size (ESS) normalized by the number of gradient evaluations used by each algorithm. The ESS of a set of M correlated samples $\theta^{1:M}$ with respect to some function $f(\theta)$ is the number of independent draws from the target distribution $p(\theta)$ that would give a Monte Carlo estimate of the mean under p of $f(\theta)$ with the same level of precision as the estimate given by the mean of f for the correlated samples $\theta^{1:M}$. That is, the ESS of a sample is a measure of how many independent samples a set of correlated samples is worth for the purposes of estimating the mean of some function; a more efficient sampler will give a larger ESS for less computation. We use the number of gradient evaluations performed by an algorithm as a proxy for the total amount of computation performed; in all of the models and distributions we tested the computational overhead of both HMC and NUTS is dominated by the cost of computing gradients. Details of the method we use to estimate ESS are provided in appendix A. In each experiment, we discarded the first 1000 samples as warmup when estimating ESS.

ESS is inherently a univariate statistic, but all of the distributions we test HMC and NUTS on are multivariate. Following Girolami and Calderhead (2011) we compute ESS separately for each dimension and report the minimum ESS across all dimensions, since we want our samplers to effectively explore all dimensions of the target distribution. For each dimension we compute ESS in terms of the variance of the estimator of that dimension's mean and second central moment (where the estimate of the mean used to compute the second central moment is taken from a separate long run of 50,000 iterations of NUTS with $\delta = 0.5$), reporting whichever statistic has a lower effective sample size. We include the second central moment as well as the mean in order to measure each algorithm's ability to estimate uncertainty.

4.1 Models and Data Sets

To evaluate NUTS and HMC, we used the two algorithms to sample from four target distributions, one of which was synthetic and the other three of which are posterior distributions arising from real data sets.

4.1.1 250-dimensional Multivariate Normal (MVN)

In these experiments the target distribution was a zero-mean 250-dimensional multivariate normal with known precision matrix A, that is,

$$p(\theta) \propto \exp\{-\tfrac{1}{2}\theta^T A \theta\}.$$

The matrix A was generated from a Wishart distribution with identity scale matrix and 250 degrees of freedom. This yields a target distribution with many strong correlations. The same matrix A was used in all experiments.

4.1.2 Bayesian Logistic Regression (LR)

In these experiments the target distribution is the posterior of a Bayesian logistic regression model fit to the German credit data set available from the UCI repository (Frank and Asuncion, 2010). The target distribution is

$$\begin{aligned} p(\alpha, \beta | x, y) &\propto p(y|x, \alpha, \beta) p(\alpha) p(\beta) \\ &\propto \exp\{-\textstyle\sum_i \log(1 + \exp\{-y_i(\alpha + x_i \cdot \beta\}) - \frac{1}{2\sigma^2}\alpha^2 - \frac{1}{2\sigma^2}\beta \cdot \beta\}, \end{aligned}$$

where x_i is a 24-dimensional vector of numerical predictors associated with a customer i, y_i is -1 if customer i should be denied credit and 1 if that customer should receive credit, α is an intercept term, and β is a vector of 24 regression coefficients. All predictors are normalized to have zero mean and unit variance. α and each element of β are given weak zero-mean normal priors with variance $\sigma^2 = 100$. The data set contains predictor and response data for 1000 customers.

4.1.3 Hierarchical Bayesian Logistic Regression (HLR)

In these experiments the target distribution is again the posterior of a Bayesian logistic regression model fit to the German credit data set, but this time the variance parameter in the prior on α and β is given an exponential prior and estimated as well. Also, we expand the predictor vectors by including two-way interactions, resulting in $\binom{24}{2} + 24 = 300$-dimensional vectors of predictors x and a 300-dimensional vector of coefficients β. These elaborations on the model make for a more challenging problem; the posterior is in higher dimensions, and the variance term σ^2 interacts strongly with the remaining 301 variables. The target distribution for this problem is

$$\begin{aligned} p(\alpha, \beta, \sigma^2 | x, y) &\propto p(y|x, \alpha, \beta) p(\beta|\sigma^2) p(\alpha|\sigma^2) p(\sigma^2) \\ &\propto \exp\{-\textstyle\sum_i \log(1 + \exp\{-y_i x_i \cdot \beta\}) - \frac{1}{2\sigma^2}\alpha^2 - \frac{1}{2\sigma^2}\beta \cdot \beta - \frac{N}{2}\log\sigma^2 - \lambda\sigma^2\}, \end{aligned}$$

where $N = 1000$ is the number of customers and λ is the rate parameter to the prior on σ^2. We set $\lambda = 0.01$, yielding a weak exponential prior distribution on σ^2 whose mean and standard deviation are 100.

4.1.4 Stochastic Volatility (SV)

In the final set of experiments the target distribution is the posterior of a relatively simple stochastic volatility model fit to 3000 days of returns from the S&P 500 index. The model assumes that the observed values of the index are generated by the following generative process:

$$\begin{gathered} \tau \sim \text{Exponential}(100); \quad \nu \sim \text{Exponential}(100); \quad s_1 \sim \text{Exponential}(100); \\ \log s_{i>1} \sim \text{Normal}(\log s_{i-1}, \tau^{-1}); \quad \tfrac{\log y_i - \log y_{i-1}}{s_i} \sim \mathrm{t}_\nu, \end{gathered}$$

where $s_{i>1}$ refers to a scale parameter s_i where $i > 1$. We integrate out the precision parameter τ to speed mixing, leading to the 3001-dimensional target distribution

$$\begin{aligned} p(s, \nu | y) \propto e^{-0.01\nu} e^{-0.01 s_1} (\textstyle\prod_{i=1}^{3000} \mathrm{t}_\nu(s_i^{-1}(\log y_i - \log y_{i-1}))) \times \\ (0.01 + 0.5 \textstyle\sum_{i=2}^{3000} (\log s_i - \log s_{i-1})^2)^{-\frac{3001}{2}}. \end{aligned}$$

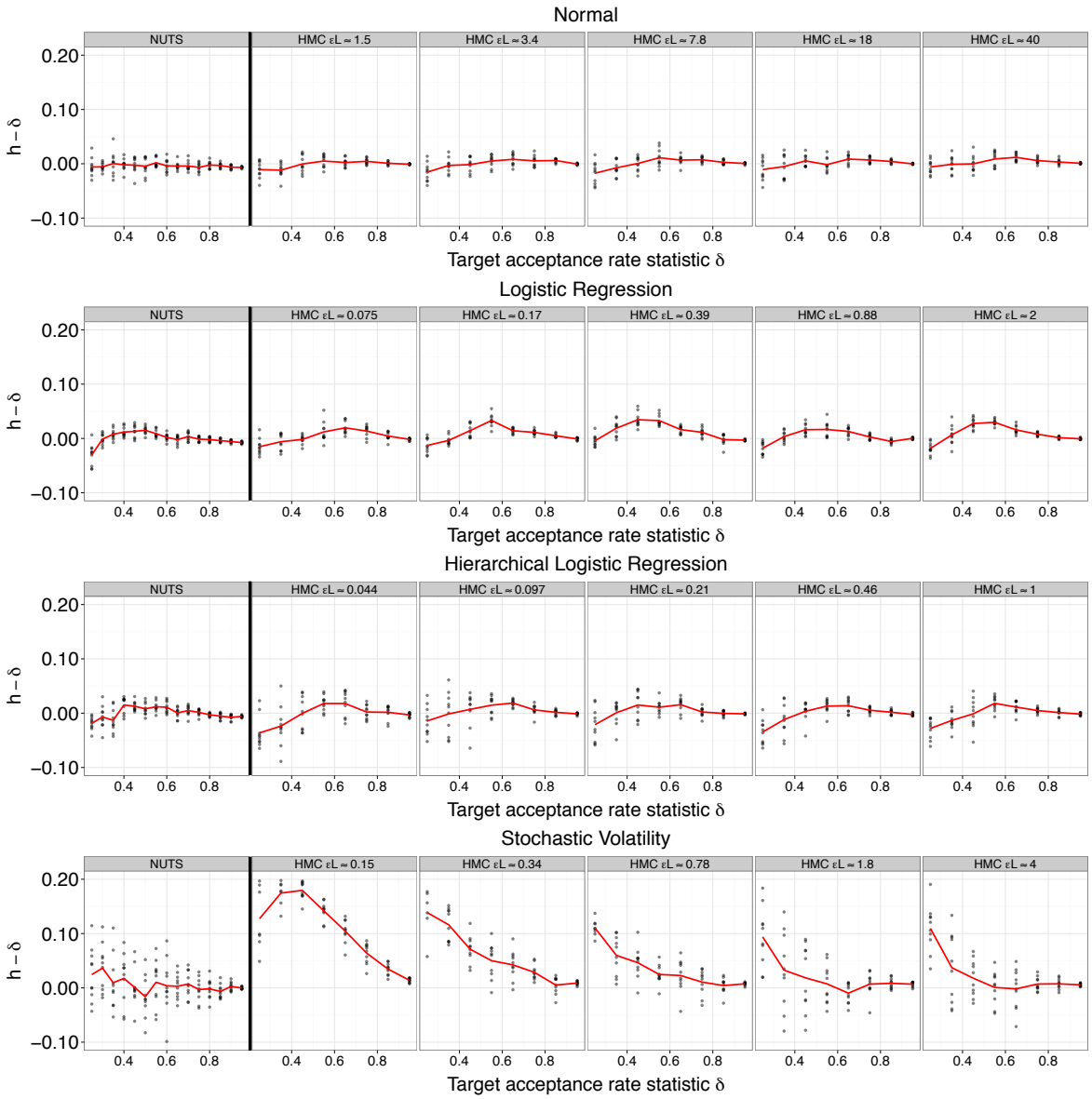


Figure 3: Discrepancies between the realized average acceptance probability statistic h and its target δ for the multivariate normal, logistic regression, hierarchical logistic regression, and stochastic volatility models. Each point's distance from the x-axis shows how effectively the dual averaging algorithm tuned the step size ϵ for a single experiment. Leftmost plots show experiments run with NUTS, other plots show experiments run with HMC with various settings of ϵL.

4.2 Convergence of Dual Averaging

Figure 3 plots the realized versus target values of the statistics h^{HMC} and h^{NUTS}. The h statistics were computed from the 1000 post-warmup samples. The dual averaging algorithm of Section 3.2 usually does a good job of coercing the statistic h to its desired value δ. It performs somewhat worse for the stochastic volatility model, which we attribute to the

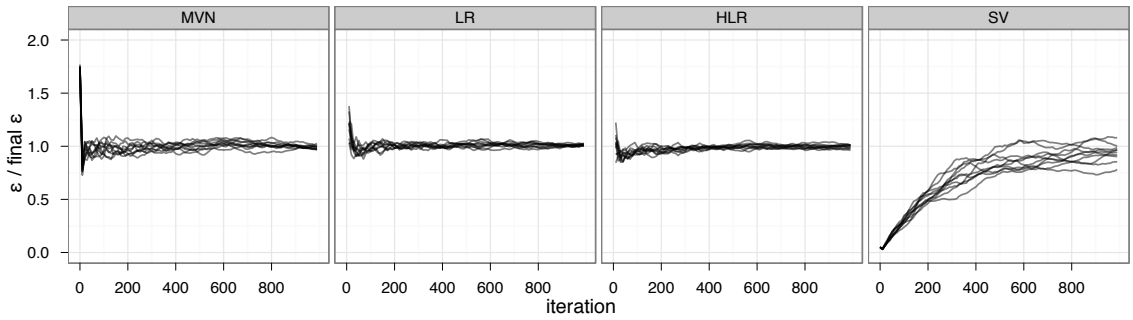


Figure 4: Plots of the convergence of $\bar{\epsilon}$ as a function of the number of iterations of NUTS with dual averaging with $\delta = 0.65$ applied to the multivariate normal (MVN), logistic regression (LR), hierarchical logistic regression (HLR), and stochastic volatility (SV) models. Each trace is from an independent run. The y-axis shows the value of $\bar{\epsilon}$, divided by one of the final values of $\bar{\epsilon}$ so that the scale of the traces for each problem can be readily compared.

longer warmup period needed for this model; since it takes more samples to reach the stationary regime for the stochastic volatility model, the adaptation algorithm has less time to tune ϵ to be appropriate for the stationary distribution. This is particularly true for HMC with small values of δ, since the overly high rejection rates caused by setting δ too small lead to slower convergence.

Figure 4 plots the convergence of the averaged iterates $\bar{\epsilon}_m$ as a function of the number of dual averaging updates for NUTS with $\delta = 0.65$. Except for the stochastic volatility model, which requires longer to warm up, $\bar{\epsilon}$ roughly converges within a few hundred iterations.

4.3 NUTS Trajectory Lengths

Figure 5 shows histograms of the trajectory lengths generated by NUTS. Most of the trajectory lengths are integer powers of two, indicating that the U-turn criterion in Equation 4 is usually satisfied only after a doubling is complete and not by one of the intermediate subtrees generated during the doubling process. This behavior is desirable insofar as it means that we only occasionally have to throw out entire half-trajectories to satisfy detailed balance.

The trajectory length (measured in number of states visited) grows as the acceptance rate target δ grows, which is to be expected since a higher δ will lead to a smaller step size ϵ, which in turn will mean that more leapfrog steps are necessary before the trajectory doubles back on itself and satisfies Equation 4.

4.4 Comparing the Efficiency of HMC and NUTS

Figure 6 compares the efficiency of HMC (with various simulation lengths $\lambda \approx \epsilon L$) and NUTS (which chooses simulation lengths automatically). The x-axis in each plot is the target δ used by the dual averaging algorithm from Section 3.2 to automatically tune the step size ϵ. The y-axis is the effective sample size (ESS) generated by each sampler, normalized by the number of gradient evaluations used in generating the samples. HMC's best performance seems to occur around $\delta = 0.65$, suggesting that this is indeed a reasonable default value

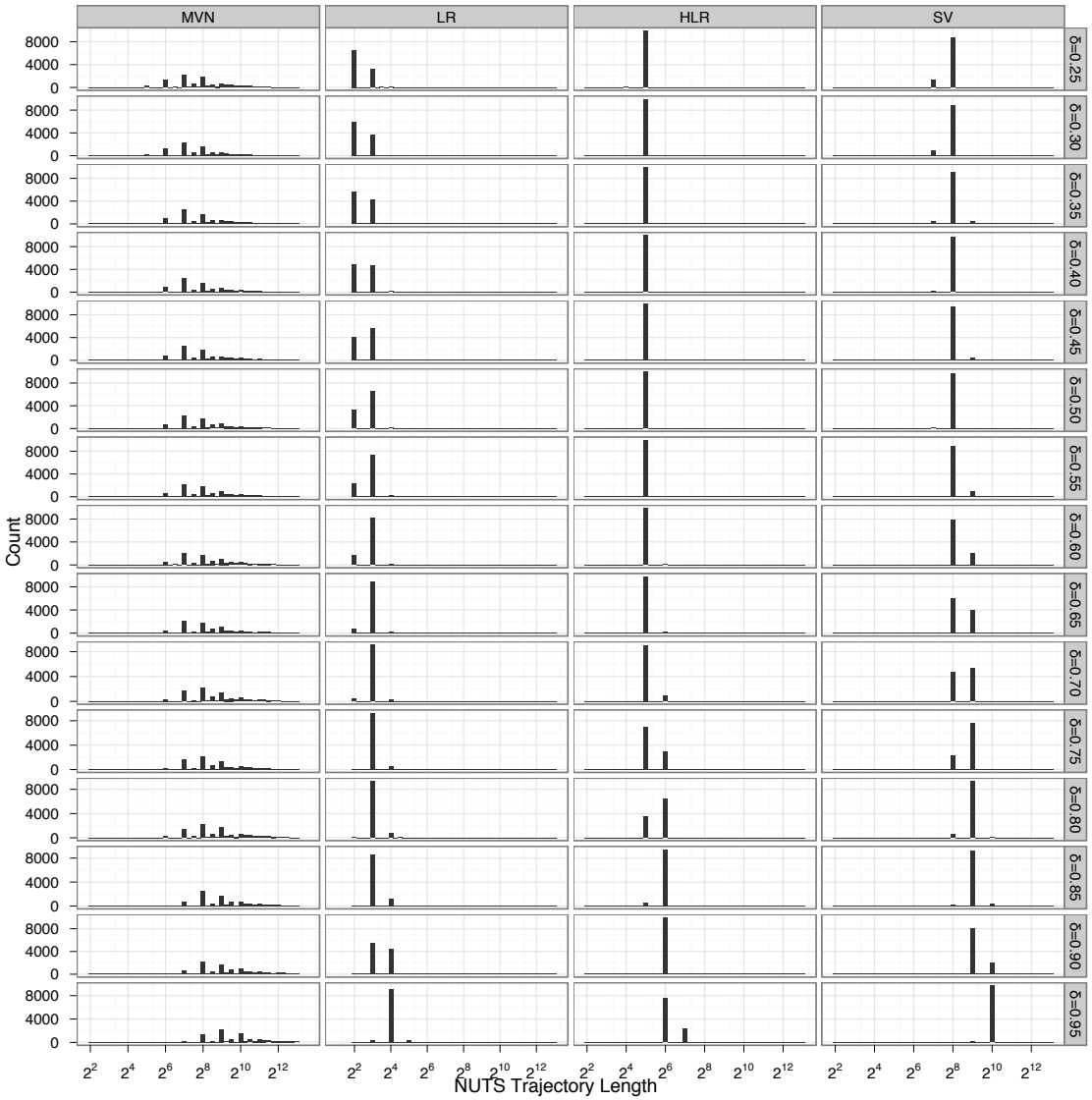


Figure 5: Histograms of the trajectory lengths generated by NUTS with various acceptance rate targets δ for the multivariate normal (MVN), logistic regression (LR), hierarchical logistic regression (HLR), and stochastic volatility (SV) models.

for a variety of problems. NUTS's best performance seems to occur around $\delta = 0.6$, but does not seem to depend strongly on δ within the range $\delta \in [0.45, 0.65]$. $\delta = 0.6$ therefore seems like a reasonable default value for NUTS.

On the two logistic regression problems NUTS is able to produce effectively independent samples about as efficiently as HMC can. On the multivariate normal and stochastic volatility problems, NUTS with $\delta = 0.6$ outperforms HMC's best ESS by about a factor of 2 and 1.5, respectively.

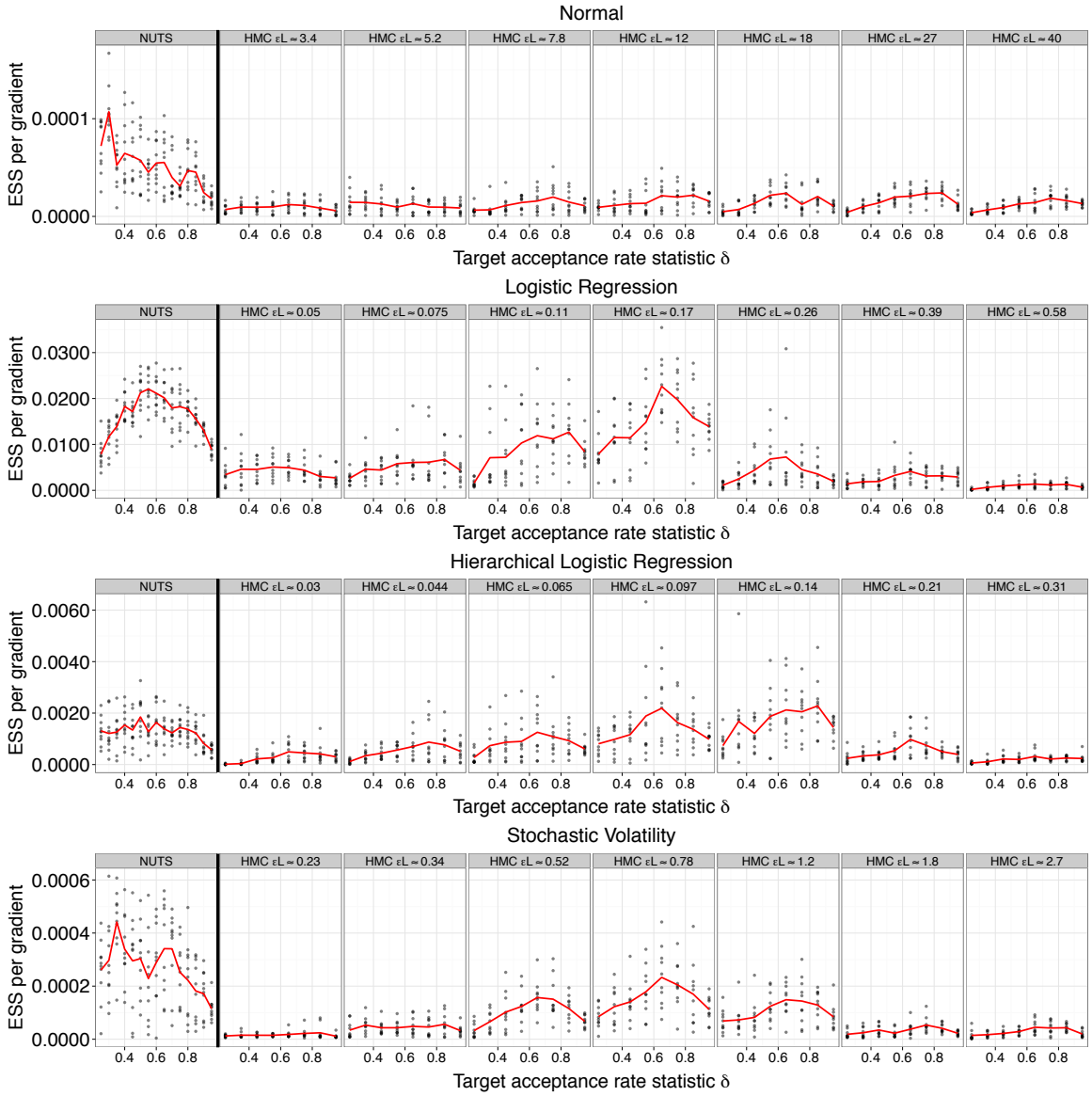


Figure 6: Effective sample size (ESS) as a function of δ and (for HMC) simulation length ϵL for the multivariate normal, logistic regression, hierarchical logistic regression, and stochastic volatility models. Each point shows the ESS divided by the number of gradient evaluations for a separate experiment; lines denote the average of the points' y-values for a particular δ. Leftmost plots are NUTS's performance, other plots shows HMC's performance for various settings of ϵL.

As expected, HMC's performance degrades if an inappropriate simulation length is chosen. Across the four target distributions we tested, the best simulation lengths λ for HMC varied by about a factor of 100, with the longest optimal λ being about 18 (for the multivariate normal) and the shortest optimal λ being about 0.14 (for the hierarchical logistic regression). In practice, finding a good simulation length for HMC will usually require

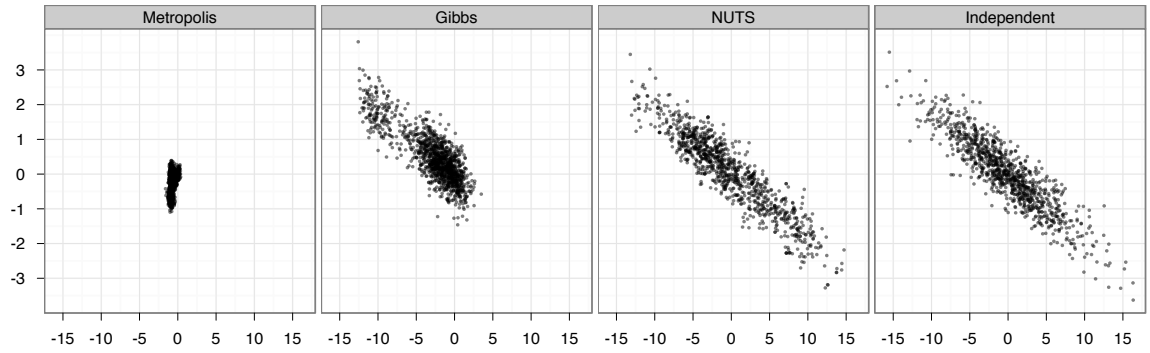


Figure 7: Samples generated by random-walk Metropolis, Gibbs sampling, and NUTS. The plots compare 1,000 independent draws from a highly correlated 250-dimensional distribution (right) with 1,000,000 samples (thinned to 1,000 samples for display) generated by random-walk Metropolis (left), 1,000,000 samples (thinned to 1,000 samples for display) generated by Gibbs sampling (second from left), and 1,000 samples generated by NUTS (second from right). Only the first two dimensions are shown here.

some number of preliminary runs. The results in Figure 6 suggest that NUTS can generate samples at least as efficiently as HMC, even discounting the cost of any preliminary runs needed to tune HMC's simulation length.

4.5 Qualitative Comparison of NUTS, Random-Walk Metropolis, and Gibbs

In Section 4.4, we compared the efficiency of NUTS and HMC. In this section, we informally demonstrate the advantages of NUTS over the popular random-walk Metropolis (RWM) and Gibbs sampling algorithms. We ran NUTS, RWM, and Gibbs sampling on the 250-dimensional multivariate normal distribution described in Section 4.1. NUTS was run with $\delta = 0.5$ for 2,000 iterations, with the first 1,000 iterations being used as warmup and to adapt ϵ. This required about 1,000,000 gradient and likelihood evaluations in total. We ran RWM for 1,000,000 iterations with an isotropic normal proposal distribution whose variance was selected beforehand to produce the theoretically optimal acceptance rate of 0.234 (Gelman et al., 1996). The cost per iteration of RWM is effectively identical to the cost per gradient evaluation of NUTS, and the two algorithms ran for about the same amount of time. We ran Gibbs sampling for 1,000,000 sweeps over the 250 parameters. This took longer to run than NUTS and RWM, since for the multivariate normal each Gibbs sweep costs more than a single gradient evaluation; we chose to nonetheless run the same number of Gibbs sweeps as RWM iterations, since for some other models Gibbs sweeps can be done more efficiently.

Figure 7 visually compares independent samples (projected onto the first two dimensions) from the target distribution with samples generated by the three MCMC algorithms. RWM has barely begun to explore the space. Gibbs does better, but still has left parts

of the space unexplored. NUTS, on the other hand, is able to generate many effectively independent samples.

We use this simple example to visualize the relative performance of NUTS, Gibbs, and RWM on a moderately high-dimensional distribution exhibiting strong correlations. For the multivariate normal, Gibbs or RWM would of course work much better after an appropriate rotation of the parameter space. But finding and applying an appropriate rotation can be expensive when the number of parameters D gets large, and RWM and Gibbs both require $O(D^2)$ operations per effectively independent sample even under the highly optimistic assumption that a transformation can be found that renders all parameters i.i.d. and can be applied cheaply (e.g., in $O(D)$ rather than the usual $O(D^2)$ cost of matrix-vector multiplication and the $O(D^3)$ cost of matrix inversion). This is shown for RWM by Creutz (1988), and for Gibbs is the result of needing to apply a transformation requiring $O(D)$ operations D times per Gibbs sweep. For complicated models, even more expensive transformations often cannot render the parameters sufficiently independent to make RWM and Gibbs run efficiently. NUTS, on the other hand, is able to efficiently sample from high-dimensional target distributions without needing to be tuned to the shape of those distributions.

5. Discussion

We have presented the No-U-Turn Sampler (NUTS), a variant of the powerful Hamiltonian Monte Carlo (HMC) Markov chain Monte Carlo (MCMC) algorithm that eliminates HMC's dependence on a number-of-steps parameter L but retains (and in some cases improves upon) HMC's ability to generate effectively independent samples efficiently. We also developed a method for automatically adapting the step size parameter ϵ shared by NUTS and HMC via an adaptation of the dual averaging algorithm of Nesterov (2009), making it possible to run NUTS with no hand tuning at all. The dual averaging approach we developed in this paper could also be applied to other MCMC algorithms in place of more traditional adaptive MCMC approaches based on the Robbins-Monro stochastic approximation algorithm (Andrieu and Thoms, 2008; Robbins and Monro, 1951).

In this paper we have only compared NUTS with the basic HMC algorithm, and not its extensions, several of which are reviewed by Neal (2011). We only considered simple kinetic energy functions of the form $\frac{1}{2}r \cdot r$, but both NUTS and HMC can benefit from introducing a "mass" matrix M and using the kinetic energy function $\frac{1}{2}r^T M^{-1} r$. If M^{-1} approximates the covariance matrix of $p(\theta)$, then this kinetic energy function will reduce the negative impacts strong correlations and bad scaling have on the efficiency of both NUTS and HMC; indeed, experiments with hierarchical regression models with high correlations show substantial reduction in total computation time from nonidentity and nondiagonal mass matrices. Fitting an appropriate mass matrix can only be done during the warmup stage, and care must be taken to ensure the stability of the fitting procedure. Since using a mass matrix is equivalent to linearly transforming the parameter space (Neal, 2011), the no-U-turn condition should be computed on the transformed parameters instead of in the original space.

Another extension of HMC introduced by Neal (1994) considers windows of proposed states rather than simply the state at the end of the trajectory to allow for larger step sizes

without sacrificing acceptance rates (at the expense of introducing a window size parameter that must be tuned). The effectiveness of the windowed HMC algorithm suggests that NUTS's lack of a single accept/reject step may be responsible for some of its performance gains over vanilla HMC.

Girolami and Calderhead (2011) recently introduced Riemannian Manifold Hamiltonian Monte Carlo (RMHMC), a variant on HMC that simulates Hamiltonian dynamics in Riemannian rather than Euclidean spaces, effectively allowing for position-dependent mass matrices. Although the worst-case $O(D^3)$ matrix inversion costs associated with this algorithm often make it expensive to apply in high dimensions, when these costs are not too onerous RMHMC's ability to adapt its kinetic energy function makes it very efficient. There are no technical barriers that stand in the way of combining NUTS's ability to adapt its trajectory lengths with RMHMC's ability to adapt its mass matrices, although naively applying the no-U-turn condition (which is tied to Euclidean geometry) to Riemannian algorithms may be suboptimal (Betancourt, 2013).

Like HMC, NUTS can only be used to resample unconstrained continuous-valued variables with respect to which the target distribution is differentiable almost everywhere. HMC and NUTS can deal with simple constraints such as nonnegativity or restriction to the simplex by an appropriate change of variable, but discrete variables must either be summed out or handled by other algorithms such as Gibbs sampling. In models with discrete variables, NUTS's ability to automatically choose a trajectory length may make it more effective than HMC when discrete variables are present, since it is not tied to a single simulation length that may be appropriate for one setting of the discrete variables but not for others.

Some models include hard constraints that are too complex to eliminate by a simple change of variables. Such models will have regions of the parameter space with zero posterior probability. When HMC encounters such a region, the best it can do is stop short and restart with a new momentum vector, wasting any work done before violating the constraints (Neal, 2011). By contrast, when NUTS encounters a zero-probability region it stops short and samples from the set of points visited up to that point, making at least some progress.

NUTS with dual averaging makes it possible for Bayesian data analysts to obtain the efficiency of HMC without spending time and effort hand-tuning HMC's parameters. This is desirable even for those practitioners who have experience using and tuning HMC, but it is especially valuable for those who lack this experience. In particular, NUTS's ability to operate efficiently without user intervention makes it well suited for use in generic inference engines in the mold of BUGS (Gilks and Spiegelhalter, 1992), which until now have largely relied on much less efficient algorithms such as Gibbs sampling. We are currently developing an automatic Bayesian inference system called Stan, which uses NUTS as its core inference algorithm for continuous-valued parameters (Stan Development Team, 2013). Stan promises to be able to generate effectively independent samples from complex models' posteriors orders of magnitude faster than previous systems such as BUGS and JAGS (Plummer, 2003).

In summary, NUTS makes it possible to efficiently perform Bayesian posterior inference on a large class of complex, high-dimensional models with minimal human intervention. It is our hope that NUTS will allow researchers and data analysts to spend more time developing and testing models and less time worrying about how to fit those models to data.

Acknowledgments

We thank Bob Carpenter, Michael Betancourt, and Radford Neal for helpful comments. This work was partially supported by Institute of Education Sciences grant ED-GRANTS-032309-005, Department of Energy grant DE-SC0002099, National Science Foundation grant ATM-0934516, and National Science Foundation grant SES-1023189.

Appendix A. Estimating Effective Sample Size

For a function $f(\theta)$, a target distribution $p(\theta)$, and a Markov chain Monte Carlo (MCMC) sampler that produces a set of M correlated samples drawn from some distribution $q(\theta^{1:M})$ such that $q(\theta^m) = p(\theta^m)$ for any $m \in \{1, \ldots, M\}$, the effective sample size (ESS) of $\theta^{1:M}$ is the number of independent samples that would be needed to obtain a Monte Carlo estimate of the mean of f with equal variance to the MCMC estimate of the mean of f:

$$\mathrm{ESS}_{q,f}(\theta^{1:M}) = M \frac{\mathbb{V}_q[\frac{1}{M}\sum_{s=1}^{M} f(\theta^s)]}{\frac{\mathbb{V}_p[f(\theta)]}{M}} = \frac{M}{1 + 2\sum_{s=1}^{M-1}(1 - \frac{s}{M})\rho_s^f};$$
$$\rho_s^f \equiv \frac{\mathbb{E}_q[(f(\theta^t) - \mathbb{E}_p[f(\theta)])(f(\theta^{t-s}) - \mathbb{E}_p[f(\theta)])]}{\mathbb{V}_p[f(\theta)]},$$

where ρ_s^f denotes the autocorrelation under q of f at lag s and $\mathbb{V}_p[x]$ denotes the variance of a random variable x under the distribution $p(x)$.

To estimate ESS, we first compute the following estimate of the autocorrelation spectrum for the function $f(\theta)$:

$$\hat{\rho}_s^f = \frac{1}{\hat{\sigma}_f^2 (M - s)} \sum_{m=s+1}^{M} (f(\theta^m) - \hat{\mu}_f)(f(\theta^{m-s}) - \hat{\mu}_f),$$

where the estimates $\hat{\mu}_f$ and $\hat{\sigma}_f^2$ of the mean and variance of the function f are computed with high precision from a separated 50,000-sample run of NUTS with $\delta = 0.5$. We do not take these estimates from the chain whose autocorrelations we are trying to estimate—doing so can lead to serious underestimates of the level of autocorrelation (and thus a serious overestimate of the number of effective samples) if the chain has not yet converged or has not yet generated a fair number of effectively independent samples.

Any estimator of ρ_s^f is necessarily noisy for large lags s, so using the naive estimator $\hat{\mathrm{ESS}}_{q,f}(\theta^{1:M}) = \frac{M}{1+2\sum_{s=1}^{M-1}(1-\frac{s}{M})\hat{\rho}_s^f}$ will yield bad results. Instead, we truncate the sum over the autocorrelations when the autocorrelations first dip below 0.05, yielding the estimator

$$\hat{\mathrm{ESS}}_{q,f}(\theta^{1:M}) = \frac{M}{1 + 2\sum_{s=1}^{M_f^{\mathrm{cutoff}}}(1 - \frac{s}{M})\hat{\rho}_s^f}; \quad M_f^{\mathrm{cutoff}} \equiv \min_s s \quad \text{s.t. } \hat{\rho}_s^f < 0.05.$$

We found that this method for estimating ESS gave more reliable confidence intervals for MCMC estimators than the autoregressive approach used by CODA (Plummer et al., 2006). (The more accurate estimator comes at the expense of needing to compute a costly high-quality estimate of the true mean and variance of the target distribution.) The 0.05 cutoff is somewhat arbitrary; in our experiments we did not find the results to be very sensitive to the precise value of this cutoff.

References

C. Andrieu and J. Thoms. A tutorial on adaptive MCMC. *Statistics and Computing*, 18 (4):343–373, 2008.

A. Beskos, N. Pillai, G. Roberts, J. Sanz-Serna, and A. Stuart. Optimal tuning of the hybrid monte-carlo algorithm. *Arxiv preprint arXiv:1001.4460*, 2010.

M. Betancourt. Generalizing the no-U-turn sampler to Riemannian manifolds. *ArXiv preprint arXiv:1304.1920*, 2013.

M. Creutz. Global Monte Carlo algorithms for many-fermion systems. *Physical Review D*, 38(4):1228–1238, 1988.

H. Daume III. HBC: Hierarchical Bayes compiler, 2007. URL `http://hal3.name/HBC`.

A. Duane, A. Kennedy, B. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

A. Gelman, G. Roberts, and W. Gilks. Efficient Metropolis jumping rules. *Bayesian Statistics*, 5:599–608, 1996.

A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis*. Chapman & Hall, 2004.

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

W. Gilks and D. Spiegelhalter. A language and program for complex Bayesian modelling. *The Statistician*, 3:169–177, 1992.

M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73 (2):123–214, 2011.

A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics (SIAM), 2008.

S. Lan, V. Stathopoulos, B. Shahbaba, and M. Girolami. Lagrangian Dynamical Monte Carlo. *ArXiv preprint arXiv:1211.3759*, 2012.

B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics*, volume 14. Cambridge University Press, 2004.

N. Metropolis, A. Rosenbluth, M. Rosenbluth, M. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

T. Minka, J. Winn, J. Guiver, and D. Knowles. Infer.NET 2.4, Microsoft Research Cambridge, 2010., utl=http://research.microsoft.com/infernet.

R. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.

R. Neal. An improved acceptance procedure for the hybrid Monte Carlo algorithm. *Journal of Computational Physics*, 111:194–203, 1994.

R. Neal. Slice sampling. *Annals of Statistics*, 31(3):705–741, 2003.

R. Neal. *Handbook of Markov Chain Monte Carlo*, chapter 5: MCMC Using Hamiltonian Dynamics. CRC Press, 2011.

Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009.

A. Patil, D. Huard, and C. Fonnesbeck. PyMC: Bayesian stochastic modelling in python. *Journal of Statistical Software*, 35(4):1–81, 2010.

M. Plummer. JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling, 2003. URL `http://sourceforge.net/projects/mcmc-jags/`.

M. Plummer, N. Best, K. Cowles, and K. Vines. CODA: Convergence diagnosis and output analysis for MCMC. *R News*, 6(1):7–11, March 2006.

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

Stan Development Team. Stan: A C++ library for probability and sampling, version 1.1, 2013. URL `http://mc-stan.org/`.

L. Tierney and A. Mira. Some adaptive Monte Carlo methods for Bayesian inference. *Statistics in Medicine*, 18:2507–2515, 1999.

M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

Journal of Machine Learning Research 15 (2014) 1625-1651 Submitted 11/13; Revised 3/14; Published 5/14

Confidence Intervals for Random Forests: The Jackknife and the Infinitesimal Jackknife

Stefan Wager SWAGER@STANFORD.EDU
Trevor Hastie HASTIE@STANFORD.EDU
Bradley Efron BRAD@STAT.STANFORD.EDU

Department of Statistics
Stanford University
Stanford, CA 94305, USA

Editor: Nicolai Meinshausen

Abstract

We study the variability of predictions made by bagged learners and random forests, and show how to estimate standard errors for these methods. Our work builds on variance estimates for bagging proposed by Efron (1992, 2013) that are based on the jackknife and the infinitesimal jackknife (IJ). In practice, bagged predictors are computed using a finite number B of bootstrap replicates, and working with a large B can be computationally expensive. Direct applications of jackknife and IJ estimators to bagging require $B = \Theta(n^{1.5})$ bootstrap replicates to converge, where n is the size of the training set. We propose improved versions that only require $B = \Theta(n)$ replicates. Moreover, we show that the IJ estimator requires 1.7 times less bootstrap replicates than the jackknife to achieve a given accuracy. Finally, we study the sampling distributions of the jackknife and IJ variance estimates themselves. We illustrate our findings with multiple experiments and simulation studies.

Keywords: bagging, jackknife methods, Monte Carlo noise, variance estimation

1. Introduction

Bagging (Breiman, 1996) is a popular technique for stabilizing statistical learners. Bagging is often conceptualized as a variance reduction technique, and so it is important to understand how the sampling variance of a bagged learner compares to the variance of the original learner. In this paper, we develop and study methods for estimating the variance of bagged predictors and random forests (Breiman, 2001), a popular extension of bagged trees. These variance estimates only require the bootstrap replicates that were used to form the bagged prediction itself, and so can be obtained with moderate computational overhead. The results presented here build on the jackknife-after-bootstrap methodology introduced by Efron (1992) and on the infinitesimal jackknife for bagging (IJ) (Efron, 2013).

Figure 1 shows the results from applying our method to a random forest trained on the "Auto MPG" data set, a regression task where we aim to predict the miles-per-gallon (MPG) gas consumption of an automobile based on 7 features including weight and horsepower. The error bars shown in Figure 1 give an estimate of the sampling variance of the random forest; in other words, they tell us how much the random forest's predictions might change if we

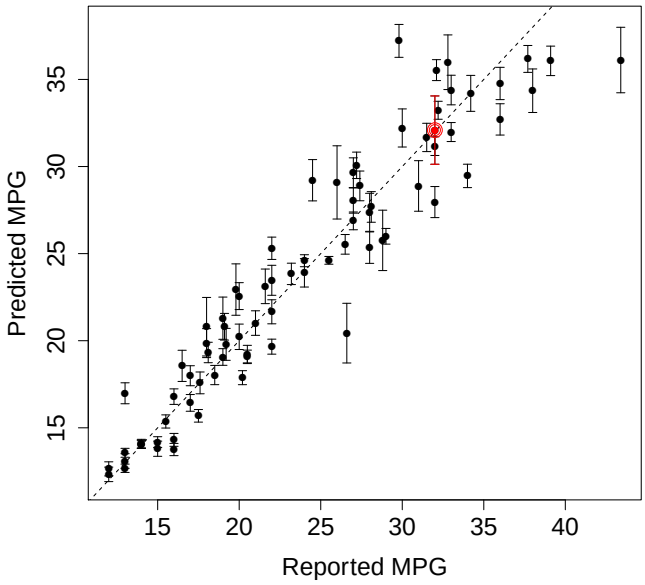


Figure 1: Random forest predictions on the "Auto MPG" data set. The random forest was trained using 314 examples; the graph shows results on a test set of size 78. The error bars are 1 standard error in each direction. Because this is a fairly small data set, we estimated standard errors for the random forest using the averaged estimator from Section 5.2. A more detailed description of the experiment is provided in Appendix C.

trained it on a new training set. The fact that the error bars do not in general cross the prediction-equals-observation diagonal suggests that there is some residual noise in the MPG of a car that cannot be explained by a random forest model based on the available predictor variables.[1]

Figure 1 tells us that the random forest was more confident about some predictions than others. Rather reassuringly, we observe that the random forest was in general less confident about the predictions for which the reported MPG and predicted MPG were very different. There is not a perfect correlation, however, between the error level and the size of the error bars. One of the points, circled in red near (32, 32), appears particularly surprising: the random forest got the prediction almost exactly right, but gave the prediction large error bars of ± 2. This curious datapoint corresponds to the 1982 Dodge Rampage, a two-door Coupe Utility that is a mix between a passenger car and a truck with a cargo tray. Perhaps our random forest had a hard time confidently estimating the mileage of the Rampage

1. Our method produces standard error estimates $\hat{\sigma}$ for random forest predictions. We then represent these standard error estimates as Gaussian confidence intervals $\hat{y} \pm z_\alpha \hat{\sigma}$, where z_α is a quantile of the normal distribution.

because it could not quite decide whether to cluster it with cars or with trucks. We present experiments on larger data sets in Section 3.

Estimating the variance of bagged learners based on the preexisting bootstrap replicates can be challenging, as there are two distinct sources of noise. In addition to the sampling noise (i.e., the noise arising from randomness during data collection), we also need to control the Monte Carlo noise arising from the use of a finite number of bootstrap replicates. We study the effects of both sampling noise and Monte Carlo noise.

In our experience, the errors of the jackknife and IJ estimates of variance are often dominated by Monte Carlo effects. Monte Carlo bias can be particularly troublesome: if we are not careful, the jackknife and IJ estimators can conflate Monte Carlo noise with the underlying sampling noise and badly overestimate the sampling variance. We show how to estimate the magnitude of this Monte Carlo bias and develop bias-corrected versions of the jackknife and IJ estimators that outperform the original ones. We also show that the IJ estimate of variance is able to use the preexisting bootstrap replicates more efficiently than the jackknife estimator by having a lower Monte Carlo variance, and needs 1.7 times less bootstrap replicates than the jackknife to achieve a given accuracy.

If we take the number of bootstrap replicates to infinity, Monte Carlo effects disappear and only sampling errors remain. We compare the sampling biases of both the jackknife and IJ rules and present some evidence that, while the jackknife rule has an upward sampling bias and the IJ estimator can have a downward bias, the arithmetic mean of the two variance estimates can be close to unbiased. We also propose a simple method for estimating the sampling variance of the IJ estimator itself.

Our paper is structured as follows. We first present an overview of our main results in Section 2, and apply them to random forest examples in Section 3. We then take a closer look at Monte Carlo effects in Section 4 and analyze the sampling distribution of the limiting IJ and jackknife rules with $B \to \infty$ in Section 5. We spread simulation experiments throughout Sections 4 and 5 to validate our theoretical analysis.

1.1 Related Work

In this paper, we focus on methods based on the jackknife and the infinitesimal jackknife for bagging (Efron, 1992, 2013) that let us estimate standard errors based on the pre-existing bootstrap replicates. Other approaches that rely on forming second-order bootstrap replicates have been studied by Duan (2011) and Sexton and Laake (2009). Directly bootstrapping a random forest is usually not a good idea, as it requires forming a large number of base learners. Sexton and Laake (2009), however, propose a clever work-around to this problem. Their approach, which could have been called a bootstrap of little bags, involves bootstrapping small random forests with around $B = 10$ trees and then applying a bias correction to remove the extra Monte Carlo noise.

There has been considerable interest in studying classes of models for which bagging can achieve meaningful variance reduction, and also in outlining situations where bagging can fail completely (e.g., Skurichina and Duin, 1998; Bühlmann and Yu, 2002; Chen and Hall, 2003; Buja and Stuetzle, 2006; Friedman and Hall, 2007). The problem of producing practical estimates of the sampling variance of bagged predictors, however, appears to have received somewhat less attention in the literature so far.

2. Estimating the Variance of Bagged Predictors

This section presents our main result: estimates of variance for bagged predictors that can be computed from the same bootstrap replicates that give the predictors. Section 3 then applies the result to random forests, which can be analyzed as a special class of bagged predictors.

Suppose that we have training examples $Z_1 = (x_1, y_1)$, ..., $Z_n = (x_n, y_n)$, an input x to a prediction problem, and a base learner $\hat{\theta}(x) = t(x; Z_1, ..., Z_n)$. To make things concrete, the Z_i could be a list of e-mails x_i paired with labels y_i that catalog the e-mails as either spam or non-spam, $t(x; Z_i)$ could be a decision tree trained on these labeled e-mails, and x could be a new e-mail that we seek to classify. The quantity $\hat{\theta}(x)$ would then be the output of the tree predictor on input x.

With bagging, we aim to stabilize the base learner t by resampling the training data. In our case, the bagged version of $\hat{\theta}(x)$ is defined as

$$\hat{\theta}^{\infty}(x) = \mathbb{E}_*[t(x; Z_1^*, ..., Z_n^*)], \tag{1}$$

where the Z_i^* are drawn independently with replacement from the original data (i.e., they form a bootstrap sample). The expectation $\mathbb{E}_*$ is taken with respect to the bootstrap measure.

The expectation in (1) cannot in general be evaluated exactly, and so we form the bagged estimator by Monte Carlo

$$\hat{\theta}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} t_b^*(x), \text{ where } t_b^*(x) = t(x; Z_{b1}^*, ..., Z_{bn}^*) \tag{2}$$

and the Z_{bi}^* are elements in the b^{th} bootstrap sample. As $B \to \infty$, we recover the perfectly bagged estimator $\hat{\theta}^{\infty}(x)$.

2.1 Basic Variance Estimates

The goal of our paper is to study the sampling variance of bagged learners

$$V(x) = \text{Var}\left[\hat{\theta}^{\infty}(x)\right].$$

In other words, we ask how much variance $\hat{\theta}^B$ would have once we make B large enough to eliminate the bootstrap effects. We consider two basic estimates of V: The *Infinitesimal Jackknife* estimate (Efron, 2013), which results in the simple expression

$$\widehat{V}_{IJ}^{\infty} = \sum_{i=1}^{n} \text{Cov}_*[N_i^*, t^*(x)]^2, \tag{3}$$

where $\text{Cov}_*[N_i^*, t^*(x)]$ is the covariance between $t^*(x)$ and the number of times N_i^* the i^{th} training example appears in a bootstrap sample; and the *Jackknife-after-Bootstrap* estimate (Efron, 1992)

$$\widehat{V}_{J}^{\infty} = \frac{n-1}{n}\sum_{i=1}^{n}\left(\bar{t}_{(-i)}^*(x) - \bar{t}^*(x)\right)^2, \tag{4}$$

where $\bar{t}^*_{(-i)}(x)$ is the average of $t^*(x)$ over all the bootstrap samples not containing the i^{th} example and $\bar{t}^*(x)$ is the mean of all the $t^*(x)$.

The jackknife-after-bootstrap estimate $\widehat{V}_J^\infty$ arises directly by applying the jackknife to the bootstrap distribution. The infinitesimal jackknife (Jaeckel, 1972), also called the non-parametric delta method, is an alternative to the jackknife where, instead of studying the behavior of a statistic when we remove one observation at a time, we look at what happens to the statistic when we individually down-weight each observation by an infinitesimal amount. When the infinitesimal jackknife is available, it sometimes gives more stable predictions than the regular jackknife. Efron (2013) shows how an application of the infinitesimal jackknife principle to the bootstrap distribution leads to the simple estimate $\widehat{V}_{IJ}^\infty$.

2.2 Finite-B Bias

In practice, we can only ever work with a finite number B of bootstrap replicates. The natural Monte Carlo approximations to the estimators introduced above are

$$\widehat{V}_{IJ}^B = \sum_{i=1}^n \widehat{\mathrm{Cov}}_i^2 \text{ with } \widehat{\mathrm{Cov}}_i = \frac{\sum_b (N^*_{bi} - 1)(t^*_b(x) - \bar{t}^*(x))}{B}, \tag{5}$$

and

$$\widehat{V}_J^B = \frac{n-1}{n} \sum_{i=1}^n \hat{\Delta}_i^2, \text{ where } \hat{\Delta}_i = \hat{\theta}^B_{(-i)}(x) - \hat{\theta}^B(x) \tag{6}$$

$$\text{and } \hat{\theta}^B_{(-i)}(x) = \frac{\sum_{\{b\,:\,N^*_{bi}=0\}} t^*_b(x)}{\left|\{N^*_{bi} = 0\}\right|}.$$

Here, N^*_{bi} indicates the number of times the i^{th} observation appears in the bootstrap sample b.

In our experience, these finite-B estimates of variance are often badly biased upwards if the number of bootstrap samples B is too small. Fortunately, bias-corrected versions are available:

$$\widehat{V}_{IJ-U}^B = \widehat{V}_{IJ}^B - \frac{n}{B^2} \sum_{b=1}^B (t^*_b(x) - \bar{t}^*(x))^2, \text{ and} \tag{7}$$

$$\widehat{V}_{J-U}^B = \widehat{V}_J^B - (e-1)\,\frac{n}{B^2} \sum_{b=1}^B (t^*_b(x) - \bar{t}^*(x))^2. \tag{8}$$

These bias corrections are derived in Section 4. In many applications, the simple estimators (5) and (6) require $B = \Theta(n^{1.5})$ bootstrap replicates to reduce Monte Carlo noise down to the level of the inherent sampling noise, whereas our bias-corrected versions only require $B = \Theta(n)$ replicates. The bias-corrected jackknife (8) was also discussed by Sexton and Laake (2009).

In Figure 2, we show how $\widehat{V}_{IJ-U}^B$ can be used to accurately estimate the variance of a bagged tree. We compare the true sampling variance of a bagged regression tree with our variance estimate. The underlying signal is a step function with four jumps that are reflected

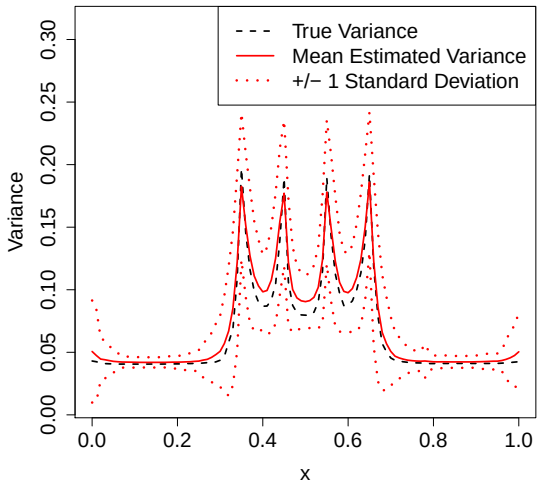


Figure 2: Testing the performance of the bias-corrected infinitesimal jackknife estimate of variance for bagged predictors, as defined in (15), on a bagged regression tree. We compare the true sampling error with the average standard error estimate produced by our method across multiple runs; the dotted lines indicate one-standard-error-wide confidence bands for our standard error estimate.

as spikes in the variance of the bagged tree. On average, our variance estimator accurately identifies the location and magnitude of these spikes.

Figure 3 compares the performance of the four considered variance estimates on a bagged adaptive polynomial regression example described in detail in Section 4.4. We see that the uncorrected estimators $\widehat{V}_J^B$ and $\widehat{V}_{IJ}^B$ are badly biased: the lower whiskers of their boxplots do not even touch the limiting estimate with $B \to \infty$. We also see that that $\widehat{V}_{IJ-U}^B$ has about half the variance of $\widehat{V}_{J-U}^B$. This example highlights the importance of using estimators that use available bootstrap replicates efficiently: with $B = 500$ bootstrap replicates, $\widehat{V}_{IJ-U}^B$ can give us a reasonable estimate of V, whereas $\widehat{V}_J^B$ is quite unstable and biased upwards by a factor 2.

The figure also suggests that the Monte Carlo noise of $\widehat{V}_{IJ}^B$ decays faster (as a function of B) than that of $\widehat{V}_J^B$. This is no accident: as we show in Section 4.2, the infinitesimal jackknife requires 1.7 times less bootstrap replicates than the jackknife to achieve a given level of level of Monte Carlo error.

2.3 Limiting Sampling Distributions

The performance of $\widehat{V}_J^B$ and $\widehat{V}_{IJ}^B$ depends on both sampling noise and Monte Carlo noise. In order for $\widehat{V}_J^B$ (and analogously $\widehat{V}_{IJ}^B$) to be accurate, we need both the sampling error of $\widehat{V}_J^\infty$, namely $\widehat{V}_J^\infty - V$, and the Monte Carlo error $\widehat{V}_J^B - \widehat{V}_J^\infty$ to be small.

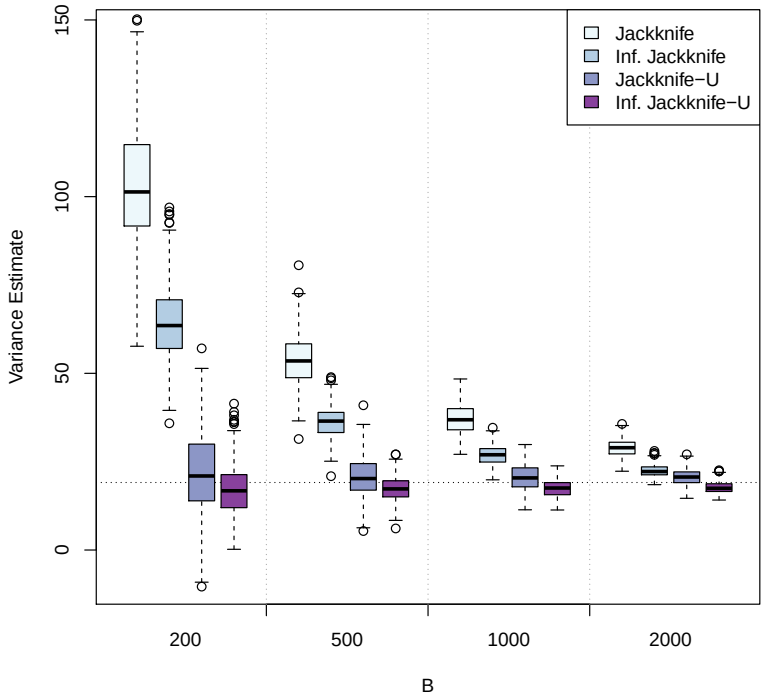


Figure 3: Performance, as a function of B, of the jackknife and IJ estimators and their bias-corrected modifications on the cholesterol data set of Efron and Feldman (1991). The boxplots depict bootstrap realizations of each estimator. The dotted line indicates the mean of all the realizations of the IJ-U and J-U estimators (weighted by B).

It is well known that jackknife estimates of variance are in general biased upwards (Efron and Stein, 1981). This phenomenon also holds for bagging: $\widehat{V}_J^\infty$ is somewhat biased upwards for V. We present some evidence suggesting that $\widehat{V}_{IJ}^\infty$ is biased downwards by a similar amount, and that the arithmetic mean of $\widehat{V}_J^\infty$ and $\widehat{V}_{IJ}^\infty$ is closer to being unbiased for V than either of the two estimators alone.

We also develop a simple estimator for the variance of $\widehat{V}_{IJ}^\infty$ itself:

$$\widehat{\mathrm{Var}}\left[\widehat{V}_{IJ}^\infty\right] = \sum_{i=1}^{n}\left(C_i^{*,2} - \overline{C_i^{*,2}}\right)^2,$$

where $C_i^* = \mathrm{Cov}_*[N_{bi}^*,\, t_b^*(x)]$ and $\overline{C_i^{*,2}}$ is the mean of the $C_i^{*,2}$.

3. Random Forest Experiments

Random forests (Breiman, 2001) are a widely used extension of bagged trees. Suppose that we have a tree-structured predictor t and training data Z_1, ..., Z_n. Using notation from (2), the bagged version of this tree predictor is

$$\hat{\theta}^B(x) = \frac{1}{B}\sum_{b=1}^{B} t_b^*(x;\, Z_{b1}^*,\, ...,\, Z_{bn}^*).$$

Random forests extend bagged trees by allowing the individual trees t_b^* to depend on an auxiliary noise source ξ_b. The main idea is that the auxiliary noise ξ_b encourages more diversity among the individual trees, and allows for more variance reduction than bagging. Several variants of random forests have been analyzed theoretically by, e.g., Biau et al. (2008), Biau (2012), Lin and Jeon (2006), and Meinshausen (2006).

Standard implementations of random forests use the auxiliary noise ξ_b to randomly restrict the number of variables on which the bootstrapped trees can split at any given training step. At each step, m features are randomly selected from the pool of all p possible features and the tree predictor must then split on one of these m features. If $m = p$ the tree can always split on any feature and the random forest becomes a bagged tree; if $m = 1$, then the tree has no freedom in choosing which feature to split on.

Following Breiman (2001), random forests are usually defined more abstractly for theoretical analysis: any predictor of the form

$$\hat{\theta}^{RF}(x) = \frac{1}{B}\sum_{b=1}^{B} t_b^*(x;\, \xi_b,\, Z_{b1}^*,\, ...,\, Z_{bn}^*) \text{ with } \xi_b \overset{\text{iid}}{\sim} \Xi \tag{9}$$

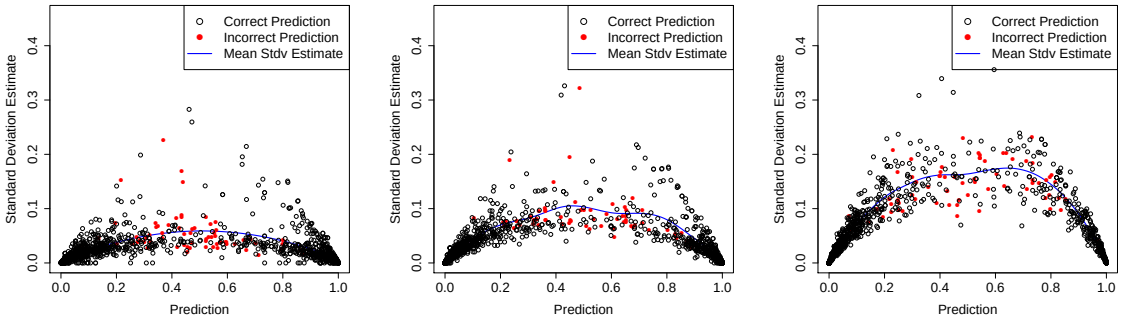
is called a *random forest*. Various choices of noise distribution Ξ lead to different random forest predictors. In particular, trivial noise sources are allowed and so the class of random forests includes bagged trees as a special case. In this paper, we only consider random forests of type (9) where individual trees are all trained on bootstrap samples of the training data. We note, however, that that variants of random forests that do not use bootstrap noise have also been found to work well (e.g., Dietterich, 2000; Geurts et al., 2006).

All our results about bagged predictors apply directly to random forests. The reason for this is that random forests can also be defined as bagged predictors with different base learners. Suppose that, on each bootstrap replicate, we drew K times from the auxiliary noise distribution Ξ instead of just once. This would give us a predictor of the form

$$\hat{\theta}^{RF}(x) = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{K}\sum_{k=1}^{K} t_b^*(x;\, \xi_{kb},\, Z_{b1}^*,\, ...,\, Z_{bn}^*) \text{ with } \xi_{kb} \overset{\text{iid}}{\sim} \Xi.$$

Adding the extra draws from Ξ to the random forest does not change the $B \to \infty$ limit of the random forest. If we take $K \to \infty$, we effectively marginalize over the noise from Ξ, and get a predictor

$$\hat{\theta}^{\widetilde{RF}}(x) = \frac{1}{B}\sum_{b=1}^{B} \tilde{t}_b^*(x;\, Z_{b1}^*,\, ...,\, Z_{bn}^*), \text{ where}$$
$$\tilde{t}(x;\, Z_1,\, ...,\, Z_n) = \mathbb{E}_{\xi\sim\Xi}\left[t(x;\, \xi,\, Z_1,\, ...,\, Z_n)\right].$$



(a) $m = 5$ (b) $m = 19$ (c) $m = 57$ (bagged tree)

Figure 4: Standard errors of random forest predictions on the e-mail spam data. The random forests with $m = 5$, 19, and 57 splitting variables were all trained on a train set of size $n = 3,065$; the panels above show class predictions and IJ-U estimates for standard errors on a test set of size 1,536. The solid curves are smoothing splines ($df = 4$) fit through the data (including both correct and incorrect predictions).

In other words, the random forest $\hat{\theta}^{RF}$ as defined in (9) is just a noisy estimate of a bagged predictor with base learner $\tilde{t}$.

It is straight-forward to check that our results about $\widehat{V}_{IJ}^B$ and $\widehat{V}_J^B$ also hold for bagged predictors with randomized base learners. The extra noise from using $t(\cdot; \xi)$ instead of $\tilde{t}(\cdot)$ does not affect the limiting correlations in (3) and (4); meanwhile, the bias corrections from (7) and (8) do not depend on how we produced the t^* and remain valid with random forests. Thus, we can estimate confidence intervals for random forests from N^* and t^* using exactly the same formulas as for bagging.

In the rest of this section, we show how the variance estimates studied in this paper can be used to gain valuable insights in applications of random forests. We use the $\widehat{V}_{IJ-U}^B$ variance estimate (7) to minimize the required computational resources. We implemented the IJ-U estimator for random forests on top of the `R` package `randomForest` (Liaw and Wiener, 2002).

3.1 E-mail Spam Example

The e-mail spam data set (spambase) is part of a standard classification task, the goal of which is to distinguish spam e-mail (1) from non-spam (0) using $p = 57$ features. Here, we investigate the performance of random forests on this data set.

We fit the spam data using random forests with $m = 5$, 19 and 57 splitting variables. With $m = 5$, the trees were highly constrained in their choice of splitting variables, while $m = 57$ is just a bagged tree. The three random forests obtained test-set accuracies of 95.1%, 95.2% and 94.7% respectively, and it appears that the $m = 5$ or 19 forests are best. We can use the IJ-U variance formula to gain deeper insight into these numbers, and get a better understanding about what is constraining the accuracy of each predictor.

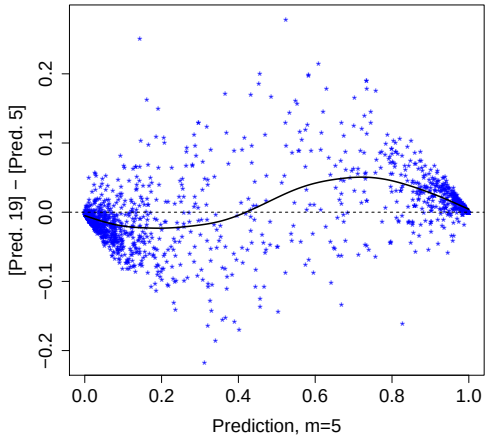
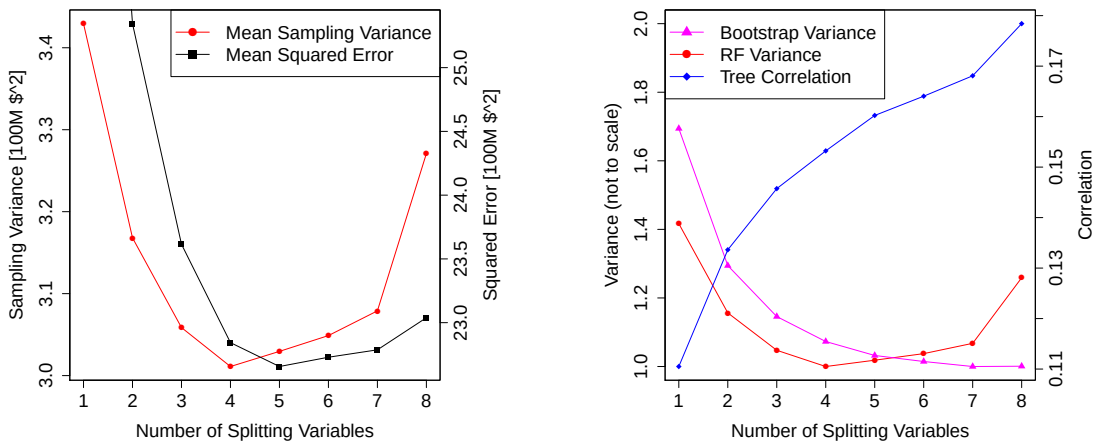


Figure 5: Comparison of the predictions made by the $m = 5$ and $m = 19$ random forests. The stars indicate pairs of test set predictions; the solid line is a smoothing spline ($df = 6$) fit through the data.

In Figure 4, we plot test-set predictions against IJ-U estimates of standard error for all three random forests. The $m = 57$ random forest appears to be quite unstable, in that the estimated errors are high. Because many of its predictions have large standard errors, it is plausible that the predictions made by the random forest could change drastically if we got more training data. Thus, the $m = 57$ forest appears to suffer from overfitting, and the quality of its predictions could improve substantially with more data.

Conversely, predictions made by the $m = 5$ random forest appear to be remarkably stable, and almost all predictions have standard errors that lie below 0.1. This suggests that the $m = 5$ forest may be mostly constrained by bias: if the predictor reports that a certain e-mail is spam with probability 0.5 ± 0.1, then the predictor has effectively abandoned any hope of unambiguously classifying the e-mail. Even if we managed to acquire much more training data, the class prediction for that e-mail would probably not converge to a strong vote for spam or non-spam.

The $m = 19$ forest appears to have balanced the bias-variance trade-off well. We can further corroborate our intuition about the bias problem faced by the $m = 5$ forest by comparing its predictions with those of the $m = 19$ forest. As shown in Figure 5, whenever the $m = 5$ forest made a cautious prediction that an e-mail might be spam (e.g., a prediction of around 0.8), the $m = 19$ forest made the same classification decision but with more confidence (i.e., with a more extreme class probability estimate $\hat{p}$). Similarly, the $m = 19$ forest tended to lower cautious non-spam predictions made by the $m = 5$ forest. In other words, the $m = 5$ forest appears to have often made lukewarm predictions with mid-range values of $\hat{p}$ on e-mails for which there was sufficient information in the data to make confident predictions. This analysis again suggests that the $m = 5$ forest was constrained by bias and was not able to efficiently use all the information present in the data set.



(a) MSE and mean variance

(b) Drivers of sampling variance

Figure 6: Performance of random forests on the California housing data. The left panel plots MSE and mean sampling variance as a function of the number m of splitting variables. The MSE estimate is the out-of bag error, while the mean sampling variance is the average estimate of variance $\widehat{V}^B_{IJ-U}$ computed over all training examples. The right panel displays the drivers of sampling variance, namely the variance of the individual bootstrapped trees (bootstrap variance v) and their correlation (tree correlation ρ).

3.2 California Housing Example

In the previous example, we saw that the varying accuracy of random forests with different numbers m of splitting variables primarily reflected a bias-variance trade-off. Random forests with small m had high bias, while those with large m had high variance. This bias-variance trade-off does not, however, underlie all random forests. The California housing data set—a regression task with $n = 20,460$ and $p = 8$—provides a contrasting example.

In Figure 6a, we plot the random forest out-of-bag MSE and IJ-U estimate of average sampling variance across all training examples, with m between 1 and 8. We immediately notice that the sampling variance is not monotone increasing in m. Rather, the sampling variance is high if m is too big or too small, and attains a minimum at $m = 4$. Meanwhile, in terms of MSE, the optimal choice is $m = 5$. Thus, there is no bias-variance trade-off here: picking a value of m around 4 or 5 is optimal both from the MSE minimization and the variance minimization points of view.

We can gain more insight into this phenomenon using ideas going back to Breiman (2001), who showed that the sampling variance of a random forest is governed by two factors: the variance v of the individual bootstrapped trees and their correlation ρ. The variance of the ensemble is then ρv. In Figure 6b, we show how both v and ρ react when we vary m. Trees with large m are fairly correlated, and so the random forest does not get as substantial a variance reduction over the base learner as with a smaller m. With a very small m, however,

the variance v of the individual trees shoots up, and so the decrease in ρ is no longer sufficient to bring down the variance of the whole forest. The increasing ρ-curve and the decreasing v-curve thus jointly produce a U-shaped relationship between m and the variance of the random forest. The $m = 4$ forest achieves a low variance by matching fairly stable base learners with a small correlation ρ.

4. Controlling Monte Carlo Error

In this section, we analyze the behavior of both the IJ and jackknife estimators under Monte Carlo noise. We begin by discussing the Monte Carlo distribution of the infinitesimal jackknife estimate of variance with a finite B; the case of the jackknife-after-bootstrap estimate of variance is similar but more technical and is presented in Appendix A. We show that the jackknife estimator needs 1.7 times more bootstrap replicates than the IJ estimator to control Monte Carlo noise at a given level. We also highlight a bias problem for both estimators, and recommend a bias correction. When there is no risk of ambiguity, we use the short-hand t^* for $t^*(x)$.

4.1 Monte Carlo Error for the IJ Estimator

We first consider the Monte Carlo bias of the infinitesimal jackknife for bagging. Let

$$\widehat{V}_{IJ}^{\infty} = \sum_{i=1}^{n} \mathrm{Cov}_*[N_i^*,\, t^*]^2 \tag{10}$$

be the perfect IJ estimator with $B = \infty$ (Efron, 2013). Then, the Monte Carlo bias of $\widehat{V}_{IJ}^{B}$ is

$$\mathbb{E}_*\left[\widehat{V}_{IJ}^{B}\right] - \widehat{V}_{IJ}^{\infty} = \sum_{i=1}^{n} \mathrm{Var}_*[C_i], \text{ where } C_i = \frac{\sum_b (N_{bi}^* - 1)(t_b^* - \bar{t}^*)}{B}$$

is the Monte Carlo estimate of the bootstrap covariance. Since t_b^* depends on all n observations, N_{bi}^* and t_b^* can in practice be treated as independent for computing $\mathrm{Var}_*[C_i]$, especially when n is large (see remark below). Thus, as $\mathrm{Var}_*[N_{bi}^*] = 1$, we see that

$$\mathbb{E}_*\left[\widehat{V}_{IJ}^{B}\right] - \widehat{V}_{IJ}^{\infty} \approx \frac{n\,\hat{v}}{B}, \text{ where } \hat{v} = \frac{1}{B}\sum_{b=1}^{B} (t_b^* - \bar{t}^*)^2. \tag{11}$$

Notice that $\hat{v}$ is the standard bootstrap estimate for the variance of the base learner $\hat{\theta}(x)$. Thus, the bias of $\widehat{V}_{IJ}^{B}$ grows linearly in the variance of the original estimator that is being bagged.

Meanwhile, by the central limit theorem, C_i converges to a Gaussian random variable as B gets large. Thus, the Monte Carlo asymptotic variance of C_i^2 is approximately $2\,\mathrm{Var}_*[C_i]^2 + 4\,\mathbb{E}_*[C_i]^2\,\mathrm{Var}_*[C_i]$. The C_i can be treated as roughly independent, and so the limiting distribution of the IJ estimate of variance has approximate moments

$$\widehat{V}_{IJ}^{B} - \widehat{V}_{IJ}^{\infty} \dot{\sim} \left(\frac{n\,\hat{v}}{B},\, 2\,\frac{n\,\hat{v}^2}{B^2} + 4\,\frac{\widehat{V}_{IJ}^{\infty}\,\hat{v}}{B}\right). \tag{12}$$

Interestingly, the Monte Carlo mean squared error (MSE) of $\widehat{V}_{IJ}^{B}$ mostly depends on the problem through $\hat{v}$, where $\hat{v}$ is the bootstrap estimate of the variance of the base learner. In other words, the computational difficulty of obtaining confidence intervals for bagged learners depends on the variance of the base learner.

4.1.1 Remark: The IJ Estimator for Sub-bagging

We have focused on the case where each bootstrap replicate contains exactly n samples. However, in some applications, bagging with subsamples of size $m \neq n$ has been found to work well (e.g., Bühlmann and Yu, 2002; Buja and Stuetzle, 2006; Friedman, 2002; Strobl et al., 2007). Our results directly extend to the case where $m \neq n$ samples are drawn with replacement from the original sample. We can check that (10) still holds, but now $\operatorname{Var} N_{bi}^{*} = m/n$. Carrying out the same analysis as above, we can establish an analogue to (12):

$$\widehat{V}_{IJ}^{B}(m) - \widehat{V}_{IJ}^{\infty}(m) \dot{\sim} \left(\frac{m\,\hat{v}}{B},\, 2\,\frac{m^2\,\hat{v}^2}{nB^2} + 4\,\frac{m\widehat{V}_{IJ}^{\infty}\,\hat{v}}{nB} \right). \tag{13}$$

For simplicity of exposition, we will restrict our analysis to the case $m = n$ for the rest of this paper.

4.1.2 Remark: Approximate Independence

In the above derivation, we used the approximation

$$\operatorname{Var}_{*}\left[(N_{bi}^{*}-1)(t_b^{*}-\bar{t}^{*})\right] \approx \operatorname{Var}_{*}\left[N_{bi}^{*}\right] \operatorname{Var}_{*}\left[t_b^{*}\right].$$

We can evaluate the accuracy of this approximation using the formula

$$\begin{aligned} &\operatorname{Var}_{*}\left[(N_{bi}^{*}-1)(t_b^{*}-\bar{t}^{*})\right] - \operatorname{Var}_{*}\left[N_{bi}^{*}\right] \operatorname{Var}\left[t_b^{*}\right] \\ &\quad = \operatorname{Cov}_{*}\left[(N_{bi}^{*}-1)^2,\, (t_b^{*}-\bar{t}^{*})^2\right] - \operatorname{Cov}_{*}\left[(N_{bi}^{*}-1),\, (t_b^{*}-\bar{t}^{*})\right]^2. \end{aligned}$$

In the case of the sample mean $t(Z_1^{*}, ..., Z_n^{*}) = \frac{1}{n}\sum_i Z_i^{*}$ paired with the Poisson bootstrap, this term reduces to

$$\operatorname{Cov}_{*}\left[(N_{bi}^{*}-1)^2,\, (t_b^{*}-\bar{t}^{*})^2\right] - \operatorname{Cov}_{*}\left[(N_{bi}^{*}-1),\, (t_b^{*}-\bar{t}^{*})\right]^2 = 2\frac{\left(Z_i-\bar{Z}\right)^2}{n^2},$$

and the correction to (11) would be $2\hat{v}/(nB) \ll n\hat{v}/B$.

4.2 Comparison of Monte Carlo Errors

As shown in Appendix A, the Monte Carlo error for the jackknife-after-bootstrap estimate of variance has approximate moments

$$\widehat{V}_{J}^{B} - \widehat{V}_{J}^{\infty} \dot{\sim} \left((e-1)\,\frac{n\,\hat{v}}{B},\, 2\,(e-1)^2\frac{n\,\hat{v}^2}{B^2} + 4\,(e-1)\,\frac{\widehat{V}_{J}^{\infty}\,\hat{v}}{B} \right), \tag{14}$$

where $\widehat{V}_{J}^{\infty}$ is the jackknife estimate computed with $B = \infty$ bootstrap replicates. The Monte Carlo stability of $\widehat{V}_{J}^{B}$ again primarily depends on $\hat{v}$.

By comparing (12) with (14), we notice that the IJ estimator makes better use of a finite number B of bootstrap replicates than the jackknife estimator. For a fixed value of B, the Monte Carlo bias of $\widehat{V}_J^B$ is about $e-1$ or 1.7 times as large as that of $\widehat{V}_{IJ}^B$; the ratio of Monte Carlo variance starts off at 3 for small values of B and decays down to 1.7 as B gets much larger than n. Alternatively, we see that the IJ estimate with B bootstrap replicates has errors on the same scale as the jackknife estimate with $1.7 \cdot B$ replicates.

This suggests that if computational considerations matter and there is a desire to perform as few bootstrap replicates B as possible while controlling Monte Carlo error, the infinitesimal jackknife method may be preferable to the jackknife-after-bootstrap.

4.3 Correcting for Monte Carlo Bias

The Monte Carlo MSEs of $\widehat{V}_{IJ}^B$ and $\widehat{V}_J^B$ are in practice dominated by bias, especially for large n. Typically, we would like to pick B large enough to keep the Monte Carlo MSE on the order of $1/n$. For both (12) and (14), we see that performing $B = \Theta(n)$ bootstrap iterations is enough to control the variance. To reduce the bias to the desired level, namely $\mathcal{O}(n^{-0.5})$, we would need to take $B = \Theta(n^{1.5})$ bootstrap samples.

Although the Monte Carlo bias for both $\widehat{V}_{IJ}^B$ and $\widehat{V}_J^B$ is large, this bias only depends on $\hat{v}$ and so is highly predictable. This suggests a bias-corrected modification of the IJ and jackknife estimators respectively:

$$\widehat{V}_{IJ-U}^B = \widehat{V}_{IJ}^B - \frac{n\,\hat{v}}{B}, \text{ and} \tag{15}$$

$$\widehat{V}_{J-U}^B = \widehat{V}_J^B - (e-1)\,\frac{n\,\hat{v}}{B}. \tag{16}$$

Here $\widehat{V}_{IJ}^B$ and $\widehat{V}_J^B$ are as defined in (5), and $\hat{v}$ is the bootstrap estimate of variance from (11). The letter U stands for unbiased. This transformation effectively removes the Monte Carlo bias in our experiments without noticeably increasing variance. The bias corrected estimates only need $B = \Theta(n)$ bootstrap replicates to control Monte Carlo MSE at level $1/n$.

4.4 A Numerical Example

To validate the observations made in this section, we re-visit the cholesterol data set used by Efron (2013) as a central example in developing the IJ estimate of variance. The data set (introduced by Efron and Feldman, 1991) contains records for $n = 164$ participants in a clinical study, all of whom received a proposed cholesterol-lowering drug. The data contains a measure d of the cholesterol level decrease observed for each subject, as well as a measure c of compliance (i.e. how faithful the subject was in taking the medication). Efron and Feldman originally fit d as a polynomial function of c; the degree of the polynomial was adaptively selected by minimizing Mallows' C_p criterion (1973).

We here follow Efron (2013) and study the bagged adaptive polynomial fit of d against c for predicting the cholesterol decrease of a new subject with a specific compliance level. The degree of the polynomial is selected among integers between 1 and 6 by C_p minimization. Efron (2013) gives a more detailed description of the experiment. We restrict our attention

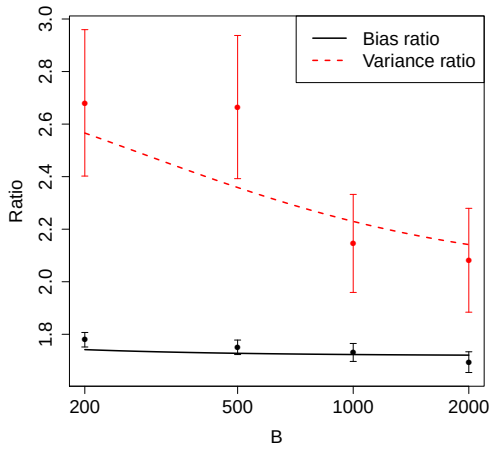


Figure 7: Predicted and actual performance ratios for the uncorrected $\widehat{V}_J^B$ and $\widehat{V}_{IJ}^B$ estimators in the cholesterol compliance example. The plot shows both $\mathrm{Var}[\widehat{V}_J^B]/\mathrm{Var}[\widehat{V}_{IJ}^B]$ and $\mathrm{Bias}[\widehat{V}_J^B]/\mathrm{Bias}[\widehat{V}_{IJ}^B]$. The observations are derived from the data presented in Figure 3; the error bars are one standard deviation in each direction. The solid lines are theoretical predictions obtained from (12) and (14).

to predicting the cholesterol decrease of a new patient with compliance level $c = -2.25$; this corresponds to the patient with the lowest observed compliance level.

In Figure 3, we compare the performance of the variance estimates for bagged predictors studied in this paper. The boxplots depict repeated realizations of the variance estimates with a finite B. We can immediately verify the qualitative insights presented in this section. Both the jackknife and IJ rules are badly biased for small B, and this bias goes away more slowly than the Monte Carlo variance. Moreover, at any given B, the jackknife estimator is noticeably less stable than the IJ estimator.

The J-U and IJ-U estimators appear to fix the bias problem without introducing instability. The J-U estimator has a slightly higher mean than the IJ-U one. As discussed in Section 5.2, this is not surprising, as the limiting ($B \to \infty$) jackknife estimator has an upward sampling bias while the limiting IJ estimator can have a downward sampling bias. The fact that the J-U and IJ-U estimators are so close suggests that both methods work well for this problem.

The insights developed here also appear to hold quantitatively. In Figure 7, we compare the ratios of Monte Carlo bias and variance for the jackknife and IJ estimators with theoretical approximations implied by (12) and (14). The theoretical formulas appear to present a credible picture of the relative merits of the jackknife and IJ rules.

5. Sampling Distribution of Variance Estimates

In practice, the $\widehat{V}_{IJ}^B$ and $\widehat{V}_J^B$ estimates are computed with a finite number B of bootstrap replicates. In this section, however, we let B go to infinity, and study the sampling properties of the IJ and jackknife variance estimates in the absence of Monte Carlo errors. In other words, we study the impact of noise in the data itself. Recall that we write $\widehat{V}_{IJ}^\infty$ and $\widehat{V}_J^\infty$ for the limiting estimators with $B = \infty$ bootstrap replicates.

We begin by developing a simple formula for the sampling variance of $\widehat{V}_{IJ}^\infty$ itself. In the process of developing this variance formula, we obtain an ANOVA expansion of $\widehat{V}_{IJ}^\infty$ that we then use in Section 5.2 to compare the sampling biases of the jackknife and infinitesimal jackknife estimators.

5.1 Sampling Variance of the IJ Estimate of Variance

If the data Z_i are independently drawn from a distribution F, then the variance of the IJ estimator is very nearly given by

$$\mathrm{Var}_F\left[\widehat{V}_{IJ}^\infty\right] \approx n\,\mathrm{Var}_F\left[h_F^2(Z)\right], \text{ where} \tag{17}$$

$$h_F(Z) = \mathbb{E}_F\left[\hat{\theta}^\infty | Z_1 = Z\right] - \mathbb{E}_F\left[\hat{\theta}^\infty\right]. \tag{18}$$

This expression suggests a natural plug-in estimator

$$\widehat{\mathrm{Var}[\widehat{V}_{IJ}^B]} = \sum_{i=1}^n \left(C_i^{*,2} - \overline{C_i^{*,2}}\right)^2, \tag{19}$$

where $C_i^* = \mathrm{Cov}_*[N_{bi}^*, t_b^*]$ is a bootstrap estimate for $h_F(Z_i)$ and $\overline{C_i^{*,2}}$ is the mean of the $C_i^{*,2}$. The rest of the notation is as in Section 2.

The relation (17) arises from a general connection between the infinitesimal jackknife and the theory of Hájek projections. The Hájek projection of an estimator is the best approximation to that estimator that only considers first-order effects. In our case, the Hájek projection of $\hat{\theta}^\infty$ is

$$\hat{\theta}_H^\infty = \mathbb{E}_F\left[\hat{\theta}^\infty\right] + \sum_{i=1}^n h_F(Z_i), \tag{20}$$

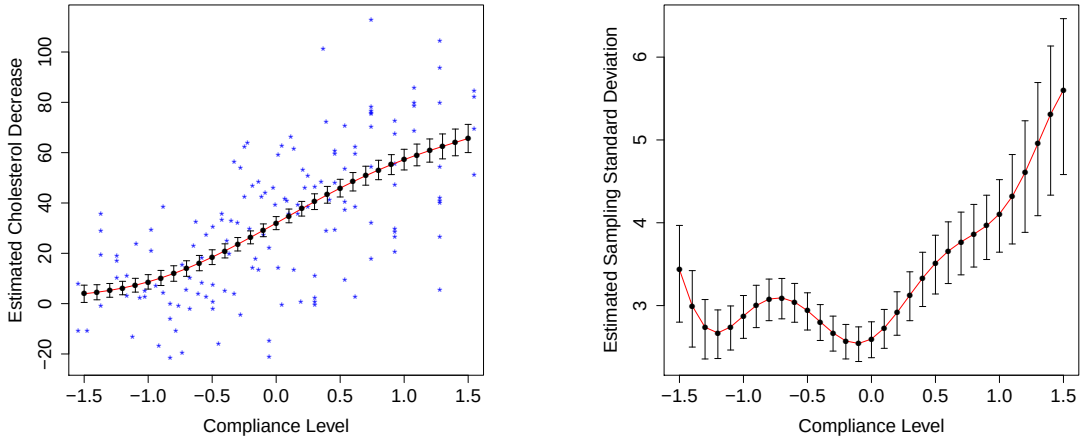
where $h_F(Z_i)$ is as in (18). The variance of the Hájek projection is $\mathrm{Var}\left[\hat{\theta}_H^\infty\right] = n\,\mathrm{Var}\left[h_F(Z)\right]$.

The key insight behind (17) is that the IJ estimator is effectively trying to estimate the variance of the Hájek projection of $\hat{\theta}^B$, and that

$$\widehat{V}_{IJ}^\infty \approx \sum_{i=1}^n h_F^2(Z_i). \tag{21}$$

The approximation (17) then follows immediately, as the right-hand side of the above expression is a sum of independent random variables. Note that we cannot apply this right-hand side expression directly, as h depends on the unknown underlying distribution F.

The connections between Hájek projections and the infinitesimal jackknife have been understood for a long time. Jaeckel (1972) originally introduced the infinitesimal jackknife



(a) Bagged polynomial fit.

(b) Errors for the IJ estimator

Figure 8: Stability of the IJ estimate of variance on the cholesterol data. The left panel shows the bagged fit to the data, along with error bars generated by the IJ method; the stars denote the data (some data points have x-values that exceed the range of the plot). In the right panel, we use (19) to estimate error bars for the error bars in the first panel. All error bars are one standard deviation in each direction.

as a practical approximation to the first-order variance of an estimator (in our case, the right-hand side of (21)). More recently, Efron (2013) showed that $\widehat{V}_{IJ}^{\infty}$ is equal to the variance of a "bootstrap Hájek projection." In Appendix B, we build on these ideas and show that, in cases where a plug-in approximation is valid, (21) holds very nearly for bagged estimators.

We apply our variance formula to the cholesterol data set of Efron (2013), following the methodology described in Section 4.4. In Figure 8, we use the formula (19) to study the sampling variance of $\widehat{V}_{IJ}^{\infty}$ as a function of the compliance level c. The main message here is rather reassuring: as seen in Figure 8b, the coefficient of variation of $\widehat{V}_{IJ}^{\infty}$ appears to be fairly low, suggesting that the IJ variance estimates can be trusted in this example. Note that, the formula from (19) can require many bootstrap replicates to stabilize and suffers from an upward Monte Carlo bias just like $\widehat{V}_{IJ}^{B}$. We used $B = 100,000$ bootstrap replicates to generate Figure 8.

5.2 Sampling Bias of the Jackknife and IJ Estimators

We can understand the sampling biases of both the jackknife and IJ estimators in the context of the ANOVA decomposition of Efron and Stein (1981). Suppose that we have data Z_1, ..., Z_n drawn independently from a distribution F, and compute our estimate $\hat{\theta}^{\infty}$ based on this data. Then, we can decompose its variance as

$$\mathrm{Var}_F\left[\hat{\theta}^{\infty}\right] = V_1 + V_2 + ... + V_n, \tag{22}$$

where

$$V_1 = n \operatorname{Var}_F \left[\mathbb{E}_F \left[\hat{\theta}^\infty | Z_1 \right] \right]$$

is the variance due to first-order effects, V_2 is the variance due to second-order effects of the form

$$\mathbb{E}_F \left[\hat{\theta}^\infty | Z_1,\, Z_2 \right] - \mathbb{E}_F \left[\hat{\theta}^\infty | Z_1 \right] - \mathbb{E}_F \left[\hat{\theta}^\infty | Z_2 \right] + \mathbb{E}_F \left[\hat{\theta}^\infty \right],$$

and so on. Note that all the terms V_k are non-negative.

Efron and Stein (1981) showed that, under general conditions, the jackknife estimate of variance is biased upwards. In our case, their result implies that the jackknife estimator computed on $n+1$ data points has variance

$$\mathbb{E}_F \left[\widehat{V}_J^\infty \right] = V_1 + 2V_2 + 3V_3 + ... + nV_n. \tag{23}$$

Meanwhile, (21) suggests that

$$\mathbb{E}_F \left[\widehat{V}_{IJ}^\infty \right] \approx V_1. \tag{24}$$

In other words, on average, both the jackknife and IJ estimators get the first-order variance term right. The jackknife estimator then proceeds to double the second-order term, triple the third-order term etc, while the IJ estimator just drops the higher order terms.

By comparing (23) and (24), we see that the upward bias of $\widehat{V}_J^\infty$ and the downward bias of $\widehat{V}_{IJ}^\infty$ partially cancel out. In fact,

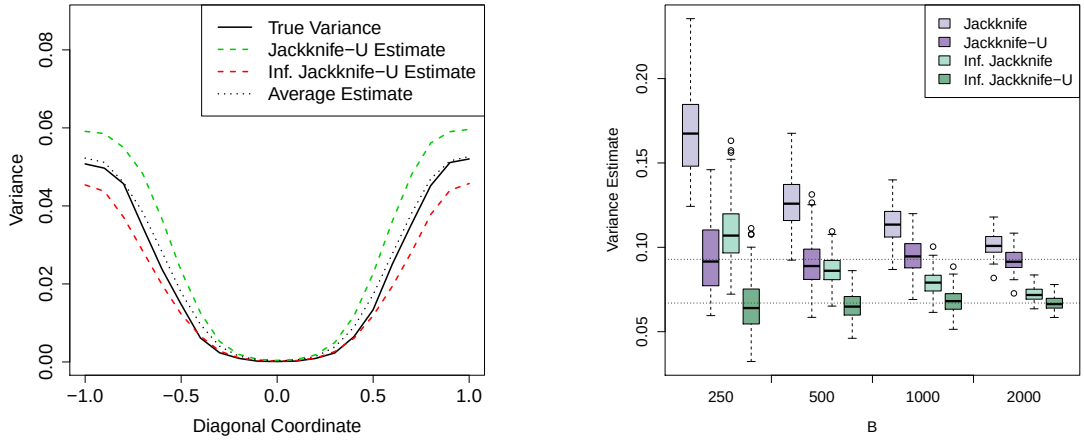
$$\mathbb{E}_F \left[\frac{\widehat{V}_J^\infty + \widehat{V}_{IJ}^\infty}{2} \right] \approx V_1 + V_2 + \frac{3}{2} V_3 + ... + \frac{n}{2} V_n, \tag{25}$$

and so the arithmetic mean of $\widehat{V}_J^\infty$ and $\widehat{V}_{IJ}^\infty$ has an upward bias that depends only on third- and higher-order effects. Thus, we might expect that in small-sample situations where $\widehat{V}_J^\infty$ and $\widehat{V}_{IJ}^\infty$ exhibit some bias, the mean of the two estimates may work better than either of them taken individually.

To test this idea, we used both the jackknife and IJ methods to estimate the variance of a bagged tree trained on a sample of size $n = 25$. (See Appendix C for details.) Since the sample size is so small, both the jackknife and IJ estimators exhibit some bias as seen in Figure 9a. However, the mean of the two estimators is nearly unbiased for the true variance of the bagged tree. (It appears that this mean has a very slight upward bias, just as we would expect from (25).)

This issue can arise in real data sets too. When training bagged forward stepwise regression on a prostate cancer data set discussed by Hastie et al. (2009), the jackknife and IJ methods give fairly different estimates of variance: the jackknife estimator converged to 0.093, while the IJ estimator stabilized at 0.067 (Figure 9b). Based on the discussion in this section, it appears that $(0.093 + 0.067)/2 = 0.08$ should be considered a more unbiased estimate of variance than either of the two numbers on their own.

In the more extensive simulations presented in Table 1, averaging $\widehat{V}_{IJ-U}^B$ and $\widehat{V}_{J-U}^B$ is in general less biased than either of the original estimators (although the "AND" experiment seems to provide an exception to this rule, suggesting that most of the bias of $\widehat{V}_{J-U}^B$ for this function is due to higher-order interactions). However, $\widehat{V}_{IJ-U}^B$ has systematically lower



(a) Bagged tree sampling bias.

(b) Prostate cancer example.

Figure 9: Sampling bias of the jackknife and IJ rules. In the left panel, we compare the expected values of the jackknife and IJ estimators as well as their mean with the true variance of a bagged tree. In this example, the features take values in $(x_1, x_2) \in [-1, 1]^2$; we depict variance estimates along the diagonal $x_1 = x_2$. The prostate cancer plot can be interpreted in the same way as Figure 3, except that the we now indicate the weighted means of the J-U and IJ-U estimators separately.

variance, which allows it to win in terms of overall mean squared error. Thus, if unbiasedness is important, averaging $\widehat{V}^B_{IJ-U}$ and $\widehat{V}^B_{J-U}$ seems like a promising idea, but $\widehat{V}^B_{IJ-U}$ appears to be the better rule in terms of raw MSE minimization.

Finally, we emphasize that this relative bias result relies on the heuristic relationship (24). While this approximation does not seem problematic for the first-order analysis presented in Section 5.1, we may be concerned that the plug-in argument from Appendix B used to justify it may not give us correct second- and higher-order terms. Thus, although our simulation results seem promising, developing a formal and general understanding of the relative biases of $\widehat{V}^\infty_{IJ}$ and $\widehat{V}^\infty_{J}$ remains an open topic for follow-up research.

6. Conclusion

In this paper, we studied the jackknife-after-bootstrap and infinitesimal jackknife (IJ) methods (Efron, 1992, 2013) for estimating the variance of bagged predictors. We demonstrated that both estimators suffer from considerable Monte Carlo bias, and we proposed bias-corrected versions of the methods that appear to work well in practice. We also provided a simple formula for the sampling variance of the IJ estimator, and showed that from a sampling bias point of view the arithmetic mean of the jackknife and IJ estimators is often preferable to either of the original methods. Finally, we applied these methods in numerous

Function	n	p	B	ERR	$\widehat{V}^B_{IJ-U}$	$\widehat{V}^B_{J-U}$	$\frac{1}{2}(\widehat{V}^B_{IJ-U} + \widehat{V}^B_{J-U})$
Cosine	50	2	200	Bias	$-0.15\,(\pm 0.03)$	$0.14\,(\pm 0.02)$	$\mathbf{-0.01}\,(\pm\mathbf{0.02})$
				Var	$\mathbf{0.08}\,(\pm\mathbf{0.02})$	$0.41\,(\pm 0.13)$	$0.2\,(\pm 0.06)$
				MSE	$\mathbf{0.11}\,(\pm\mathbf{0.03})$	$0.43\,(\pm 0.13)$	$0.2\,(\pm 0.06)$
Cosine	200	2	500	Bias	$-0.05\,(\pm 0.01)$	$0.07\,(\pm 0.01)$	$\mathbf{0.01}\,(\pm\mathbf{0.01})$
				Var	$\mathbf{0.02}\,(\pm\mathbf{0})$	$0.07\,(\pm 0.01)$	$0.04\,(\pm 0.01)$
				MSE	$\mathbf{0.02}\,(\pm\mathbf{0})$	$0.07\,(\pm 0.01)$	$0.04\,(\pm 0.01)$
XOR	50	50	200	Bias	$-0.3\,(\pm 0.03)$	$0.37\,(\pm 0.04)$	$\mathbf{0.03}\,(\pm\mathbf{0.03})$
				Var	$\mathbf{0.48}\,(\pm\mathbf{0.03})$	$1.82\,(\pm 0.12)$	$0.89\,(\pm 0.05)$
				MSE	$\mathbf{0.58}\,(\pm\mathbf{0.03})$	$1.96\,(\pm 0.13)$	$0.89\,(\pm 0.05)$
XOR	200	50	500	Bias	$\mathbf{-0.08}\,(\pm\mathbf{0.02})$	$0.24\,(\pm 0.03)$	$\mathbf{0.08}\,(\pm\mathbf{0.02})$
				Var	$\mathbf{0.26}\,(\pm\mathbf{0.02})$	$0.77\,(\pm 0.04)$	$0.4\,(\pm 0.02)$
				MSE	$\mathbf{0.27}\,(\pm\mathbf{0.01})$	$0.83\,(\pm 0.04)$	$0.41\,(\pm 0.02)$
AND	50	500	200	Bias	$\mathbf{-0.23}\,(\pm\mathbf{0.04})$	$0.65\,(\pm 0.05)$	$\mathbf{0.21}\,(\pm\mathbf{0.04})$
				Var	$\mathbf{1.15}\,(\pm\mathbf{0.05})$	$4.23\,(\pm 0.18)$	$2.05\,(\pm 0.09)$
				MSE	$\mathbf{1.21}\,(\pm\mathbf{0.06})$	$4.64\,(\pm 0.21)$	$2.09\,(\pm 0.09)$
AND	200	500	500	Bias	$\mathbf{-0.04}\,(\pm\mathbf{0.04})$	$0.32\,(\pm 0.04)$	$0.14\,(\pm 0.03)$
				Var	$\mathbf{0.55}\,(\pm\mathbf{0.07})$	$1.71\,(\pm 0.22)$	$0.85\,(\pm 0.11)$
				MSE	$\mathbf{0.57}\,(\pm\mathbf{0.08})$	$1.82\,(\pm 0.24)$	$0.88\,(\pm 0.11)$
Auto	314	7	1000	Bias	$-0.11\,(\pm 0.02)$	$0.23\,(\pm 0.05)$	$\mathbf{0.06}\,(\pm\mathbf{0.03})$
				Var	$\mathbf{0.13}\,(\pm\mathbf{0.04})$	$0.49\,(\pm 0.19)$	$0.27\,(\pm 0.1)$
				MSE	$\mathbf{0.15}\,(\pm\mathbf{0.04})$	$0.58\,(\pm 0.24)$	$0.29\,(\pm 0.11)$

Table 1: Simulation study. We evaluate the mean bias, variance, and MSE of different variance estimates $\widehat{V}$ for random forests. Here, n is the number of test examples used, p is the number of features, and B is the number of trees grown; the numbers in parentheses are 95% confidence errors from sampling. The best methods for each evaluation metric are highlighted in bold. The data-generating functions are described in Appendix C.

experiments, including some random forest examples, and showed how they can be used to gain valuable insights in realistic problems.

Acknowledgments

The authors are grateful for helpful suggestions from the action editor and three anonymous referees. S.W. is supported by a B.C. and E.J. Eaves Stanford Graduate Fellowship.

Appendix A. The Effect of Monte Carlo Noise on the Jackknife Estimator

In this section, we derive expressions for the finite-B Monte Carlo bias and variance of the jackknife-after-bootstrap estimate of variance. Recall from (6) that

$$\widehat{V}_J^B = \frac{n-1}{n}\sum_{i=1}^{n}\hat{\Delta}_i^2, \text{ where } \hat{\Delta}_i = \frac{\sum_{\{b\,:\,N_{bi}^*=0\}} t_b^*}{\left|\{N_{bi}^*=0\}\right|} - \frac{\sum_b t_b^*}{B}$$

and N_{bi}^* indicates the number of times the i^{th} observation appears in the bootstrap sample b. If $\hat{\Delta}_i$ is not defined because $N_{bi}^* = 0$ for either all or none of the $b = 1, ..., B$, then just set $\hat{\Delta}_i = 0$.

Now $\widehat{V}_J^B$ is the sum of squares of noisy quantities, and so $\widehat{V}_J^B$ will be biased upwards. Specifically,

$$\mathbb{E}_*\left[\widehat{V}_J^B\right] - \widehat{V}_J^\infty = \frac{n-1}{n}\sum_{i=1}^{n}\mathrm{Var}_*\left[\hat{\Delta}_i\right],$$

where $\widehat{V}_J^\infty$ is the jackknife estimate computed with $B = \infty$ bootstrap replicates. For convenience, let

$$B_i = \left|\{b : N_{bi} = 0\}\right|,$$

and recall that

$$\mathrm{Var}_*\left[\hat{\Delta}_i\right] = \mathbb{E}_*\left[\mathrm{Var}_*\left[\hat{\Delta}_i|B_i\right]\right] + \mathrm{Var}_*\left[\mathbb{E}_*\left[\hat{\Delta}_i|B_i\right]\right].$$

For all $B_i \neq 0$ or B, the conditional expectation is

$$\mathbb{E}_*[\hat{\Delta}_i|B_i] = \left(1 - \frac{B_i - \mathbb{E}\left[B_i\right]}{B}\right)\Delta_i, \text{ where } \Delta_i = \mathbb{E}_*\left[t_b^*|N_{bi}^*=0\right] - \mathbb{E}_*\left[t_b^*\right];$$

$\mathbb{E}_*[\hat{\Delta}_i|B_i] = 0$ in the degenerate cases with $B_i \in \{0,\, B\}$. Thus,

$$\mathrm{Var}_*\left[\mathbb{E}_*\left[\hat{\Delta}_i|B_i\right]\right] = \mathcal{O}\left(\Delta_i^2/B\right),$$

and so

$$\mathrm{Var}_*\left[\hat{\Delta}_i\right] = \mathbb{E}_*\left[\mathrm{Var}_*\left[\hat{\Delta}_i|B_i\right]\right] + \mathcal{O}\left(\Delta_i^2/B\right).$$

Meanwhile, for $B_i \notin \{0,\, B\}$,

$$\begin{aligned}\mathrm{Var}_*\left[\hat{\Delta}_i|B_i\right] &= \frac{1}{B^2}\left(\left(\frac{B}{B_i}-1\right)^2 B_i\tilde{v}_i^{(0)} + (B-B_i)\tilde{v}_i^{(+)}\right)\\ &= \frac{1}{B}\left(\frac{(B-B_i)^2}{BB_i}\,\tilde{v}_i^{(0)} + \frac{B-B_i}{B}\tilde{v}_i^{(+)}\right)\end{aligned}$$

where

$$\tilde{v}_i^{(0)} = \mathrm{Var}_*\left[t_b^*|N_{bi}^*=0\right] \text{ and } \tilde{v}_i^{(+)} = \mathrm{Var}_*\left[t_b^*|N_{bi}^*\neq 0\right].$$

Thus,

$$\mathrm{Var}_*\left[\hat{\Delta}_i\right] = \frac{1}{B}\left(\mathbb{E}_*\left[\frac{(B-B_i)^2}{BB_i}1_i\right]\tilde{v}_i^{(0)} + \mathbb{E}_*\left[\frac{B-B_i}{B}1_i\right]\tilde{v}_i^{(+)}\right) + \mathcal{O}\left(\Delta_i^2/B\right),$$

where $1_i = 1(\{B_i \notin \{0,\, B\}\})$.

As n and B get large, B_i converges in law to a Gaussian random variable

$$\frac{B_i - Be^{-1}}{\sqrt{B}} \Rightarrow \left(0,\, e^{-1}(1-e^{-1})\right)$$

and the above expressions are uniformly integrable. We can verify that

$$\mathbb{E}_*\left[\frac{(B-B_i)^2}{BB_i}1_i\right] = e - 2 + e^{-1} + \mathcal{O}\left(\frac{1}{B}\right),$$

and

$$\mathbb{E}_*\left[\frac{B-B_i}{B}1_i\right] = \frac{e-1}{e} + \mathcal{O}\left(\left(1-e^{-1}\right)^B\right).$$

Finally, this lets us conclude that

$$\mathbb{E}_*\left[\widehat{V}_J^B\right] - \widehat{V}_J^\infty = \frac{1}{B}\frac{n-1}{n}\sum_{i=1}^n\left(\left(\frac{(e-1)^2}{e}\right)\tilde{v}_i^{(0)} + \left(\frac{e-1}{e}\right)\tilde{v}_i^{(+)}\right) + \mathcal{O}\left(\frac{1}{B} + \frac{n}{B^2}\right),$$

where the error term depends on $\tilde{v}_i^{(0)}$, $\tilde{v}_i^{(+)}$, and $\widehat{V}_J^\infty = (n-1)/n\,\sum_{i=1}^n \Delta_i^2$.

We now address Monte Carlo variance. By the central limit theorem, $\hat{\Delta}_i$ converges to a Gaussian random variable as B gets large. Thus, the asymptotic Monte Carlo variance of $\hat{\Delta}_i^2$ is approximately $2\,\mathrm{Var}_*[\hat{\Delta}_i]^2 + 4\mathbb{E}_*[\hat{\Delta}_i]^2\,\mathrm{Var}_*[\hat{\Delta}_i]$, and so

$$\begin{aligned}\mathrm{Var}_*\left[\widehat{V}_J^B\right] \approx\; & 2\left(\frac{1}{B}\frac{n-1}{n}\right)^2\sum_{i=1}^n\left(\left(\frac{(e-1)^2}{e}\right)\tilde{v}_i^{(0)} + \left(\frac{e-1}{e}\right)\tilde{v}_i^{(+)}\right)^2 \\ & + 4\,\frac{1}{B}\frac{n-1}{n}\sum_{i=1}^n\Delta_i^2\left(\left(\frac{(e-1)^2}{e}\right)\tilde{v}_i^{(0)} + \left(\frac{e-1}{e}\right)\tilde{v}_i^{(+)}\right).\end{aligned}$$

In practice, the terms $\tilde{v}_i^{(0)}$ and $\tilde{v}_i^{(+)}$ can be well approximated by $\hat{v} = \mathrm{Var}_*[t_b^*]$, namely the bootstrap estimate of variance for the base learner. (Note that $\tilde{v}_i^{(0)}$, $\tilde{v}_i^{(+)}$, and $\hat{v}$ can always be inspected on a random forest, so this assumption can be checked in applications.) This lets us considerably simplify our expressions for Monte Carlo bias and variance:

$$\begin{aligned}&\mathbb{E}_*\left[\widehat{V}_J^B\right] - \widehat{V}_J^\infty \approx \frac{n}{B}\,(e-1)\hat{v},\ \text{and} \\ &\mathrm{Var}_*\left[\widehat{V}_J^B\right] \approx 2\,\frac{n}{B^2}\,(e-1)^2\hat{v}^2 + 4\,\frac{1}{B}\,(e-1)\,\widehat{V}_J^\infty\,\hat{v}.\end{aligned}$$

Appendix B. The IJ estimator and Hájek projections

Up to (27), the derivation below is an alternate presentation of the argument made by Efron (2013) in the proof of his Theorem 1. To establish a connection between the IJ estimate of variance for bagged estimators and the theory of Hájek projections, it is useful to consider $\hat{\theta}^B$ as a functional over distributions. Let G be a probability distribution, and let T be a functional over distributions with the following property:

$$T(G) = \mathbb{E}_G[\tau(Y_1, ..., Y_n)] \text{ for some function } \tau, \tag{26}$$

where the $Y_1, ..., Y_n$ are drawn independently from G. We call functionals T satisfying (26) *averaging*. Clearly, $\hat{\theta}^B$ can be expressed as an averaging functional applied to the empirical distribution $\widehat{F}$ defined by the observations $Z_1, ..., Z_n$.

Suppose that we have an averaging functional T, a sample $Z_1, ..., Z_n$ forming an empirical distribution $\widehat{F}$, and want to study the variance of $T(\widehat{F})$. The infinitesimal jackknife estimate for the variance of $\widehat{T}$ is given by

$$\widehat{V} = \sum_{i=1}^{n} \left(\frac{1}{n} \frac{\partial}{\partial \varepsilon} T\left(\widehat{F}_i(\varepsilon) \right) \right)^2,$$

where $\widehat{F}_i(\varepsilon)$ is the discrete distribution that places weight $1/n + (n-1)/n \cdot \varepsilon$ at Z_i and weight $1/n - \varepsilon/n$ at all the other Z_j.

We can transform samples from $\widehat{F}$ into samples from $\widehat{F}_i(\varepsilon)$ by the following method. Let $Z_1^*, ..., Z_n^*$ be a sample from $\widehat{F}$. Go through the whole sample and, independently for each j, take Z_j^* and with probability ε replace it with Z_i. The sample can now be considered a sample from $\widehat{F}_i(\varepsilon)$.

When $\varepsilon \to 0$, the probability of replacing two of the Z_i^* with this procedure becomes negligible, and we can equivalently transform our sample into a sample from $\widehat{F}_i(\varepsilon)$ by transforming a single random element from $\{Z_j^*\}$ into Z_i with probability $n\,\varepsilon$. Without loss of generality this element is the first one, and so we conclude that

$$\begin{aligned} &\lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \left(\mathbb{E}_{\widehat{F}_{i(\varepsilon)}} [\tau(Z_1^*, ..., Z_n^*)] - \mathbb{E}_{\widehat{F}} [\tau(Z_1^*, ..., Z_n^*)] \right) \\ &\qquad = n \left(\mathbb{E}_{\widehat{F}} \left[\tau(Z_1^*, ..., Z_n^*) \middle| Z_1^* = Z_i \right] - \mathbb{E}_{\widehat{F}} [\tau(Z_1^*, ..., Z_n^*)] \right), \end{aligned}$$

where τ defines T through (26). Thus,

$$\frac{1}{n} \frac{\partial}{\partial \varepsilon} T(\widehat{F}_i(\varepsilon)) = \mathbb{E}_{\widehat{F}} \left[T \middle| Z_1^* = Z_i \right] - \mathbb{E}_{\widehat{F}} [T],$$

and so

$$\widehat{V} = \sum_{i=1}^{n} \left(\mathbb{E}_{\widehat{F}} \left[T \middle| Z_1^* = Z_i \right] - \mathbb{E}_{\widehat{F}} [T] \right)^2 \tag{27}$$

$$\approx \sum_{i=1}^{n} \left(\mathbb{E}_{F} \left[T \middle| Z_1^* = Z_i \right] - \mathbb{E}_{F} [T] \right)^2, \tag{28}$$

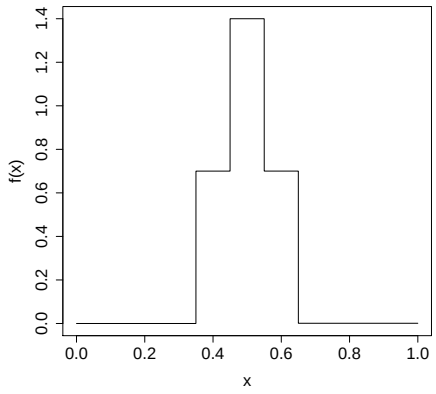


Figure 10: Underlying model for the bagged tree example from Figure 2.

where on the last line we only replaced the empirical approximation $\widehat{F}$ with its true value F. In the case of bagging, this last expression is equivalent to (21).

A crucial step in the above argument is the plug-in approximation (28). If T is just a sum, then the error of (28) is within $\mathcal{O}(1/n)$; presumably, similar statements hold whenever T is sufficiently well-behaved. That being said, it is possible to construct counter-examples where (28) fails; a simple such example is when T counts the number of times Z_1^* is matched in the rest of the training data. Establishing general conditions under which (28) holds is an interesting topic for further research.

Appendix C. Description of Experiments

This section provides a more detailed description of the experiments reported in this paper.

C.1 Auto MPG Example (Figure 1)

The Auto MPG data set, available from the UCI Machine Learning Repository (Bache and Lichman, 2013), is a regression task with 7 features. After discarding examples with missing entries, the data set had 392 rows, which we divided into a test set of size 78 and a train set of size 314. We estimated the variance of the random forest predictions using the $(\widehat{V}^B_{J-U} + \widehat{V}^B_{IJ-U})/2$ estimator advocated in Section 5.2, with $B = 10,000$ bootstrap replicates.

C.2 Bagged Tree Simulation (Figure 2)

The data for this simulation was drawn from a model $y_i = f(x_i) + \varepsilon_i$, where $x_i \sim U([0,1])$, $\varepsilon_i \sim \mathcal{N}(0, 1/2^2)$, and $f(x)$ is the step function shown in Figure 10. We modeled the data using 5-leaf regression trees generated using the `R` package `tree` (Venables and Ripley, 2002); for bagging, we used $B = 10,000$ bootstrap replicates. The reported data is compiled over $1,000$ simulation runs with $n = 500$ data points each.

C.3 Cholesterol Example (Figures 3, 7, and 8)

For the cholesterol data set (Efron and Feldman, 1991), we closely follow the methodology of Efron (2013); see his paper for details. The data set has $n = 164$ subjects and only one predictor.

C.4 E-mail Spam Example (Figures 4 and 5)

The e-mail spam data set (spambase, Bache and Lichman, 2013) is a classification problem with $n = 4,601$ e-mails and $p = 57$ features; the goal is to discern spam from non-spam. We divided the data into train and test sets of size 3,065 and 1,536 respectively. Each of the random forests described in Section 3.1 was fit on the train set using the `R` package `randomForest` (Liaw and Wiener, 2002) with $B = 40,000$ bootstrap replicates.

C.5 California Housing Example (Figure 6)

The California housing data set (described in Hastie et al., 2009, and available from StatLib) contains aggregated data from $n = 20,460$ neighborhoods. There are $p = 8$ features; the response is the median house value. We fit random forests on this data using the `R` package `randomForest` (Liaw and Wiener, 2002) with $B = 1,000$ bootstrap replicates.

C.6 Bagged Tree Simulation #2 (Figure 9a)

We drew $n = 25$ points from a model where the x_i are uniformly distributed over a square, i.e., $x_i \sim U([-1, 1]^2)$; the y_i are deterministically given by $y_i = 1(\{||x_i||_2 \geq 1\})$. We fit this data using the `R` package `tree` (Venables and Ripley, 2002). The bagged predictors were generated using $B = 1,000$ bootstrap replicates. The reported results are based on 2,000 simulation runs.

C.7 Prostate Cancer Example (Figure 9b)

The prostate cancer data (published by Stamey et al., 1989) is described in Section 1 of Hastie et al. (2009). We used forward stepwise regression as implemented by the `R` function `step` as our base learner. This data set has $n = 97$ subjects and 8 available predictor variables. In figure 9b, we display standard errors for the predicted response of a patient whose features match those of patient #41 in the data set.

C.8 Simulations for Table 1

The data generation functions used in Table 1 are defined as follows. The X_i for $i = 1, ..., p$ are all generated as independent $U([0, 1])$ random variables, and $\varepsilon \sim \mathcal{N}(0, 1)$.

- Cosine: $Y = 3 \cdot \cos(\pi \cdot (X_1 + X_2))$, with $p = 2$.
- XOR: Treating `XOR` as a function with a 0/1 return-value,

$$Y = 5 \cdot [\texttt{XOR}(X_1 > 0.6,\, X_2 > 0.6) + \texttt{XOR}(X_3 > 0.6,\, X_4 > 0.6)] + \varepsilon$$

  and $p = 50$.

- AND: With analogous notation,

$$Y = 10 \cdot \texttt{AND}\left(X_1 > 0.3,\, X_2 > 0.3,\, X_3 > 0.3,\, X_4 > 0.3\right) + \varepsilon$$

  and $p = 500$.

- Auto: This example is based on a parametric bootstrap built on the same data set as used in Figure 1. We first fit a random forest to the training set, and evaluated the MSE $\hat{\sigma}^2$ on the test set. We then generated new training sets by replacing the labels Y_i from the original training set with $\widehat{Y}_i + \hat{\sigma}\varepsilon$, where $\widehat{Y}_i$ is the original random forest prediction at the i^{th} training example and ε is fresh residual noise.

During the simulation, we first generated a random test set of size 50 (except for the auto example, where we just used the original test set of size 78). Then, while keeping the test set fixed, we generated 100 training sets and produced variance estimates $\widehat{V}$ at each test point. Table 1 reports average performance over the test set.

References

Kevin Bache and Moshe Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Gérard Biau. Analysis of a random forests model. *The Journal of Machine Learning Research*, 13(4):1063–1095, 2012.

Gérard Biau, Luc Devroye, and Gábor Lugosi. Consistency of random forests and other averaging classifiers. *The Journal of Machine Learning Research*, 9:2015–2033, 2008.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Peter Bühlmann and Bin Yu. Analyzing bagging. *The Annals of Statistics*, 30(4):927–961, 2002.

Andreas Buja and Werner Stuetzle. Observations on bagging. *Statistica Sinica*, 16(2):323, 2006.

Song Xi Chen and Peter Hall. Effects of bagging and bias correction on estimators defined by estimating equations. *Statistica Sinica*, 13(1):97–110, 2003.

Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40 (2):139–157, 2000.

Jiangtao Duan. *Bootstrap-Based Variance Estimators for a Bagging Predictor.* PhD thesis, North Carolina State University, 2011.

Bradley Efron. Jackknife-after-bootstrap standard errors and influence functions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 83–127, 1992.

Bradley Efron. Estimation and accuracy after model selection. *Journal of the American Statistical Association*, (just-accepted), 2013.

Bradley Efron and David Feldman. Compliance as an explanatory variable in clinical trials. *Journal of the American Statistical Association*, 86(413):9–17, 1991.

Bradley Efron and Charles Stein. The jackknife estimate of variance. *The Annals of Statistics*, pages 586–596, 1981.

Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

Jerome H Friedman and Peter Hall. On bagging and nonlinear estimation. *Journal of Statistical Planning and Inference*, 137(3):669–683, 2007.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. New York: Springer, 2009.

Louis A Jaeckel. *The Infinitesimal Jackknife*. 1972.

Andy Liaw and Matthew Wiener. Classification and regression by randomForest. *R News*, 2(3):18–22, 2002. URL `http://CRAN.R-project.org/doc/Rnews/`.

Yi Lin and Yongho Jeon. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590, 2006.

Colin L Mallows. Some comments on Cp. *Technometrics*, 15(4):661–675, 1973.

Nicolai Meinshausen. Quantile regression forests. *The Journal of Machine Learning Research*, 7:983–999, 2006.

Joseph Sexton and Petter Laake. Standard errors for bagged and random forest estimators. *Computational Statistics & Data Analysis*, 53(3):801–811, 2009.

Marina Skurichina and Robert PW Duin. Bagging for linear classifiers. *Pattern Recognition*, 31(7):909–930, 1998.

Thomas A Stamey, John N Kabalin, John E McNeal, Iain M Johnstone, Fuad Freiha, EA Redwine, and N Yang. Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. II. radical prostatectomy treated patients. *The Journal of Urology*, 141(5):1076–1083, 1989.

Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8(1):25, 2007.

William N Venables and Brian D Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL `http://www.stats.ox.ac.uk/pub/MASS4`. ISBN 0-387-95457-0.

Journal of Machine Learning Research 15 (2014) 1653-1674 Submitted 7/12; Revised 12/13; Published 5/14

Surrogate Regret Bounds for Bipartite Ranking via Strongly Proper Losses

Shivani Agarwal SHIVANI@CSA.IISC.ERNET.IN
Department of Computer Science and Automation
Indian Institute of Science
Bangalore 560012, India

Editor: Nicolas Vayatis

Abstract

The problem of bipartite ranking, where instances are labeled positive or negative and the goal is to learn a scoring function that minimizes the probability of mis-ranking a pair of positive and negative instances (or equivalently, that maximizes the area under the ROC curve), has been widely studied in recent years. A dominant theoretical and algorithmic framework for the problem has been to reduce bipartite ranking to pairwise classification; in particular, it is well known that the bipartite ranking regret can be formulated as a pairwise classification regret, which in turn can be upper bounded using usual regret bounds for classification problems. Recently, Kotlowski et al. (2011) showed regret bounds for bipartite ranking in terms of the regret associated with balanced versions of the standard (non-pairwise) logistic and exponential losses. In this paper, we show that such (non-pairwise) surrogate regret bounds for bipartite ranking can be obtained in terms of a broad class of proper (composite) losses that we term as *strongly proper*. Our proof technique is much simpler than that of Kotlowski et al. (2011), and relies on properties of proper (composite) losses as elucidated recently by Reid and Williamson (2010, 2011) and others. Our result yields explicit surrogate bounds (with no hidden balancing terms) in terms of a variety of strongly proper losses, including for example logistic, exponential, squared and squared hinge losses as special cases. An important consequence is that standard algorithms minimizing a (non-pairwise) strongly proper loss, such as logistic regression and boosting algorithms (assuming a universal function class and appropriate regularization), are in fact consistent for bipartite ranking; moreover, our results allow us to quantify the bipartite ranking regret in terms of the corresponding surrogate regret. We also obtain tighter surrogate bounds under certain low-noise conditions via a recent result of Clémençon and Robbiano (2011).

Keywords: bipartite ranking, area under ROC curve (AUC), statistical consistency, regret bounds, proper losses, strongly proper losses

1. Introduction

Ranking problems arise in a variety of applications ranging from information retrieval to recommendation systems and from computational biology to drug discovery, and have been widely studied in machine learning and statistics in the last several years. Recently, there has been much interest in understanding statistical consistency and regret behavior of algorithms for a variety of ranking problems, including various forms of label/subset ranking as well as instance ranking problems (Clémençon and Vayatis, 2007; Clémençon et al., 2008;

Cossock and Zhang, 2008; Balcan et al., 2008; Ailon and Mohri, 2008; Xia et al., 2008; Duchi et al., 2010; Ravikumar et al., 2011; Buffoni et al., 2011; Clémençon and Robbiano, 2011; Kotlowski et al., 2011; Uematsu and Lee, 2011; Calauzènes et al., 2012; Lan et al., 2012; Ramaswamy and Agarwal, 2012; Ramaswamy et al., 2013).

In this paper, we study regret bounds for the bipartite instance ranking problem, where instances are labeled positive or negative and the goal is to learn a scoring function that minimizes the probability of mis-ranking a pair of positive and negative instances, or equivalently, that maximizes the area under the ROC curve (Freund et al., 2003; Agarwal et al., 2005). A popular algorithmic and theoretical approach to bipartite ranking has been to treat the problem as analogous to pairwise classification (Herbrich et al., 2000; Joachims, 2002; Freund et al., 2003; Rakotomamonjy, 2004; Burges et al., 2005; Clémençon et al., 2008). Indeed, this approach enjoys theoretical support since the bipartite ranking regret can be formulated as a pairwise classification regret, and therefore any algorithm minimizing the latter over a suitable class of functions will also minimize the ranking regret (this follows formally from results of Clémençon et al., 2008; see Section 3.1 for a summary). Nevertheless, it has often been observed that algorithms such as AdaBoost, logistic regression, and in some cases even SVMs, which minimize the exponential, logistic, and hinge losses respectively in the standard (non-pairwise) setting, also yield good bipartite ranking performance (Cortes and Mohri, 2004; Rakotomamonjy, 2004; Rudin and Schapire, 2009). For losses such as the exponential or logistic losses, this is not surprising since algorithms minimizing these losses (but not the hinge loss) are known to effectively estimate conditional class probabilities (Zhang, 2004); since the class probability function provides the optimal ranking (Clémençon et al., 2008), it is intuitively clear (and follows formally from results in Clémençon et al., 2008; Clémençon and Robbiano, 2011) that any algorithm providing a good approximation to the class probability function should also produce a good ranking. However, there has been very little work so far on quantifying the ranking regret of a scoring function in terms of the regret associated with such surrogate losses.

Recently, Kotlowski et al. (2011) showed that the bipartite ranking regret of a scoring function can be upper bounded in terms of the regret associated with *balanced* versions of the standard (non-pairwise) exponential and logistic losses. However their proof technique builds on analyses involving the reduction of bipartite ranking to pairwise classification, and involves analyses specific to the exponential and logistic losses (see Section 3.2). More fundamentally, the balanced losses in their result depend on the underlying distribution and cannot be optimized directly by an algorithm; while it is possible to do so approximately, one then loses the quantitative nature of the bounds.

In this work we obtain quantitative regret bounds for bipartite ranking in terms of a broad class of proper (composite) loss functions that we term *strongly proper*. Our proof technique is considerably simpler than that of Kotlowski et al. (2011), and relies on properties of proper (composite) losses as elucidated recently for example in Reid and Williamson (2010, 2011); Gneiting and Raftery (2007); Buja et al. (2005). Our result yields explicit surrogate bounds (with no hidden balancing terms) in terms of a variety of strongly proper (composite) losses, including for example logistic, exponential, squared and squared hinge losses as special cases. An immediate consequence is that standard algorithms minimizing such losses, such as standard logistic regression and boosting algorithms (assuming a universal function class and appropriate regularization), are in fact consistent for bipartite

ranking. We also obtain tighter surrogate bounds under certain low-noise conditions via a recent result of Clémençon and Robbiano (2011).

The paper is organized as follows. In Section 2 we formally set up the bipartite instance ranking problem and definitions related to loss functions and regret, and provide background on proper (composite) losses. Section 3 summarizes related work that provides the background for our study, namely the reduction of bipartite ranking to pairwise binary classification and the result of Kotlowski et al. (2011). In Section 4 we define and characterize strongly proper losses. Section 5 contains our main result, namely a bound on the bipartite ranking regret in terms of the regret associated with any strongly proper loss, together with several examples. Section 6 gives a tighter bound under certain low-noise conditions via a recent result of Clémençon and Robbiano (2011). We conclude with a brief discussion and some open questions in Section 7.

2. Formal Setup, Preliminaries, and Background

This section provides background on the bipartite ranking problem, binary loss functions and regret, and proper (composite) losses.

2.1 Bipartite Ranking

As in binary classification, in bipartite ranking there is an instance space $\mathcal{X}$ and binary labels $\mathcal{Y} = \{\pm 1\}$, with an unknown distribution D on $\mathcal{X} \times \{\pm 1\}$. For $(X, Y) \sim D$ and $x \in \mathcal{X}$, we denote $\eta(x) = \mathbf{P}(Y = 1 \mid X = x)$ and $p = \mathbf{P}(Y = 1)$. Given i.i.d. examples $(X_1, Y_1), \ldots, (X_n, Y_n) \sim D$, the goal is to learn a scoring function $f : \mathcal{X} \to \bar{\mathbb{R}}$ (where $\bar{\mathbb{R}} = [-\infty, \infty]$) that assigns higher scores to positive instances than to negative ones.[1] Specifically, the goal is to learn a scoring function f with low *ranking error* (or *ranking risk*), defined as[2]

$$\mathrm{er}_D^{\mathrm{rank}}[f] \;=\; \mathbf{E}\Big[\mathbf{1}\big((Y - Y')(f(X) - f(X')) < 0\big) + \tfrac{1}{2}\,\mathbf{1}\big(f(X) = f(X')\big) \;\Big|\; Y \neq Y'\Big], \quad (1)$$

where $(X, Y), (X', Y')$ are assumed to be drawn i.i.d. from D, and $\mathbf{1}(\cdot)$ is 1 if its argument is true and 0 otherwise; thus the ranking error of f is simply the probability that a randomly drawn positive instance receives a lower score under f than a randomly drawn negative instance, with ties broken uniformly at random. The *optimal ranking error* (or *Bayes ranking error* or *Bayes ranking risk*) can be seen to be

$$\begin{aligned} \mathrm{er}_D^{\mathrm{rank},*} &= \inf_{f:\mathcal{X}\to\bar{\mathbb{R}}} \mathrm{er}_D^{\mathrm{rank}}[f] & (2)\\ &= \frac{1}{2p(1-p)} \mathbf{E}_{X,X'}\Big[\min\Big(\eta(X)(1-\eta(X')),\, \eta(X')(1-\eta(X))\Big)\Big]. & (3) \end{aligned}$$

The *ranking regret* of a scoring function $f : \mathcal{X} \to \bar{\mathbb{R}}$ is then simply

$$\mathrm{regret}_D^{\mathrm{rank}}[f] \;=\; \mathrm{er}_D^{\mathrm{rank}}[f] - \mathrm{er}_D^{\mathrm{rank},*}. \quad (4)$$

1. Most algorithms learn real-valued functions; we also allow values $-\infty$ and ∞ for technical reasons.
2. We assume measurability conditions where necessary.

We will be interested in upper bounding the ranking regret of a scoring function f in terms of its regret with respect to certain other (binary) loss functions. In particular, the loss functions we consider will belong to the class of *proper* (composite) loss functions. Below we briefly review some standard notions related to loss functions and regret, and then discuss some properties of proper (composite) losses.

2.2 Loss Functions, Regret, and Conditional Risks and Regret

Assume again a probability distribution D on $\mathcal{X} \times \{\pm 1\}$ as above. Given a prediction space $\widehat{\mathcal{Y}} \subseteq \bar{\mathbb{R}}$, a binary *loss function* $\ell : \{\pm 1\} \times \widehat{\mathcal{Y}} \to \bar{\mathbb{R}}_+$ (where $\bar{\mathbb{R}}_+ = [0, \infty]$) assigns a penalty $\ell(y, \widehat{y})$ for predicting $\widehat{y} \in \widehat{\mathcal{Y}}$ when the true label is $y \in \{\pm 1\}$.[3] For any such loss ℓ, the *ℓ-error* (or *ℓ-risk*) of a function $f : \mathcal{X} \to \widehat{\mathcal{Y}}$ is defined as

$$\mathrm{er}_D^{\ell}[f] = \mathbf{E}_{(X,Y)\sim D}[\ell(Y, f(X))]\,, \tag{5}$$

and the *optimal ℓ-error* (or *optimal ℓ-risk* or *Bayes ℓ-risk*) is defined as

$$\mathrm{er}_D^{\ell,*} = \inf_{f:\mathcal{X}\to\widehat{\mathcal{Y}}} \mathrm{er}_D^{\ell}[f]\,. \tag{6}$$

The *ℓ-regret* of a function $f : \mathcal{X} \to \widehat{\mathcal{Y}}$ is the difference of its ℓ-error from the optimal ℓ-error:

$$\mathrm{regret}_D^{\ell}[f] = \mathrm{er}_D^{\ell}[f] - \mathrm{er}_D^{\ell,*}\,. \tag{7}$$

The *conditional ℓ-risk* $L_\ell : [0,1] \times \widehat{\mathcal{Y}} \to \bar{\mathbb{R}}_+$ is defined as[4]

$$L_\ell(\eta, \widehat{y}) = \mathbf{E}_{Y\sim\eta}[\ell(Y, \widehat{y})] = \eta\,\ell(1, \widehat{y}) + (1-\eta)\,\ell(-1, \widehat{y})\,, \tag{8}$$

where $Y \sim \eta$ denotes a $\{\pm 1\}$-valued random variable taking value $+1$ with probability η. The *conditional Bayes ℓ-risk* $H_\ell : [0,1] \to \bar{\mathbb{R}}_+$ is defined as

$$H_\ell(\eta) = \inf_{\widehat{y}\in\widehat{\mathcal{Y}}} L_\ell(\eta, \widehat{y})\,. \tag{9}$$

The *conditional ℓ-regret* $R_\ell : [0,1] \times \widehat{\mathcal{Y}} \to \bar{\mathbb{R}}_+$ is then simply

$$R_\ell(\eta, \widehat{y}) = L_\ell(\eta, \widehat{y}) - H_\ell(\eta)\,. \tag{10}$$

Clearly, we have for $f : \mathcal{X} \to \widehat{\mathcal{Y}}$,

$$\mathrm{er}_D^{\ell}[f] = \mathbf{E}_X[L_\ell(\eta(X), f(X))]\,, \tag{11}$$

and

$$\mathrm{er}_D^{\ell,*} = \mathbf{E}_X[H_\ell(\eta(X))]\,. \tag{12}$$

We note the following:

Lemma 1 *For any $\widehat{\mathcal{Y}} \subseteq \bar{\mathbb{R}}$ and binary loss $\ell : \{\pm 1\} \times \widehat{\mathcal{Y}} \to \bar{\mathbb{R}}_+$, the conditional Bayes ℓ-risk H_ℓ is a concave function on $[0,1]$.*

The proof follows simply by observing that H_ℓ is defined as the pointwise infimum of a family of linear (and therefore concave) functions, and therefore is itself concave.

3. Most loss functions take values in $\mathbb{R}_+$, but some loss functions (such as the logistic loss, described later) can assign a loss of ∞ to certain label-prediction pairs.
4. Note that we overload notation by using η here to refer to a number in $[0,1]$; the usage should be clear from context.

2.3 Proper and Proper Composite Losses

In this section we review some background material related to proper and proper composite losses, as studied recently in Reid and Williamson (2010, 2011); Gneiting and Raftery (2007); Buja et al. (2005). While the material is meant to be mostly a review, some of the exposition is simplified compared to previous presentations, and we include a new, simple proof of an important fact (Theorem 4).

2.3.1 Proper Losses

We start by considering binary class probability estimation (CPE) loss functions that operate on the prediction space $\widehat{\mathcal{Y}} = [0,1]$. A binary CPE loss function $c : \{\pm 1\} \times [0,1] \to \mathbb{R}_+$ is said to be *proper* if for all $\eta \in [0,1]$,

$$\eta \in \operatorname*{arg\,min}_{\widehat{\eta} \in [0,1]} L_c(\eta, \widehat{\eta}) \,, \tag{13}$$

and *strictly proper* if the minimizer is unique for all $\eta \in [0,1]$. Equivalently, c is proper if for all $\eta \in [0,1]$, $H_c(\eta) = L_c(\eta, \eta)$, and strictly proper if $H_c(\eta) < L_c(\eta, \widehat{\eta})$ for all $\widehat{\eta} \neq \eta$. We have the following basic result:

Lemma 2 (Gneiting and Raftery (2007); Schervish (1989)) *Let $c : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$ be a binary CPE loss. If c is proper, then $c(1,\cdot)$ is a decreasing function on $[0,1]$ and $c(-1,\cdot)$ is an increasing function. If c is strictly proper, then $c(1,\cdot)$ is strictly decreasing on $[0,1]$ and $c(-1,\cdot)$ is strictly increasing.*

We will find it useful to consider *regular* proper losses. As in Gneiting and Raftery (2007), we say a binary CPE loss $c : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$ is *regular* if $c(1,\widehat{\eta}) \in \mathbb{R}_+ \; \forall \widehat{\eta} \in (0,1]$ and $c(-1,\widehat{\eta}) \in \mathbb{R}_+ \; \forall \widehat{\eta} \in [0,1)$, i.e., if $c(y,\widehat{\eta})$ is finite for all $y, \widehat{\eta}$ except possibly for $c(1,0)$ and $c(-1,1)$, which are allowed to be infinite. The following characterization of regular proper losses is well known (see also Gneiting and Raftery, 2007):

Theorem 3 (Savage (1971)) *A regular binary CPE loss $c : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$ is proper if and only if for all $\eta, \widehat{\eta} \in [0,1]$ there exists a superderivative $H_c'(\widehat{\eta})$ of H_c at $\widehat{\eta}$ such that*[5]

$$L_c(\eta, \widehat{\eta}) \;=\; H_c(\widehat{\eta}) + (\eta - \widehat{\eta}) \cdot H_c'(\widehat{\eta}) \,.$$

The following is a characterization of strict properness of a proper loss c in terms of its conditional Bayes risk H_c:

Theorem 4 *A proper loss $c : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$ is strictly proper if and only if H_c is strictly concave.*

This result can be proved in several ways. A proof in Gneiting and Raftery (2007) is attributed to an argument in Hendrickson and Buehler (1971). If H_c is twice differentiable, an alternative proof follows from a result in Buja et al. (2005); Schervish (1989), which shows that a proper loss c is strictly proper if and only if its 'weight function' $w_c = -H_c''$ satisfies $w_c(\eta) > 0$ for all except at most countably many points $\eta \in [0,1]$; by a very recent

5. Here $u \in \mathbb{R}$ is a superderivative of H_c at $\widehat{\eta}$ if for all $\eta \in [0,1]$, $H_c(\widehat{\eta}) - H_c(\eta) \geq u(\widehat{\eta} - \eta)$.

result of Stein (2012), this condition is equivalent to strict convexity of the function $-H_c$, or equivalently, strict concavity of H_c. Here we give a third, self-contained proof of the above result that is derived from first principles, and that will be helpful when we study strongly proper losses in Section 4.

Proof [of Theorem 4] Let $c : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$ be a proper loss. For the 'if' direction, assume H_c is strictly concave. Let $\eta, \widehat{\eta} \in [0,1]$ such that $\widehat{\eta} \neq \eta$. Then we have

$$\begin{aligned}
L_c(\eta, \widehat{\eta}) - H_c(\eta) &= L_c(\eta, \widehat{\eta}) + H_c(\widehat{\eta}) - H_c(\widehat{\eta}) - H_c(\eta) \\
&= L_c(\eta, \widehat{\eta}) + H_c(\widehat{\eta}) - 2\Big(\tfrac{1}{2}H_c(\eta) + \tfrac{1}{2}H_c(\widehat{\eta})\Big) \\
&> L_c(\eta, \widehat{\eta}) + H_c(\widehat{\eta}) - 2H_c\Big(\frac{\eta + \widehat{\eta}}{2}\Big) \\
&= 2\left(\Big(\frac{\eta + \widehat{\eta}}{2}\Big)c(1, \widehat{\eta}) + \Big(1 - \frac{\eta + \widehat{\eta}}{2}\Big)c(-1, \widehat{\eta})\right) - 2H_c\Big(\frac{\eta + \widehat{\eta}}{2}\Big) \\
&= 2\left(L_c\Big(\frac{\eta + \widehat{\eta}}{2}, \widehat{\eta}\Big) - H_c\Big(\frac{\eta + \widehat{\eta}}{2}\Big)\right) \\
&\geq 0\,.
\end{aligned}$$

Thus c is strictly proper.

Conversely, to prove the 'only if' direction, assume c is strictly proper. Let $\eta_1, \eta_2 \in [0,1]$ such that $\eta_1 \neq \eta_2$, and let $t \in (0,1)$. Then we have

$$\begin{aligned}
H_c\big(t\eta_1 + (1-t)\eta_2\big) &= L_c\big(t\eta_1 + (1-t)\eta_2,\, t\eta_1 + (1-t)\eta_2\big) \\
&= t\, L_c\big(\eta_1,\, t\eta_1 + (1-t)\eta_2\big) + (1-t)\, L_c\big(\eta_2,\, t\eta_1 + (1-t)\eta_2\big) \\
&> t\, H_c(\eta_1) + (1-t)\, H_c(\eta_2)\,.
\end{aligned}$$

Thus H_c is strictly concave. ■

2.3.2 Proper Composite Losses

The notion of properness can be extended to binary loss functions operating on prediction spaces $\widehat{\mathcal{Y}}$ other than $[0,1]$ via composition with a *link* function $\psi : [0,1] \to \widehat{\mathcal{Y}}$. Specifically, for any $\widehat{\mathcal{Y}} \subseteq \bar{\mathbb{R}}$, a loss function $\ell : \{\pm 1\} \times \widehat{\mathcal{Y}} \to \mathbb{R}_+$ is said to be *proper composite* if it can be written as

$$\ell(y, \widehat{y}) = c(y, \psi^{-1}(\widehat{y})) \tag{14}$$

for some proper loss $c : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$ and strictly increasing (and therefore invertible) link function $\psi : [0,1] \to \widehat{\mathcal{Y}}$. Proper composite losses have been studied recently in Reid and Williamson (2010, 2011); Buja et al. (2005), and include several widely used losses such as squared, squared hinge, logistic, and exponential losses.

It is worth noting that for a proper composite loss ℓ formed from a proper loss c, $H_\ell = H_c$. Moreover, any property associated with the underlying proper loss c can also be used to describe the composite loss ℓ; thus we will refer to a proper composite loss ℓ formed from a regular proper loss c as *regular proper composite*, a composite loss formed from a strictly proper loss as *strictly proper composite*, etc. In Section 4, we will define and

characterize *strongly proper* (composite) losses, which we will use to obtain regret bounds for bipartite ranking.

3. Related Work

As noted above, a popular theoretical and algorithmic framework for bipartite ranking has been to reduce the problem to pairwise classification. Below we describe this reduction in the context of our setting and notation, and then review the result of Kotlowski et al. (2011) which builds on this pairwise reduction.

3.1 Reduction of Bipartite Ranking to Pairwise Binary Classification

For any distribution D on $\mathcal{X} \times \{\pm 1\}$, consider the distribution $\widetilde{D}$ on $(\mathcal{X} \times \mathcal{X}) \times \{\pm 1\}$ defined as follows:

1. Sample (X, Y) and (X', Y') i.i.d. from D;
2. If $Y = Y'$, then go to step 1; else set[6]

$$\widetilde{X} = (X, X')\,, \quad \widetilde{Y} = \operatorname{sign}(Y - Y')$$

and return $(\widetilde{X}, \widetilde{Y})$.

Then it is easy to see that, under $\widetilde{D}$,

$$\mathbf{P}\big(\widetilde{X} = (x, x')\big) \;=\; \frac{\mathbf{P}(X = x)\,\mathbf{P}(X' = x')\,\big(\eta(x)(1 - \eta(x')) + \eta(x')(1 - \eta(x))\big)}{2p(1-p)} \tag{15}$$

$$\widetilde{\eta}((x, x')) \;=\; \mathbf{P}\big(\widetilde{Y} = 1 \mid \widetilde{X} = (x, x')\big) \;=\; \frac{\eta(x)(1 - \eta(x'))}{\eta(x)(1 - \eta(x')) + \eta(x')(1 - \eta(x))} \tag{16}$$

$$\widetilde{p} \;=\; \mathbf{P}\big(\widetilde{Y} = 1\big) \;=\; \tfrac{1}{2}\,. \tag{17}$$

Moreover, for the 0-1 loss $\ell_{0\text{-}1} : \{\pm 1\} \times \{\pm 1\} \to \{0, 1\}$ given by $\ell_{0\text{-}1}(y, \widehat{y}) = \mathbf{1}(\widehat{y} \neq y)$, we have the following for any pairwise binary classifier $h : \mathcal{X} \times \mathcal{X} \to \{\pm 1\}$:

$$\operatorname{er}_{\widetilde{D}}^{0\text{-}1}[h] \;=\; \mathbf{E}_{(\widetilde{X}, \widetilde{Y}) \sim \widetilde{D}}\Big[\mathbf{1}\big(h(\widetilde{X}) \neq \widetilde{Y}\big)\Big] \tag{18}$$

$$\operatorname{er}_{\widetilde{D}}^{0\text{-}1,*} \;=\; \mathbf{E}_{\widetilde{X}}\Big[\min\Big(\widetilde{\eta}(\widetilde{X}), 1 - \widetilde{\eta}(\widetilde{X})\Big)\Big] \tag{19}$$

$$\operatorname{regret}_{\widetilde{D}}^{0\text{-}1}[h] \;=\; \operatorname{er}_{\widetilde{D}}^{0\text{-}1}[h] - \operatorname{er}_{\widetilde{D}}^{0\text{-}1,*}\,. \tag{20}$$

Now for any scoring function $f : \mathcal{X} \to \bar{\mathbb{R}}$, define $f_{\text{diff}} : \mathcal{X} \times \mathcal{X} \to \bar{\mathbb{R}}$ as

$$f_{\text{diff}}(x, x') \;=\; f(x) - f(x')\,. \tag{21}$$

Then it is easy to see that:

$$\operatorname{er}_{D}^{\text{rank}}[f] \;=\; \operatorname{er}_{\widetilde{D}}^{0\text{-}1}[\operatorname{sign} \circ f_{\text{diff}}] \tag{22}$$

$$\operatorname{er}_{D}^{\text{rank},*} \;=\; \operatorname{er}_{\widetilde{D}}^{0\text{-}1,*}\,, \tag{23}$$

6. Throughout the paper, $\operatorname{sign}(u) = +1$ if $u > 0$ and -1 otherwise.

where $(g \circ f)(u) = g(f(u))$. The equality in Eq. (23) follows from the fact that the classifier $h^*(x, x') = \text{sign}(\eta(x) - \eta(x'))$ achieves the Bayes 0-1 risk, i.e., $\text{er}^{0\text{-}1}_{\widetilde{D}}[h^*] = \text{er}^{0\text{-}1,*}_{\widetilde{D}}$ (Clémençon et al., 2008). Thus

$$\text{regret}^{\text{rank}}_{D}[f] \;\; = \;\; \text{regret}^{0\text{-}1}_{\widetilde{D}}[\text{sign} \circ f_{\text{diff}}]\,, \tag{24}$$

and therefore the ranking regret of a scoring function $f : \mathcal{X} \to \bar{\mathbb{R}}$ can be analyzed via upper bounds on the 0-1 regret of the pairwise classifier $(\text{sign} \circ f_{\text{diff}}) : \mathcal{X} \times \mathcal{X} \to \{\pm 1\}$.[7]

In particular, as noted in Clémençon et al. (2008), applying a result of Bartlett et al. (2006), we can upper bound the pairwise 0-1 regret above in terms of the pairwise ℓ_ϕ-regret associated with any classification-calibrated margin loss $\ell_\phi : \{\pm 1\} \times \bar{\mathbb{R}} \to \bar{\mathbb{R}}_+$, i.e., any loss of the form $\ell_\phi(y, \widehat{y}) = \phi(y\widehat{y})$ for some function $\phi : \bar{\mathbb{R}} \to \bar{\mathbb{R}}_+$ satisfying $\forall\, \eta \in [0, 1], \eta \neq \frac{1}{2}$,[8]

$$\widehat{y}^* \in \arg\min_{\widehat{y} \in \bar{\mathbb{R}}} L_\phi(\eta, \widehat{y}) \;\; \Longrightarrow \;\; \widehat{y}^*(\eta - \tfrac{1}{2}) > 0\,. \tag{25}$$

We note in particular that for every proper composite margin loss, the associated link function ψ satisfies $\psi(\frac{1}{2}) = 0$ (Reid and Williamson, 2010), and therefore every strictly proper composite margin loss is classification-calibrated in the sense above.[9]

Theorem 5 (Bartlett et al. (2006); see also Clémençon et al. (2008)) *Let $\phi : \bar{\mathbb{R}} \to \bar{\mathbb{R}}_+$ be such that the margin loss $\ell_\phi : \{\pm 1\} \times \bar{\mathbb{R}} \to \bar{\mathbb{R}}_+$ defined as $\ell_\phi(y, \widehat{y}) = \phi(y\widehat{y})$ is classification-calibrated as above. Then $\exists$ strictly increasing function $g_\phi : \bar{\mathbb{R}}_+ \to [0, 1]$ with $g_\phi(0) = 0$ such that for any $\widetilde{f} : \mathcal{X} \times \mathcal{X} \to \bar{\mathbb{R}}$,*

$$\text{regret}^{0\text{-}1}_{\widetilde{D}}[\text{sign} \circ \widetilde{f}] \;\leq\; g_\phi\Big(\text{regret}^{\phi}_{\widetilde{D}}[\widetilde{f}]\Big)\,.$$

Bartlett et al. (2006) give a construction for g_ϕ; in particular, for the exponential loss given by $\phi_{\text{exp}}(u) = e^{-u}$ and logistic loss given by $\phi_{\log}(u) = \ln(1 + e^{-u})$, both of which are strictly proper composite losses (see Section 5.2) and are therefore classification-calibrated, one has

$$g_{\text{exp}}(z) \;\leq\; \sqrt{2z} \tag{26}$$

$$g_{\log}(z) \;\leq\; \sqrt{2z}\,. \tag{27}$$

As we describe below, Kotlowski et al. (2011) build on these observations to bound the ranking regret in terms of the regret associated with balanced versions of the exponential and logistic losses.

7. Note that the setting here is somewhat different from that of Balcan et al. (2008) and Ailon and Mohri (2008), who consider a *subset* version of bipartite ranking where each instance consists of some finite subset of objects to be ranked; there also the problem is reduced to a (subset) pairwise classification problem, and it is shown that given any (subset) pairwise classifier h, a subset ranking function f can be constructed such that the resulting subset ranking regret is at most twice the subset pairwise classification regret of h (Balcan et al., 2008), or in expectation at most equal to the pairwise classification regret of h (Ailon and Mohri, 2008).
8. We abbreviate $L_\phi = L_{\ell_\phi}$, $\text{er}^{\phi}_{D} = \text{er}^{\ell_\phi}_{D}$, etc.
9. We note that in general, every strictly proper (composite) loss is classification-calibrated with respect to any cost-sensitive zero-one loss, using a more general definition of classification calibration with an appropriate threshold (e.g., see (Reid and Williamson, 2010)).

3.2 Result of Kotlowski et al. (2011)

For any binary loss $\ell : \{\pm 1\} \times \widehat{\mathcal{Y}} \to \bar{\mathbb{R}}_+$, consider defining a *balanced* loss $\ell_{\text{bal}} : \{\pm 1\} \times \widehat{\mathcal{Y}} \to \bar{\mathbb{R}}_+$ as

$$\ell_{\text{bal}}(y, \widehat{y}) \;=\; \frac{1}{2p}\ell(1, \widehat{y}) \cdot \mathbf{1}(y = 1) + \frac{1}{2(1-p)}\ell(-1, \widehat{y}) \cdot \mathbf{1}(y = -1)\,. \tag{28}$$

Note that such a balanced loss depends on the underlying distribution D via $p = \mathbf{P}(Y = 1)$. Then Kotlowski et al. (2011) show the following, via analyses specific to the exponential and logistic losses:

Theorem 6 (Kotlowski et al. (2011)) *For any $f : \mathcal{X} \to \bar{\mathbb{R}}$,*

$$\begin{aligned} \text{regret}^{\text{exp}}_{\widetilde{D}}[f_{\text{diff}}] &\leq \frac{9}{4}\text{regret}^{\text{exp,bal}}_{D}[f] \\ \text{regret}^{\text{log}}_{\widetilde{D}}[f_{\text{diff}}] &\leq 2\,\text{regret}^{\text{log,bal}}_{D}[f]\,. \end{aligned}$$

Combining this with the results of Eq. (24), Theorem 5, and Eqs. (26-27) then gives the following bounds on the ranking regret of any scoring function $f : \mathcal{X} \to \bar{\mathbb{R}}$ in terms of the (non-pairwise) balanced exponential and logistic regrets of f:

$$\text{regret}^{\text{rank}}_{D}[f] \;\leq\; \frac{3}{\sqrt{2}}\sqrt{\text{regret}^{\text{exp,bal}}_{D}[f]} \tag{29}$$

$$\text{regret}^{\text{rank}}_{D}[f] \;\leq\; 2\sqrt{\text{regret}^{\text{log,bal}}_{D}[f]}\,. \tag{30}$$

This suggests that an algorithm that produces a function $f : \mathcal{X} \to \bar{\mathbb{R}}$ with low balanced exponential or logistic regret will also have low ranking regret. Unfortunately, since the balanced losses depend on the unknown distribution D, they cannot be optimized by an algorithm directly.[10] Kotlowski et al. (2011) provide some justification for why in certain situations, minimizing the usual exponential or logistic loss may also minimize the balanced versions of these losses; however, by doing so, one loses the quantitative nature of the above bounds. Below we obtain upper bounds on the ranking regret of a function f directly in terms of its loss-based regret (with no balancing terms) for a wide range of proper (composite) loss functions that we term *strongly proper*, including the exponential and logistic losses as special cases.

4. Strongly Proper Losses

We define strongly proper losses as follows:

Definition 7 *Let $c : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$ be a binary CPE loss and let $\lambda > 0$. We say c is* λ-strongly proper *if for all $\eta, \widehat{\eta} \in [0,1]$,*

$$L_c(\eta, \widehat{\eta}) - H_c(\eta) \;\geq\; \frac{\lambda}{2}(\eta - \widehat{\eta})^2\,.$$

We have the following necessary and sufficient conditions for strong properness:

10. We note it is possible to optimize approximately balanced losses, e.g., by estimating p from the data.

Lemma 8 *Let $\lambda > 0$. If $c : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$ is λ-strongly proper, then H_c is λ-strongly concave.*

Proof The proof is similar to the 'only if' direction in the proof of Theorem 4. Let c be λ-strongly proper. Let $\eta_1, \eta_2 \in [0,1]$ such that $\eta_1 \neq \eta_2$, and let $t \in (0,1)$. Then we have

$$\begin{aligned}
H_c\big(t\eta_1 + (1-t)\eta_2\big) &= L_c\big(t\eta_1 + (1-t)\eta_2,\, t\eta_1 + (1-t)\eta_2\big) \\
&= t\, L_c\big(\eta_1,\, t\eta_1 + (1-t)\eta_2\big) + (1-t)\, L_c\big(\eta_2,\, t\eta_1 + (1-t)\eta_2\big) \\
&\geq t\left(H_c(\eta_1) + \frac{\lambda}{2}(1-t)^2(\eta_1-\eta_2)^2\right) \\
&\qquad + (1-t)\left(H_c(\eta_2) + \frac{\lambda}{2}t^2(\eta_1-\eta_2)^2\right) \\
&= t\, H_c(\eta_1) + (1-t)\, H_c(\eta_2) + \frac{\lambda}{2}t(1-t)(\eta_1-\eta_2)^2 .
\end{aligned}$$

Thus H_c is λ-strongly concave. ■

Lemma 9 *Let $\lambda > 0$ and let $c : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$ be a regular proper loss. If H_c is λ-strongly concave, then c is λ-strongly proper.*

Proof Let $\eta, \widehat{\eta} \in [0,1]$. By Theorem 3, there exists a superderivative $H_c'(\widehat{\eta})$ of H_c at $\widehat{\eta}$ such that

$$L_c(\eta, \widehat{\eta}) \;=\; H_c(\widehat{\eta}) + (\eta - \widehat{\eta}) \cdot H_c'(\widehat{\eta})\,.$$

This gives

$$\begin{aligned}
L_c(\eta, \widehat{\eta}) - H_c(\eta) &= H_c(\widehat{\eta}) - H_c(\eta) + (\eta - \widehat{\eta}) \cdot H_c'(\widehat{\eta}) \\
&\geq \frac{\lambda}{2}(\widehat{\eta} - \eta)^2\,, \qquad \text{since } H_c \text{ is } \lambda\text{-strongly concave.}
\end{aligned}$$

Thus c is λ-strongly proper. ■

This gives us the following characterization of strong properness for regular proper losses:

Theorem 10 *Let $\lambda > 0$ and let $c : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$ be a regular proper loss. Then c is λ-strongly proper if and only if H_c is λ-strongly concave.*

We note that from Lemma 8, another way to think about strongly proper losses is that the weight function $w(\eta) = -H''(\eta)$, used to express the proper loss as a weighted mixture of cost-sensitive misclassification losses (e.g., Buja et al., 2005; Reid and Williamson, 2010), is bounded below by a positive constant.

Several examples of strongly proper (composite) losses will be provided in Section 5.2 and Section 5.3. Theorem 10 will form our main tool in establishing strong properness of many of these loss functions.

5. Regret Bounds via Strongly Proper Losses

We start by recalling the following result of Clémençon et al. (2008) (adapted to account for ties, and for the conditioning on $Y \neq Y'$):

Theorem 11 (Clémençon et al. (2008)) *For any $f : \mathcal{X} \to \bar{\mathbb{R}}$,*

$$\begin{aligned}\mathrm{regret}_D^{\mathrm{rank}}[f] \;=\; \frac{1}{2p(1-p)}\mathbf{E}_{X,X'}\Big[\big|\eta(X)-\eta(X')\big| \cdot \Big(\mathbf{1}\big((f(X)-f(X'))(\eta(X)-\eta(X')) < 0\big) \\ +\; \tfrac{1}{2}\mathbf{1}\big(f(X)=f(X')\big)\Big)\Big].\end{aligned}$$

As noted by Clémençon and Robbiano (2011), this leads to the following corollary on the regret of any plug-in ranking function based on an estimate $\widehat{\eta}$:

Corollary 12 *For any $\widehat{\eta} : \mathcal{X} \to [0,1]$,*

$$\mathrm{regret}_D^{\mathrm{rank}}\big[\,\widehat{\eta}\,\big] \;\leq\; \frac{1}{p(1-p)}\mathbf{E}_X\big[\big|\widehat{\eta}(X)-\eta(X)\big|\big]\,.$$

For completeness, a proof is given in Appendix A. We now give our main result.

5.1 Main Result

Theorem 13 *Let $\widehat{\mathcal{Y}} \subseteq \bar{\mathbb{R}}$ and let $\lambda > 0$. Let $\ell : \{\pm 1\} \times \widehat{\mathcal{Y}} \to \bar{\mathbb{R}}_+$ be a λ-strongly proper composite loss. Then for any $f : \mathcal{X} \to \widehat{\mathcal{Y}}$,*

$$\mathrm{regret}_D^{\mathrm{rank}}[f] \;\leq\; \frac{\sqrt{2}}{p(1-p)\sqrt{\lambda}}\sqrt{\mathrm{regret}_D^{\ell}[f]}\,.$$

Proof Let $c : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$ be a λ-strongly proper loss and $\psi : [0,1] \to \widehat{\mathcal{Y}}$ be a (strictly increasing) link function such that $\ell(y,\widehat{y}) = c(y,\psi^{-1}(\widehat{y}))$ for all $y \in \{\pm 1\}, \widehat{y} \in \widehat{\mathcal{Y}}$. Let $f : \mathcal{X} \to \widehat{\mathcal{Y}}$. Then we have

$$\begin{aligned}
\mathrm{regret}_D^{\mathrm{rank}}[f] &= \mathrm{regret}_D^{\mathrm{rank}}[\psi^{-1} \circ f]\,, \quad \text{since } \psi \text{ is strictly increasing} \\
&\leq \frac{1}{p(1-p)}\mathbf{E}_X\big[\big|\psi^{-1}(f(X)) - \eta(X)\big|\big]\,, \quad \text{by Corollary 12} \\
&= \frac{1}{p(1-p)}\sqrt{\Big(\mathbf{E}_X\big[\big|\psi^{-1}(f(X)) - \eta(X)\big|\big]\Big)^2} \\
&\leq \frac{1}{p(1-p)}\sqrt{\mathbf{E}_X\Big[\big(\psi^{-1}(f(X)) - \eta(X)\big)^2\Big]}\,, \\
&\qquad\qquad \text{by convexity of } \phi(u) = u^2 \text{ and Jensen's inequality} \\
&\leq \frac{1}{p(1-p)}\sqrt{\frac{2}{\lambda}\,\mathbf{E}_X\big[R_c(\eta(X), \psi^{-1}(f(X)))\big]}\,, \quad \text{since } c \text{ is } \lambda\text{-strongly proper} \\
&= \frac{1}{p(1-p)}\sqrt{\frac{2}{\lambda}\,\mathbf{E}_X\big[R_\ell(\eta(X), f(X))\big]} \\
&= \frac{\sqrt{2}}{p(1-p)\sqrt{\lambda}}\sqrt{\mathrm{regret}_D^{\ell}[f]}.
\end{aligned}$$

■

Theorem 13 shows that for any strongly proper composite loss $\ell : \{\pm 1\} \times \widehat{\mathcal{Y}} \to \bar{\mathbb{R}}_+$, a function $f : \mathcal{X} \to \widehat{\mathcal{Y}}$ with low ℓ-regret will also have low ranking regret. Below we give several examples of such strongly proper (composite) loss functions; properties of some of these losses are summarized in Table 1.

5.2 Examples

Example 1 (Exponential loss) *The exponential loss $\ell_{\exp} : \{\pm 1\} \times \bar{\mathbb{R}} \to \bar{\mathbb{R}}_+$ defined as*

$$\ell_{\exp}(y, \widehat{y}) \;=\; e^{-y\widehat{y}}$$

is a proper composite loss with associated proper loss $c_{\exp} : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$ and link function $\psi_{\exp} : [0,1] \to \bar{\mathbb{R}}$ given by

$$c_{\exp}(y, \widehat{\eta}) \;=\; \left(\frac{1-\widehat{\eta}}{\widehat{\eta}}\right)^{y/2} ; \quad \psi_{\exp}(\widehat{\eta}) \;=\; \frac{1}{2} \ln\left(\frac{\widehat{\eta}}{1-\widehat{\eta}}\right) .$$

It is easily verified that $c_{\exp}$ is regular. Moreover, it can be seen that

$$H_{\exp}(\eta) \;=\; 2\sqrt{\eta(1-\eta)}\,,$$

with

$$-H''_{\exp}(\eta) \;=\; \frac{1}{2(\eta(1-\eta))^{3/2}} \;\geq\; 4 \quad \forall \eta \in [0,1]\,.$$

Thus $H_{\exp}$ is 4-strongly concave, and so by Theorem 10, we have $\ell_{\exp}$ is 4-strongly proper composite. Therefore applying Theorem 13 we have for any $f : \mathcal{X} \to \bar{\mathbb{R}}$,

$$\mathrm{regret}_D^{\mathrm{rank}}[f] \;\leq\; \frac{1}{\sqrt{2}\,p(1-p)} \sqrt{\mathrm{regret}_D^{\exp}[f]}\,.$$

Example 2 (Logistic loss) *The logistic loss $\ell_{\exp} : \{\pm 1\} \times \bar{\mathbb{R}} \to \bar{\mathbb{R}}_+$ defined as*

$$\ell_{\log}(y, \widehat{y}) \;=\; \ln(1 + e^{-y\widehat{y}})$$

is a proper composite loss with associated proper loss $c_{\log} : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$ and link function $\psi_{\log} : [0,1] \to \bar{\mathbb{R}}$ given by

$$c_{\log}(1, \widehat{\eta}) \;=\; -\ln \widehat{\eta}\,; \quad c_{\log}(-1, \widehat{\eta}) \;=\; -\ln(1-\widehat{\eta})\,; \quad \psi_{\log}(\widehat{\eta}) \;=\; \ln\left(\frac{\widehat{\eta}}{1-\widehat{\eta}}\right) .$$

Again, it is easily verified that $c_{\log}$ is regular. Moreover, it can be seen that

$$H_{\log}(\eta) \;=\; -\eta \ln \eta - (1-\eta)\ln(1-\eta)\,,$$

with

$$-H''_{\log}(\eta) \;=\; \frac{1}{\eta(1-\eta)} \;\geq\; 4 \quad \forall \eta \in [0,1]\,.$$

Thus $H_{\log}$ is 4-strongly concave, and so by Theorem 10, we have $\ell_{\log}$ is 4-strongly proper composite. Therefore applying Theorem 13 we have for any $f : \mathcal{X} \to \bar{\mathbb{R}}$,

$$\mathrm{regret}_D^{\mathrm{rank}}[f] \;\leq\; \frac{1}{\sqrt{2}\,p(1-p)} \sqrt{\mathrm{regret}_D^{\log}[f]}\,.$$

Example 3 (Squared and squared hinge losses) *The (binary) squared loss* $(1 - y\widehat{y})^2$ *and squared hinge loss* $((1-y\widehat{y})_+)^2$ *(where* $u_+ = \max(u, 0)$*) are generally defined for* $\widehat{y} \in \mathbb{R}$*. To obtain class probability estimates from a predicted value* $\widehat{y} \in \mathbb{R}$*, one then truncates* $\widehat{y}$ *to* $[-1, 1]$*, and uses* $\widehat{\eta} = \frac{\widehat{y}+1}{2}$ *(Zhang, 2004). To obtain a proper loss, we can take* $\widehat{\mathcal{Y}} = [-1, 1]$*; in this range, both losses coincide, and we can define* $\ell_{\text{sq}} : \{\pm 1\} \times [-1, 1] \to [0, 4]$ *as*

$$\ell_{\text{sq}}(y, \widehat{y}) \;=\; (1 - y\widehat{y})^2 \,.$$

This is a proper composite loss with associated proper loss $c_{\text{sq}} : \{\pm 1\} \times [-1, 1] \to [0, 4]$ *and link function* $\psi_{\text{sq}} : [0, 1] \to [-1, 1]$ *given by*

$$c_{\text{sq}}(1, \widehat{\eta}) \;=\; 4(1 - \widehat{\eta})^2 \,; \quad c_{\text{sq}}(-1, \widehat{\eta}) \;=\; 4\widehat{\eta}^2 \,; \quad \psi_{\text{sq}}(\widehat{\eta}) \;=\; 2\widehat{\eta} - 1 \,.$$

It can be seen that

$$L_{\text{sq}}(\eta, \widehat{\eta}) \;=\; 4\eta(1 - \widehat{\eta})^2 + 4(1 - \eta)\widehat{\eta}^2$$

and

$$H_{\text{sq}}(\eta) \;=\; 4\eta(1 - \eta) \,,$$

so that

$$L_{\text{sq}}(\eta, \widehat{\eta}) - H_{\text{sq}}(\eta) \;=\; 4(\eta - \widehat{\eta})^2 \,.$$

Thus ℓ_{sq} *is 8-strongly proper composite, and so applying Theorem 13 we have for any* $f : \mathcal{X} \to [-1, 1]$*,*

$$\text{regret}_D^{\text{rank}}[f] \;\leq\; \frac{1}{2\,p(1-p)} \sqrt{\text{regret}_D^{\text{sq}}[f]} \,.$$

Note that, if a function $f : \mathcal{X} \to \mathbb{R}$ *is learned, then our bound in terms of* ℓ_{sq}*-regret applies to the ranking regret of an appropriately transformed function* $\bar{f} : \mathcal{X} \to [-1, 1]$*, such as that obtained by truncating values* $f(x) \notin [-1, 1]$ *to the appropriate endpoint* -1 *or* 1*:*

$$\bar{f}(x) \;=\; \begin{cases} -1 & \text{if } f(x) < -1 \\ f(x) & \text{if } f(x) \in [-1, 1] \\ 1 & \text{if } f(x) > 1. \end{cases}$$

5.3 Constructing Strongly Proper Losses

In general, given any concave function $H : [0, 1] \to \mathbb{R}_+$, one can construct a proper loss $c : \{\pm 1\} \times [0, 1] \to \bar{\mathbb{R}}_+$ with $H_c = H$ as follows:

$$c(1, \widehat{\eta}) \;=\; H(\widehat{\eta}) + (1 - \widehat{\eta})H'(\widehat{\eta}) \tag{31}$$

$$c(-1, \widehat{\eta}) \;=\; H(\widehat{\eta}) - \widehat{\eta}H'(\widehat{\eta}) \,, \tag{32}$$

where $H'(\widehat{\eta})$ denotes any superderivative of H at $\widehat{\eta}$. It can be verified that this gives $L_c(\eta, \widehat{\eta}) = H(\widehat{\eta}) + (\eta - \widehat{\eta})H'(\widehat{\eta})$ for all $\eta, \widehat{\eta} \in [0, 1]$, and therefore $H_c(\eta) = H(\eta)$ for all $\eta \in [0, 1]$. Moreover, if H is such that $H(\widehat{\eta}) + (1-\widehat{\eta})H'(\widehat{\eta}) \in \mathbb{R}_+ \; \forall \widehat{\eta} \in (0, 1]$ and $H(\widehat{\eta}) - \widehat{\eta}H'(\widehat{\eta}) \in \mathbb{R}_+ \; \forall \widehat{\eta} \in [0, 1)$, then the loss c constructed above is also regular. Thus, starting with any λ-strongly concave function $H : [0, 1] \to \mathbb{R}_+$ satisfying these regularity conditions, any proper composite loss ℓ formed from the loss function c constructed according to Eqs. (31-32) (and any link function ψ) is λ-strongly proper composite.

Loss	$\widehat{\mathcal{Y}}$	$\ell(y,\widehat{y})$	$c(y,\widehat{\eta})$		$\psi(\widehat{\eta})$	λ
			$y=1$	$y=-1$		
Exponential	$\bar{\mathbb{R}}$	$e^{-y\widehat{y}}$	$\sqrt{\frac{1-\widehat{\eta}}{\widehat{\eta}}}$	$\sqrt{\frac{\widehat{\eta}}{1-\widehat{\eta}}}$	$\frac{1}{2}\ln\left(\frac{\widehat{\eta}}{1-\widehat{\eta}}\right)$	4
Logistic	$\bar{\mathbb{R}}$	$\ln(1+e^{-y\widehat{y}})$	$-\ln\widehat{\eta}$	$-\ln(1-\widehat{\eta})$	$\ln\left(\frac{\widehat{\eta}}{1-\widehat{\eta}}\right)$	4
Squared	$[-1,1]$	$(1-y\widehat{y})^2$	$4(1-\widehat{\eta})^2$	$4\widehat{\eta}^2$	$2\widehat{\eta}-1$	8
Spherical	$[0,1]$	$c(y,\widehat{y})$	$1-\frac{\widehat{\eta}}{\sqrt{\widehat{\eta}^2+(1-\widehat{\eta})^2}}$	$1-\frac{1-\widehat{\eta}}{\sqrt{\widehat{\eta}^2+(1-\widehat{\eta})^2}}$	$\widehat{\eta}$	1
Canonical 'exponential'	$\bar{\mathbb{R}}$	$\sqrt{1+\left(\frac{\widehat{y}}{2}\right)^2}-\frac{y\widehat{y}}{2}$	$\sqrt{\frac{1-\widehat{\eta}}{\widehat{\eta}}}$	$\sqrt{\frac{\widehat{\eta}}{1-\widehat{\eta}}}$	$\frac{2\widehat{\eta}-1}{\sqrt{\widehat{\eta}(1-\widehat{\eta})}}$	4
Canonical squared	$[-1,1]$	$\frac{1}{4}(1-y\widehat{y})^2$	$(1-\widehat{\eta})^2$	$\widehat{\eta}^2$	$2\widehat{\eta}-1$	2
Canonical spherical	$[-1,1]$	$1-\frac{1}{2}\left(\sqrt{2-\widehat{y}^2}+y\widehat{y}\right)$	$1-\frac{\widehat{\eta}}{\sqrt{\widehat{\eta}^2+(1-\widehat{\eta})^2}}$	$1-\frac{1-\widehat{\eta}}{\sqrt{\widehat{\eta}^2+(1-\widehat{\eta})^2}}$	$\frac{2\widehat{\eta}-1}{\sqrt{\widehat{\eta}^2+(1-\widehat{\eta})^2}}$	1

Table 1: Examples of strongly proper composite losses $\ell : \{\pm 1\} \times \widehat{\mathcal{Y}} \to \bar{\mathbb{R}}_+$ satisfying the conditions of Theorem 13, together with prediction space $\widehat{\mathcal{Y}}$, proper loss $c : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$, link function $\psi : [0,1] \to \widehat{\mathcal{Y}}$, and strong properness parameter λ.

Example 4 (Spherical loss) *Consider starting with the function* $H_{\text{spher}} : [0,1] \to \mathbb{R}$ *defined as*

$$H_{\text{spher}}(\eta) \;=\; 1 - \sqrt{\eta^2 + (1-\eta)^2}\,.$$

Then

$$H'_{\text{spher}}(\eta) \;=\; \frac{-(2\eta-1)}{\sqrt{\eta^2+(1-\eta)^2}}$$

and

$$-H''_{\text{spher}}(\eta) \;=\; \frac{1}{(\eta^2+(1-\eta)^2)^{3/2}} \;\geq\; 1 \qquad \forall \eta \in [0,1]\,,$$

and therefore H_{spher} *is 1-strongly concave. Moreover, since* H_{spher} *and* H'_{spher} *are both bounded, the conditions for regularity are also satisfied. Thus we can use Eqs. (31-32) to construct a 1-strongly proper loss* $c_{\text{spher}} : \{\pm 1\} \times [0,1] \to \mathbb{R}$ *as follows:*

$$c_{\text{spher}}(1,\widehat{\eta}) \;=\; H_{\text{spher}}(\widehat{\eta}) + (1-\widehat{\eta})H'_{\text{spher}}(\widehat{\eta}) \;=\; 1 - \frac{\widehat{\eta}}{\sqrt{\widehat{\eta}^2+(1-\widehat{\eta})^2}}$$

$$c_{\text{spher}}(-1,\widehat{\eta}) \;=\; H_{\text{spher}}(\widehat{\eta}) - \widehat{\eta}H'_{\text{spher}}(\widehat{\eta}) \;=\; 1 - \frac{1-\widehat{\eta}}{\sqrt{\widehat{\eta}^2+(1-\widehat{\eta})^2}}\,.$$

Therefore by Theorem 13, we have for any $f : \mathcal{X} \to [0,1]$,

$$\text{regret}_D^{\text{rank}}[f] \;\leq\; \frac{\sqrt{2}}{p(1-p)}\sqrt{\text{regret}_D^{\text{spher}}[f]}\,.$$

The loss c_{spher} above corresponds to the spherical scoring rule *described in Gneiting and Raftery (2007).*

We also note that, for every strictly proper loss $c : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$, there is an associated 'canonical' link function $\psi : [0,1] \to \widehat{\mathcal{Y}}$ defined as

$$\psi(\widehat{\eta}) \;=\; c(-1, \widehat{\eta}) - c(1, \widehat{\eta})\,, \tag{33}$$

where $\widehat{\mathcal{Y}} = \{\psi(\widehat{\eta}) : \widehat{\eta} \in [0,1]\}$. We refer to composite losses comprised of such a strictly proper loss c with the corresponding canonical link ψ as *canonical* proper composite losses. Clearly, multiplying c by a factor $\alpha > 0$ results in the corresponding canonical link ψ also being multiplied by α; adding a constant (or a function $\theta(y, \widehat{\eta}) = \theta(\widehat{\eta})$ that is independent of y) to c has no effect on ψ. Conversely, given any $\widehat{\mathcal{Y}} \subseteq \bar{\mathbb{R}}$ and any (strictly increasing) link function $\psi : [0,1] \to \widehat{\mathcal{Y}}$, there is a unique strictly proper loss $c : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$ (up to addition of constants or functions of the form $\theta(y, \widehat{\eta}) = \theta(\widehat{\eta})$) for which ψ is canonical; this is obtained using Eqs. (31-32) with H satisfying $H'(\widehat{\eta}) = -\psi(\widehat{\eta})$ (with possible addition of a term $\theta(\widehat{\eta})$ to both $c(1, \widehat{\eta})$ and $c(-1, \widehat{\eta})$ thus constructed). Canonical proper composite losses $\ell(y, \widehat{y})$ have some desirable properties, including for example convexity in their second argument $\widehat{y}$ for each $y \in \{\pm 1\}$; we refer the reader to Buja et al. (2005); Reid and Williamson (2010) for further discussion of such properties.

We note that the logistic loss in Example 2 is a canonical proper composite loss. On the other hand, as noted in Buja et al. (2005), the link ψ_{exp} associated with the exponential loss in Example 1 is not the canonical link for the proper loss c_{exp} (see Example 5). The squared loss in Example 3 is almost canonical, modulo a scaling factor; one needs to scale either the link function or the loss appropriately (Example 6).

Example 5 (Canonical proper composite loss associated with c_{exp}) *Let $c_{\text{exp}} : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$ be as in Example 1. The corresponding canonical link $\psi_{\text{exp,can}} : [0,1] \to \bar{\mathbb{R}}$ is given by*

$$\psi_{\text{exp,can}}(\widehat{\eta}) \;=\; \sqrt{\frac{\widehat{\eta}}{1-\widehat{\eta}}} - \sqrt{\frac{1-\widehat{\eta}}{\widehat{\eta}}} \;=\; \frac{2\widehat{\eta} - 1}{\sqrt{\widehat{\eta}(1-\widehat{\eta})}}\,.$$

With a little algebra, it can be seen that the resulting canonical proper composite loss $\ell_{\text{exp,can}} : \{\pm 1\} \times \bar{\mathbb{R}} \to \bar{\mathbb{R}}_+$ is given by

$$\ell_{\text{exp,can}}(y, \widehat{y}) \;=\; \sqrt{1 + \left(\frac{\widehat{y}}{2}\right)^2} - \frac{y\widehat{y}}{2}\,.$$

Since we saw c_{exp} is 4-strongly proper, we have $\ell_{\text{exp,can}}$ is 4-strongly proper composite, and therefore we have from Theorem 13 that for any $f : \mathcal{X} \to \bar{\mathbb{R}}$,

$$\text{regret}_D^{\text{rank}}[f] \;\leq\; \frac{1}{\sqrt{2}\,p(1-p)} \sqrt{\text{regret}_D^{\text{exp,can}}[f]}\,.$$

Example 6 (Canonical squared loss) *For $c_{\text{sq}} : \{\pm 1\} \times [0,1] \to [0,4]$ defined as in Example 3, the canonical link $\psi_{\text{sq,can}} : [0,1] \to \widehat{\mathcal{Y}}$ is given by*

$$\psi_{\text{sq,can}}(\widehat{\eta}) \;=\; 4\widehat{\eta}^2 - 4(1-\widehat{\eta})^2 \;=\; 4(2\widehat{\eta} - 1)\,,$$

with $\widehat{\mathcal{Y}} = [-4, 4]$*, and the resulting canonical squared loss* $\ell_{\text{sq,can}} : \{\pm 1\} \times [-4, 4] \to [0, 4]$ *is given by*

$$\ell_{\text{sq,can}}(y, \widehat{y}) \;=\; \Big(1 - \frac{y\widehat{y}}{4}\Big)^2 .$$

Since we saw c_{sq} *is 4-strongly proper, we have* $\ell_{\text{sq,can}}$ *is 4-strongly proper composite, giving for any* $f : \mathcal{X} \to [-4, 4]$*,*

$$\text{regret}_D^{\text{rank}}[f] \;\leq\; \frac{1}{2\,p(1-p)} \sqrt{\text{regret}_D^{\text{sq,can}}[f]} \,.$$

For practical purposes, this is equivalent to using the loss $\ell_{\text{sq}} : \{\pm 1\} \times [-1, 1] \to [0, 4]$ *defined in Example 3. Alternatively, we can start with a scaled version of the squared proper loss* $c_{\text{sq}'} : \{\pm 1\} \times [0, 1] \to [0, 1]$ *defined as*

$$c_{\text{sq}'}(1, \widehat{\eta}) \;=\; (1 - \widehat{\eta})^2 \,; \qquad c_{\text{sq}'}(-1, \widehat{\eta}) \;=\; \widehat{\eta}^2 \,,$$

for which the associated canonical link $\psi_{\text{sq}',\text{can}} : [0, 1] \to \widehat{\mathcal{Y}}$ *is given by*

$$\psi_{\text{sq}',\text{can}}(\widehat{\eta}) \;=\; \widehat{\eta}^2 - (1 - \widehat{\eta})^2 \;=\; 2\widehat{\eta} - 1 \,,$$

with $\widehat{\mathcal{Y}} = [-1, 1]$*, and the resulting canonical squared loss* $\ell_{\text{sq}',\text{can}} : \{\pm 1\} \times [-1, 1] \to [0, 1]$ *is given by*

$$\ell_{\text{sq}',\text{can}}(y, \widehat{y}) \;=\; \frac{(1 - y\widehat{y})^2}{4} \,.$$

Again, it can be verified that $c_{\text{sq}'}$ *is regular; in this case* $H_{\text{sq}'}(\eta) = \eta(1-\eta)$ *which is 2-strongly concave, giving that* $H_{\text{sq}',\text{can}}$ *is 4-strongly proper composite. Therefore applying Theorem 13 we have for any* $f : \mathcal{X} \to [-1, 1]$*,*

$$\text{regret}_D^{\text{rank}}[f] \;\leq\; \frac{1}{p(1-p)} \sqrt{\text{regret}_D^{\text{sq}',\text{can}}[f]} \,.$$

Again, for practical purposes, this is equivalent to using the loss ℓ_{sq} *defined in Example 3.*

Example 7 (Canonical spherical loss) *For* $c_{\text{spher}} : \{\pm 1\} \times [0, 1] \to \mathbb{R}$ *defined as in Example 4, the canonical link* $\psi_{\text{spher,can}} : [0, 1] \to \widehat{\mathcal{Y}}$ *is given by*

$$\psi_{\text{spher,can}}(\widehat{\eta}) \;=\; \frac{2\widehat{\eta} - 1}{\sqrt{\widehat{\eta}^2 + (1 - \widehat{\eta})^2}} \,,$$

with $\widehat{\mathcal{Y}} = [-1, 1]$*. The resulting canonical spherical loss* $\ell_{\text{spher,can}} : \{\pm 1\} \times [-1, 1] \to \mathbb{R}$ *is given by*

$$\ell_{\text{spher,can}}(y, \widehat{y}) \;=\; 1 - \frac{1}{2}\Big(\sqrt{2 - \widehat{y}^2} + y\widehat{y}\Big) \,.$$

Since we saw c_{spher} *is 1-strongly proper, we have* $\ell_{\text{spher,can}}$ *is 1-strongly proper composite, and therefore we have from Theorem 13 that for any* $f : \mathcal{X} \to [-1, 1]$*,*

$$\text{regret}_D^{\text{rank}}[f] \;\leq\; \frac{\sqrt{2}}{p(1-p)} \sqrt{\text{regret}_D^{\text{spher,can}}[f]} \,.$$

6. Tighter Bounds under Low-Noise Conditions

In essence, our results exploit the fact that for any scoring function $f : \mathcal{X} \to \bar{\mathbb{R}}$, given a strongly proper composite loss ℓ, one can construct a class probability estimator from f whose distance from the true class probability function η can be upper bounded in terms of the ℓ-regret of f. Specifically, if ℓ is a strongly proper composite loss with underlying strongly proper loss c and link function ψ, then the L_2 (and therefore L_1) distance between $\psi^{-1}(f(X))$ and $\eta(X)$ (with respect to μ, the marginal density of D on $\mathcal{X}$) can be upper bounded precisely in terms of the ℓ-regret of f. From this perspective, $\widehat{\eta} = \psi^{-1} \circ f$ can be treated as a class probability estimator and therefore a 'plug-in' scoring function, which we analyzed via Corollary 12.

Recently, Clémençon and Robbiano (2011) showed that, under certain low-noise assumptions, one can obtain tighter bounds on the ranking risk of a plug-in scoring function $\widehat{\eta} : \mathcal{X} \to [0,1]$ than that offered by Corollary 12. Specifically, Clémençon and Robbiano (2011) consider the following noise assumption for bipartite ranking (inspired by the noise condition studied in Tsybakov (2004) for binary classification):

Noise Assumption NA(α) ($\alpha \in [0,1]$): *A distribution D on $\mathcal{X} \times \{\pm 1\}$ satisfies assumption* NA(α) *if $\exists$ a constant $C > 0$ such that for all $x \in \mathcal{X}$ and $t \in [0,1]$,*

$$\mathbf{P}_X\big(\big|\eta(X) - \eta(x)\big| \leq t\big) \;\leq\; C \cdot t^{\alpha}\,.$$

Note that $\alpha = 0$ imposes no restriction on D, while larger values of α impose greater restrictions. Clémençon and Robbiano (2011) showed the following result (adapted slightly to our setting, where the ranking risk is conditioned on $Y \neq Y'$):

Theorem 14 (Clémençon and Robbiano (2011)) *Let $\alpha \in [0,1)$ and $q \in [1,\infty)$. Then $\exists$ a constant $C_{\alpha,q} > 0$ such that for any distribution D on $\mathcal{X} \times \{\pm 1\}$ satisfying noise assumption* NA(α) *and any $\widehat{\eta} : \mathcal{X} \to [0,1]$,*

$$\mathrm{regret}_D^{\mathrm{rank}}[\,\widehat{\eta}\,] \;\leq\; \frac{C_{\alpha,q}}{p(1-p)}\Big(\mathbf{E}_X\big[\big|\widehat{\eta}(X) - \eta(X)\big|^q\big]\Big)^{\frac{1+\alpha}{q+\alpha}}\,.$$

This allows us to obtain the following tighter version of our regret bound in terms of strongly proper losses under the same noise assumption:

Theorem 15 *Let $\widehat{\mathcal{Y}} \subseteq \bar{\mathbb{R}}$ and $\lambda > 0$, and let $\alpha \in [0,1)$. Let $\ell : \{\pm 1\} \times \widehat{\mathcal{Y}} \to \bar{\mathbb{R}}_+$ be a λ-strongly proper composite loss. Then $\exists$ a constant $C_\alpha > 0$ such that for any distribution D on $\mathcal{X} \times \{\pm 1\}$ satisfying noise assumption* NA(α) *and any $f : \mathcal{X} \to \widehat{\mathcal{Y}}$,*

$$\mathrm{regret}_D^{\mathrm{rank}}[f] \;\leq\; \frac{C_\alpha}{p(1-p)}\left(\frac{2}{\lambda}\right)^{\frac{1+\alpha}{2+\alpha}}\Big(\mathrm{regret}_D^{\ell}[f]\Big)^{\frac{1+\alpha}{2+\alpha}}\,.$$

Proof Let $c : \{\pm 1\} \times [0,1] \to \bar{\mathbb{R}}_+$ be a λ-strongly proper loss and $\psi : [0,1] \to \widehat{\mathcal{Y}}$ be a (strictly increasing) link function such that $\ell(y, \widehat{y}) = c(y, \psi^{-1}(\widehat{y}))$ for all $y \in \{\pm 1\}, \widehat{y} \in \widehat{\mathcal{Y}}$. Let D be

a distribution on $\mathcal{X} \times \{\pm 1\}$ satisfying noise assumption NA(α) and let $f : \mathcal{X} \to \widehat{\mathcal{Y}}$. Then

$$\begin{aligned}
\text{regret}_D^{\text{rank}}[f] &= \text{regret}_D^{\text{rank}}[\psi^{-1} \circ f], \quad \text{since } \psi \text{ is strictly increasing} \\
&\leq \frac{C_{\alpha,2}}{p(1-p)} \left(\mathbf{E}_X \left[\left(\psi^{-1}(f(X)) - \eta(X) \right)^2 \right] \right)^{\frac{1+\alpha}{2+\alpha}}, \\
&\qquad \text{by Theorem 14, taking } q = 2 \\
&\leq \frac{C_{\alpha,2}}{p(1-p)} \left(\frac{2}{\lambda} \mathbf{E}_X \left[R_c(\eta(X), \psi^{-1}(f(X))) \right] \right)^{\frac{1+\alpha}{2+\alpha}}, \\
&\qquad \text{since } c \text{ is } \lambda\text{-strongly proper} \\
&= \frac{C_{\alpha,2}}{p(1-p)} \left(\frac{2}{\lambda} \mathbf{E}_X \left[R_\ell(\eta(X), f(X)) \right] \right)^{\frac{1+\alpha}{2+\alpha}} \\
&= \frac{C_{\alpha,2}}{p(1-p)} \left(\frac{2}{\lambda} \right)^{\frac{1+\alpha}{2+\alpha}} \left(\text{regret}_D^\ell[f] \right)^{\frac{1+\alpha}{2+\alpha}}.
\end{aligned}$$

The result follows by setting $C_\alpha = C_{\alpha,2}$. ■

For $\alpha = 0$, as noted above, there is no restriction on D, and so the above result gives the same dependence on $\text{regret}_D^\ell[f]$ as that obtained from Theorem 13. On the other hand, as α approaches 1, the exponent of the $\text{regret}_D^\ell[f]$ term in the above bound approaches $\frac{2}{3}$, which improves over the exponent of $\frac{1}{2}$ in Theorem 13.

7. Conclusion and Open Questions

We have obtained upper bounds on the bipartite ranking regret of a scoring function in terms of the (non-pairwise) regret associated with a broad class of proper (composite) losses that we have termed *strongly proper* (composite) losses. This class includes several widely used losses such as exponential, logistic, squared and squared hinge losses as special cases.

The definition and characterization of strongly proper losses may be of interest in its own right and may find applications elsewhere. As one example, we recently found strongly proper losses to be useful in analyzing consistency of algorithms for binary classification in class imbalance settings (Menon et al., 2013). An open question concerns the necessity of the regularity condition in the characterization of strong properness of a proper loss in terms of strong concavity of the conditional Bayes risk (Theorem 10). The characterization of strict properness of a proper loss in terms of strict concavity of the conditional Bayes risk (Theorem 4) does not require such an assumption, and one wonders whether it may be possible to remove the regularity assumption in the case of strong properness as well.

Many of the strongly proper composite losses that we have considered, such as the exponential, logistic, squared and spherical losses, are margin-based losses, which means the bipartite ranking regret can also be upper bounded in terms of the regret associated with pairwise versions of these losses via the reduction to pairwise classification (Section 3.1). A natural question that arises is whether it is possible to characterize conditions on the distribution under which algorithms based on one of the two approaches (minimizing a pairwise form of the loss as in RankBoost/pairwise logistic regression, or minimizing the

standard loss as in AdaBoost/standard logistic regression) lead to faster convergence than those based on the other. We hope the tools and results established here may help in studying such questions in the future.

Acknowledgments

Thanks to Harish G. Ramaswamy and Arun Rajkumar for helpful discussions, and to the anonymous reviewers of this work for helpful comments. Thanks also to Yoonkyung Lee for inviting me to give a talk at the ASA Conference on Statistical Learning and Data Mining held in Ann Arbor, Michigan, in June 2012; part of this work was done while preparing for that talk. This research was supported in part by a Ramanujan Fellowship from the Department of Science and Technology, Government of India.

Appendix A. Proof of Corollary 12

Proof Let $\widehat{\eta} : \mathcal{X} \to [0,1]$. By Theorem 11, we have

$$\text{regret}_D^{\text{rank}}[\widehat{\eta}] \;\leq\; \frac{1}{2p(1-p)} \mathbf{E}_{X,X'} \Big[\big|\eta(X) - \eta(X')\big| \cdot \mathbf{1}\big((\widehat{\eta}(X) - \widehat{\eta}(X'))(\eta(X) - \eta(X')) \leq 0\big) \Big] .$$

The result follows by observing that for any $x, x' \in \mathcal{X}$,

$$(\widehat{\eta}(x) - \widehat{\eta}(x'))(\eta(x) - \eta(x')) \leq 0 \implies |\eta(x) - \eta(x')| \leq |\widehat{\eta}(x) - \eta(x)| + |\widehat{\eta}(x') - \eta(x')| .$$

To see this, note the statement is trivially true if $\eta(x) = \eta(x')$. If $\eta(x) > \eta(x')$, we have

$$\begin{aligned}
(\widehat{\eta}(x) - \widehat{\eta}(x'))(\eta(x) - \eta(x')) \leq 0 &\implies \widehat{\eta}(x) \leq \widehat{\eta}(x') \\
&\implies \eta(x) - \eta(x') \leq (\eta(x) - \widehat{\eta}(x)) + (\widehat{\eta}(x') - \eta(x')) \\
&\implies \eta(x) - \eta(x') \leq |\eta(x) - \widehat{\eta}(x)| + |\widehat{\eta}(x') - \eta(x')| \\
&\implies |\eta(x) - \eta(x')| \leq |\widehat{\eta}(x) - \eta(x)| + |\widehat{\eta}(x') - \eta(x')| .
\end{aligned}$$

The case $\eta(x) < \eta(x')$ can be proved similarly. Thus we have

$$\begin{aligned}
\text{regret}_D^{\text{rank}}[\widehat{\eta}] &\leq \frac{1}{2p(1-p)} \mathbf{E}_{X,X'} \Big[\big|\widehat{\eta}(X) - \eta(X)\big| + \big|\widehat{\eta}(X') - \eta(X')\big| \Big] \\
&= \frac{1}{p(1-p)} \mathbf{E}_X \Big[\big|\widehat{\eta}(X) - \eta(X)\big| \Big] .
\end{aligned}$$

■

References

Shivani Agarwal, Thore Graepel, Ralf Herbrich, Sariel Har-Peled, and Dan Roth. Generalization bounds for the area under the ROC curve. *Journal of Machine Learning Research*, 6:393–425, 2005.

Nir Ailon and Mehryar Mohri. An efficient reduction of ranking to classification. In *Proceedings of the 21st Annual Conference on Learning Theory*, 2008.

Maria-Florina Balcan, Nikhil Bansal, Alina Beygelzimer, Don Coppersmith, John Langford, and Gregory B. Sorkin. Robust reductions from ranking to classification. *Machine Learning*, 72:139–153, 2008.

Peter Bartlett, Michael Jordan, and Jon McAuliffe. Convexity, classification and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

David Buffoni, Clément Calauzenes, Patrick Gallinari, and Nicolas Usunier. Learning scoring functions with order-preserving losses and standardized supervision. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

Andreas Buja, Werner Stuetzle, and Yi Shen. Loss functions for binary class probability estimation: Structure and applications. Technical report, University of Pennsylvania, November 2005.

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 22nd International Conference on Machine Learning*, 2005.

Clément Calauzènes, Nicolas Usunier, and Patrick Gallinari. On the (non-)existence of convex, calibrated surrogate losses for ranking. In *Advances in Neural Information Processing Systems 25*, pages 197–205. 2012.

Stéphan Clémençon and Sylvain Robbiano. Minimax learning rates for bipartite ranking and plug-in rules. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

Stéphan Clémençon and Nicolas Vayatis. Ranking the best instances. *Journal of Machine Learning Research*, 8:2671–2699, 2007.

Stéphan Clémençon, Gabor Lugosi, and Nicolas Vayatis. Ranking and empirical minimization of U-statistics. *Annals of Statistics*, 36:844–874, 2008.

Corinna Cortes and Mehryar Mohri. AUC optimization vs. error rate minimization. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

David Cossock and Tong Zhang. Statistical analysis of Bayes optimal subset ranking. *IEEE Transactions on Information Theory*, 54(11):5140–5154, 2008.

John Duchi, Lester Mackey, and Michael I. Jordan. On the consistency of ranking algorithms. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Arlo D. Hendrickson and Robert J. Buehler. Proper scores for probability forecasters. *The Annals of Mathematical Statistics*, 42:1916–1921, 1971.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers*, pages 115–132, 2000.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM Conference on Knowledge Discovery and Data Mining*, 2002.

Wojciech Kotlowski, Krzysztof Dembczynski, and Eyke Huellermeier. Bipartite ranking through minimization of univariate loss. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

Yanyan Lan, Jiafeng Guo, Xueqi Cheng, and Tie-Yan Liu. Statistical consistency of ranking methods in a rank-differentiable probability space. In *Advances in Neural Information Processing Systems 25*, pages 1241–1249. 2012.

Aditya K. Menon, Harikrishna Narasimhan, Shivani Agarwal, and Sanjay Chawla. On the statistical consistency of algorithms for binary classification under class imbalance. In *Proceedings of the 30th International Conference on Machine Learning*, 2013.

Alain Rakotomamonjy. Optimizing area under ROC curves with SVMs. In *Proceedings of the ECAI-2004 Workshop on ROC Analysis in AI*, 2004.

Harish G. Ramaswamy and Shivani Agarwal. Classification calibration dimension for general multiclass losses. In *Advances in Neural Information Processing Systems 25*, pages 2087–2095. 2012.

Harish G. Ramaswamy, Shivani Agarwal, and Ambuj Tewari. Convex calibrated surrogates for low-rank loss matrices with applications to subset ranking losses. In *Advances in Neural Information Processing Systems 26*, pages 1475–1483. 2013.

Pradeep Ravikumar, Ambuj Tewari, and Eunho Yang. On NDCG consistency of listwise ranking methods. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, 2010*, volume 15 of *JMLR Workshop and Conference Proceedings*, pages 618–626, 2011.

Mark D. Reid and Robert C. Williamson. Composite binary losses. *Journal of Machine Learning Research*, 11:2387–2422, 2010.

Mark D. Reid and Robert C. Williamson. Information, divergence and risk for binary experiments. *Journal of Machine Learning Research*, 12:731–817, 2011.

Cynthia Rudin and Robert E. Schapire. Margin-based ranking and an equivalence between adaboost and rankboost. *Journal of Machine Learning Research*, 10:2193–2232, 2009.

Leonard J. Savage. Elicitation of personal probabilities and expectations. *Journal of the American Statistical Association*, 66(336):783–801, 1971.

Mark J. Schervish. A general method for comparing probability assessors. *The Annals of Statistics*, 17:1856–1879, 1989.

Oliver Stein. Twice differentiable characterizations of convexity notions for functions on full dimensional convex sets. *Schedae Informaticae*, 21:55–63, 2012.

Alexandre B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *The Annals of Statistics*, 32(1):135–166, 2004.

Kazuki Uematsu and Yoonkyung Lee. On theoretically optimal ranking functions in bipartite ranking. Technical Report 863, Department of Statistics, The Ohio State University, December 2011.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: Theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

Tong Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32(1):56–134, 2004.

Journal of Machine Learning Research 15 (2014) 1675-1711 Submitted 2/13; Revised 4/14; Published 5/14

Adaptive Minimax Regression Estimation over Sparse ℓ_q-Hulls

Zhan Wang ZWANG@FULCRM.COM
Fulcrum Analytics
Fairfield, CT 06824, USA

Sandra Paterlini SANDRA.PATERLINI@EBS.EDU
Department of Finance & Accounting
EBS Universität für Wirtschaft und Recht
Gustav-Stresemann-Ring 3
65189 Wiesbaden, Germany

Fuchang Gao FUCHANG@UIDAHO.EDU
Department of Mathematics
University of Idaho
Moscow, ID 83844, USA

Yuhong Yang YYANG@STAT.UMN.EDU
School of Statistics
University of Minnesota
313 Ford Hall
224 Church Street
Minneapolis, MN 55455, USA

Editor: Bin Yu

Abstract

Given a dictionary of M_n predictors, in a random design regression setting with n observations, we construct estimators that target the best performance among all the linear combinations of the predictors under a sparse ℓ_q-norm ($0 \leq q \leq 1$) constraint on the linear coefficients. Besides identifying the optimal rates of convergence, our universal aggregation strategies by model mixing achieve the optimal rates simultaneously over the full range of $0 \leq q \leq 1$ for any M_n and without knowledge of the ℓ_q-norm of the best linear coefficients to represent the regression function.

To allow model misspecification, our upper bound results are obtained in a framework of aggregation of estimates. A striking feature is that no specific relationship among the predictors is needed to achieve the upper rates of convergence (hence permitting basically arbitrary correlations between the predictors). Therefore, whatever the true regression function (assumed to be uniformly bounded), our estimators automatically exploit any sparse representation of the regression function (if any), to the best extent possible within the ℓ_q-constrained linear combinations for any $0 \leq q \leq 1$.

A sparse approximation result in the ℓ_q-hulls turns out to be crucial to adaptively achieve minimax rate optimal aggregation. It precisely characterizes the number of terms needed to achieve a prescribed accuracy of approximation to the best linear combination in an ℓ_q-hull for $0 \leq q \leq 1$. It offers the insight that the minimax rate of ℓ_q-aggregation is basically determined by an effective model size, which is a sparsity index that depends on

q, M_n, n, and the ℓ_q-norm bound in an easily interpretable way based on a classical model selection theory that deals with a large number of models.

Keywords: high-dimensional sparse learning, minimax rate of convergence, model selection, optimal aggregation, sparse ℓ_q-constraint

1. Introduction

Learning a high-dimensional function has become a central research topic in machine learning. In this paper, we intend to provide a theoretical understanding on how well one can adaptively estimate a regression function by sparse linear combinations of a number of predictors based on i.i.d observations.

1.1 Motivation

Sparse modeling has become a popular area of research to handle high-dimensional linear regression learning. One notable approach is to exploit the assumption that the "true" linear coefficients have a bounded ℓ_q-norm (or simply q-norm) for some $0 \leq q \leq 1$, which implies that the parameter space is necessarily sparse in a proper sense. A major recent theoretical advancement is made by Raskutti, Wainwright, and Yu (2012), who derive minimax rates of convergence both for estimating the regression function and for estimating the parameter vector, which spells out how the sparsity parameter q (in conjunction with the number of predictors, say, M_n) affects the intrinsic capability of estimation for the ℓ_q-hulls. The results confirm that even if M_n is much larger than the sample size n, relatively fast rates of convergence in learning the linear regression function are possible. In a Gaussian sequence model framework, Donoho and Johnstone (1994) identify precisely how the ℓ_q-constraint ($q > 0$) on the mean vector affects estimation accuracy under the ℓ_p-loss ($p \geq 1$).

In this paper, differently from the fixed design setting in Raskutti et al. (2012); Negahban et al. (2012), under a random design, we examine the issue of minimax optimal estimation of a linear regression function in the ℓ_q-hulls for $0 \leq q \leq 1$. Besides confirming the same role of q on determining the minimax rate of convergence for estimation of the regression function also for the random design, we prove that the minimax rate can be adaptively achieved without *any* knowledge of q or the ℓ_q-radius of the linear coefficients. The adaptation results show that in high-dimensional linear regression learning, theoretically speaking, should the regression function happen to depend on just a few predictors (i.e., hard sparsity or $q = 0$) or only a small number of coefficients really matter (i.e., soft sparsity or $0 < q \leq 1$), the true sparsity nature is automatically exploited, leading to whatever the optimal rate of convergence for the situation. No restriction is imposed on M_n. To our knowledge, this is the most general result on minimax learning in the ℓ_q-hulls for $0 \leq q \leq 1$.

In reality, obviously, the soft or hard sparsity is only an approximation that hopefully captures the nature of the target function. To deal with possible model misspecification (i.e., the sparsity assumption may or may not be suitable for the data), our upper bound results on regression estimation will be given in a framework that permits the regression function to be outside of the ℓ_q-hulls. The risk bounds show that whichever soft or hard sparse representation of the true regression function by linear combination of the predictors

best describes the truth, the corresponding optimal performance (in rate) is automatically achieved (with some additional conditions for deriving matching minimax lower bounds).

Our aim of simultaneous adaptive estimation over the ℓ_q-hulls for all $0 \leq q \leq 1$ and positive ℓ_q-radius, especially with possible model misspecification, requires a deeper understanding of sparse approximation of functions in the ℓ_q-hulls than what is available in the literature. As a solution, we provide a sharp sparse approximation error bound for ℓ_q-hulls with $0 \leq q \leq 1$, which may also be relevant for studying other linear representation based high-dimensional sparse learning methods.

The aforementioned flexible approach to optimal estimation that allows model misspecification is done in the framework of aggregation of estimates. Besides the aspect of not assuming the true target function to have any specific relationship to the predictors/the initial estimates to be aggregated, the theory of aggregation emphasizes that the predictors/the initial estimates are basically arbitrary in their association with each other. With this characteristic in sight, the minimax rate of aggregation is properly determined by finding a specific set of initial estimates with known relationship (e.g., independence) under which an existing upper bound can be shown to be un-improvable up to a constant factor. In contrast, for minimax optimal regression, one works with whatever (hopefully weak) assumptions imposed on the predictors and tries to achieve the minimax rate of convergence for the function class of interest. With the above, the problem of aggregation of estimates is closely related to the usual regression estimation: A risk upper bound on aggregation of estimates readily gives a risk upper bound for regression estimation, but one has to derive minimax lower bounds for the specific regression learning settings. In this work, we will first give results on aggregation of estimates (where most work is on the upper bounds under minimal assumptions on the initial estimates) and then present results on minimax regression in ℓ_q-hulls (where most work is on deriving lower rates of convergence). The focus is on random design and additions results on fixed design are in Wang et al. (2011).

1.2 Aggregation of Estimates

The idea of sharing strengths of different learning procedures by combining them instead of choosing a single one has led to fruitful and exciting research results in statistics and machine learning. The theoretical advances have centered on optimal risk or loss bounds that require almost no assumption on the behaviors of the individual estimators to be aggregated. See, e.g., Yang (1996, 2000a); Catoni (1997, 2004); Juditsky and Nemirovski (2000); Nemirovski (2000); Yang (2004); Tsybakov (2003); Leung and Barron (2006) for early representative work (the reader is referred to Cesa-Bianchi and Lugosi 2006 for interesting results and references from an individual sequence perspective). While there are many different ways that one can envision to combine the advantages of the candidate procedures, the combining methods can be put into two main categories: those intended for *combining for adaptation*, which aim at combining the procedures to perform adaptively as well as the best candidate procedure no matter what the truth is, and those for *combining for improvement*, which aim at improving over the performance of all the candidate procedures in certain ways. Whatever the goal is, for the purpose of estimating the regression function, we expect to pay a price: the risk of the combined procedure is typically larger than the target risk. The

difference between the two risks (or a proper upper bound on the difference) is henceforth called *risk regret* of the combining method.

The research attention is often focused on one but the main step in the process of combining procedures, namely, *aggregation of estimates*, wherein one has already obtained estimates by all the candidate procedures (based on initial data, most likely from data splitting or previous studies; some exceptions are in e.g., Leung and Barron 2006; Dalalyan and Salmon 2012), and is trying to aggregate these estimates into a single one based on data that are independent of the initial data. The performance of the aggregated estimator (conditional on the initial estimates) plays the most important role in determining the total risk of the whole combined procedure, although proportion of the initial data and the later one certainly also influences the overall performance. In this work, we will mainly focus on the aggregation step.

It is now well-understood that given a collection of procedures, one only needs to pay a relatively small price for aggregation for adaptation (Yang 2000b; Catoni 2004; Tsybakov 2003). In contrast, aggregation for improvement under a convex constraint or ℓ_1-constraint on coefficients is associated with a higher risk regret (as shown in Juditsky and Nemirovski 2000; Nemirovski 2000; Yang 2004; Tsybakov 2003). Several other directions of aggregation for improvement, defined via proper constraints imposed on the ℓ_0-norm alone or in conjunction with the ℓ_1-norm of the linear coefficients, have also been studied, including linear aggregation (no constraint, Tsybakov 2003), aggregation to achieve the best performance of a linear combination of no more than a given number of initial estimates (Bunea et al. 2007) and also under an additional constraint on the ℓ_1-norm of these coefficients (Lounici 2007). Interestingly, combining for adaptation plays a fundamental role in combining for improvement: it serves as an effective tool in constructing multi-directional (or universal) aggregation methods, that is methods which simultaneously achieve the best performance in multiple specific directions of aggregation for improvement. This strategy was taken in, e.g., Yang (2004), Tsybakov (2003), Bunea et al. (2007), Rigollet and Tsybakov (2010), and Dalalyan and Tsybakov (2012b).

The goal of our work on aggregation is to propose aggregation methods that achieve the performance (in risk with/without a multiplying factor), up to a multiple of the optimal risk regret as defined in Tsybakov (2003), of the best linear combination of the initial estimates under the constraint that the ℓ_q-norm ($0 \le q \le 1$) of the linear coefficients is no larger than some positive number t_n (henceforth the *ℓ_q-constraint*). We call this type of aggregation *ℓ_q-aggregation*. It turns out that the optimal rate is simply determined by an *effective model size* m_*, which roughly means that only m_* terms are really needed for effective estimation. We strive to achieve the optimal ℓ_q-aggregation simultaneously for all q ($0 \le q \le 1$) and t_n ($t_n > 0$).

It is useful to note that the ℓ_q-aggregation provides a general framework: our proposed strategies enable one to reach the optimal bounds automatically and simultaneously for the major state-of-art aggregation strategies and more, as will be seen.

1.3 Plan of the Paper

The paper is organized as follows. In Section 2, we introduce notation and some preliminaries of the estimators and aggregation algorithms that will be used in our strategies for

learning. In Section 3, we derive optimal rates of ℓ_q-aggregation and show that our methods achieve multi-directional aggregation. In Section 4, we derive the minimax rate for linear regression with ℓ_q-constrained coefficients. A discussion is then reported in Section 5. Finally, we report in Appendix A the derivation of metric entropy and approximation error bounds for $\ell_{q,t_n}^{M_n}$-hulls, while Appendix B provides an insight from the sparse approximation bound based on classical model selection theory. The proofs of the results in Sections 3 and 4 are then provided in the Appendix C.

2. Preliminaries

Consider the regression problem where a dictionary of M_n prediction functions ($M_n \geq 2$ unless stated otherwise) are given as initial estimates of the unknown true regression function. The goal is to construct a linearly combined estimator using these estimates to pursue the performance of the best (possibly constrained) linear combinations. A learning strategy with two building blocks will be considered. First, we construct candidate estimators from subsets of the given estimates. Second, we aggregate the candidate estimators using aggregation algorithms to obtain the final estimator.

2.1 Notation and Definition

Let $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$ be n ($n \geq 2$) i.i.d. observations where $\mathbf{X}_i = (X_{i,1}, \ldots, X_{i,d})$, $1 \leq i \leq n$, take values in $\mathcal{X}$ with a probability distribution P_X. We assume the regression model

$$Y_i = f_0(\mathbf{X}_i) + \varepsilon_i, \qquad i = 1, \ldots n, \tag{1}$$

where f_0 is the unknown true regression function to be estimated. The random errors ε_i, $1 \leq i \leq n$, are independent of each other and of $\mathbf{X}_i$, and have the probability density function $h(x)$ (with respect to the Lebesgue measure or a general measure μ) such that $E(\varepsilon_i) = 0$ and $E(\varepsilon_i^2) = \sigma^2 < \infty$. The quality of estimating f_0 by using the estimator $\hat{f}$ is measured by the squared L_2 risk (with respect to P_X)

$$R(\hat{f}; f_0; n) = E\|\hat{f} - f_0\|^2 = E\left(\int (\hat{f} - f_0)^2 dP_X\right),$$

where, as in the rest of the paper, $\|\cdot\|$ denotes the L_2-norm with respect to the distribution of P_X.

Let $F_n = \{f_1, f_2, \ldots, f_{M_n}\}$ be a dictionary of M_n initial estimates of f_0. In this paper, unless stated otherwise, $\|f_j\| \leq 1$, $1 \leq j \leq M_n$. The condition is satisfied, possibly after a scaling, if the f_j's are uniformly bounded between known constants, and it may require additional assumptions on the distribution of P_X to check its validity for a general case. Consider the constrained linear combinations of the estimates $\mathcal{F} = \left\{f_\theta = \sum_{j=1}^{M_n} \theta_j f_j : \theta \in \Theta_n, f_j \in F_n\right\}$, where Θ_n is a subset of $\mathbb{R}^{M_n}$. Let

$$d^2(f_0; \mathcal{F}) = \inf_{f_\theta \in \mathcal{F}} \|f_\theta - f_0\|^2$$

denote the smallest approximation error to f_0 over a function class $\mathcal{F}$.

The problem of constructing an estimator $\hat{f}$ that pursues the best performance in $\mathcal{F}$ is called *aggregation of estimates*. We consider aggregation of estimates with sparsity constraints on θ. For any $\theta = (\theta_1, \ldots, \theta_{M_n})'$, define the ℓ_0-norm and the ℓ_q-norm $(0 < q \leq 1)$ by

$$\|\theta\|_0 = \sum_{j=1}^{M_n} I(\theta_j \neq 0), \text{ and } \|\theta\|_q = \left(\sum_{j=1}^{M_n} |\theta_j|^q \right)^{1/q},$$

where $I(\cdot)$ is the indicator function. Note that for $0 < q < 1$, $\|\cdot\|_q$ is not a norm but a quasinorm, and for $q = 0$, $\|\cdot\|_0$ is not even a quasinorm. However, we choose to refer them as norms for ease of exposition. For any $0 \leq q \leq 1$ and $t_n > 0$, define the ℓ_q-ball

$$B_q(t_n; M_n) = \left\{ \theta = (\theta_1, \theta_2, \ldots, \theta_{M_n})' : \|\theta\|_q \leq t_n \right\}.$$

When $q = 0$, t_n is understood to be an integer between 1 and M_n, and sometimes denoted by k_n to be distinguished from t_n when $q > 0$. Define the $\ell_{q,t_n}^{M_n}$*-hull* of F_n to be the class of linear combinations of functions in F_n with the ℓ_q-constraint

$$\mathcal{F}_q(t_n) = \mathcal{F}_q(t_n; M_n; F_n) = \left\{ f_\theta = \sum_{j=1}^{M_n} \theta_j f_j : \theta \in B_q(t_n; M_n), f_j \in F_n \right\}, 0 \leq q \leq 1, t_n > 0.$$

One of our goals is to propose an estimator $\hat{f}_{F_n} = \sum_{j=1}^{M_n} \hat{\theta}_j f_j$ such that its risk is upper bounded by a multiple of the smallest risk over the class $\mathcal{F}_q(t_n)$ plus a small risk regret term

$$R(\hat{f}_{F_n}; f_0; n) \leq C \inf_{f_\theta \in \mathcal{F}_q(t_n)} \|f_\theta - f_0\|^2 + REG_q(t_n; M_n),$$

where C is a constant that does not depend on f_0, n, and M_n, or $C = 1$ for some estimators. In various situations (e.g., adaptive estimation with data splitting as in Yang 2000a), the initial estimates can be made such that $\inf_{f_\theta \in \mathcal{F}_q(t_n)} \|f_\theta - f_0\|^2$ approaches zero as $n \to \infty$ in a proper manner. Thus, results with $C > 1$ (especially under heavy tailed random errors) are also of interest.

2.2 Two Starting Estimators

A key step of our strategy is the construction of candidate estimators using subsets of the initial estimates. The T- and AC-estimators, described below, were chosen because of the relatively mild assumptions for them to work with respect to the squared L_2 risk (each of them gives cleaner results in different aspects). Under the data generating model (1) and i.i.d. observations $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$, suppose we are given m terms $\{g_1, \ldots, g_m\}$ (i.e., m functions of the original explanatory variables) as regressors.

When working on the minimax upper bounds in random design settings, we always make the following assumption on the true regression function.

Assumption BD: There exists a known constant $L > 0$ such that $\|f_0\|_\infty \leq L < \infty$.

To our knowledge, Assumption BD is typically assumed in the literature of aggregation of estimates. A recent work of Birgé (2014) successfully removes this limitation in a density estimation framework.

(T-estimator) Birgé (2006, 2004) constructed the T-estimator and derived its L_2 risk bounds under the Gaussian regression setting. The following proposition is a simple consequence of Theorem 3 of Birgé (2004). Suppose
T1. The error distribution $h(\cdot)$ is normal;
T2. $0 < \sigma < \infty$ is known.

Proposition 1 *Suppose Assumptions* BD *and* T1, T2 *hold. We can construct a T-estimator* $\hat{f}^{(T)}$ *such that*

$$E\|\hat{f}^{(T)} - f_0\|^2 \le C_{L,\sigma}\left(\inf_{\vartheta\in\mathbb{R}^m}\left\|\sum_{j=1}^m \vartheta_j g_j - f_0\right\|^2 + \frac{m}{n}\right),$$

where $C_{L,\sigma}$ *is a constant depending only on* L *and* σ*.*

(AC-estimator) For our purpose, consider the class of linear combinations with the ℓ_1-constraint $\mathcal{G}_s = \{g = \sum_{j=1}^m \vartheta_j g_j : \|\vartheta\|_1 \le s\}$ for some $s > 0$. Audibert and Catoni proposed a sophisticated AC-estimator $\hat{f}_s^{(AC)}$ (Audibert and Catoni 2010, page 25). The following proposition is a direct result from Theorem 4.1 in Audibert and Catoni (2010) under the following conditions.
AC1. There exists a constant $H > 0$ such that $\sup_{g,g'\in\mathcal{G}_s,\mathbf{x}\in\mathcal{X}} |g(\mathbf{x}) - g'(\mathbf{x})| = H < \infty$.
AC2. There exists a constant $\sigma' > 0$ such that $\sup_{\mathbf{x}\in\mathcal{X}} E\left((Y - g_s^*(\mathbf{X}))^2|\mathbf{X} = \mathbf{x}\right) \le (\sigma')^2 < \infty$, where $g_s^* = \inf_{g\in\mathcal{G}_s}\|g - f_0\|^2$.

Proposition 2 *Suppose Assumptions* AC1 *and* AC2 *hold. For any* $s > 0$*, we can construct an AC-estimator* $\hat{f}_s^{(AC)}$ *(that may depend on* H *and* σ'*) such that*

$$E\|\hat{f}_s^{(AC)} - f_0\|^2 \le \inf_{g\in\mathcal{G}_s}\|g - f_0\|^2 + c\left(2\sigma' + H\right)^2\frac{m}{n},$$

where c *is a pure constant.*

The risk bound for the AC-estimator improves over that for the T-estimator in terms of i) reducing the multiplying constant in front of the optimal approximation error to the best possible; ii) relaxing the normality assumption on the errors. But this is achieved at the expense of restricting the ℓ_1-norm of the linear coefficients in approximation. Note also that under the assumption $\|f_0\|_\infty \le L$, we can always enforce the estimators $\hat{f}^{(T)}$ and $\hat{f}_s^{(AC)}$ to be in the range of $[-L, L]$ with the same risk bounds in the propositions.

2.3 Two Aggregation Algorithms for Adaptation

Suppose N estimates $\check{f}_1, \ldots, \check{f}_N$ are obtained from N candidate procedures based on some initial data. Two aggregation algorithms, the ARM algorithm (Adaptive Regression by Mixing, Yang 2001) and Catoni's algorithm (Catoni 2004, 1999), can be used to construct the final estimator $\hat{f}$ by aggregating the candidate estimates $\check{f}_1, \ldots, \check{f}_N$ based on n additional i.i.d. observations $(\mathbf{X}_i, Y_i)_{i=1}^n$. The ARM algorithm requires knowing the form of the error distribution but it allows heavy tail cases. In contrast, Catoni's algorithm does not assume any functional form of the error distribution, but demands exponential decay of the tail probability.

(The ARM algorithm) Suppose
Y1. There exist two known constants $\underline{\sigma}$ and $\overline{\sigma}$ such that $0 < \underline{\sigma} \le \sigma \le \overline{\sigma} < \infty$;
Y2. The error density has the form $h(x) = h_0(x/\sigma)/\sigma$, where h_0 is known and has mean zero, variance 1, and a finite fourth moment. In addition, for each pair of constants $R_0 > 0$ and $0 < S_0 < 1$, there exists a constant B_{S_0,R_0} (depending on S_0 and R_0) such that for all $|R| < R_0$ and $S_0 \le S \le S_0^{-1}$,

$$\int h_0(x) \log \frac{h_0(x)}{S^{-1} h_0((x-R)/S)} dx \le B_{S_0,R_0}((1-S)^2 + R^2).$$

The condition Y2 can be shown to hold for Gaussian, Laplace and Student's t (with at least 3 degrees of freedom) distributions. We can construct an estimator $\hat{f}^Y$ which aggregates $\check{f}_1, \ldots, \check{f}_N$ by the ARM algorithm with prior probabilities π_k ($\sum_{k=1}^N \pi_k = 1$) on the procedures.

Proposition 3 *(Yang 2004, Proposition 1) Suppose Assumptions* BD *and* Y1, Y2 *hold, and* $\|\check{f}_k\|_\infty \le L < \infty$ *with probability* 1, $1 \le k \le N$. *The estimator* $\hat{f}^Y$ *by the ARM algorithm has the risk*

$$R(\hat{f}^Y; f_0; n) \le C_Y \inf_{1 \le k \le N} \left(\|\check{f}_k - f_0\|^2 + \frac{\sigma^2}{n} \left(1 + \log \frac{1}{\pi_k}\right)\right),$$

where C_Y *is a constant that depends on* $\underline{\sigma}, \overline{\sigma}, L$, *and also* h *(through the fourth moment of the random error and* B_{S_0,R_0} *with* $S_0 = \underline{\sigma}/\overline{\sigma}, R_0 = L$*).*

(Catoni's algorithm) Suppose for some positive constant $\alpha < \infty$, there exist known constants $U_\alpha, V_\alpha < \infty$ such that
C1. $E(\exp(\alpha|\varepsilon_i|)) \le U_\alpha$;
C2. $\frac{E(\varepsilon_i^2 \exp(\alpha|\varepsilon_i|))}{E(\exp(\alpha|\varepsilon_i|))} \le V_\alpha$.
Let $\lambda_C = \min\{\frac{\alpha}{2L}, (U_\alpha(17L^2 + 3.4V_\alpha))^{-1}\}$ and π_k be the prior for $\check{f}_k$, $1 \le k \le N$.

Proposition 4 *(Catoni 2004, Theorem 3.6.1) Suppose Assumptions* BD *and* C1, C2 *hold, and* $\|\check{f}_k\|_\infty \le L < \infty$, $1 \le k \le N$. *The estimator* $\hat{f}^C$ *that aggregates* $\check{f}_1, \ldots, \check{f}_N$ *by Catoni's algorithm has the risk*

$$R(\hat{f}^C; f_0; n) \le \inf_{1 \le k \le N} \left(\|\check{f}_k - f_0\|^2 + \frac{2}{n\lambda_C} \log \frac{1}{\pi_k} \right).$$

Juditsky et al. (2008) (p. 2200) give a similar result under simplified conditions.

3. ℓ_q-Aggregation of Estimates

Consider the setup from Section 2.1. We focus on the problem of aggregating the estimates in F_n to pursue the best performance in $\mathcal{F}_q(t_n)$ for $0 \le q \le 1$, $t_n > 0$, which we call *ℓ_q-aggregation of estimates*. To be more precise, when needed, it will be called $\ell_q(t_n)$-aggregation, and for the special case of $q = 0$, we call it $\ell_0(k_n)$-aggregation for $1 \le k_n \le M_n$.

3.1 The Strategy

For each $1 \le m \le M_n \wedge n$ and each subset model $J_m \subset \{1, 2, \ldots, M_n\}$ of size m, define $\mathcal{F}_{J_m} = \{\sum_{j\in J_m} \theta_j f_j : \theta_j \in \mathbb{R}, j \in J_m\}$. Let $\mathcal{F}^L_{J_m,s} = \{f_\theta = \sum_{j\in J_m} \theta_j f_j : \|\theta\|_1 \le s, \|f_\theta\|_\infty \le L\}$ $(s = 1, 2, ...)$ be the class of ℓ_1-constrained linear combinations in F_n with a sup-norm bound on f_θ. Our strategy is as follows.

Step I. Divide the data into two parts: $Z^{(1)} = (\mathbf{X}_i, Y_i)_{i=1}^{n_1}$ and $Z^{(2)} = (\mathbf{X}_i, Y_i)_{i=n_1+1}^{n}$.

Step II. Based on data $Z^{(1)}$, obtain a T-estimator for each function class $\mathcal{F}_{J_m}$, or obtain an AC-estimator for each function class $\mathcal{F}^L_{J_m,s}$ with $s \in \mathbb{N}$.

Step III. Based on data $Z^{(2)}$, combine all estimators obtained in step II and the null model ($f \equiv 0$) using Catoni's or the ARM algorithm. Let p_0 be a small positive number in $(0, 1)$. In all, we have to combine $\sum_{m=1}^{M_n\wedge n} \binom{M_n}{m}$ T-estimators with the weight $\pi_{J_m} = (1 - p_0)\left((M_n \wedge n)\binom{M_n}{m}\right)^{-1}$ and the null model with the weight $\pi_0 = p_0$, or combine countably many AC-estimators with the weight $\pi_{J_m,s} = (1 - p_0)\left((1 + s)^2(M_n \wedge n)\binom{M_n}{m}\right)^{-1}$ and the null model with the weight $\pi_0 = p_0$. (Note that sub-probabilities on the models do not affect the validity of the risk bounds to be given.)

For simplicity of exposition, from now on and when relevant, we assume n is even and choose $n_1 = n/2$ in our strategy. However, similar results hold for other values of n and n_1.

We use the expression "**E-G** strategy" for ease of presentation where $\mathbf{E} = \mathbf{T}$ or $\mathbf{AC}$ represents the estimators constructed in Step II, and $\mathbf{G} = \mathbf{C}$ or $\mathbf{Y}$ stands for the aggregation algorithm used in Step III. By our construction, Assumption AC1 is automatically satisfied: for each J_m, $H_{J_m,s} = \sup_{f,f'\in\mathcal{F}^L_{J_m,s},\mathbf{x}\in\mathcal{X}} |f(\mathbf{x}) - f'(\mathbf{x})| \le 2L$. Assumption AC2 is met with $(\sigma')^2 = \sigma^2 + 4L^2$.

We assume the following conditions are satisfied for each strategy, respectively.

$$\begin{aligned} \mathrm{A}_{\mathbf{T-C}} \text{ and } \mathrm{A}_{\mathbf{T-Y}} &: \text{BD, T1, T2.} \\ \mathrm{A}_{\mathbf{AC-C}} &: \text{BD, C1, C2.} \\ \mathrm{A}_{\mathbf{AC-Y}} &: \text{BD, Y1, Y2.} \end{aligned}$$

Given that T1, T2 are stronger than C1, C2 and Y1, Y2, it is enough to require their satisfaction in $\mathrm{A}_{\mathbf{T-C}}$ and $\mathrm{A}_{\mathbf{T-Y}}$.

3.2 Minimax Rates for ℓ_q-Aggregation of Estimates

Consider general M_n, t_n and $0 < q \leq 1$. The *ideal model size* (in order) that balances the approximation error and the estimation error under the ℓ_q-constraint over $1 \leq m \leq M_n \wedge n$ is

$$m^* = m^*(q, t_n) = \left\lceil 2\left(nt_n^2\tau\right)^{q/2} \right\rceil \wedge M_n \wedge n,$$

where $\tau = \sigma^{-2}$ is the precision parameter. The *effective model size* (in order) that yields the optimal rate of convergence, as will be shown, is

$$m_* = m_*(q, t_n) = \begin{cases} m^* & \text{if } m^* = M_n \wedge n, \\ \left\lceil \frac{m^*}{\left(1+\log \frac{M_n}{m^*}\right)^{q/2}} \right\rceil & \text{otherwise.} \end{cases}$$

See Appendix B for an explanation on why m_* is expected to yield the minimax rate of convergence. Let $\mathcal{F}_q^L(t_n) = \mathcal{F}_q(t_n) \cap \{f : \|f\|_\infty \leq L\}$ for $0 \leq q \leq 1$, and define

$$m_*^{\mathcal{F}} = \begin{cases} m_*(q, t_n) & \text{for case 1: } \mathcal{F} = \mathcal{F}_q(t_n), 0 < q \leq 1, \\ k_n \wedge n & \text{for case 2: } \mathcal{F} = \mathcal{F}_0(k_n), \\ m_*(q, t_n) \wedge k_n & \text{for case 3: } \mathcal{F} = \mathcal{F}_q(t_n) \cap \mathcal{F}_0(k_n), 0 < q \leq 1. \end{cases}$$

Note that in the third case, we are simply taking the smaller one between the effective model sizes from the soft sparsity constraint (ℓ_q-constraint with $0 < q \leq 1$) and the hard sparsity one (ℓ_0-constraint), and this smaller size determines the final sparsity. Define

$$REG(m_*^{\mathcal{F}}) = \sigma^2 \left(1 \wedge \frac{m_*^{\mathcal{F}} \cdot \left(1 + \log\left(\frac{M_n}{m_*^{\mathcal{F}}}\right)\right)}{n} \right),$$

which will be shown to be typically the optimal rate of the risk regret for ℓ_q-aggregation.

For case 3, we intend to achieve the best performance of linear combinations when both ℓ_0- and ℓ_q-constraints are imposed on the linear coefficients, which results in ℓ_q-aggregation using just a subset of the initial estimates and is called $\ell_0 \cap \ell_q$-aggregation. For the special case of $q = 1$, this $\ell_0 \cap \ell_1$-aggregation is studied in Yang (2004) (page 36) for multi-directional aggregation and in Lounici (2007) (called D-convex aggregation) more formally, giving also lower bounds. Our results below not only handle $q < 1$ but also close a gap of a logarithmic factor in upper and lower bounds in Lounici (2007).

For ease of presentation, we may use the same symbol (e.g., C) to denote possibly different constants of the same nature.

Theorem 5 *Suppose* $\mathrm{A}_{\mathbf{E-G}}$ *holds for the* ***E-G*** *strategy respectively. Our estimator* $\hat{f}_{F_n}$ *simultaneously has the following properties.*

(i) For ***T-*** *strategies, for* $\mathcal{F} = \mathcal{F}_q(t_n)$ *with* $0 < q \leq 1$, *or* $\mathcal{F} = \mathcal{F}_0(k_n)$, *or* $\mathcal{F} = \mathcal{F}_q(t_n) \cap \mathcal{F}_0(k_n)$ *with* $0 < q \leq 1$, *we have*

$$R(\hat{f}_{F_n}; f_0; n) \leq \left[C_0 d^2(f_0; \mathcal{F}) + C_1 REG(m_*^{\mathcal{F}})\right] \wedge \left[C_0 \left(\|f_0\|^2 \vee \frac{C_2\sigma^2}{n}\right)\right].$$

(ii) For ***AC-*** *strategies, for $\mathcal{F} = \mathcal{F}_q(t_n)$ with $0 < q \le 1$, or $\mathcal{F} = \mathcal{F}_0(k_n)$, or $\mathcal{F} = \mathcal{F}_q(t_n) \cap \mathcal{F}_0(k_n)$ with $0 < q \le 1$, we have*

$$R(\hat{f}_{F_n}; f_0; n) \le C_1 REG(m_*^{\mathcal{F}}) + C_0 \begin{cases} d^2(f_0; \mathcal{F}_q^L(t_n)) + \frac{C_2\sigma^2 \log(1+t_n)}{n} & \text{for case 1,} \\ \inf_{s\ge 1} \left(d^2(f_0; \mathcal{F}_1(s) \cap \mathcal{F}_0^L(k_n)) + \frac{C_2\sigma^2 \log(1+s)}{n} \right) & \text{for case 2,} \\ d^2(f_0; \mathcal{F}_q^L(t_n) \cap \mathcal{F}_0^L(k_n)) + \frac{C_2\sigma^2 \log(1+t_n)}{n} & \text{for case 3.} \end{cases}$$

Also, $R(\hat{f}_{F_n}; f_0; n) \le C_0 \left(\|f_0\|^2 \vee \frac{C_2\sigma^2}{n} \right)$.

For all these cases, C_0, C_1, and C_2 do not depend on n, f_0, t_n, q, k_n, M_n. These constants may depend on L, p_0, σ^2 or $\overline{\sigma}^2/\underline{\sigma}^2$, α, U_α, V_α when relevant. An exception is that $C_0 = 1$ for the ***AC-C*** *strategy.*

Remark 6 *For case 2, the boundedness assumption of $\|f_j\| \le 1$, $1 \le j \le M_n$ is not necessary.*

Remark 7 *If the true function f_0 happens to have a small L_2-norm such that $\|f_0\|^2 \vee \frac{\sigma^2}{n}$ is of a smaller order than $REG(m_*^{\mathcal{F}})$, then its inclusion in the risk bounds may improve the rate of convergence.*

Discussion of the bounds. Note that an extra term of $\log(1 + t_n)/n$ is present in the upper bounds of the estimator obtained by **AC-** strategies. For case 1, let us focus on the high-dimensional situation of M_n between order n and order $e^{O(n)}$. When t_n is no larger (in order) than $\sigma n^{1/q-1/2}$, the extra price $\log(1 + t_n)/n$ does not damage the rate of convergence if $\frac{\log(1+t_n)}{t_n^q}$ is no larger in order than $\frac{n^{q/2}}{\sigma^q}(\log M_n)^{1-q/2}$, which does hold as $n \to \infty$ when q is fixed. When t_n is at least of order $\sigma n^{1/q-1/2}$, $REG(m_*^{\mathcal{F}})$ is of order 1, and from Proposition 15 in the Appendix, it can be seen that under the conditions of the theorem, the risks of the AC- strategies are also of order 1. For case 2, the extra term in Theorem 5 is harmless in rate if for some $s \le e^{cn} \wedge e^{ck_n(1+\log(M_n/k_n))}$ for some constant $c > 0$, the ℓ_1-norm constraint does not enlarge the approximation error order.

Comparison to the existing literature. When $q = 1$, our theorem covers some important previous aggregation results. With $t_n = 1$, Juditsky and Nemirovski (2000) obtained the optimal result for large M_n; Yang (2004) gave upper bounds for all M_n, but the rate is slightly sub-optimal (by a logarithmic factor) when $M_n = O(\sqrt{n})$ and with a factor larger than 1 in front of the approximation error; Tsybakov (2003) derived the optimal rates for both large and small M_n (and also for linear aggregation) but under the assumption that the joint distribution of $\{f_j(\mathbf{X}), j = 1, ..., M_n\}$ is known. For the case $M_n = O(\sqrt{n})$, Audibert and Catoni (2010) have improved over Yang (2004) and Tsybakov (2003) by giving an optimal risk bound. Thus in the special case of $q = 1$, our result overcomes the aforementioned shortcomings by integrating the previous understandings together, with the additional generality of t_n. In the direction of adaptive aggregation, Dalalyan and Tsybakov (2012b) give risk bounds, up to a logarithmic factor, suitable for $q = 0, 1$. Our result here closes the logarithmic factor gap and also handles q between 0 and

1. Note also that Rigollet and Tsybakov (2010) obtain adaptive optimal ℓ_q-aggregation for $q = 0, 1$ under a fixed design Gaussian regression setting.

Next, we establish lower bounds for the aggregation problems that match (up to a multiplicative constant) the upper bounds above, which then implies that the estimators by our strategies are indeed minimax adaptive for ℓ_q-aggregation of estimates (with respect to both q and the ℓ_q-radius). Let $f_1, \ldots, f_{M_n}$ be an orthonormal basis with respect to the distribution of $\mathbf{X}$. Since the earlier upper bounds are obtained under the assumption that the true regression function f_0 satisfies $\|f_0\|_\infty \le L$ for some known (possibly large) constant $L > 0$, for our lower bound result below, this assumption will also be considered. For the last result in part (iii) below under the sup-norm constraint on f_0, the functions $f_1, \ldots, f_{M_n}$ are specially constructed on $[0, 1]$ and P_X is the uniform distribution on $[0, 1]$. See the proof for details.

In order to give minimax lower bounds without any norm assumption on f_0, let $\widetilde{m}_*^{\mathcal{F}}$ be defined the same as $m_*^{\mathcal{F}}$ except that the ceiling of n is removed. Define

$$\overline{REG}(\widetilde{m}_*^{\mathcal{F}}) = \frac{\sigma^2 \widetilde{m}_*^{\mathcal{F}} \cdot \left(1 + \log\left(\frac{M_n}{\widetilde{m}_*^{\mathcal{F}}}\right)\right)}{n} \wedge \begin{cases} t_n^2 & \text{for cases 1 and 3,} \\ \infty & \text{for case 2,} \end{cases}$$

$$\underline{REG}(m_*^{\mathcal{F}}) = REG(m_*^{\mathcal{F}}) \wedge \begin{cases} t_n^2 & \text{for cases 1 and 3,} \\ \infty & \text{for case 2.} \end{cases}$$

Theorem 8 *Suppose the noise ε follows a normal distribution with mean 0 and variance $\sigma^2 > 0$.*

(i) *For any aggregated estimator $\hat{f}_{F_n}$ based on an orthonormal dictionary $F_n = \{f_1, \ldots, f_{M_n}\}$, for $\mathcal{F} = \mathcal{F}_q(t_n)$, or $\mathcal{F} = \mathcal{F}_0(k_n)$, or $\mathcal{F} = \mathcal{F}_q(t_n) \cap \mathcal{F}_0(k_n)$ with $0 < q \le 1$, one can find a regression function f_0 (that may depend on $\mathcal{F}$) such that*

$$R(\hat{f}_{F_n}; f_0; n) - d^2(f_0; \mathcal{F}) \ge C \cdot \overline{REG}(\widetilde{m}_*^{\mathcal{F}}),$$

where C may depend on q (and only q) for cases 1 and 3 and is an absolute constant for case 2.

(ii) *Under the additional assumption that $\|f_0\| \le L$ for a known $L > 0$, the above lower bound becomes $C' \cdot \underline{REG}(m_*^{\mathcal{F}})$ for the three cases, where C' may depend on q and L for cases 1 and 3 and on L for case 2.*

(iii) *With the additional knowledge $\|f_0\|_\infty \le L$ for a known $L > 0$, the lower bound $C'' \cdot \underline{REG}(m_*^{\mathcal{F}})$ also holds for the following situations: 1) for $\mathcal{F} = \mathcal{F}_q(t_n)$ with $0 < q \le 1$, if $\sup_{f_\theta \in \mathcal{F}_q(t_n)} \|f_\theta\|_\infty \le L$; 2) for $\mathcal{F} = \mathcal{F}_0(k_n)$, if $\sup_{1 \le j \le M_n} \|f_j\|_\infty \le L < \infty$ and $\frac{k_n^2}{n}(1 + \log \frac{M_n}{k_n})$ are bounded above; 3) for $\mathcal{F} = \mathcal{F}_0(k_n)$, if $M_n / \left(1 + \log \frac{M_n}{k_n}\right) \le bn$ for some constant $b > 0$ and the orthonormal basis is specially chosen.*

Remark 9 *Consider the interesting high-dimensional case of M_n between order n and order $e^{O(n)}$. Then $\underline{REG}(m_*^{\mathcal{F}})$ is of the same order as $REG(m_*^{\mathcal{F}})$ unless t_n^2 is of a smaller order than $\log M_n / n$. Thus, except this situation of small t_n, the lower bounds above match the orders of the upper bounds in the previous theorem.*

For satisfaction of $\sup_{f_\theta \in \mathcal{F}_q(t_n)} \|f_\theta\|_\infty \le L$, consider uniformly bounded functions f_j, then for $0 < q \le 1$,

$$\|\sum_{j=1}^{M_n} \theta_j f_j\|_\infty \le \sum_{j=1}^{M_n} |\theta_j| \|f_j\|_\infty \le \left(\sup_{1\le j\le M_n} \|f_j\|_\infty\right) \|\theta\|_1 \le \left(\sup_{1\le j\le M_n} \|f_j\|_\infty\right) \|\theta\|_q.$$

Thus, under the condition that $\left(\sup_{1\le j\le M_n} \|f_j\|_\infty\right) t_n$ is upper bounded, $\sup_{f_\theta \in \mathcal{F}_q(t_n)} \|f_\theta\|_\infty \le L$ is met.

The lower bounds given in part (iii) of the theorem for the three cases of ℓ_q-aggregation of estimates are of the same order of the upper bounds in the previous theorem, respectively, unless t_n is too small. Hence, under the given conditions, the minimax rates for ℓ_q-aggregation are identified according to the definition of the minimax rate of aggregation in Tsybakov (2003). When no restriction is imposed on the norm of f_0, the lower bounds can certainly approach infinity (e.g., when t_n is really large). That is why $\overline{REG}(\widetilde{m}_*^{\mathcal{F}})$ is introduced. The same can be said for later lower bounds.

For the new case $0 < q < 1$, the ℓ_q-constraint imposes a type of soft-sparsity more stringent than $q = 1$: even more coefficients in the linear expression are pretty much negligible. For the discussion below, assume $m^* < n$. When the radius t_n increases or $q \to 1$, m^* increases given that the ℓ_q-ball enlarges. When $m_* = m^* = M_n < n$, the ℓ_q-constraint is not tight enough to impose sparsity: ℓ_q-aggregation is then simply equivalent to linear aggregation and the risk regret term corresponds to the estimation price of the full model, $M_n\sigma^2/n$. In contrast, when $1 < m_* < M_n \wedge n$, the rate for ℓ_q-aggregation is

$$\sigma^{2-q} t_n^q \left(\frac{\log\left(1 + \frac{M_n}{(n\tau t_n^2)^{q/2}}\right)}{n}\right)^{1-q/2}.$$

When $m^* \le (1 + \log(M_n/m^*))^{q/2}$ or equivalently $m_* = 1$, the ℓ_q-constraint restricts the search space of the optimization problem so much that it suffices to consider at most one f_j and the null model may provide a better risk.

Now let us explain that our ℓ_q-aggregation includes the commonly studied aggregation problems in the literature. First, when $q = 1$, we have the well-known convex or ℓ_1-aggregation (but now with the ℓ_1-norm bound allowed to be general). Second, when $q = 0$, with $k_n = M_n \le n$, we have the linear aggregation. For other $k_n < M_n \wedge n$, we have the aggregation to achieve the best linear performance of only k_n initial estimates. The case $q = 0$ and $k_n = 1$ has a special implication. Observe that from Theorem 5, we deduce that for both the **T-** strategies and **AC-** strategies, under the assumption $\sup_j \|f_j\|_\infty \le L$, our estimator satisfies

$$R(\hat{f}_{F_n}; f_0; n) \le C_0 \inf_{1\le j\le M_n} \|f_j - f_0\|^2 + C_1\sigma^2 \left(1 \wedge \frac{1 + \log M_n}{n}\right),$$

where $C_0 = 1$ for the **AC-C** strategy. Together with the lower bound of the order $\sigma^2\left(1 \wedge \frac{1+\log M_n}{n}\right)$ on the risk regret of aggregation for adaptation given in Tsybakov (2003), we conclude that $\ell_0(1)$-aggregation directly implies the aggregation for adaptation (model selection aggregation). As mentioned earlier, $\ell_0(k_n) \cap \ell_q(t_n)$-aggregation pursues the best

performance of the linear combination of at most k_n initial estimates with coefficients satisfying the ℓ_q-constraint, which includes the D-convex aggregation as a special case (with $q = 1$).

Some additional interesting results on combining procedures are in Audibert (2007, 2009); Birgé (2006); Bunea and Nobel (2008); Catoni (2012); Dalalyan and Tsybakov (2007, 2012a); Giraud (2008); Goldenshluger (2009); Györfi et al. (2002); Györfi and Ottucsák (2007); Wegkamp (2003); Yang (2001).

4. Linear Regression with ℓ_q-Constrained Coefficients under Random Design

Let's consider the linear regression model with M_n predictors $X_1, \ldots, X_{M_n}$. Suppose the data are drawn i.i.d. from the following model

$$Y = f_0(\mathbf{X}) + \varepsilon = \sum_{j=1}^{M_n} \theta_j X_j + \varepsilon. \tag{2}$$

As previously defined, for a function $f(x_1, \ldots, x_{M_n}) : \mathcal{X} \to \mathbb{R}$, the L_2-norm $\|f\|$ is the square root of $Ef^2(X_1, \ldots, X_{M_n})$, where the expectation is taken with respect to P_X, the distribution of $\mathbf{X}$. Denote the $\ell_{q,t_n}^{M_n}$-hull in this context by

$$\mathcal{F}_q(t_n; M_n) = \left\{ f_\theta = \sum_{j=1}^{M_n} \theta_j x_j : \|\theta\|_q \le t_n \right\}, \ 0 \le q \le 1, \ t_n > 0.$$

For linear regression, we assume coefficients of the true regression function f_0 have a sparse ℓ_q-representation ($0 < q \le 1$) or ℓ_0-representation or both, i.e., $f_0 \in \mathcal{F}$ where $\mathcal{F} = \mathcal{F}_q(t_n; M_n)$, $\mathcal{F}_0(k_n; M_n)$ or $\mathcal{F}_q(t_n; M_n) \bigcap \mathcal{F}_0(k_n; M_n)$.

Assumptions BD and $A_{\mathbf{E-G}}$ are still relevant in this section. As in the previous section, for AC-estimators, we consider ℓ_1- and sup-norm constraints.

For each $1 \le m \le M_n \wedge n$ and each subset J_m of size m, let $\mathcal{G}_{J_m} = \{\sum_{j \in J_m} \theta_j x_j : \theta \in \mathbb{R}^m\}$ and $\mathcal{G}_{J_m,s}^L = \{\sum_{j \in J_m} \theta_j x_j : \|\theta\|_1 \le s, \|f_\theta\|_\infty \le L\}$. We introduce now the adaptive estimator $\hat{f}_A$, built with the same strategy used to construct $\hat{f}_{F_n}$ except that we now consider $\mathcal{G}_{J_m}$ and $\mathcal{G}_{J_m,s}^L$ instead of $\mathcal{F}_{J_m}$ and $\mathcal{F}_{J_m,s}^L$.

4.1 Upper Bounds

We give upper bounds for the risk of our estimator assuming $f_0 \in \mathcal{F}_q^L(t_n; M_n)$, $\mathcal{F}_0^L(k_n; M_n)$, or $\mathcal{F}_q^L(t_n; M_n) \cap \mathcal{F}_0^L(k_n; M_n)$, where $\mathcal{F}^L = \{f : f \in \mathcal{F}, \|f\|_\infty \le L\}$ for a positive constant L. Let $\alpha_n = \sup_{f \in \mathcal{F}_0^L(k_n; M_n)} \inf\{\|\theta\|_1 : f_\theta = f\}$ be the maximum smallest ℓ_1-norm needed to represent the functions in $\mathcal{F}_0^L(k_n; M_n)$.

Recall $m^* = \left\lceil 2\left(nt_n^2/\sigma^2\right)^{q/2} \right\rceil \wedge M_n \wedge n$ and m_* equals m^* when $m^* = M_n \wedge n$ and $\left\lceil \frac{m^*}{\left(1+\log \frac{M_n}{m^*}\right)^{q/2}} \right\rceil$ otherwise. For ease of presentation, define $\Psi^{\mathcal{F}}$ as follows:

$$\Psi^{\mathcal{F}_q^L(t_n;M_n)} = \begin{cases} \sigma^2 & \text{if } m_* = n, \\ \frac{\sigma^2 M_n}{n} & \text{if } m_* = M_n < n, \\ \sigma^{2-q} t_n^q \left(\frac{1+\log \frac{M_n}{(nt_n^2\tau)^{q/2}}}{n} \right)^{1-q/2} \wedge \sigma^2 & \text{if } 1 < m_* < M_n \wedge n, \\ \left(t_n^2 \vee \frac{\sigma^2}{n}\right) \wedge \sigma^2 & \text{if } m_* = 1, \end{cases}$$

$$\Psi^{\mathcal{F}_0^L(k_n;M_n)} = \sigma^2 \left(1 \wedge \frac{k_n\left(1+\log \frac{M_n}{k_n}\right)}{n} \right),$$

$$\Psi^{\mathcal{F}_q^L(t_n;M_n)\cap\mathcal{F}_0^L(k_n;M_n)} = \Psi^{\mathcal{F}_q^L(t_n;M_n)} \wedge \Psi^{\mathcal{F}_0^L(k_n;M_n)}.$$

In addition, for lower bound results, let $\underline{\Psi}^{\mathcal{F}_q^L(t_n;M_n)}$ $(0 \le q \le 1)$ and $\underline{\Psi}^{\mathcal{F}_q^L(t_n;M_n)\cap\mathcal{F}_0^L(k_n;M_n)}$ $(0 < q \le 1)$ be the same as $\Psi^{\mathcal{F}_q^L(t_n;M_n)}$ and $\Psi^{\mathcal{F}_q^L(t_n;M_n)\cap\mathcal{F}_0^L(k_n;M_n)}$, respectively, except that when $0 < q \le 1$ and $m_* = 1$, $\underline{\Psi}^{\mathcal{F}_q^L(t_n;M_n)}$ takes the value $\sigma^2 \wedge t_n^2$ instead of $\sigma^2 \wedge \left(t_n^2 \vee \frac{\sigma^2}{n}\right)$ and $\underline{\Psi}^{\mathcal{F}_q^L(t_n;M_n)\cap\mathcal{F}_0^L(k_n;M_n)}$ is modified the same way.

Corollary 10 *Suppose $A_{\mathbf{E-G}}$ holds for the **E-G** strategy respectively, and $\sup_{1\le j\le M_n} \|X_j\|_\infty \le 1$. The estimator $\hat{f}_A$ simultaneously has the following properties.*

*(i) For **T-** strategies, for $\mathcal{F} = \mathcal{F}_q^L(t_n; M_n)$ with $0 < q \le 1$, or $\mathcal{F} = \mathcal{F}_0^L(k_n; M_n)$, or $\mathcal{F} = \mathcal{F}_q^L(t_n; M_n) \cap \mathcal{F}_0^L(k_n; M_n)$ with $0 < q \le 1$, we have*

$$\sup_{f_0\in\mathcal{F}} R(\hat{f}_A; f_0; n) \le C_1 \Psi^{\mathcal{F}},$$

where the constant C_1 does not depend on n.

*(ii) For **AC-** strategies, for $\mathcal{F} = \mathcal{F}_q^L(t_n; M_n)$ with $0 < q \le 1$, or $\mathcal{F} = \mathcal{F}_0^L(k_n; M_n)$, or $\mathcal{F} = \mathcal{F}_q^L(t_n; M_n) \cap \mathcal{F}_0^L(k_n; M_n)$ with $0 < q \le 1$, we have*

$$\sup_{f_0\in\mathcal{F}} R(\hat{f}_A; f_0; n) \le C_1 \Psi^{\mathcal{F}} + C_2 \begin{cases} \frac{\sigma^2 \log(1+\alpha_n)}{n} & \text{for } \mathcal{F} = \mathcal{F}_0^L(k_n; M_n), \\ \frac{\sigma^2 \log(1+t_n)}{n} & \text{otherwise,} \end{cases}$$

where the constants C_1 and C_2 do not depend on n.

We need to point out that closely related work has been done under a fixed design setting. While determining the minimax rate of convergence, Raskutti et al. (2012) derive optimal estimators of a function (only at the design points) in the ℓ_q-hulls for $0 \le q \le 1$, but the estimators require knowledge of q and t_n. Negahban et al. (2012), when applying their general M-estimation methodology to the same fixed design regression setting, give

an optimal estimator of the true coefficient vector (as opposed to the regression function) assumed in any ℓ_q-ball for $0 \le q \le 1$ under the squared error loss. It requires that the predictors satisfy a restricted eigenvalue (RE) condition. Raskutti et al. (2010) show that under broad Gaussian random designs, the RE condition holds with exponentially small exception probability. Therefore, the fixed design performance bounds in Negahban et al. (2012), as well as those in Raskutti et al. (2012) (which do not need the RE condition), can be used to draw some conclusions for Gaussian random designs or more general random design in the latter case. Our risk upper bounds (directly under the global squared L_2 loss) do not require the Gaussian assumption on the covariates nor the RE condition. In addition, differently from the aforementioned two papers, our performance bounds hold without any restriction on the relationship between M and n.

4.2 Lower Bounds

The lower bounds used in the previous section for deriving the minimax rate of aggregation are not suitable for obtaining the minimax rate of convergence for the current regression estimation problem. We make the following near orthogonality assumption on sparse subcollections of the predictors. Such an assumption, similar to the sparse Riesz condition (SRC) (Zhang 2010) under fixed design, is used only for lower bounds but not for upper bounds.

Assumption SRC: For some $\gamma > 0$, there exist two positive constants $\underline{a}$ and $\overline{a}$ that do not depend on n such that for every θ with $\|\theta\|_0 \le \min(2\gamma, M_n)$ we have

$$\underline{a}\|\theta\|_2 \le \|f_\theta\| \le \overline{a}\|\theta\|_2.$$

Theorem 11 *Suppose the noise ε follows a normal distribution with mean 0 and variance $0 < \sigma^2 < \infty$.*

(i) For $0 < q \le 1$, under Assumption SRC with $\gamma = m_*$, we have

$$\inf_{\hat{f}} \sup_{f_0 \in \mathcal{F}_q(t_n; M_n)} E\|\hat{f} - f_0\|^2 \ge c\underline{\Psi}^{\mathcal{F}_q^L(t_n; M_n)}.$$

(ii) Under Assumption SRC with $\gamma = k_n$, we have

$$\inf_{\hat{f}} \sup_{f_0 \in \mathcal{F}_0(k_n; M_n) \cap \{f_\theta : \|\theta\|_2 \le a_n\}} E\|\hat{f} - f_0\|^2 \ge c' \begin{cases} \underline{\Psi}^{\mathcal{F}_0^L(k_n; M_n)} & \text{if } a_n \ge \tilde{c}\sigma\sqrt{\frac{k_n\left(1+\log\frac{M_n}{k_n}\right)}{n}}, \\ a_n^2 & \text{if } a_n < \tilde{c}\sigma\sqrt{\frac{k_n\left(1+\log\frac{M_n}{k_n}\right)}{n}}. \end{cases}$$

where $\tilde{c}$ is a pure constant.

(iii) For any $0 < q \le 1$, under Assumption SRC with $\gamma = k_n \wedge m_*$, we have

$$\inf_{\hat{f}} \sup_{f_0 \in \mathcal{F}_0(k_n; M_n) \cap \mathcal{F}_q(t_n; M_n)} E\|\hat{f} - f_0\|^2 \ge c'' \underline{\Psi}^{\mathcal{F}_q^L(t_n; M_n) \cap \mathcal{F}_0^L(k_n; M_n)}.$$

For all cases, $\inf_{\hat{f}}$ *is over all estimators and the constants* c, $c^{'}$ *and* $c^{''}$ *may depend on* $\underline{a}$, $\overline{a}$, q *and* σ^2.

For the second case (ii), the lower bound is stated in a more informative way because the effect of the bound on $\|\theta\|_2$ is clearly seen.

4.3 The Minimax Rates of Convergence

Combining the upper and lower bounds, we give a representative minimax rate result with the roles of the key quantities n, M_n, q, and k_n explicitly seen in the rate expressions. Below "$\asymp$" means of the same order when L, L_0, q, $t_n = t$, and $\overline{\sigma}^2$ ($\overline{\sigma}^2$ is defined in Corollary 12 below) are held constant in the relevant expressions.

Corollary 12 *Suppose the noise* ε *follows a normal distribution with mean* 0 *and variance* σ^2, *and there exists a known constant* $\overline{\sigma}$ *such that* $0 < \sigma \le \overline{\sigma} < \infty$. *Also assume there exists a known constant* $L_0 > 0$ *such that* $\sup_{1\le j\le M_n} \|X_j\|_\infty \le L_0 < \infty$.

(i) For $0 < q \le 1$, *under Assumption SRC with* $\gamma = m_*$,

$$\inf_{\hat{f}} \sup_{f_0 \in \mathcal{F}_q^L(t;M_n)} E\|\hat{f} - f_0\|^2 \asymp 1 \wedge \begin{cases} 1 & \text{if } m_* = n, \\ \frac{M_n}{n} & \text{if } m_* = M_n < n, \\ \left(\frac{1+\log \frac{M_n}{(nt^2\tau)^{q/2}}}{n} \right)^{1-q/2} & \text{if } 1 \le m_* < M_n \wedge n. \end{cases}$$

(ii) If there exists a constant $K_0 > 0$ *such that* $\frac{k_n^2\left(1+\log \frac{M_n}{k_n}\right)}{n} \le K_0$, *then under Assumption SRC with* $\gamma = k_n$,

$$\inf_{\hat{f}} \sup_{f_0 \in \mathcal{F}_0^L(k_n;M_n) \cap \{f_\theta : \|\theta\|_\infty \le L_0\}} E\|\hat{f} - f_0\|^2 \asymp 1 \wedge \frac{k_n\left(1+\log \frac{M_n}{k_n}\right)}{n}.$$

(iii) If $\sigma > 0$ *is actually known, then under the condition* $\frac{k_n^2\left(1+\log \frac{M_n}{k_n}\right)}{n} \le K_0$ *and Assumption SRC with* $\gamma = k_n$, *we have*

$$\inf_{\hat{f}} \sup_{f_0 \in \mathcal{F}_0^L(k_n;M_n)} E\|\hat{f} - f_0\|^2 \asymp 1 \wedge \frac{k_n\left(1+\log \frac{M_n}{k_n}\right)}{n},$$

and for any $0 < q \le 1$, *under Assumption SRC with* $\gamma = k_n \wedge m_*$, *we have*

$$\inf_{\hat{f}} \sup_{f_0 \in \mathcal{F}_0^L(k_n;M_n) \cap \mathcal{F}_q^L(t;M_n)} E\|\hat{f} - f_0\|^2 \asymp 1 \wedge \begin{cases} \frac{k_n\left(1+\log \frac{M_n}{k_n}\right)}{n} & \text{if } m_* > k_n, \\ \left(\frac{1+\log \frac{M_n}{(nt^2\tau)^{q/2}}}{n} \right)^{1-q/2} & \text{if } 1 \le m_* \le k_n. \end{cases}$$

5. Conclusion

Sparse modeling by imposing an ℓ_q-constraint on the coefficients of a linear representation of a target function to be learned has found consensus among academics and practitioners in many application fields, among which, just to mention a few, compressed sensing, signal and image compression, gene-expression, cryptography and recovery of loss data. The ℓ_q-constraints promote sparsity essential for high-dimensional learning and they also are often approximately satisfied on natural classes of signal and images, such as the bounded variation model for images and the bump algebra model for spectra (see Donoho 2006).

In the direction of using the ℓ_1-constraints in constructing estimators, algorithmic and theoretical results have been well developed. Both the Lasso and the Dantzig selector have been shown to achieve the rate $k_n \log(M_n)/n$ under different conditions on correlations of predictors and the hard sparsity constraint on the linear coefficients (see van de Geer and Bühlmann 2009 for a discussion about the sufficient conditions for deriving oracle inequalities for the Lasso). Our upper bound results do not require any of those conditions, but we do assume the sparse Riesz condition for deriving the lower bounds. Computational issues aside, we have seen that the approach of model selection/combination with descriptive complexity penalty has provided the most general adaptive estimators that automatically exploit the sparsity characteristics of the target function in terms of linear approximations subject to ℓ_q-constraints.

In our results, the effective model size m_* (as defined in Section 3.2 and further explained in Appendix B) plays a key role in determining the minimax rate of ℓ_q-aggregation for $0 < q \leq 1$. With the extended definition of the effective model size m_* to be simply the number of nonzero components k_n when $q = 0$ and re-defining m_* to be $m_* \wedge k_n$ under both ℓ_q- $(0 < q \leq 1)$ and ℓ_0-constraints, the minimax rate of aggregation is unified to be the simple form $1 \wedge \frac{m_*\left(1+\log\left(\frac{M_n}{m_*}\right)\right)}{n}$.

The ℓ_q-aggregation includes as special cases the state-of-art aggregation problems, namely aggregation for adaptation, convex and D-convex aggregations, linear aggregation, and subset selection aggregation, and all of them can be defined (or essentially so) by considering linear combinations under ℓ_0- and/or ℓ_1-constraints. Our investigation provides optimal rates of aggregation, which not only agrees with (and, in some cases, improves over) previous findings for the mostly studied aggregation problems, but also holds for a much larger set of linear combination classes. Indeed, we have seen that ℓ_0-aggregation includes aggregation for adaptation over the initial estimates (or model selection aggregation) ($\ell_0(1)$-aggregation), linear aggregation when $M_n \leq n$ ($\ell_0(M_n)$-aggregation), and aggregation to achieve the best performance of linear combination of k_n estimates in the dictionary for $1 < k_n < M_n$ (sometimes called subset selection aggregation) ($\ell_0(k_n)$-aggregation). When M_n is large, aggregating a subset of the dictionary under an ℓ_q-constraint for $0 < q \leq 1$ can be advantageous, which is just $\ell_0(k_n) \cap \ell_q(t_n)$-aggregation. Since the optimal rates of aggregation as defined in Tsybakov (2003), can differ substantially in different directions of aggregation and typically one does not know which direction works the best for the unknown regression function, multi-directional or universal aggregation is important so that the final estimator is automatically conservative and aggressive, whichever is better (see Yang 2004). Our aggregation strategy is indeed multi-directional , achieving the optimal rates over all ℓ_q-aggregation for $0 \leq q \leq 1$ and $\ell_0 \cap \ell_q$-aggregation for all $0 < q \leq 1$.

Our focus in this work is of a theoretical nature to provide an understanding of the fundamental theoretical issues about ℓ_q-aggregation or linear regression under ℓ_q-constraints. Computational aspects will be studied in the future.

Acknowledgments

We thank Yannick Baraud for helpful discussions and for pointing out the work of Audibert and Catoni (2011) that helped us to remove a logarithmic factor in some of our previous aggregation risk bounds. Three anonymous referees are sincerely thanked for their valuable comments on improving our work.

Sandra Paterlini conducted part of this research while visiting the School of Mathematics, University of Minnesota. Her research was partially supported by Fondazione Cassa di Risparmio di Modena for REFIGLO. Yuhong Yang's research was partially supported by NSF grant DMS-1106576.

Appendix A. Metric Entropy and Sparse Approximation Error of $\ell_{q,t_n}^{M_n}$-Hulls

It is well-known that the metric entropy plays a fundamental role in determining minimax-rates of convergence, as shown, e.g., in Birgé (1986); Yang and Barron (1999).

For each $1 \le m \le M_n$ and each subset $J_m \subset \{1, 2, \ldots, M_n\}$ of size m, recall $\mathcal{F}_{J_m} = \{\sum_{j\in J_m} \theta_j f_j : \theta_j \in \mathbb{R}, j \in J_m\}$. Recall also

$$d^2(f_0; \mathcal{F}) = \inf_{f_\theta \in \mathcal{F}} \|f_\theta - f_0\|^2$$

is the smallest approximation error to f_0 over a function class $\mathcal{F}$.

Theorem 13 *(Metric entropy and sparse approximation bound for $\ell_{q,t_n}^{M_n}$-hulls) Suppose $F_n = \{f_1, f_2, ..., f_{M_n}\}$ with $\|f_j\|_{L^2(\nu)} \le 1$, $1 \le j \le M_n$, where ν is a σ-finite measure.*

(i) For $0 < q \le 1$, there exists a positive constant c_q depending only on q, such that for any $0 < \epsilon < t_n$, $\mathcal{F}_q(t_n)$ contains an ϵ-net $\{e_j\}_{j=1}^{N_\epsilon}$ in the $L_2(\nu)$ distance with $\|e_j\|_0 \le 5(t_n\epsilon^{-1})^{2q/(2-q)} + 1$ for $j = 1, 2, ..., N_\epsilon$, where N_ϵ satisfies

$$\log N_\epsilon \le \begin{cases} c_q \left(t_n\epsilon^{-1}\right)^{\frac{2q}{2-q}} \log(1 + M_n^{\frac{1}{q}-\frac{1}{2}} t_n^{-1}\epsilon) & \text{if } \epsilon > t_n M_n^{\frac{1}{2}-\frac{1}{q}}, \\ c_q M_n \log(1 + M_n^{\frac{1}{2}-\frac{1}{q}} t_n\epsilon^{-1}) & \text{if } \epsilon \le t_n M_n^{\frac{1}{2}-\frac{1}{q}}. \end{cases} \tag{3}$$

(ii) For any $1 \le m \le M_n$, $0 < q \le 1$, $t_n > 0$, there exists a subset J_m and $f_{\theta^m} \in \mathcal{F}_{J_m}$ with $\|\theta^m\|_1 \le t_n$ such that the sparse approximation error is upper bounded as follows

$$\|f_{\theta^m} - f_0\|^2 - d^2(f_0; \mathcal{F}_q(t_n)) \le 2^{2/q-1} t_n^2 m^{1-2/q}. \tag{4}$$

Remark 14 *The metric entropy estimate (3) is the best possible. Indeed, if f_j, $1 \le j \le M_n$, are orthonormal functions, then (3) is sharp in order for any ϵ satisfying that ϵ/t_n is bounded away from 1 (see Kühn 2001). Part (i) of Theorem 13 implies Lemma 3 of Raskutti et al. (2012), with an improvement of a $\log(M_n)$ factor when $\epsilon \approx t_n M_n^{\frac{1}{2}-\frac{1}{q}}$, and an improvement from $(t_n\epsilon^{-1})^{\frac{2q}{2-q}}\log(M_n)$ to $M_n \log(1 + M_n^{\frac{1}{q}-\frac{1}{2}} t_n\epsilon^{-1})$ when $\epsilon < t_n M_n^{\frac{1}{2}-\frac{1}{q}}$. These improvements are needed to derive the exact minimax rates for some of the possible situations in terms of M_n, q, and t_n.*

A.1 Proof of Theorem 13

(i) Because $\{e_j\}_{j=1}^{N_\epsilon}$ is an ϵ-net of $\mathcal{F}_q(t_n)$ if and only if $\{t_n^{-1}e_j\}_{j=1}^{N_\epsilon}$ is an ϵ/t_n-net of $\mathcal{F}_q(1)$, we only need to prove the theorem for the case $t_n = 1$. Recall that for any positive integer k, the unit ball of $\ell_q^{M_n}$ can be covered by 2^{k-1} balls of radius ϵ_k in ℓ_1 distance, where

$$\epsilon_k \le c \begin{cases} 1 & 1 \le k \le \log_2(2M_n) \\ \left(\frac{\log_2(1+\frac{2M_n}{k})}{k}\right)^{\frac{1}{q}-1} & \log_2(2M_n) \le k \le 2M_n \\ 2^{-\frac{k}{2M_n}}(2M_n)^{1-\frac{1}{q}} & k \ge 2M_n \end{cases}$$

(c.f., Edmunds and Triebel 1998, page 98). Thus, there are 2^{k-1} functions g_j, $1 \le j \le 2^{k-1}$, such that

$$\mathcal{F}_q(1) \subset \bigcup_{j=1}^{2^{k-1}} (g_j + \mathcal{F}_1(\epsilon_k)).$$

Note that without loss of generality, g_j can be assumed to belong to $\mathcal{F}_q(1)$ (because if not we can replace it by a member in $\mathcal{F}_q(1)$ closest to it in ℓ_1 distance on the coefficient vectors (which is a real distance), the effect of which is merely a change of the constant c above). For any $g \in \mathcal{F}_1(\epsilon_k)$, g can be expressed as $g = \sum_{i=1}^{M_n} c_i f_i$ with $\sum_{i=1}^{M_n} |c_i| \le \epsilon_k$. Following the idea of Maurey's empirical method (see, e.g., Pisier 1981), we define a random function U, such that

$$\mathbb{P}(U = \mathrm{sign}(c_i)\epsilon_k f_i) = |c_i|/\epsilon_k, \quad \mathbb{P}(U = 0) = 1 - \sum_{i=1}^{M_n} |c_i|/\epsilon_k.$$

Then, we have $\|U\|_2 \le \epsilon_k$ a.s. and $\mathbb{E}U = g$ under the randomness just introduced. Let $U_1, U_2, ..., U_m$ be i.i.d. copies of U, and let $V = \frac{1}{m}\sum_{i=1}^m U_i$. We have

$$\mathbb{E}\|V - g\|_2 \le \sqrt{\frac{1}{m}\|\mathbb{V}ar(U)\|_2} \le \sqrt{\frac{1}{m}\mathbb{E}\|U\|_2^2} \le \frac{\epsilon_k}{\sqrt{m}}.$$

In particular, there exists a realization of V, such that $\|V - g\|_2 \le \epsilon_k/\sqrt{m}$. Note that V can be expressed as $\epsilon_k m^{-1}(k_1 f_1 + k_2 f_2 + \cdots + k_{M_n} f_{M_n})$, where k_1, k_2, ..., k_{M_n} are integers, and $|k_1| + |k_2| + \cdots + |k_{M_n}| \le m$. Thus, the total number of different realizations of V is upper bounded by $\binom{2M_n+m}{m}$. Furthermore, $\|V\|_0 \le m$.

If $\log_2(2M_n) \le k \le 2M_n$, we choose m to be the largest integer such that $\binom{2M_n+m}{m} \le 2^k$. Then we have

$$\frac{1}{m} \le \frac{c'}{k}\log_2\left(1 + \frac{2M_n}{k}\right)$$

for some positive constant c'. Hence, $\mathcal{F}_q(1)$ can be covered by 2^{2k-1} balls of radius

$$\epsilon_k \sqrt{c' k^{-1} \log_2 \left(1 + \frac{2M_n}{k}\right)}$$

in L^2 distance.

If $k \geq 2M_n$, we choose $m = M_n$. Then $\mathcal{F}_q(1)$ can be covered by $2^{k-1}\binom{2M_n+m}{m}$ balls of radius $\epsilon_k M_n^{-1/2}$ in L^2 distance. Consequently, there exists a positive constant c'' such that $\mathcal{F}_q(1)$ can be covered by 2^{l-1} balls of radius r_l, where

$$r_l \leq c'' \begin{cases} 1 & 1 \leq l \leq \log_2(2M_n), \\ l^{\frac{1}{2}-\frac{1}{q}} [\log_2(1 + \frac{2M_n}{l})]^{\frac{1}{q}-\frac{1}{2}} & \log_2(2M_n) \leq l \leq 2M_n, \\ 2^{-\frac{l}{2M_n}} (2M_n)^{\frac{1}{2}-\frac{1}{q}} & l \geq 2M_n. \end{cases}$$

For any given $0 < \epsilon < 1$, by choosing the smallest l such that $r_l < \epsilon/2$, we find an $\epsilon/2$-net $\{u_i\}_{i=1}^N$ of $\mathcal{F}_q(1)$ in L^2 distance, where

$$N = 2^{l-1} \leq \begin{cases} \exp\left(c''' \epsilon^{-\frac{2q}{2-q}} \log(1 + M_n^{\frac{1}{q}-\frac{1}{2}} \epsilon)\right) & \epsilon > M_n^{\frac{1}{2}-\frac{1}{q}}, \\ \exp\left(c''' M_n \log(1 + M_n^{\frac{1}{2}-\frac{1}{q}} \epsilon^{-1})\right) & \epsilon < M_n^{\frac{1}{2}-\frac{1}{q}}, \end{cases}$$

and c''' is some positive constant.

It remains to show that for each $1 \leq i \leq N$, we can find a function e_i so that $\|e_i\|_0 \leq 5\epsilon^{2q/(q-2)} + 1$ and $\|e_i - u_i\|_2 \leq \epsilon/2$.

Suppose $u_i = \sum_{j=1}^{M_n} c_{ij} f_j$, $1 \leq i \leq N$, with $\sum_{j=1}^{M_n} |c_{ij}|^q \leq 1$. Let $L_i = \{j : |c_{ij}| > \epsilon^{2/(2-q)}\}$. Then, $|L_i|\epsilon^{2q/(2-q)} \leq \sum |c_{ij}|^q \leq 1$, which implies $|L_i| \leq \epsilon^{2q/(q-2)}$ and also

$$\sum_{j \notin L_i} |c_{ij}| \leq \sum_{j \notin L_i} |c_{ij}|^q [\epsilon^{2/(2-q)}]^{1-q} \leq \epsilon^{\frac{2-2q}{2-q}}.$$

Define $v_i = \sum_{j \in L_i} c_{ij} f_j$ and $w_i = \sum_{j \notin L_i} c_{ij} f_j$. We have $w_i \in \mathcal{F}_1(\epsilon^{\frac{2-2q}{2-q}})$. By the probability argument above, we can find a function w_i' such that $\|w_i'\|_0 \leq m$ and $\|w_i - w_i'\|_2 \leq \epsilon^{\frac{2-2q}{2-q}}/\sqrt{m}$. In particular, if we choose m to be the smallest integer such that $m \geq 4\epsilon^{2q/(q-2)}$. Then, $\|w_i - w_i'\|_2 \leq \epsilon/2$.

We define $e_i = v_i + w_i'$, we have $\|u_i - e_i\|_2 \leq \epsilon/2$, and then we can show that

$$\|e_i\|_0 = \|v_i\|_0 + \|w_i'\|_0 \leq |L_i| + m \leq 5\epsilon^{2q/(q-2)} + 1.$$

(ii) Let $f_\theta^* = \sum_{j=1}^{M_n} c_j f_j = \arg\inf_{f_\theta \in \mathcal{F}_q(t_n)} \|f_\theta - f_0\|^2$ be a best approximation of f_0 over the class $\mathcal{F}_q(t_n)$. For any $1 \leq m \leq M_n$, let $L^* = \{j : |c_j| > t_n m^{-1/q}\}$. Because $\sum_{j=1}^{M_n} |c_j|^q \leq t_n^q$, we have $|L^*| t_n^q / m < \sum |c_j|^q \leq t_n^q$. So, $|L^*| < m$. Also,

$$D := \sum_{j \notin L^*} |c_j| \leq \sum_{j \notin L^*} |c_j|^q [t_n (1/m)^{1/q}]^{1-q} = \sum_{j \notin L^*} |c_j|^q t_n^{1-q} (1/m)^{(1-q)/q} \leq t_n m^{1-1/q}.$$

Define $v^* = \sum_{j\in L^*} c_j f_j$ and $w^* = \sum_{j\notin L^*} c_j f_j$. We have $w^* \in \mathcal{F}_1(D)$. Define a random function U so that $\mathbb{P}(U = D\text{sign}(c_j)f_j) = |c_j|/D$, $j \notin L^*$. Thus, $\mathbb{E}U = w^*$, where $\mathbb{E}$ denotes expectation with respect to the randomness $\mathbb{P}$ (just introduced). Also, $\|U\| \leq D \sup_{1\leq j\leq M_n} \|f_j\| \leq D$. Let $U_1, U_2, ..., U_m$ be i.i.d. copies of U, then $\forall \mathbf{x} \in \mathcal{X}$,

$$\mathbb{E}\left(f_0(\mathbf{x}) - v^*(\mathbf{x}) - \frac{1}{m}\sum_{i=1}^m U_i(\mathbf{x})\right)^2 = \left(f_\theta^*(\mathbf{x}) - f_0(\mathbf{x})\right)^2 + \frac{1}{m}\mathbb{V}ar\left(U(\mathbf{x})\right).$$

Together with Fubini,

$$\mathbb{E}\left\|f_0 - v^* - \frac{1}{m}\sum_{i=1}^m U_i\right\|^2 \leq \|f_\theta^* - f_0\|^2 + \frac{1}{m}E\|U\|^2 \leq \|f_\theta^* - f_0\|^2 + t_n^2 m^{1-2/q}.$$

In particular, there exists a realization of $v^* + \frac{1}{m}\sum_{i=1}^m U_i$, denoted by f_{θ^m}, such that $\|f_{\theta^m} - f_0\|^2 \leq \|f_\theta^* - f_0\|^2 + t_n^2 m^{1-2/q}$. Note that $\|\theta^m\|_1 \leq t_n$ and $\|\theta^m\|_0 \leq 2m-1$. If we consider $\widetilde{m} = \lfloor (m+1)/2 \rfloor$ instead, we have $2\widetilde{m} - 1 \leq m$ and $\widetilde{m} \geq m/2$. The conclusion then follows. This completes the proof of the theorem.

Appendix B. An Insight from the Sparse Approximation Bound Based on Classical Model Selection Theory

Consider general M_n, t_n and $0 < q \leq 1$. With the approximation error bound in Theorem 13, classical model selection theories can provide key insight on what to expect regarding the minimax rate of convergence for estimating a function in the $\ell_{q,t_n}^{M_n}$-hull.

Suppose J_m is the best subset model of size m in terms of having the smallest L_2 approximation error to f_0. Then, the estimator based on J_m is expected to have the risk (under some squared error loss) of order

$$2^{2/q} t_n^2 m^{1-2/q} + \frac{\sigma^2 m}{n}.$$

Minimizing this bound over m, we get the best choice (in order) in the range $1 \leq m \leq M_n \wedge n$:

$$m^* = m^*(q, t_n) = \left\lceil 2\left(nt_n^2\tau\right)^{q/2}\right\rceil \wedge M_n \wedge n,$$

where $\tau = \sigma^{-2}$ is the precision parameter. When $q = 0$ with $t_n = k_n$, m^* should be taken to be $k_n \wedge n$. It is the *ideal model size* (in order) under the ℓ_q-constraint because it provides the best possible trade-off between the approximation error and estimation error when $1 \leq m \leq M_n \wedge n$. The calculation of balancing the approximation error and the estimation error is well-known to lead to the minimax rate of convergence for general full approximation sets of functions with pre-determined order of the terms in an approximation system (see section 4 of Yang and Barron 1999). However, when the terms are not pre-ordered, there are many models of the same size m^*, and one must pay a price for dealing with exponentially many or more models (see, e.g., section 5 of Yang and Barron 1999). The classical model selection theory that deals with searching over a large number of models tells us that the price of searching over $\binom{M_n}{m^*}$ many models is the addition of the term

$\log \binom{M_n}{m^*}/n$ (e.g., Barron and Cover 1991; Yang and Barron 1998; Barron et al. 1999; Yang 1999; Baraud 2000; Birgé and Massart 2001; Baraud 2002; Massart 2007). That is, the risk (under squared error type of loss) of the estimator based on subset selection with a model descriptive complexity term of order $\log \binom{M_n}{m}$ added to the AIC-type of criteria is typically upper bounded in order by the smallest value of

$$\text{(squared) approximation error}_m + \frac{\sigma^2 m}{n} + \frac{\sigma^2 \log \binom{M_n}{m}}{n}$$

over all the subset models, which is called the index of the resolvability of the function to be estimated. Note that $\frac{m}{n} + \frac{\log \binom{M_n}{m}}{n}$ is uniformly of order $m\left(1 + \log\left(\frac{M_n}{m}\right)\right)/n$ over $0 \le m \le M_n$. Evaluating the above bound at m^* in our context yields a quite sensible rate of convergence. Note also that $\log \binom{M_n}{m^*}/n$ (price of searching) is of a higher order than $\frac{m^*}{n}$ (price of estimation) when $m^* \le M_n/2$. Define

$$SER(m) = 1 + \log\left(\frac{M_n}{m}\right) \asymp \frac{m + \log \binom{M_n}{m}}{m}, \qquad 1 \le m \le M_n,$$

to be the ratio of the price with searching to that without searching (i.e., only the price of estimation of the parameters in the model). Here "$\asymp$" means of the same order as $n \to \infty$. Observe that reducing m^* slightly will reduce the order of searching price $\frac{m^* SER(m^*)}{n}$ (since $x(1 + \log(M_n/x))$ is an increasing function for $0 < x < M_n$) and increase the order of the squared bias plus variance (i.e., $2^{2/q} t_n^2 m^{1-2/q} + \frac{\sigma^2 m}{n}$). The best choice will typically make the approximation error $2^{2/q} t_n^2 m^{1-2/q}$ of the same order as $\frac{m(1+\log \frac{M_n}{m})}{n}$ (as also pointed out in Raskutti et al. 2012 from a different analysis). Define

$$m_* = m_*(q, t_n) = \begin{cases} m^* & \text{if } m^* = M_n \wedge n, \\ \left\lceil \frac{m^*}{\left(1+\log \frac{M_n}{m^*}\right)^{q/2}} \right\rceil = \left\lceil \frac{m^*}{SER(m^*)^{q/2}} \right\rceil & \text{otherwise.} \end{cases}$$

We call this the *effective model size* (in order) under the ℓ_q-constraint because evaluating the index of resolvability expression from our general oracle inequality (see Proposition 15 in the Appendix) at the best model of this size gives the minimax rate of convergence, as shown in this work. When $m^* = n$, the minimax risk is of order 1 (or higher sometimes) and thus does not converge. Note that the down-sizing factor $SER(m^*)^{q/2}$ from m^* to m_* depends on q: it becomes more severe as q increases; when $q = 1$, the down-sizing factor reaches the order $\left(1 + \log\left(\frac{M_n}{m^*}\right)\right)^{1/2}$. Since the risk of the ideal model and that by a good model selection rule differ only by a factor of $\log(M_n/m^*)$, as long as M_n is not too large, the price of searching over many models of the same size is small, which is a fact well known in the model selection literature (see, e.g., Yang and Barron 1998, section III.D).

For $q = 0$, under the assumption of at most $k_n \le M_n \wedge n$ nonzero terms in the linear representation of the true regression function, the risk bound immediately yields the rate $\left(1 + \log \binom{M_n}{k_n}\right)/n \asymp \frac{k_n\left(1+\log \frac{M_n}{k_n}\right)}{n}$. Thus, from all above, we expect that $\frac{m_* SER(m_*)}{n} \wedge 1$ is the unifying optimal rate of convergence for regression under the ℓ_q-constraint for $0 \le q \le 1$.

Appendix C.

In this appendix, the theorems in Sections 3-4 are proved, with additional results given as preparations.

C.1 Some General Oracle Inequalities

The proofs of the upper bound results rely on some oracle inequalities, which may be of interest in other applications. Consider the setting in Section 3.2 of the main paper.

Proposition 15 *Suppose $A_{\mathbf{E-G}}$ holds for the **E-G** strategy, respectively. Then, the following oracle inequalities hold for the estimator $\hat{f}_{F_n}$.*

*(i) For **T-C** and **T-Y** strategies,*

$$\begin{aligned} & R(\hat{f}_{F_n}; f_0; n) \\ \leq\ & c_0 \inf_{1\leq m\leq M_n\wedge n}\left(c_1 \inf_{J_m} d^2(f_0;\mathcal{F}_{J_m}) + c_2\frac{m}{n_1} + c_3\frac{1+\log\binom{M_n}{m}+\log(M_n\wedge n)-\log(1-p_0)}{n-n_1}\right) \\ & \wedge c_0\left(\|f_0\|^2 + c_3\frac{1-\log p_0}{n-n_1}\right), \end{aligned}$$

*where $c_0=1$, $c_1=c_2=C_{L,\sigma}$, $c_3=\frac{2}{\lambda_C}$ for the **T-C** strategy; $c_0=C_Y$, $c_1=c_2=C_{L,\sigma}$, $c_3=\sigma^2$ for the **T-Y** strategy.*

*(ii) For **AC-C** and **AC-Y** strategies,*

$$\begin{aligned} & R(\hat{f}_{F_n}; f_0; n) \\ \leq\ & c_0 \inf_{1\leq m\leq M_n\wedge n}\left(R(f_0,m,n) + c_2\frac{m}{n_1} + c_3\frac{1+\log\binom{M_n}{m}+\log(M_n\wedge n)-\log(1-p_0)}{n-n_1}\right) \\ & \wedge c_0\left(\|f_0\|^2 + c_3\frac{1-\log p_0}{n-n_1}\right), \end{aligned}$$

where

$$R(f_0,m,n) = c_1 \inf_{J_m}\inf_{s\geq 1}\left(d^2(f_0;\mathcal{F}^L_{J_m,s}) + 2c_3\frac{\log(1+s)}{n-n_1}\right),$$

*and $c_0=c_1=1$, $c_2=8c(\sigma^2+5L^2)$, $c_3=\frac{2}{\lambda_C}$ for the **AC-C** strategy; $c_0=C_Y$, $c_1=1$, $c_2=8c(\sigma^2+5L^2)$, $c_3=\sigma^2$ for the **AC-Y** strategy.*

From the proposition above, the risk $R(\hat{f}_{F_n}; f_0; n)$ is upper bounded by a multiple of the best trade-off of the different sources of errors (approximation error, estimation error due to estimating the linear coefficients, and error associated with searching over many models of the same dimension). For a model J, let $IR(f_0; J)$ generically denote the sum of these three sources of errors. Then, the best trade-off is $IR(f_0) = \inf_J IR(f_0; J)$, where the infimum is over all the candidate models. Following the terminology in Barron and Cover (1991), $IR(f_0)$ is the so-called index of resolvability of the true function f_0 by the estimation method over the candidate models. We call $IR(f_0; J)$ the index of resolvability at model J. The utility of the index of resolvability is that for f_0 with a given characteristic,

an evaluation of the index of resolvability at the best J immediately tells us how well the unknown function is "resolved" by the estimation method at the current sample size. Thus, accurate index of resolvability bounds often readily show minimax optimal performance of the model selection based estimator.

Proof of Proposition 15.

(i) For the **T-C** strategy,

$$R(\hat{f}_{F_n}; f_0; n) \le \inf_{1\le m\le M_n\wedge n}\left\{C_{L,\sigma}\left(\inf_{J_m} d^2(f_0;\mathcal{F}_{J_m})+\frac{m}{n_1}\right)+ \frac{2}{\lambda_C}\left(\frac{\log(M_n\wedge n)+\log\binom{M_n}{m}-\log(1-p_0)}{n-n_1}\right)\right\}\wedge\left\{\|f_0\|^2-\frac{2}{\lambda_C}\frac{\log p_0}{n-n_1}\right\}.$$

For the **T-Y** strategy,

$$R(\hat{f}_{F_n}; f_0; n) \le C_Y \inf_{1\le m\le M_n\wedge n}\left\{C_{L,\sigma}\inf_{J_m} d^2(f_0;\mathcal{F}_{J_m})+C_{L,\sigma}\frac{m}{n_1}+ \sigma^2\left(\frac{1+\log(M_n\wedge n)+\log\binom{M_n}{m}-\log(1-p_0)}{n-n_1}\right)\right\}\wedge C_Y\left\{\|f_0\|^2+\sigma^2\frac{1-\log p_0}{n-n_1}\right\}.$$

(ii) For the **AC-C** strategy,

$$\begin{aligned} & R(\hat{f}_{F_n}; f_0; n)\\ \le\ & \inf_{1\le m\le M_n\wedge n}\left\{\inf_{J_m}\inf_{s\ge1}\left(d^2(f_0;\mathcal{F}^L_{J_m,s})+c(2\sigma'+H)^2\frac{m}{n_1}+\frac{2}{\lambda_C}\times \left(\frac{\log(M_n\wedge n)+\log\binom{M_n}{m}-\log(1-p_0)}{n-n_1}+\frac{2\log(1+s)}{n-n_1}\right)\right)\right\}\wedge\left\{\|f_0\|^2-\frac{2}{\lambda_C}\frac{\log p_0}{n-n_1}\right\}\\ \le\ & \inf_{1\le m\le M_n\wedge n}\left\{\inf_{J_m}\inf_{s\ge1}\left(d^2(f_0;\mathcal{F}^L_{J_m,s})+8c(\sigma^2+5L^2)\frac{m}{n_1}+\frac{2}{\lambda_C}\times \left(\frac{\log(M_n\wedge n)+\log\binom{M_n}{m}-\log(1-p_0)}{n-n_1}+\frac{2\log(1+s)}{n-n_1}\right)\right)\right\}\wedge\left\{\|f_0\|^2-\frac{2}{\lambda_C}\frac{\log p_0}{n-n_1}\right\}. \end{aligned}$$

For the **AC-Y** strategy,

$$\begin{aligned} & R(\hat{f}_{F_n}; f_0; n)\\ \le\ & C_Y\inf_{1\le m\le M_n\wedge n}\left\{\inf_{J_m}\inf_{s\ge1}\left(d^2(f_0;\mathcal{F}^L_{J_m,s})+c(2\sigma'+H)^2\frac{m}{n_1}+\sigma^2\left(\frac{1+\log(M_n\wedge n)}{n-n_1}+ +\frac{\log\binom{M_n}{m}}{n-n_1}+\frac{-\log(1-p_0)+2\log(1+s)}{n-n_1}\right)\right)\right\}\wedge C_Y\left\{\|f_0\|^2+\sigma^2\frac{1-\log p_0}{n-n_1}\right\}\\ \le\ & C_Y\inf_{1\le m\le M_n\wedge n}\left\{\inf_{J_m}\inf_{s\ge1}\left(d^2(f_0;\mathcal{F}^L_{J_m,s})+8c(\sigma^2+5L^2)\frac{m}{n_1}+\sigma^2\left(\frac{1+\log(M_n\wedge n)}{n-n_1}+ +\frac{\log\binom{M_n}{m}}{n-n_1}+\frac{-\log(1-p_0)+2\log(1+s)}{n-n_1}\right)\right)\right\}\wedge C_Y\left\{\|f_0\|^2+\sigma^2\frac{1-\log p_0}{n-n_1}\right\}. \end{aligned}$$

This completes the proof of Proposition 15.

C.2 Proof of Theorem 5

To derive the upper bounds, we only need to examine the index of resolvability for each strategy. The nature of the constants in Theorem 5 follows from Proposition 15.

(i) For **T-** strategies, according to Theorem 13 and the general oracle inequalities in Proposition 15, for each $1 \le m \le M_n \wedge n$, there exists a subset J_m and the best $f_{\theta^m} \in \mathcal{F}_{J_m}$ such that

$$\begin{aligned} R(\hat{f}_{F_n}; f_0; n) &\le c_0 \left(c_1 \|f_{\theta^m} - f_0\|^2 + 2c_2 \frac{m}{n} + 2c_3 \frac{1 + \log \binom{M_n}{m} + \log(M_n \wedge n) - \log(1-p_0)}{n} \right) \\ &\quad \wedge c_0 \left(\|f_0\|^2 + 2c_3 \frac{1 - \log p_0}{n} \right). \end{aligned}$$

Under the assumption that f_0 has sup-norm bounded, the index of resolvability evaluated at the null model $f_\theta \equiv 0$ leads to the fact that the risk is always bounded above by $C_0 \left(\|f_0\|^2 + \frac{C_2 \sigma^2}{n} \right)$ for some constant $C_0, C_2 > 0$.

For $\mathcal{F} = \mathcal{F}_q(t_n)$, and when $m_* = m^* = M_n < n$, evaluating the index of resolvability at the full model J_{M_n}, we get

$$R(\hat{f}_{F_n}; f_0; n) \le c_0 c_1 d^2(f_0; \mathcal{F}_q(t_n)) + \frac{CM_n}{n} \quad \text{with} \quad \frac{CM_n}{n} = \frac{Cm_* \left(1 + \log\left(\frac{M_n}{m_*}\right)\right)}{n}.$$

Thus, the upper bound is proved when $m_* = m^* = M_n$.

For $\mathcal{F} = \mathcal{F}_q(t_n)$, and when $m_* = m^* = n < M_n$, then clearly $m_* \left(1 + \log\left(\frac{M_n}{m_*}\right)\right)/n$ is larger than 1, and then the risk bound given in the theorem in this case holds.

For $\mathcal{F} = \mathcal{F}_q(t_n)$, and when $1 \le m_* \le m^* < M_n \wedge n$, for $1 \le m < M_n$, and from Theorem 13, we have

$$\begin{aligned} R(\hat{f}_{F_n}; f_0; n) &\le c_0 \Big(c_1 d^2(f_0; \mathcal{F}_q(t_n)) + c_1 2^{2/q-1} t_n^2 m^{1-2/q} + 2c_2 \frac{m}{n} \\ &\quad + 2c_3 \frac{1 + \log \binom{M_n}{m} + \log(M_n \wedge n)}{n} - 2c_3 \frac{\log(1-p_0)}{n} \Big). \end{aligned}$$

Since $\log \binom{M_n}{m} \le m \log\left(\frac{eM_n}{m}\right) = m\left(1 + \log \frac{M_n}{m}\right)$, then

$$\begin{aligned} R(\hat{f}_{F_n}; f_0; n) &\le c_0 c_1 d^2(f_0; \mathcal{F}_q(t_n)) + C \left(2^{2/q} t_n^2 m^{1-2/q} + \frac{m\left(1 + \log \frac{M_n}{m}\right)}{n} + \frac{\log(M_n \wedge n)}{n} \right) \\ &\le c_0 c_1 d^2(f_0; \mathcal{F}_q(t_n)) + C' \left(2^{2/q} t_n^2 m^{1-2/q} + \frac{m\left(1 + \log \frac{M_n}{m}\right)}{n} \right), \end{aligned}$$

where C and C' are constants that do not depend on n, q, t_n, and M_n (but may depend on σ^2, p_0 and L). Choosing $m = m_*$, we have

$$2^{2/q} t_n^2 m^{1-2/q} + \frac{m\left(1+\log\frac{M_n}{m}\right)}{n} \leq C'' \frac{m_*\left(1+\log\left(\frac{M_n}{m_*}\right)\right)}{n},$$

where C'' is an absolute constant. The upper bound for this case then follows.

For $\mathcal{F} = \mathcal{F}_0(k_n)$, by evaluating the index of resolvability from Proposition 15 at $m = k_n$, the upper bound immediately follows.

For $\mathcal{F} = \mathcal{F}_q(t_n) \cap \mathcal{F}_0(k_n)$, both ℓ_q- and ℓ_0-constraints are imposed on the coefficients, the upper bound will go with the faster rate from the tighter constraint. The result follows.

(ii) For **AC-** strategies, three constraints $\|\theta\|_1 \leq s$ ($s > 0$), $\|\theta\|_q \leq t_n$ ($0 \leq q \leq 1$, $t_n > 0$) and $\|f_\theta\|_\infty \leq L$ are imposed on the coefficients. Notice that $\|\theta\|_1 \leq \|\theta\|_q$ when $0 < q \leq 1$, then the ℓ_1-constraint is satisfied by default as long as $s \geq t_n$ and $\|\theta\|_q \leq t_n$ with $0 < q \leq 1$. Using similar arguments as used for **T-**strategies, the desired upper bounds can be easily derived. This completes the proof of Theorem 5.

C.3 Global Metric Entropy and Local Metric Entropy

The derivations of the lower bounds in the main paper require some preparations.

Consider estimating a regression function f_0 in a general function class $\mathcal{F}$ based on i.i.d. observations $(\mathbf{X}_i, Y_i)_{i=1}^n$ from the model

$$Y = f_0(\mathbf{X}) + \sigma \cdot \varepsilon, \tag{5}$$

where $\sigma > 0$ and ε follows a standard normal distribution and is independent of $\mathbf{X}$.

Given $\mathcal{F}$, we say $G \subset \mathcal{F}$ is an *ϵ-packing set* in $\mathcal{F}$ ($\epsilon > 0$) if any two functions in G are more than ϵ apart in the L_2 distance. Let $0 < \alpha < 1$ be a constant.

Definition 1: (*Global metric entropy*) The packing ϵ-entropy of $\mathcal{F}$ is the logarithm of the largest ϵ-packing set in $\mathcal{F}$. The packing ϵ-entropy of $\mathcal{F}$ is denoted by $M(\epsilon)$.

Definition 2: (*Local metric entropy*) The α-local ϵ-entropy at $f \in \mathcal{F}$ is the logarithm of the largest $(\alpha\epsilon)$-packing set in $\mathcal{B}(f, \epsilon) = \{f' \in \mathcal{F} : \| f' - f \| \leq \epsilon\}$. The α-local ϵ-entropy at f is denoted by $M_\alpha(\epsilon \mid f)$. The α-local ϵ-entropy of $\mathcal{F}$ is defined as $M_\alpha^{\text{loc}}(\epsilon) = \max_{f \in \mathcal{F}} M_\alpha(\epsilon \mid f)$.

Suppose that $M_\alpha^{\text{loc}}(\epsilon)$ is lower bounded by $\underline{M}_\alpha^{\text{loc}}(\epsilon)$ (a continuous function), and assume that $M(\epsilon)$ is upper bounded by $\overline{M}(\epsilon)$ and lower bounded by $\underline{M}(\epsilon)$ (with $\overline{M}(\epsilon)$ and $\underline{M}(\epsilon)$ both being continuous).

Suppose there exist ϵ_n, $\bar{\epsilon}_n$, and $\underline{\epsilon}_n$ such that

$$\underline{M}_\alpha^{\text{loc}}(\sigma\epsilon_n) \geq n\epsilon_n^2 + 2\log 2, \tag{6}$$

$$\overline{M}(\sqrt{2}\sigma\bar{\epsilon}_n) = n\bar{\epsilon}_n^2, \tag{7}$$

$$\underline{M}(\sigma\underline{\epsilon}_n) = 4n\bar{\epsilon}_n^2 + 2\log 2. \tag{8}$$

Proposition 16 *(Yang and Barron 1999) The minimax risk for estimating f_0 from model (5) in the function class $\mathcal{F}$ is lower-bounded as the following*

$$\inf_{\hat{f}} \sup_{f_0 \in \mathcal{F}} E\|\hat{f} - f_0\|^2 \geq \frac{\alpha^2\sigma^2\epsilon_n^2}{8},$$

$$\inf_{\hat{f}} \sup_{f_0 \in \mathcal{F}} E\|\hat{f} - f_0\|^2 \geq \frac{\sigma^2 \underline{\epsilon}_n^2}{8}.$$

Let $\underline{\mathcal{F}}$ be a subset of $\mathcal{F}$. If a packing set in $\mathcal{F}$ of size at least $\exp(\underline{M}_\alpha^{loc}(\sigma\epsilon_n))$ or $\exp(\underline{M}(\sigma\underline{\epsilon}_n))$ is actually contained in $\underline{\mathcal{F}}$, then $\inf_{\hat{f}} \sup_{f_0 \in \underline{\mathcal{F}}} E\|\hat{f} - f_0\|^2$ is lower bounded by $\frac{\alpha^2\sigma^2\epsilon_n^2}{8}$ or $\frac{\sigma^2\underline{\epsilon}_n^2}{8}$, respectively.

Proof. The result is essentially given in Yang and Barron (1999), but not in the concrete forms. The second lower bound is given in Yang (2004). We briefly derive the first one.

Let N be an $(\alpha\epsilon_n)$-packing set in $\mathcal{B}(f, \sigma\epsilon_n) = \{f' \in \mathcal{F} : \| f' - f \| \leq \sigma\epsilon_n\}$. Let Θ denote a uniform distribution on N. Then, by applying the upper bound on mutual information displayed in the middle of page 1571 of Yang and Barron (1999), together with the specific form of the K-L divergence between the Gaussian regression densities (see the first paragraph of the proof of Theorem 6 of Yang and Barron 1999 on page 1583), the mutual information between Θ and the observations $(\mathbf{X}_i, Y_i)_{i=1}^n$ is upper bounded by $\frac{n}{2}\epsilon_n^2$, and an application of Fano's inequality (see the proof of Theorem 1 in Yang and Barron 1999, particularly Equation 1 on page 1571) to the regression problem gives the minimax lower bound

$$\frac{\alpha^2\sigma^2\epsilon_n^2}{4}\left(1 - \frac{I\left(\Theta; (\mathbf{X}_i, Y_i)_{i=1}^n\right) + \log 2}{\log|N|}\right),$$

where $|N|$ denotes the size of N. By our way of defining ϵ_n, the conclusion of the first lower bound follows.

For the last statement, we prove for the global entropy case and the argument for the local entropy case similarly follows. Observe that the upper bound on $I\left(\Theta; (\mathbf{X}_i, Y_i)_{i=1}^n\right)$ by $\log(|G|) + n\bar{\epsilon}_n^2$, where G is an $\bar{\epsilon}_n$-net of $\mathcal{F}$ under the square root of the Kullback-Leibler divergence (see Yang and Barron 1999, page 1571), continues to be an upper bound on $I\left(\underline{\Theta}; (\mathbf{X}_i, Y_i)_{i=1}^n\right)$, where $\underline{\Theta}$ is the uniform distribution on a packing set in $\underline{\mathcal{F}}$. Therefore, by the derivation of Theorem 1 in Yang and Barron (1999), the same lower bound holds for $\underline{\mathcal{F}}$ as well. This completes the proof.

C.4 Proof of Theorem 8

Assume $f_0 \in \mathcal{F}$ in each case of $\mathcal{F}$ so that $d^2(f_0; \mathcal{F}) = 0$. Without loss of generality, assume $\sigma = 1$.

(i) We first derive the lower bounds without L_2 or L_∞ upper bound assumption on f_0. To prove case 1 (i.e., $\mathcal{F} = \mathcal{F}_q(t_n)$), it is enough to show that

$$\inf_{\hat{f}} \sup_{f_0 \in \mathcal{F}_q(t_n)} E\|\hat{f} - f_0\|^2 \geq C_q \begin{cases} \frac{M_n}{n} & \text{if } \widetilde{m}^* = M_n, \\ t_n^q \left(\frac{1+\log\frac{M_n}{(nt_n^2)^{q/2}}}{n}\right)^{1-q/2} & \text{if } 1 < \widetilde{m}_* \leq \widetilde{m}^* < M_n, \\ t_n^2 & \text{if } \widetilde{m}_* = 1, \end{cases}$$

in light of the fact that, by definition, when $\widetilde{m}^* = M_n$, $\widetilde{m}_* = M_n$ and when $1 < \widetilde{m}_* \leq \widetilde{m}^* < M_n$, we have $\frac{\widetilde{m}_*(1+\log\frac{M_n}{\widetilde{m}_*})}{n}$ is upper and lower bounded by multiples (depending only on q)

of $t_n^q \left(\frac{1+\log \frac{M_n}{(nt_n^2)^{q/2}}}{n} \right)^{1-q/2}$. Note that $\widetilde{m}^*$ and $\widetilde{m}_*$ are defined as m^* and m_* except that no ceiling of n is imposed there.

Given that the basis functions are orthonormal, the L_2 distance on $\mathcal{F}_q(t_n)$ is the same as the ℓ_2 distance on the coefficients in $B_q(t_n; M_n) = \{\theta : \|\theta\|_q \le t_n\}$. Thus, the entropy of $\mathcal{F}_q(t_n)$ under the L_2 distance is the same as that of $B_q(t_n; M_n)$ under the ℓ_2 distance.

When $\widetilde{m}^* = M_n$, we use the lower bound tool in terms of local metric entropy. Given the ℓ_q-ℓ_2-relationship $\|\theta\|_q \le {M_n}^{1/q-1/2}\|\theta\|_2$ for $0 < q \le 2$, for $\epsilon \le \sqrt{M_n/n}$, taking $f_0^* \equiv 0$, we have

$$\mathcal{B}(f_0^*; \epsilon) = \{f_\theta : \|f_\theta - f_0^*\| \le \epsilon, \|\theta\|_q \le t_n\} = \{f_\theta : \|\theta\|_2 \le \epsilon, \|\theta\|_q \le t_n\} = \{f_\theta : \|\theta\|_2 \le \epsilon\},$$

where the last equality holds because when $\epsilon \le \sqrt{M_n/n}$, for $\|\theta\|_2 \le \epsilon$, $\|\theta\|_q \le t_n$ is always satisfied. Consequently, for $\epsilon \le \sqrt{M_n/n}$, the $(\epsilon/2)$-packing of $\mathcal{B}(f_0^*; \epsilon)$ under the L_2 distance is equivalent to the $(\epsilon/2)$-packing of $B_\epsilon = \{\theta : \|\theta\|_2 \le \epsilon\}$ under the ℓ_2 distance. Note that the size of the maximum packing set is at least the ratio of volumes of the balls B_ϵ and $B_{\epsilon/2}$, which is 2^{M_n}. Thus, the local entropy $M_{1/2}^{\mathrm{loc}}(\epsilon)$ of $\mathcal{F}_q(t)$ under the L_2 distance is at least $\underline{M}_{1/2}^{\mathrm{loc}}(\epsilon) = M_n \log 2$ for $\epsilon \le \sqrt{M_n/n}$. The minimax lower bound for the case of $\widetilde{m}^* = M_n$ then directly follows from Proposition 16.

When $1 < \widetilde{m}_* \le \widetilde{m}^* < M_n$, the use of global entropy is handy. Applying the minimax lower bound in terms of global entropy in Proposition 16, with the metric entropy order for larger ϵ (which is tight in our case of orthonormal functions in the dictionary) from Theorem 13 the minimax lower rate is readily obtained. Indeed, for the class $\mathcal{F}_q(t_n)$, with $\epsilon > t_n M_n^{\frac{1}{2}-\frac{1}{q}}$, there are constants c' and $\underline{c}'$ (depending only on q) such that

$$\underline{c}' \left(t_n \epsilon^{-1}\right)^{\frac{2q}{2-q}} \log(1 + M_n^{\frac{1}{q}-\frac{1}{2}} t_n^{-1}\epsilon) \le \underline{M}(\epsilon) \le \overline{M}(\epsilon) \le c' \left(t_n \epsilon^{-1}\right)^{\frac{2q}{2-q}} \log(1 + M_n^{\frac{1}{q}-\frac{1}{2}} t_n^{-1}\epsilon).$$

Thus, we see that $\underline{\epsilon}_n$ determined by (8.4) is lower bounded by $c''' t_n^{\frac{q}{2}} \left((1 + \log \frac{M_n}{(nt_n^2)^{q/2}})/n \right)^{\frac{1}{2}-\frac{q}{4}}$, where c''' is a constant depending only on q.

When $\widetilde{m}_* = 1$, note that with $f_0^* = 0$ and $\epsilon \le t_n$,

$$\mathcal{B}(f_0^*; \epsilon) = \{f_\theta : \|\theta\|_2 \le \epsilon, \|\theta\|_q \le t_n\} \supset \{f_\theta : \|\theta\|_q \le \epsilon\}.$$

Observe that the $(\epsilon/2)$-packing of $\{f_\theta : \|\theta\|_q \le \epsilon\}$ under the L_2 distance is equivalent to the $(1/2)$-packing of $\{f_\theta : \|\theta\|_q \le 1\}$ under the same distance. Thus, by applying Theorem 13 with $t_n = 1$ and $\epsilon = 1/2$, we know that the $(\epsilon/2)$-packing entropy of $\mathcal{B}(f_0^*; \epsilon)$ is lower bounded by $\underline{c}'' \log(1 + \frac{1}{2} M_n^{1/q-1/2})$ for some constant $\underline{c}''$ depending only on q, which is at least a multiple of nt_n^2 when $\widetilde{m}^* \le \left(1 + \log \frac{M_n}{\widetilde{m}^*}\right)^{q/2}$. Therefore we can choose $0 < \delta < 1$ small enough (depending only on q) such that

$$\underline{c}'' \log(1 + \frac{1}{2} M_n^{1/q-1/2}) \ge n\delta^2 t_n^2 + 2 \log 2.$$

The conclusion then follows from applying the first lower bound of Proposition 16.

To prove case 2 (i.e., $\mathcal{F} = \mathcal{F}_0(k_n)$), noticing that for $M_n/2 \le k_n \le M_n$, we have $(1+\log 2)/2M_n \le k_n\left(1+\log\frac{M_n}{k_n}\right) \le M_n$, together with the monotonicity of the minimax risk in the function class, it suffices to show the lower bound for $k_n \le M_n/2$. Let $B_{k_n}(\epsilon) = \{\theta : \|\theta\|_2 \le \epsilon, \|\theta\|_0 \le k_n\}$. As in case 1, we only need to understand the local entropy of the set $B_{k_n}(\epsilon)$ for the critical ϵ that gives the claimed lower rate. Let $\eta = \epsilon/\sqrt{k_n}$. Then $B_{k_n}(\epsilon)$ contains the set $D_{k_n}(\eta)$, where

$$D_k(\eta) = \{\theta = \eta I : I \in \{1,0,-1\}^{M_n}, \|I\|_0 \le k\}.$$

Clearly $\|\eta I_1 - \eta I_2\|_2 \ge \eta\left(d_{HM}(I_1,I_2)\right)^{1/2}$, where $d_{HM}(I_1,I_2)$ is the Hamming distance between $I_1, I_2 \in \{1,0,-1\}^{M_n}$. From Lemma 4 of Raskutti et al. (2012) (the result there actually also holds when requiring the pairwise Hamming distance to be strictly larger than $k/2$; see also Lemma 4 of Birgé and Massart 2001 or the derivation of a metric entropy lower bound in Kühn 2001), there exists a subset of $\{I : I \in \{1,0,-1\}^{M_n}, \|I\|_0 \le k\}$ with more than $\exp\left(\frac{k}{2}\log\frac{2(M_n-k)}{k}\right)$ points that have pairwise Hamming distance larger than $k/2$. Consequently, we know the local entropy $M^{loc}_{1/\sqrt{2}}(\epsilon)$ of $\mathcal{F}_0(k_n)$ is lower bounded by $\frac{k_n}{2}\log\frac{2(M_n-k_n)}{k_n}$. The result follows.

To prove case 3 (i.e., $\mathcal{F}_q(t_n) \cap \mathcal{F}_0(k_n)$), for the larger k_n case, from the proof of case 1, we have used fewer than k_n nonzero components to derive the minimax lower bound there. Thus, the extra ℓ_0-constraint does not change the problem in terms of lower bound. For the smaller k_n case, note that for θ with $\|\theta\|_0 \le k_n$, $\|\theta\|_q \le k_n^{1/q-1/2}\|\theta\|_2 \le k_n^{1/q-1/2}\cdot\sqrt{Ck_n\left(1+\log\frac{M_n}{k_n}\right)/n}$ for θ with $\|\theta\|_2 \le \sqrt{Ck_n\left(1+\log\frac{M_n}{k_n}\right)/n}$ for some constant $C > 0$. Therefore the ℓ_q-constraint is automatically satisfied when $\|\theta\|_2$ is no larger than the critical order $\sqrt{k_n\left(1+\log\frac{M_n}{k_n}\right)/n}$, which is sufficient for the lower bound via local entropy techniques. The conclusion follows.

(ii) Now, we turn to the lower bounds under the L_2-norm condition. When the regression function f_0 satisfies the boundedness condition in L_2-norm, the estimation risk is obviously upper bounded by L^2 by taking the trivial estimator $\hat{f} = 0$. In all of the lower boundings in (i) through local entropy argument, if the critical radius ϵ is of order 1 or lower, the extra condition $\|f_0\| \le L$ does not affect the validity of the lower bound. Otherwise, we take ϵ to be L. Then, since the local entropy stays the same, it directly follows from the first lower bound in Proposition 16 that L^2 is a lower order of the minimax risk. The only case remained is that of $\left(1+\log\frac{M_n}{m^*}\right)^{q/2} \le m^* < M_n$. If $t_n^q\left((1+\log\frac{M_n}{(nt^2)^{q/2}})/n\right)^{1-q/2}$ is upper bounded by a constant, from the proof of the lower bound of the metric entropy of the ℓ_q-ball in Kühn (2001), we know that the functions in the special packing set satisfy the L_2 bound. Indeed, consider $\{f_\theta : \theta \in D_{m_n}(\eta)\}$ with m_n being a multiple of $\left(nt_n^2/\left(1+\log\frac{M_n}{(nt_n^2)^{q/2}}\right)\right)^{q/2}$ and η being a (small enough) multiple of $\sqrt{(1+\log\frac{M_n}{(nt_n^2)^{q/2}})/n}$. Then these f_θ have $\|f_\theta\|$ upper bounded by a multiple of $t_n^q\left((1+\log\frac{M_n}{(nt_n^2)^{q/2}})/n\right)^{1-q/2}$ and the minimax lower bound follows from the last statement of Proposition 16. If $t_n^q\left((1+\log\frac{M_n}{(nt_n^2)^{q/2}})/n\right)^{1-q/2}$ is not upper bounded,

we reduce the packing radius to L (i.e., choose η so that $\eta\sqrt{m_n}$ is bounded by a multiple of L). Then the functions in the packing set satisfy the L_2 bound and furthermore, the number of points in the packing set is of a larger order than $nt_n^q\left((1+\log\frac{M_n}{(nt_n^2)^{q/2}})/n\right)^{1-q/2}$. Again, adding the L_2 condition on $f_0 \in \mathcal{F}_q(t)$ does not increase the mutual information bound in our application of Fano's inequality. We conclude that the minimax risk is lower bounded by a constant.

(iii) Finally, we prove the lower bounds under the sup-norm bound condition. For 1), under the direct sup-norm assumption, the lower bound is obvious. For the general M_n case 2), note that the functions f_θ's in the critical packing set satisfies that $\|\theta\|_2 \le \epsilon$ with ϵ being a multiple of $\sqrt{\frac{k_n\left(1+\log\frac{M_n}{k_n}\right)}{n}}$. Then together with $\|\theta\|_0 \le k_n$, we have $\|\theta\|_1 \le \sqrt{k_n}\|\theta\|_2$, which is bounded by assumption. The lower bound conclusion then follows from the last part of Proposition 16. To prove the results for the case $M_n/\left(1+\log\frac{M_n}{k_n}\right) \le bn$, as in Tsybakov (2003), we consider the special dictionary $F_n = \{f_i : 1 \le i \le M_n\}$ on $[0,1]$, where

$$f_i(\mathbf{x}) = \sqrt{M_n} I_{[\frac{i-1}{M_n}, \frac{i}{M_n})}(\mathbf{x}), \qquad i = 1, ..., M_n.$$

Clearly, these functions are orthonormal. By the last statement of Proposition 16, we only need to verify that the functions in the critical packing set in each case do have the sup-norm bound condition satisfied. Note that for any f_θ with $\theta \in D_{k_n}(\eta)$ (as defined earlier), we have $\|f_\theta\| \le \eta\sqrt{k_n}$ and $\|f_\theta\|_\infty \le \eta\sqrt{M_n}$. Thus, it suffices to show that the critical packing sets for the previous lower bounds without the sup-norm bound can be chosen with θ in $D_{k_n}(\eta)$ for some $\eta = O\left(M_n^{-1/2}\right)$. Consider η to be a (small enough) multiple of $\sqrt{\left(1+\log\frac{M_n}{k_n}\right)/n} = O\left(M_n^{-1/2}\right)$ (which holds under the assumption $\frac{M_n}{1+\log\frac{M_n}{k_n}} \le bn$). From the proof of part (ii) without constraint, we know that there is a subset of $D_{k_n}(\eta)$ that with more than $\exp(\frac{k_n}{2}\log\frac{2(M_n-k_n)}{k_n})$ points that are separated in ℓ_2 distance by at least $\sqrt{k_n\left(1+\log\frac{M_n}{k_n}\right)/n}$. This completes the proof.

C.5 Proof of Corollary 10

Since f_0 belongs to $\mathcal{F}_q^L(t_n; M_n)$, or $\mathcal{F}_0^L(k_n; M_n)$, or both, thus $d^2(f_0, \mathcal{F})$ is equal to zero for all cases (except for **AC-** strategies when $\mathcal{F} = \mathcal{F}_0^L(k_n; M_n)$, which we discuss later).

(i) For **T-** strategies and $\mathcal{F} = \mathcal{F}_q^L(t_n; M_n)$. For each $1 \le m \le M_n \wedge n$, according to the general oracle inequalities in the proof of Theorem 5, the adaptive estimator $\hat{f}_A$ has

$$\begin{aligned}\sup_{f_0\in\mathcal{F}} R(\hat{f}_A; f_0; n) \quad &\le \quad c_0\left(2c_2\frac{m}{n} + 2c_3\frac{1+\log\binom{M_n}{m} + \log(M_n\wedge n) - \log(1-p_0)}{n}\right)\\ &\quad \wedge c_0\left(\|f_0\|^2 - 2c_3\frac{\log p_0}{n}\right).\end{aligned}$$

When $m_* = m^* = M_n < n$, the full model J_{M_n} results in an upper bound of order M_n/n.

When $m_* = m^* = n < M_n$, we choose the null model and the upper bound is simply of order one.

When $1 < m_* \le m^* < M_n \wedge n$, the similar argument of Theorem 5 leads to an upper bound of order $1 \wedge \frac{m_*}{n}\left(1+\log\frac{M_n}{m_*}\right)$. Since $(nt_n^2)^{q/2}\left(1+\log\frac{M_n}{(nt_n^2)^{q/2}}\right)^{-q/2} \le m_* \le 4(nt_n^2)^{q/2}\left(1+\log\frac{M_n}{2(nt_n^2)^{q/2}}\right)^{-q/2}$, then the upper bound is further upper bounded by $c_q t_n^q \cdot \left(\frac{1+\log\frac{M_n}{(nt_n^2)^{q/2}}}{n}\right)^{1-q/2}$ for some constant c_q only depending on q.

When $m_* = 1$, the null model leads to an upper bound of order $\|f_0\|^2 + \frac{1}{n} \le t_n^2 + \frac{1}{n} \le 2(t_n^2 \vee \frac{1}{n})$ if $f_0 \in \mathcal{F}_q^L(t_n; M_n)$.

For $\mathcal{F} = \mathcal{F}_0^L(k_n; M_n)$ or $\mathcal{F} = \mathcal{F}_q^L(t_n; M_n) \cap \mathcal{F}_0^L(k_n; M_n)$, one can use the same argument as in Theorem 5.

(ii) For **AC-** strategies, for $\mathcal{F} = \mathcal{F}_q^L(t_n; M_n)$ or $\mathcal{F} = \mathcal{F}_q^L(t_n; M_n) \cap \mathcal{F}_0^L(k_n; M_n)$, again one can use the same argument as in the proof of Theorem 5. For $\mathcal{F} = \mathcal{F}_0^L(k_n; M_n)$, the approximation error is $\inf_{s\ge 1}\left(\inf_{\{\theta:\|\theta\|_1\le s, \|\theta\|_0\le k_n, \|f_\theta\|_\infty \le L\}} \|f_\theta - f_0\|^2 + 2c_3\frac{\log(1+s)}{n}\right) \le \inf_{\{\theta:\|\theta\|_1\le \alpha_n, \|\theta\|_0\le k_n, \|f_\theta\|_\infty \le L\}} \|f_\theta - f_0\|^2 + 2c_3\frac{\log(1+\alpha_n)}{n} = 2c_3\frac{\log(1+\alpha_n)}{n}$ if $f_0 \in \mathcal{F}_0^L(k_n; M_n)$. The upper bound then follows. This completes the proof.

C.6 Proof of Theorem 11

Without loss of generality, we assume $\sigma^2 = 1$ for the error variance. First, we give a simple fact. Let $B_k(\eta) = \{\theta : \|\theta\|_2 \le \eta, \|\theta\|_0 \le k\}$ and $\mathcal{B}_k(f_0; \epsilon) = \{f_\theta : \|f_\theta\| \le \epsilon, \|\theta\|_0 \le k\}$ (take $f_0 = 0$). Then, under Assumption SRC with $\gamma = k$, the $\frac{a}{2\overline{a}}$-local ϵ-packing entropy of $\mathcal{B}_k(f_0; \epsilon)$ is lower bounded by the $\frac{1}{2}$-local η-packing entropy of $B_k(\eta)$ with $\eta = \epsilon/\overline{a}$.

(i) The proof is essentially the same as that of Theorem 8. When $m^* = M_n$, the previous lower bounding method works with a slight modification. When $\left(1+\log\frac{M_n}{m^*}\right)^{q/2} < m^* < M_n$, we again use the global entropy to derive the lower bound based on Proposition 16. The key is to realize that in the derivation of the metric entropy lower bound for $\{\theta : \|\theta\|_q \le t_n\}$ in Kühn (2001), an optimal size packing set is constructed in which every member has at most m_* non-zero coefficients. Assumption SRC with $\gamma = m_*$ ensures that the L_2 distance on this packing set is equivalent to the ℓ_2 distance on the coefficients and then we know the metric entropy of $\mathcal{F}_q(t_n; M_n)$ under the L_2 distance is at the order given. The result follows as before. When $m^* \le \left(1+\log\frac{M_n}{m^*}\right)^{q/2}$, observe that $\mathcal{F}_q(t_n; M_n) \supset \{\beta x_j : |\beta| \le t_n\}$ for any $1 \le j \le M_n$. The use of the local entropy result in Proposition 16 readily gives the desired result.

(ii) As in the proof of Theorem 8, without loss of generality, we can assume $k_n \le M_n/2$. Together with the simple fact given at the beginning of the proof, for $B_{k_n}(\epsilon/\overline{a}) = \{\theta : \|\theta\|_2 \le \epsilon/\overline{a}, \|\theta\|_0 \le k_n\}$, with $\eta' = \epsilon/(\overline{a}\sqrt{k_n})$, we know $B_{k_n}(\epsilon/\overline{a})$ contains the set

$$\{\theta = \eta' I : I \in \{1, 0, -1\}^{M_n}, \|I\|_0 \le k_n\}.$$

For $\theta_1 = \eta' I_1, \theta_2 = \eta' I_2$ both in the above set, by Assumption SRC, $\|f_{\theta_1} - f_{\theta_2}\|^2 \ge \underline{a}^2 \eta'^2 d_{HM}(I_1, I_2) \ge \underline{a}^2\epsilon^2/(2\overline{a}^2)$ when the Hamming distance $d_{HM}(I_1, I_2)$ is larger than $k_n/2$.

With the derivation in the proof of part (i) of Theorem 8 (case 2), we know the local entropy $M^{loc}_{\underline{a}/(\sqrt{2}\bar{a})}(\epsilon)$ of $\mathcal{F}_0(k_n; M_n) \cap \{f_\theta : \|\theta\|_2 \le a_n\}$ with $a_n \ge \epsilon$ is lower bounded by $\frac{k_n}{2} \log \frac{2(M_n - k_n)}{k_n}$. Then, under the condition $a_n \ge C\sqrt{k_n \left(1 + \log \frac{M_n}{k_n}\right)/n}$ for some constant C, the minimax lower rate $k_n \left(1 + \log \frac{M_n}{k_n}\right)/n$ follows from a slight modification of the proof of Theorem 8 with $\epsilon = C'\sqrt{k_n \left(1 + \log \frac{M_n}{k_n}\right)/n}$ for some constant $C' > 0$. When $0 < a_n < C\sqrt{k_n \left(1 + \log \frac{M_n}{k_n}\right)/n}$, with ϵ of order a_n, the lower bound follows.

(iii) For the larger k_n case, from the proof of part (i) of the theorem, we have used fewer than k_n nonzero components to derive the minimax lower bound there. Thus, the extra ℓ_0-constraint does not change the problem in terms of lower bound. For the smaller k_n case, note that for θ with $\|\theta\|_0 \le k_n$, $\|\theta\|_q \le k_n^{1/q-1/2}\|\theta\|_2 \le k_n^{1/q-1/2}\sqrt{Ck_n \left(1 + \log \frac{M_n}{k_n}\right)/n}$ for θ with $\|\theta\|_2 \le \sqrt{Ck_n \left(1 + \log \frac{M_n}{k_n}\right)/n}$. Therefore the ℓ_q-constraint is automatically satisfied when $\|\theta\|_2$ is no larger than the critical order $\sqrt{k_n \left(1 + \log \frac{M_n}{k_n}\right)/n}$, which is sufficient for the lower bound via local entropy techniques. The conclusion follows. This completes the proof.

C.7 Proof of Corollary 12

(i) We only need to derive the lower bound part. Under the assumptions that $\sup_j \|X_j\|_\infty \le L_0 < \infty$ for some constant $L_0 > 0$, for a fixed $t_n = t > 0$, we have $\forall f_\theta \in \mathcal{F}_q(t_n; M_n)$, $\|f_\theta\|_\infty \le \sup_j \|X_j\|_\infty \cdot \sum_{j=1}^{M_n} |\theta_j| \le L_0\|\theta\|_1 \le L_0\|\theta\|_q \le L_0 t$. Then the conclusion follows directly from Theorem 11 (Part (i)). Note that when t_n is fixed, the case $m_* = 1$ does not need to be separately considered.

(ii) For the upper rate part, we use the **AC-C** upper bound. For f_θ with $\|\theta\|_\infty \le L_0$, clearly, we have $\|\theta\|_1 \le M_n L_0$, and consequently, since $\log(1 + M_n L_0)$ is upper bounded by a multiple of $k_n \left(1 + \log \frac{M_n}{k_n}\right)$, the upper rate $\frac{k_n}{n}\left(1 + \log \frac{M_n}{k_n}\right) \wedge 1$ is obtained from Corollary 10. Under the assumptions that $\sup_j \|X_j\|_\infty \le L_0 < \infty$ and $k_n\sqrt{\left(1 + \log \frac{M_n}{k_n}\right)/n} \le \sqrt{K_0}$, we know that $\forall f_\theta \in \mathcal{F}_0(k_n; M_n) \bigcap \{f_\theta : \|\theta\|_2 \le a_n\}$ with $a_n = C\sqrt{k_n \left(1 + \log \frac{M_n}{k_n}\right)/n}$ for some constant $C > 0$, the sup-norm of f_θ is upper bounded by

$$\|\sum_{j=1}^{M_n} \theta_j x_j\|_\infty \le L_0\|\theta\|_1 \le L_0\sqrt{k_n} a_n = CL_0 k_n \sqrt{\frac{1 + \log \frac{M_n}{k_n}}{n}} \le C\sqrt{K_0} L_0.$$

Then the functions in $\mathcal{F}_0(k_n; M_n) \bigcap \{f : \|\theta\|_2 \le a_n\}$ have sup-norm uniformly bounded. Note that for bounded a_n, $\|\theta\|_2 \le a_n$ implies that $\|\theta\|_\infty \le a_n$. Thus, the extra restriction $\|\theta\|_\infty \le L_0$ does not affect the minimax lower rate established in part (ii) of Theorem 11.

(iii) The upper and lower rates follow similarly from Corollary 10 and Theorem 11. The details are thus skipped. This completes the proof.

References

J.-Y. Audibert. No fast exponential deviation inequalities for the progressive mixture rule. Available at `http://arxiv.org/abs/math.ST/0703848v1`, 2007.

J.-Y. Audibert. Fast learning rates in statistical inference through aggregation. *Annals of Statistics*, 37:1591–1646, 2009.

J.-Y. Audibert and O. Catoni. Risk bounds in linear regression through PAC-Bayesian truncation. Preprint at arXiv:1010.0072, 2010.

Y. Baraud. Model selection for regression on a fixed design. *Probability Theory Related Fields*, 117:467–493, 2000.

Y. Baraud. Model selection for regression on a random design. *ESAIM Probab.Statist.*, 6: 127–146, 2002.

A. R. Barron, L. Birgé, and P. Massart. Risk bounds for model selection via penalization. *Probability Theory and Related Fields*, 113:301–413, 1999.

A.R. Barron and T.M. Cover. Minimum complexity density estimation. *IEEE Transactions on Information Theory*, 37:1034–1054, 1991.

L. Birgé. On estimating a density using Hellinger distance and some other strange facts. *Probab. Theory Related Fields*, 97:113–150, 1986.

L. Birgé. Model selection for Gaussian regression with random design. *Bernoulli*, 10:1039–1051, 2004.

L. Birgé. Model selection via testing: an alternative to (penalized) maximum likelihood estimators. *Ann. I. H. Poincaré*, 42:273–325, 2006.

L. Birgé. Model selection for density estimation with L_2-loss. *Probab. Theory Related Fields*, 158:533–574, 2014.

L. Birgé and P. Massart. Gaussian model selection. *Journal of European Math. Society*, 3: 203–268, 2001.

F. Bunea and A. Nobel. Sequential procedures for aggregating arbitrary estimators of a conditional mean. *IEEE Transactions on Information Theory*, 54:1725–1735, 2008.

F. Bunea, A.B. Tsybakov, and M.H. Wegkamp. Aggregation for gaussian regression. *Annals of Statistics*, 35:1674–1697, 2007.

O. Catoni. The mixture approach to universal model selection. Preprint, LMENS-97-30, Ecole Normale Supèrieure, Paris, France, 1997.

O. Catoni. Universal aggregation rules with exact bias bounds. Preprint 510, Laboratoire de Probabilitès et Modèles Aléatoires, Univ. Paris 6 and Paris 7, France. Available at `http://www.proba.jussieu.fr/mathdoc/preprints/index.html#1999`, 1999.

O. Catoni. *Statistical Learning Theory and Stochastic Optimization*, volume 1851. Springer, New York, 2004.

O. Catoni. Challenging the empirical mean and empirical variance: A deviation study. *Ann. I. H. Poincaré*, 48:909–1244, 2012.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning and Games*. Cambridge University Press, 2006.

A. S. Dalalyan and J. Salmon. Sharp oracle inequalities for aggregation of affine estimators. *Annals of Statistics*, 40:2327–2355, 2012.

A. S. Dalalyan and A.B. Tsybakov. Aggregation by exponential weighting and sharp oracle inequalities. *Lecture Notes in Computer Science*, 4539:97–111, 2007.

A. S. Dalalyan and A.B. Tsybakov. Sparse regression learning by aggregation and Langevin Monte Carlo. *Journal of Computer and System Science*, 78:1423–1443, 2012a.

A. S. Dalalyan and A.B. Tsybakov. Mirror averaging with sparsity priors. *Bernoulli*, 18: 914–944, 2012b.

D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52:1289–1306, 2006.

D. L. Donoho and I.M. Johnstone. Minimax risk over ℓ_p-balls for ℓ_q-error. *Probability Theory and Related Fields*, 99:277–303, 1994.

D. E. Edmunds and H. Triebel. Function spaces, entropy numbers, and differential operators. *Cambridge Tracts in Mathematics*, 120, 1998.

C. Giraud. Mixing least-squares estimators when the variance is unknown. *Bernoulli*, 14: 1089–1107, 2008.

A. Goldenshluger. A universal procedure for aggregating estimators. *Annals of Statistics*, 37:542–568, 2009.

L. Györfi and Gy. Ottucsák. Sequential prediction of unbounded stationary time series. *IEEE Transactions on Information Theory*, 53:1866–1872, 2007.

L. Györfi, M. Kohler, A. Krzyżak, and H. Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer, New York, 2002.

A. Juditsky and A. Nemirovski. Functional aggregation for nonparametric estimation. *Annals of Statistics*, 28:681–719, 2000.

A. Juditsky, P. Rigollet, and A.B. Tsybakov. Learning by mirror averaging. *Annals of Statistics*, 36:2183–2206, 2008.

T. Kühn. A lower estimate for entropy numbers. *Journal of Approximation Theory*, 110: 120–124, 2001.

G. Leung and A.R. Barron. Information theory and mixing least-squares regressions. *IEEE Transactions on Information Theory*, 52:3396–3410, 2006.

K. Lounici. Generalized mirror averaging and d-convex aggregation. *Mathematical methods of statistic*, 16:246–259, 2007.

P. Massart. *Concentration Inequalities and Model Selection*, volume 1896. Lecture Notes in Mathematics, Springer, Berlin/Heidelberg, 2007.

S. Negahban, P. Ravikumar, M.J. Wainwright, and B. Yu. A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. *Statistical Science*, 27:538–557, 2012.

A. Nemirovski. Topics in non-parametric statistics. Lecture Notes in Mathematics, Springer, New York, ecole d'eté de probabilités de saint-flour 1998 edition, 2000.

G. Pisier. Remarques sur un résultat non publié de b. maure. *Sém. d'Analyse Fonctionnelle 1980/1981, Exp. V.*, 1981.

G. Raskutti, M. Wainwright, and B. Yu. Restricted eigenvalue properties for correlated gaussian designs. *Journal of Machine Learning Research*, 11:2241–2259, 2010.

G. Raskutti, M. Wainwright, and B. Yu. Minimax rates of estimation for high-dimensional linear regression over ℓ_q-balls. *IEEE Transactions on Information Theory*, 57:6976–699, 2012.

P. Rigollet and A.B. Tsybakov. Exponential screening and optimal rates of sparse estimation. *Annals of Statistics*, 39:731–771, 2010.

A.B. Tsybakov. Optimal rates of aggregation. *Annals of Statistics*, 39:731–771, 2003.

S.A. van de Geer and P. Bühlmann. On the conditions used to prove oracle results for the lasso. *Electron. J. Statist*, 3:1360–1392, 2009.

Z. Wang, S. Paterlini, F. Gao, and Y. Yang. Adaptive minimax estimation over sparse ℓ_q-hulls. Arxiv preprint arXiv:1108.1961, 2011.

M. Wegkamp. Model selection in nonparametric regression. *Annals of Statistics*, 31:252–273, 2003.

Y. Yang. Minimax optimal density estimation. Ph.D. Dissertation, Department of Statistics, Yale University, 1996.

Y. Yang. Model selection for nonparametric regression. *Statistica Sinica*, 9:475–499, 1999.

Y. Yang. Combining different procedures for adaptive regression. *Journal of Multivariate Analysis*, 74:135–161, 2000a.

Y. Yang. Mixing strategies for density estimation. *Annals of Statistics*, 28:75–87, 2000b.

Y. Yang. Adaptive regression by mixing. *Journal of American Statistical Association*, 96: 574–588, 2001.

Y. Yang. Aggregating regression procedures to improve performance. *Bernoulli*, 10:25–47, 2004. An older version of the paper is Preprint #1999-17 of Department of Statistics at Iowa State University.

Y. Yang and A.R. Barron. An asymptotic property of model selection criteria. *IEEE Trans. Inform. Theory*, 44:95–116, 1998.

Y. Yang and A.R. Barron. Information theoretic determination of minimax rates of convergence. *Annals of Statistics*, 27:1564–1599, 1999.

C.H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *Annals of Statistics*, 38:894–942, 2010.

Journal of Machine Learning Research 15 (2014) 1713-1750 Submitted 5/13; Revised 3/14; Published 5/14

Graph Estimation From Multi-Attribute Data

Mladen Kolar MKOLAR@CHICAGOBOOTH.EDU
The University of Chicago Booth School of Business
Chicago, Illinois 60637, USA

Han Liu HANLIU@PRINCETON.EDU
Department of Operations Research and Financial Engineering
Princeton University
Princeton, New Jersey 08544, USA

Eric P. Xing EPXING@CS.CMU.EDU
Machine Learning Department
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213, USA

Editor: Yuan (Alan) Qi

Abstract

Undirected graphical models are important in a number of modern applications that involve exploring or exploiting dependency structures underlying the data. For example, they are often used to explore complex systems where connections between entities are not well understood, such as in functional brain networks or genetic networks. Existing methods for estimating structure of undirected graphical models focus on scenarios where each node represents a scalar random variable, such as a binary neural activation state or a continuous mRNA abundance measurement, even though in many real world problems, nodes can represent multivariate variables with much richer meanings, such as whole images, text documents, or multi-view feature vectors. In this paper, we propose a new principled framework for estimating the structure of undirected graphical models from such multivariate (or multi-attribute) nodal data. The structure of a graph is inferred through estimation of non-zero partial canonical correlation between nodes. Under a Gaussian model, this strategy is equivalent to estimating conditional independencies between random vectors represented by the nodes and it generalizes the classical problem of covariance selection (Dempster, 1972). We relate the problem of estimating non-zero partial canonical correlations to maximizing a penalized Gaussian likelihood objective and develop a method that efficiently maximizes this objective. Extensive simulation studies demonstrate the effectiveness of the method under various conditions. We provide illustrative applications to uncovering gene regulatory networks from gene and protein profiles, and uncovering brain connectivity graph from positron emission tomography data. Finally, we provide sufficient conditions under which the true graphical structure can be recovered correctly.

Keywords: graphical model selection, multi-attribute data, network analysis, partial canonical correlation

1. Introduction

Gaussian graphical models are commonly used to represent and explore conditional independencies between variables in a complex system. An edge between two nodes is present

in the graph if and only if the corresponding variables are conditionally independent given all the other variables. Current approaches to estimating the Markov network structure underlying a Gaussian graphical model focus on cases where nodes in a network represent scalar variables such as the binary voting actions of actors (Banerjee et al., 2008; Kolar et al., 2010) or the continuous mRNA abundance measurements of genes (Peng et al., 2009). However, in many modern problems, we are interested in studying a network where nodes can represent more complex and informative vector-variables or multi-attribute entities. For example, due to advances of modern data acquisition technologies, researchers are able to measure the activities of a single gene in a high-dimensional space, such as an image of the spatial distribution of the gene expression, or a multi-view snapshot of the gene activity such as mRNA and protein abundances; when modeling a social network, a node may correspond to a person for which a vector of attributes is available, such as personal information, demographics, interests, and other features. Therefore, there is a need for methods that estimate the structure of an undirected graphical model from such multi-attribute data.

In this paper, we develop a new method for estimating the structure of undirected graphical models of which the nodes correspond to vectors, that is, multi-attribute entities. We consider the following setting. Let $X = (X_1^T, ..., X_p^T)^T$ where $X_1 \in \mathbb{R}^{k_1}, \ldots, X_p \in \mathbb{R}^{k_p}$ are random vectors that jointly follow a multivariate Gaussian distribution with mean $\mu = (\mu_1^T, \ldots, \mu_p^T)^T$ and covariance matrix Σ^*, which is partitioned as

$$\Sigma^* = \begin{pmatrix} \Sigma^*_{11} & \cdots & \Sigma^*_{1p} \\ \vdots & \ddots & \vdots \\ \Sigma^*_{p1} & \cdots & \Sigma^*_{pp} \end{pmatrix}, \tag{1}$$

with $\Sigma^*_{ij} = \mathrm{Cov}(X_i, X_j)$. Without loss of generality, we assume $\mu = 0$. Let $G = (V, E)$ be a graph with the vertex set $V = \{1, \ldots, p\}$ and the set of edges $E \subseteq V \times V$ that encodes the conditional independence relationships among $(X_a)_{a \in V}$. That is, each node $a \in V$ of the graph G corresponds to the random vector X_a and there is no edge between nodes a and b in the graph if and only if X_a is conditionally independent of X_b given all the vectors corresponding to the remaining nodes, $X_{\neg ab} = \{X_c \ : \ c \in V \backslash \{a, b\}\}$. Such a graph is known as a *Markov network* (of Markov graph), which we shall emphasize in this paper to contrast an alternative graph over V known as the *association network*, which is based on pairwise marginal independence. Conditional independence can be read from the inverse of the covariance matrix of X, as the block corresponding to X_a and X_b will be equal to zero when they are conditionally independent, whereas marginal independencies are captured by the covariance matrix itself. It is well known that estimating an association network from data can result in a hard-to-interpret dense graph due to prevalent indirect correlations among variables (for example, multiple nodes each influenced by a common single node could result in a clique over all these nodes), which can be avoided in estimating a Markov network.

Let $\mathcal{D}_n = \{x_i\}_{i=1}^n$ be a sample of n independent and identically distributed (*iid*) vectors drawn from $\mathrm{N}(0, \Sigma^*)$. For a vector x_i, we denote $x_{i,a} \in \mathbb{R}^{k_a}$ the component corresponding to the node $a \in V$. Our goal is to estimate the structure of the graph G from the sample $\mathcal{D}_n$. Note that we allow for different nodes to have different number of attributes, which is useful in many applications, for example, when a node represents a gene pathway (of

different sizes) in a regulatory network, or a brain region (of different volumes) in a neural activation network.

Learning the structure of a Gaussian graphical model, where each node represents a scalar random variable, is a classical problem, known as the covariance selection (Dempster, 1972). One can estimate the graph structure by estimating the sparsity pattern of the precision matrix $\Omega = \Sigma^{-1}$. For high dimensional problems, Meinshausen and Bühlmann (2006) propose a parallel Lasso approach for estimating Gaussian graphical models by solving a collection of sparse regression problems. This procedure can be viewed as a pseudo-likelihood based method. In contrast, Banerjee et al. (2008), Yuan and Lin (2007), and Friedman et al. (2008) take a penalized likelihood approach to estimate the sparse precision matrix Ω. To reduce estimation bias, Lam and Fan (2009), Johnson et al. (2012), and Shen et al. (2012) developed the non-concave penalties to penalize the likelihood function. More recently, Yuan (2010) and Cai et al. (2011) proposed the graphical Dantzig selector and CLIME, which can be solved by linear programming and are more amenable to theoretical analysis than the penalized likelihood approach. Under certain regularity conditions, these methods have proven to be graph-estimation consistent (Ravikumar et al., 2011; Yuan, 2010; Cai et al., 2011) and scalable software packages, such as *glasso* and *huge*, were developed to implement these algorithms (Zhao et al., 2012). For a recent survey see Pourahmadi (2011). However, these methods cannot be extended to handle multi-attribute data we consider here in an obvious way. For example, if the number of attributes is the same for each node, one may naively estimate one graph per attribute, however, it is not clear how to combine such graphs into a summary graph with a clear statistical interpretation. The situation becomes even more difficult when nodes correspond to objects that have different number of attributes.

In a related work, Katenka and Kolaczyk (2011) use canonical correlation to estimate association networks from multi-attribute data, however, such networks have different interpretation to undirected graphical models. In particular, as mentioned above, association networks are known to confound the direct interactions with indirect ones as they only represent marginal associations, whereas Markov networks represent conditional independence assumptions that are better suited for separating direct interactions from indirect confounders. Our work is related to the literature on simultaneous estimation of multiple Gaussian graphical models under a multi-task setting (Guo et al., 2011; Varoquaux et al., 2010; Honorio and Samaras, 2010; Chiquet et al., 2011; Danaher et al., 2014). However, the model given in (1) is different from models considered in various multi-task settings and the optimization algorithms developed in the multi-task literature do not extend to handle the optimization problem given in our setting.

Unlike the standard procedures for estimating the structure of Gaussian graphical models, for example, neighborhood selection (Meinshausen and Bühlmann, 2006) or glasso (Friedman et al., 2008), which infer the partial correlations between pairs of nodes, our proposed method estimates the graph structure based on the partial canonical correlation, which can naturally incorporate complex nodal observations. Under that the Gaussian model in (1), the estimated graph structure has the same probabilistic independence interpretations as the Gaussian graphical model over univariate nodes. The main contributions of the paper are the following. First, we introduce a new framework for learning structure of undirected graphical models from multi-attribute data. Second, we develop an efficient

algorithm that estimates the structure of a graph from the observed data. Third, we provide extensive simulation studies that demonstrate the effectiveness of our method and illustrate how the framework can be used to uncover gene regulatory networks from gene and protein profiles, and to uncover brain connectivity graph from functional magnetic resonance imaging data. Finally, we provide theoretical results, which give sufficient conditions for consistent structure recovery.

2. Methodology

In this section, we propose to estimate the graph by estimating non-zero partial canonical correlation between the nodes. This leads to a penalized maximum likelihood objective, for which we develop an efficient optimization procedure.

2.1 Preliminaries

Let X_a and X_b be two multivariate random vectors. Canonical correlation is defined between X_a and X_b as

$$\rho_c(X_a, X_b) = \max_{u \in \mathbb{R}^{k_a}, v \in \mathbb{R}^{k_b}} \text{corr}(u^T X_a, v^T X_b).$$

That is, computing canonical correlation between X_a and X_b is equivalent to maximizing the correlation between two linear combinations $u^T X_a$ and $v^T X_b$ with respect to vectors u and v. Canonical correlation can be used to measure association strength between two nodes with multi-attribute observations. For example, in Katenka and Kolaczyk (2011), a graph is estimated from multi-attribute nodal observations by elementwise thresholding the canonical correlation matrix between nodes, but such a graph estimator may confound the direct interactions with indirect ones.

In this paper, we exploit the partial canonical correlation to estimate a graph from multi-attribute nodal observations. A graph is going to be formed by connecting nodes with non-zero partial canonical correlation. Let $\widehat{A} = \text{argmin } E\left(||X_a - AX_{\neg ab}||_2^2\right)$ and $\widehat{B} = \text{argmin } E\left(||X_b - BX_{\neg ab}||_2^2\right)$, then the partial canonical correlation between X_a and X_b is defined as

$$\rho_c(X_a, X_b; X_{\neg ab}) = \max_{u \in \mathbb{R}^{k_a}, v \in \mathbb{R}^{k_b}} \text{corr}\{u^T(X_a - \widehat{A}X_{\neg ab}), v^T(X_b - \widehat{B}X_{\neg ab})\}, \tag{2}$$

that is, the partial canonical correlation between X_a and X_b is equal to the canonical correlation between the residual vectors of X_a and X_b after the effect of $X_{\neg ab}$ is removed (Rao, 1969).[1]

Let $\Omega^* = (\Sigma^*)^{-1}$ denote the precision matrix. A simple calculation, given in Appendix B.3, shows that

$$\rho_c(X_a, X_b; X_{\neg ab}) \neq 0 \quad \text{if and only if} \quad \max_{u \in \mathbb{R}^{k_a}, v \in \mathbb{R}^{k_b}} u^T \Omega^*_{ab} v \neq 0. \tag{3}$$

This implies that estimating whether the partial canonical correlation is zero or not can be done by estimating whether a block of the precision matrix is zero or not. Furthermore,

1. The operator $E(\cdot)$ denotes the expectation and $X_{\neg ab} = \{X_c : c \in V \backslash \{a, b\}\}$ denotes all the variables except for X_a and X_b.

under the Gaussian model in (1), vectors X_a and X_b are conditionally independent given $X_{\neg ab}$ if and only if partial canonical correlation is zero. A graph built on this type of inter-nodal relationship is known as a Markov network, as it captures both local and global Markov properties over all arbitrary subsets of nodes in the graph, even though the graph is built based on pairwise conditional independence properties. In Section 2.2, we use the above observations to design an algorithm that estimates the non-zero partial canonical correlation between nodes from data $\mathcal{D}_n$ using the penalized maximum likelihood estimation of the precision matrix.

Based on the relationship given in (3), we can motivate an alternative method for estimating the non-zero partial canonical correlation. Let $\overline{a} = \{b \ : \ b \in V \backslash \{a\}\}$ denote the set of all nodes minus the node a. Then

$$E\left(X_a \mid X_{\overline{a}} = x_{\overline{a}}\right) = \Sigma^*_{a,\overline{a}} \Sigma^{*,-1}_{\overline{a},\overline{a}} x_{\overline{a}}.$$

Since $\Omega^*_{a,\overline{a}} = -(\Sigma^*_{aa} - \Sigma^*_{a,\overline{a}} \Sigma^{*,-1}_{\overline{a},\overline{a}} \Sigma^*_{\overline{a},a})^{-1} \Sigma^*_{a,\overline{a}} \Sigma^{*,-1}_{\overline{a},\overline{a}}$, we observe that a zero block Ω_{ab} can be identified from the regression coefficients when each component of X_a is regressed on $X_{\overline{a}}$. We do not build an estimation procedure around this observation, however, we note that this relationship shows how one would develop a regression based analogue of the work presented in Katenka and Kolaczyk (2011).

2.2 Penalized Log-Likelihood Optimization

Based on the data $\mathcal{D}_n$, we propose to minimize the penalized negative Gaussian log-likelihood under the model in (1),

$$\min_{\Omega > 0} \Big\{ \operatorname{tr} S\Omega - \log |\Omega| + \lambda \sum_{a,b} ||\Omega_{ab}||_F \Big\}, \tag{4}$$

where $S = n^{-1} \sum_{i=1}^n x_i x_i^T$ is the sample covariance matrix, $||\Omega_{ab}||_F$ denotes the Frobenius norm of Ω_{ab} and λ is a user defined parameter that controls the sparsity of the solution $\widehat{\Omega}$. The first two terms in (4) correspond to the negative Gaussian log-likelihood, while the second term is the Frobenius norm penalty, which encourages blocks of the precision matrix to be equal to zero, similar to the way that the ℓ_2 penalty is used in the group Lasso (Yuan and Lin, 2006). Here we assume that the same number of samples is available per attribute. However, the same method can be used in cases when some samples are obtained on a subset of attributes. Indeed, we can simply estimate each element of the matrix S from available samples, treating non-measured attributes as missing completely at random (for more details see Kolar and Xing, 2012).

The dual problem to (4) is

$$\max_{\Sigma} \sum_{j \in V} k_j + \log |\Sigma| \qquad \text{subject to} \qquad \max_{a,b} \ ||S_{ab} - \Sigma_{ab}||_F \leqslant \lambda, \tag{5}$$

where k_j is the number attributes of node j, Σ is the dual variable to Ω and $|\Sigma|$ denotes the determinant of Σ. Note that the primal problem gives us an estimate of the precision matrix, while the dual problem estimates the covariance matrix. The proposed optimization procedure, described below, will simultaneously estimate the precision matrix and covariance matrix, without explicitly performing an expensive matrix inversion.

We propose to optimize the objective function in (4) using an inexact block coordinate descent procedure, inspired by Mazumder and Agarwal (2011). The block coordinate descent is an iterative procedure that operates on a block of rows and columns while keeping the other rows and columns fixed. We write

$$\Omega = \begin{pmatrix} \Omega_{aa} & \Omega_{a,\overline{a}} \\ \Omega_{\overline{a},a} & \Omega_{\overline{a},\overline{a}} \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{a,\overline{a}} \\ \Sigma_{\overline{a},a} & \Sigma_{\overline{a},\overline{a}} \end{pmatrix}, \quad S = \begin{pmatrix} S_{aa} & S_{a,\overline{a}} \\ S_{\overline{a},a} & S_{\overline{a},\overline{a}} \end{pmatrix},$$

and suppose that $(\widetilde{\Omega}, \widetilde{\Sigma})$ are the current estimates of the precision matrix and covariance matrix. With the above block partition, we have $\log|\Omega| = \log(\Omega_{\overline{a},\overline{a}}) + \log(\Omega_{aa} - \Omega_{a,\overline{a}}(\Omega_{\overline{a},\overline{a}})^{-1}\Omega_{\overline{a},a})$. In the next iteration, $\widehat{\Omega}$ is of the form

$$\widehat{\Omega} = \widetilde{\Omega} + \begin{pmatrix} \Delta_{aa} & \Delta_{a,\overline{a}} \\ \Delta_{\overline{a},a} & 0 \end{pmatrix} = \begin{pmatrix} \widehat{\Omega}_{aa} & \widehat{\Omega}_{a,\overline{a}} \\ \widehat{\Omega}_{\overline{a},a} & \widetilde{\Omega}_{\overline{a},\overline{a}} \end{pmatrix},$$

and is obtained by minimizing

$$\operatorname{tr} S_{aa}\Omega_{aa} + 2\operatorname{tr} S_{a,\overline{a}}\Omega_{\overline{a},a} - \log|\Omega_{aa} - \Omega_{a,\overline{a}}(\widetilde{\Omega}_{\overline{a},\overline{a}})^{-1}\Omega_{\overline{a},a}| + \lambda||\Omega_{aa}||_F + 2\lambda\sum_{b\neq a}||\Omega_{ab}||_F. \quad (6)$$

Exact minimization over the variables Ω_{aa} and $\Omega_{a,\overline{a}}$ at each iteration of the block coordinate descent procedure can be computationally expensive. Therefore, we propose to update Ω_{aa} and $\Omega_{a,\overline{a}}$ using one generalized gradient step update (see Beck and Teboulle, 2009) in each iteration. Note that the objective function in (6) is a sum of a smooth convex function and a non-smooth convex penalty so that the gradient descent method cannot be directly applied. Given a step size t, generalized gradient descent optimizes a quadratic approximation of the objective at the current iterate $\widetilde{\Omega}$, which results in the following two updates

$$\widehat{\Omega}_{aa} = \underset{\Omega_{aa}}{\operatorname{argmin}} \Big\{ \operatorname{tr}(S_{aa} - \widetilde{\Sigma}_{aa})\Omega_{aa} + \frac{1}{2t}||\Omega_{aa} - \widetilde{\Omega}_{aa}||_F^2 + \lambda||\Omega_{aa}||_F \Big\}, \quad \text{and} \quad (7)$$

$$\widehat{\Omega}_{ab} = \underset{\Omega_{ab}}{\operatorname{argmin}} \Big\{ \operatorname{tr}(S_{ab} - \widetilde{\Sigma}_{ab})\Omega_{ba} + \frac{1}{2t}||\Omega_{ab} - \widetilde{\Omega}_{ab}||_F^2 + \lambda||\Omega_{ab}||_F \Big\}, \quad \forall b \in \overline{a}. \quad (8)$$

If the resulting estimator $\widehat{\Omega}$ is not positive definite or the update does not decrease the objective, we halve the step size t and find a new update. Once the update of the precision matrix $\widehat{\Omega}$ is obtained, we update the covariance matrix $\widehat{\Sigma}$. Updates to the precision and covariance matrices can be found efficiently, without performing expensive matrix inversion. First, note that the solutions to (7) and (8) can be computed in a closed form as

$$\widehat{\Omega}_{aa} = (1 - t\lambda/||\widetilde{\Omega}_{aa} + t(\widetilde{\Sigma}_{aa} - S_{aa})||_F)_+(\widetilde{\Omega}_{aa} + t(\widetilde{\Sigma}_{aa} - S_{aa})), \quad \text{and} \quad (9)$$

$$\widehat{\Omega}_{ab} = (1 - t\lambda/||\widetilde{\Omega}_{ab} + t(\widetilde{\Sigma}_{ab} - S_{ab})||_F)_+(\widetilde{\Omega}_{ab} + t(\widetilde{\Sigma}_{ab} - S_{ab})), \quad \forall b \in \overline{a}, \quad (10)$$

where $(x)_+ = \max(0, x)$. Next, the estimate of the covariance matrix can be updated efficiently, without inverting the whole $\widehat{\Omega}$ matrix, using the matrix inversion lemma as follows

$$\begin{aligned} \widehat{\Sigma}_{\overline{a},\overline{a}} &= (\widetilde{\Omega}_{\overline{a},\overline{a}})^{-1} + (\widetilde{\Omega}_{\overline{a},\overline{a}})^{-1}\widehat{\Omega}_{\overline{a},a}(\widehat{\Omega}_{aa} - \widehat{\Omega}_{a,\overline{a}}(\widetilde{\Omega}_{\overline{a},\overline{a}})^{-1}\widehat{\Omega}_{\overline{a},a})^{-1}\widehat{\Omega}_{a,\overline{a}}(\widetilde{\Omega}_{\overline{a},\overline{a}})^{-1}, \\ \widehat{\Sigma}_{a,\overline{a}} &= -\widehat{\Omega}_{aa}\widehat{\Omega}_{a,\overline{a}}\widehat{\Sigma}_{\overline{a},\overline{a}}, \\ \widehat{\Sigma}_{aa} &= (\widehat{\Omega}_{aa} - \widehat{\Omega}_{a,\overline{a}}(\widetilde{\Omega}_{\overline{a},\overline{a}})^{-1}\widehat{\Omega}_{\overline{a},a})^{-1}, \end{aligned} \quad (11)$$

with $(\widetilde{\Omega}_{\overline{a},\overline{a}})^{-1} = \widetilde{\Sigma}_{\overline{a},\overline{a}} - \widetilde{\Sigma}_{\overline{a},a}\widetilde{\Sigma}_{aa}^{-1}\widetilde{\Sigma}_{a,\overline{a}}$.

Combining all three steps we get the following algorithm:

1. Set the initial estimator $\widetilde{\Omega} = \mathrm{diag}(S)$ and $\widetilde{\Sigma} = \widetilde{\Omega}^{-1}$. Set the step size $t = 1$.

2. For each $a \in V$ perform the following:

   Update $\widehat{\Omega}$ using (9) and (10).
   If $\widehat{\Omega}$ is not positive definite, set $t \leftarrow t/2$ and repeat the update.
   Update $\widehat{\Sigma}$ using (11).

3. Repeat Step 2 until the duality gap

$$\left|\mathrm{tr}(S\widehat{\Omega}) - \log|\widehat{\Omega}| + \lambda\sum_{a,b}||\widehat{\Omega}_{ab}||_F - \sum_{j\in V} k_j - \log|\Sigma|\right| \leqslant \epsilon,$$

   where ϵ is a prefixed precision parameter (for example, $\epsilon = 10^{-3}$).

Finally, we form a graph $\widehat{G} = (V, \widehat{E})$ by connecting nodes with $||\widehat{\Omega}_{ab}||_F \neq 0$.

Computational complexity of the procedure is given in Appendix A. Convergence of the above described procedure to the unique minimum of the objective function in (4) does not follow from the standard results on the block coordinate descent algorithm (Tseng, 2001) for two reasons. First, the minimization problem in (6) is not solved exactly at each iteration, since we only update Ω_{aa} and $\Omega_{a,\overline{a}}$ using one generalized gradient step update in each iteration. Second, the blocks of variables, over which the optimization is done at each iteration, are not completely separable between iterations due to the symmetry of the problem. The proof of the following convergence result is given in Appendix B.

Lemma 1 *For every value of $\lambda > 0$, the above described algorithm produces a sequence of estimates $\left\{\widetilde{\Omega}^{(t)}\right\}_{t\geqslant 1}$ of the precision matrix that monotonically decrease the objective values given in* (4)*. Every element of this sequence is positive definite and the sequence converges to the unique minimizer $\widehat{\Omega}$ of* (4)*.*

2.3 Efficient Identification of Connected Components

When the target graph $\widehat{G}$ is composed of smaller, disconnected components, the solution to the problem in (4) is block diagonal (possibly after permuting the node indices) and can be obtained by solving smaller optimization problems. That is, the minimizer $\widehat{\Omega}$ can be obtained by solving (4) for each connected component independently, resulting in massive computational gains. We give necessary and sufficient condition for the solution $\widehat{\Omega}$ of (4) to be block-diagonal, which can be easily checked by inspecting the empirical covariance matrix S.

Our first result follows immediately from the Karush-Kuhn-Tucker conditions for the optimization problem (4) and states that if $\widehat{\Omega}$ is block-diagonal, then it can be obtained by solving a sequence of smaller optimization problems.

Lemma 2 *If the solution to* (4) *takes the form* $\widehat{\Omega} = \mathrm{diag}(\widehat{\Omega}_1, \widehat{\Omega}_2, \ldots, \widehat{\Omega}_l)$*, that is,* $\widehat{\Omega}$ *is a block diagonal matrix with the diagonal blocks* $\widehat{\Omega}_1, \ldots, \widehat{\Omega}_l$*, then it can be obtained by solving*

$$\min_{\Omega_{l'} > 0} \left\{ \mathrm{tr}\, S_{l'} \Omega_{l'} - \log |\Omega_{l'}| + \lambda \sum_{a,b} ||\Omega_{ab}||_F \right\}$$

separately for each $l' = 1, \ldots, l$*, where* $S_{l'}$ *are submatrices of* S *corresponding to* $\Omega_{l'}$*.*

Next, we describe how to identify diagonal blocks of $\widehat{\Omega}$. Let $\mathcal{P} = \{P_1, P_2, \ldots, P_l\}$ be a partition of the set V and assume that the nodes of the graph are ordered in a way that if $a \in P_j$, $b \in P_{j'}$, $j < j'$, then $a < b$. The following lemma states that the blocks of $\widehat{\Omega}$ can be obtained from the blocks of a thresholded sample covariance matrix.

Lemma 3 *A necessary and sufficient condition for* $\widehat{\Omega}$ *to be block diagonal with blocks* $P_1, P_2, \ldots, P_l$ *is that* $||S_{ab}||_F \leqslant \lambda$ *for all* $a \in P_j$, $b \in P_{j'}$, $j \neq j'$*.*

Blocks $P_1, P_2, \ldots, P_l$ can be identified by forming a $p \times p$ matrix Q with elements $q_{ab} = \mathbb{1}\{||S_{ab}||_F > \lambda\}$ and computing connected components of the graph with adjacency matrix Q. The lemma states also that given two penalty parameters $\lambda_1 < \lambda_2$, the set of unconnected nodes with penalty parameter λ_1 is a subset of unconnected nodes with penalty parameter λ_2. The simple check above allows us to estimate graphs on data sets with large number of nodes, if we are interested in graphs with small number of edges. However, this is often the case when the graphs are used for exploration and interpretation of complex systems. Lemma 3 is related to existing results established for speeding-up computation when learning single and multiple Gaussian graphical models (Witten et al., 2011; Mazumder and Hastie, 2012; Danaher et al., 2014). Each condition is different, since the methods optimize different objective functions.

3. Consistent Graph Identification

In this section, we provide theoretical analysis of the estimator described in Section 2.2. In particular, we provide sufficient conditions for consistent graph recovery. For simplicity of presentation, we assume that $k_a = k$, for all $a \in V$, that is, we assume that the same number of attributes is observed for each node. For each $a = 1, \ldots, kp$, we assume that $(\sigma^*_{aa})^{-1/2} X_a$ is sub-Gaussian with parameter γ, where σ^*_{aa} is the ath diagonal element of Σ^*. Recall that Z is a sub-Gaussian random variable if there exists a constant $\sigma \in (0, \infty)$ such that

$$E\left(\exp(tZ)\right) \leqslant \exp(\sigma^2 t^2), \text{ for all } t \in \mathbb{R}.$$

Our assumptions involve the Hessian of the function $f(A) = \mathrm{tr}\, SA - \log |A|$ evaluated at the true Ω^*, $\mathcal{H} = \mathcal{H}(\Omega^*) = (\Omega^*)^{-1} \otimes (\Omega^*)^{-1} \in \mathbb{R}^{(pk)^2 \times (pk)^2}$, with $\otimes$ denoting the Kronecker product, and the true covariance matrix Σ^*. The Hessian and the covariance matrix can be thought of as block matrices with blocks of size $k^2 \times k^2$ and $k \times k$, respectively. We will make use of the operator $\mathcal{C}(\cdot)$ that operates on these block matrices and outputs a smaller matrix with elements that equal to the Frobenius norm of the original blocks. For example, $\mathcal{C}(\Sigma^*) \in \mathbb{R}^{p \times p}$ with elements $\mathcal{C}(\Sigma^*)_{ab} = ||\Sigma^*_{ab}||_F$. Let $\mathcal{T} = \{(a, b) : ||\Omega_{ab}||_F \neq 0\}$ and

$\mathcal{N} = \{(a,b) : ||\Omega_{ab}||_F = 0\}$. With this notation introduced, we assume that the following irrepresentable condition holds. There exists a constant $\alpha \in [0,1)$ such that

$$\|\!|\mathcal{C}\left(\mathcal{H}_{\mathcal{N}\mathcal{T}}(\mathcal{H}_{\mathcal{T}\mathcal{T}})^{-1}\right)|\!\|_\infty \leqslant 1-\alpha, \tag{12}$$

where $|\!|\!|A|\!|\!|_\infty = \max_i \sum_j |A_{ij}|$. We will also need the following quantities to specify the results $\kappa_{\Sigma *} = |\!|\!|\mathcal{C}(\Sigma^*)|\!|\!|_\infty$ and $\kappa_{\mathcal{H}} = |\!|\!|\mathcal{C}(\mathcal{H}_{\mathcal{T}\mathcal{T}}^{-1})|\!|\!|_\infty$. These conditions extend the conditions specified in Ravikumar et al. (2011) needed for estimating graphs from single attribute observations.

We have the following result that provides sufficient conditions for the exact recovery of the graph.

Proposition 4 *Let $\tau > 2$. We set the penalty parameter λ in (4) as*

$$\lambda = 8k\alpha^{-1}\left(128(1+4\gamma^2)^2(\max_a(\sigma^*_{aa})^2)n^{-1}(2\log(2k)+\tau\log(p))\right)^{1/2}.$$

If $n > C_1 s^2 k^2(1+8\alpha^{-1})^2(\tau\log p + \log 4 + 2\log k)$, where s is the maximal degree of nodes in G, $C_1 = (48\sqrt{2}(1+4\gamma^2)(\max_a \sigma^*_{aa})\max(\kappa_{\Sigma*}\kappa_{\mathcal{H}}, \kappa^3_{\Sigma*}\kappa^2_{\mathcal{H}}))^2$ and

$$\min_{(a,b)\in\mathcal{T}, a\neq b} ||\Omega_{ab}||_F > 16\sqrt{2}(1+4\gamma^2)(\max_a \sigma^*_{aa})(1+8\alpha^{-1})\kappa_{\mathcal{H}} k\left(\frac{\tau\log p+\log 4+2\log k}{n}\right)^{1/2},$$

then $\text{pr}\left(\widehat{G} = G\right) \geqslant 1 - p^{2-\tau}$.

The proof of Proposition 4 is given in Appendix B. We extend the proof of Ravikumar et al. (2011) to accommodate the Frobenius norm penalty on blocks of the precision matrix. This proposition specifies the sufficient sample size and a lower bound on the Frobenius norm of the off-diagonal blocks needed for recovery of the unknown graph. Under these conditions and correctly specified tuning parameter λ, the solution to the optimization problem in (4) correctly recovers the graph with high probability. In practice, one needs to choose the tuning parameter in a data dependent way. For example, using the Bayesian information criterion. Even though our theoretical analysis obtains the same rate of convergence as that of Ravikumar et al. (2011), our method has a significantly improved finite-sample performance, as will be shown in Section 5. It remains an open question whether the sample size requirement can be improved as in the case of group Lasso (see, for example, Lounici et al., 2011). The analysis of Lounici et al. (2011) relies heavily on the special structure of the least squares regression. Hence, their method does not carry over to the more complicated objective function as in (4).

4. Interpreting Edges

We propose a post-processing step that will allow us to quantify the strength of links identified by the method proposed in Section 2.2, as well as identify important attributes that contribute to the existence of links.

For any two nodes a and b for which $\Omega_{ab} \neq 0$, we define $\mathcal{N}(a,b) = \{c \in V\backslash\{a,b\} \ : \ \Omega_{ac} \neq 0 \text{ or } \Omega_{bc} \neq 0\}$, which is the Markov blanket for the set of nodes $\{X_a, X_b\}$. Note that the

conditional distribution of $(X_a^T, X_b^T)^T$ given $X_{\neg ab}$ is equal to the conditional distribution of $(X_a^T, X_b^T)^T$ given $X_{\mathcal{N}(a,b)}$. Now,

$$\begin{aligned}\rho_c(X_a, X_b; X_{\neg ab}) &= \rho_c(X_a, X_b; X_{\mathcal{N}(a,b)}) \\ &= \max_{w_a \in \mathbb{R}^{k_a}, w_b \in \mathbb{R}^{k_b}} \operatorname{corr}(u^T(X_a - \widetilde{A} X_{\mathcal{N}(a,b)}), v^T(X_b - \widetilde{B} X_{\mathcal{N}(a,b)})),\end{aligned}$$

where $\widetilde{A} = \operatorname{argmin}\ E\left(||X_a - A X_{\mathcal{N}(a,b)}||_2^2\right)$ and $\widetilde{B} = \operatorname{argmin}\ E\left(||X_b - B X_{\mathcal{N}(a,b)}||_2^2\right)$. Let $\overline{\Sigma}(a,b) = \operatorname{var}(X_a, X_b \mid X_{\mathcal{N}(a,b)})$. Now we can express the partial canonical correlation as

$$\rho_c(X_a, X_b; X_{\mathcal{N}(a,b)}) = \max_{w_a \in \mathbb{R}^{k_a}, w_b \in \mathbb{R}^{k_a}} \frac{w_a^T \overline{\Sigma}_{ab} w_b}{\left(w_a^T \overline{\Sigma}_{aa} w_a\right)^{1/2} \left(w_b^T \overline{\Sigma}_{bb} w_b\right)^{1/2}},$$

where

$$\overline{\Sigma}(a,b) = \begin{pmatrix} \overline{\Sigma}_{aa} & \overline{\Sigma}_{ab} \\ \overline{\Sigma}_{ba} & \overline{\Sigma}_{bb} \end{pmatrix}.$$

The weight vectors w_a and w_b can be easily found by solving the system of eigenvalue equations

$$\begin{cases} \overline{\Sigma}_{aa}^{-1} \overline{\Sigma}_{ab} \overline{\Sigma}_{bb}^{-1} \overline{\Sigma}_{ba} w_a = \phi^2 w_a \\ \overline{\Sigma}_{bb}^{-1} \overline{\Sigma}_{ba} \overline{\Sigma}_{aa}^{-1} \overline{\Sigma}_{ab} w_b = \phi^2 w_b \end{cases} \tag{13}$$

with w_a and w_b being the vectors that correspond to the maximum eigenvalue ϕ^2. Furthermore, we have $\rho_c(X_a, X_b; X_{\mathcal{N}(a,b)}) = \phi$. Following Katenka and Kolaczyk (2011), the weights w_a, w_b can be used to access the relative contribution of each attribute to the edge between the nodes a and b. In particular, the weight $(w_{a,i})^2$ characterizes the relative contribution of the ith attribute of node a to $\rho_c(X_a, X_b; X_{\mathcal{N}(a,b)})$.

Given an estimate $\widehat{\mathcal{N}}(a,b) = \{c \in V \backslash \{a,b\} \ :\ \widehat{\Omega}_{ac} \neq 0 \text{ or } \widehat{\Omega}_{bc} \neq 0\}$ of the Markov blanket $\mathcal{N}(a,b)$, we form the residual vectors

$$r_{i,a} = x_{i,a} - \breve{A} x_{i,\widehat{\mathcal{N}}(a,b)}, \qquad r_{i,b} = x_{i,b} - \breve{B} x_{i,\widehat{\mathcal{N}}(a,b)},$$

where $\breve{A}$ and $\breve{B}$ are the least square estimators of $\widetilde{A}$ and $\widetilde{B}$. Given the residuals, we form $\breve{\Sigma}(a,b)$, the empirical version of the matrix $\overline{\Sigma}(a,b)$, by setting

$$\breve{\Sigma}_{aa} = \operatorname{corr}\left(\{r_{i,a}\}_{i \in [n]}\right), \quad \breve{\Sigma}_{bb} = \operatorname{corr}\left(\{r_{i,b}\}_{i \in [n]}\right), \quad \breve{\Sigma}_{ab} = \operatorname{corr}\left(\{r_{i,a}\}_{i \in [n]}, \{r_{i,a}\}_{i \in [n]}\right).$$

Now, solving the eigenvalue system in (13) will give us estimates of the vectors w_a, w_b and the partial canonical correlation.

Note that we have described a way to interpret the elements of the off-diagonal blocks in the estimated precision matrix. The elements of the diagonal blocks, which correspond to coefficients between attributes of the same node, can still be interpreted by their relationship to the partial correlation coefficients.

5. Simulation Studies

In this section, we perform a set of simulation studies to illustrate finite sample performance of our method. We demonstrate that the scalings of (n, p, s) predicted by the theory are sharp. Furthermore, we compare against three other methods: 1) a method that uses the glasso first to estimate one graph over each of the k individual attributes and then creates an edge in the resulting graph if an edge appears in at least one of the single attribute graphs, 2) the method of Guo et al. (2011) and 3) the method of Danaher et al. (2014). We have also tried applying the glasso to estimate the precision matrix for the model in (1) and then post-processing it, so that an edge appears in the resulting graph if the corresponding block of the estimated precision matrix is non-zero. The result of this method is worse compared to the first baseline, so we do not report it here.

All the methods above require setting one or two tuning parameters that control the sparsity of the estimated graph. We select these tuning parameters by minimizing the Bayesian information criterion (Schwarz, 1978), which balances the goodness of fit of the model and its complexity, over a grid of parameter values. For our multi-attribute method, the Bayesian information criterion takes the following form

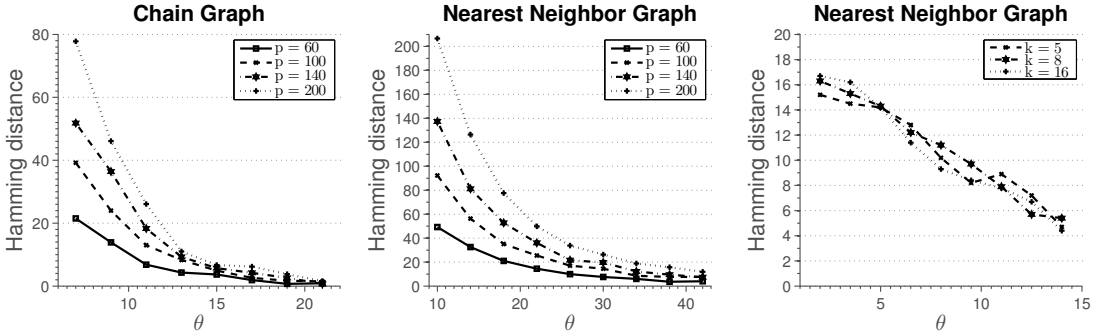
$$\mathrm{BIC}(\lambda) = \mathrm{tr}(S\widehat{\Omega}) - \log|\widehat{\Omega}| + \sum_{a<b} \mathbb{1}\{\widehat{\Omega}_{ab} \neq 0\} k_a k_b \log(n).$$

Other methods for selecting tuning parameters are possible, like minimization of cross-validation or Akaike information criterion (Akaike, 1974). However, these methods tend to select models that are too dense.

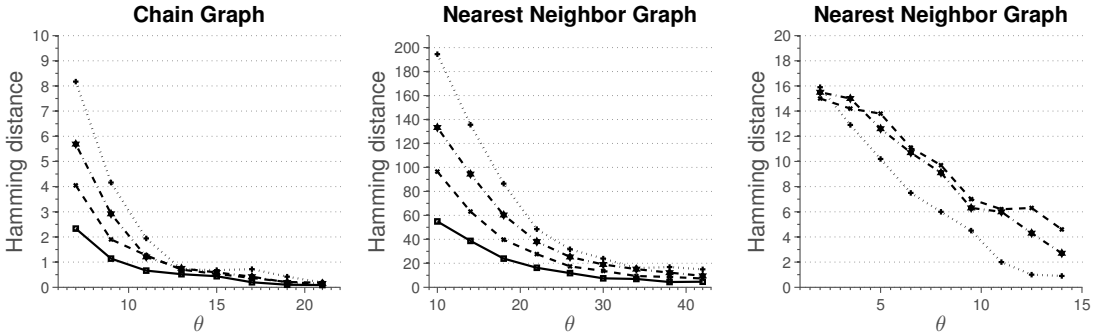
Theoretical results given in Section 3 characterize the sample size needed for consistent recovery of the underlying graph. In particular, Proposition 4 suggests that we need $n = \theta s^2 k^2 \log(pk)$ samples to estimate the graph structure consistently, for some control parameter $\theta > 0$. Therefore, if we plot the hamming distance between the true and recovered graph against θ, we expect the curves to reach zero distance for different problem sizes at a same point. We verify this on randomly generated chain and nearest-neighbors graphs.

Simulation 1. We generate data as follows. A random graph with p nodes is created by first partitioning nodes into $p/20$ connected components, each with 20 nodes, and then forming a random graph over these 20 nodes. A chain graph is formed by permuting the nodes and connecting them in succession, while a nearest-neighbor graph is constructed following the procedure outlined in Li and Gui (2006). That is, for each node, we draw a point uniformly at random on a unit square and compute the pairwise distances between nodes. Each node is then connected to $s = 4$ closest neighbors. Since some of nodes will have more than 4 adjacent edges, we randomly remove edges from nodes that have degree larger than 4 until the maximum degree of a node in a graph is 4. Once the graph is created, we construct a precision matrix, with non-zero blocks corresponding to edges in the graph. Elements of diagonal blocks are set as $0.5^{|a-b|}$, $0 \leqslant a, b \leqslant k$, while off-diagonal blocks have elements with the same value, 0.2 for chain graphs and $0.3/k$ for nearest-neighbor graph. Finally, we add ρI to the precision matrix, so that its minimum eigenvalue is equal to 0.5. Note that $s = 2$ for the chain graph and $s = 4$ for the nearest-neighbor graph. Simulation results are averaged over 100 replicates.

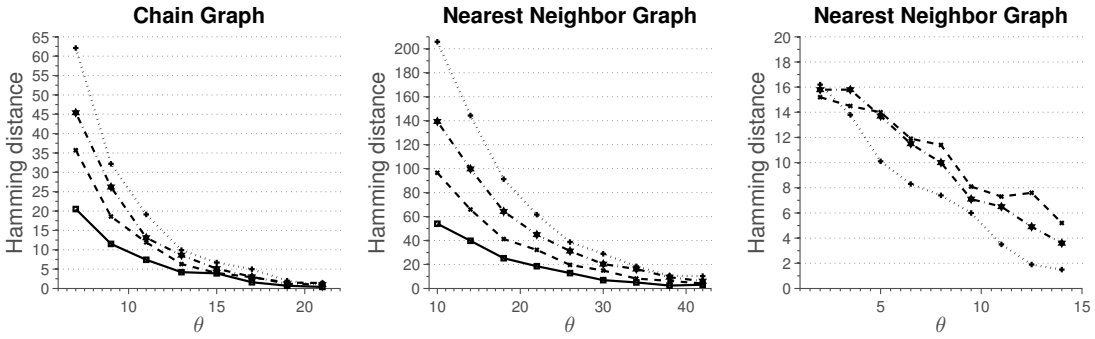
Figure 1 shows simulation results. Each row in the figure reports results for one method, while each column in the figure represents a different simulation setting. For the first two



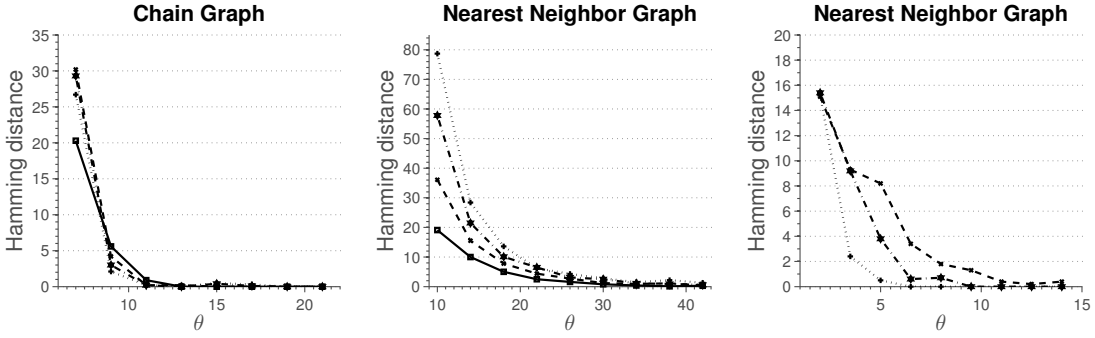
(a) glasso procedure



(b) Procedure of Danaher et al. (2014)



(c) Procedure of Guo et al. (2011)



(d) Multi-attribute procedure

Figure 1: Results of Simulation 1. Average hamming distance plotted against the rescaled sample size. Off-diagonal blocks are full matrices.

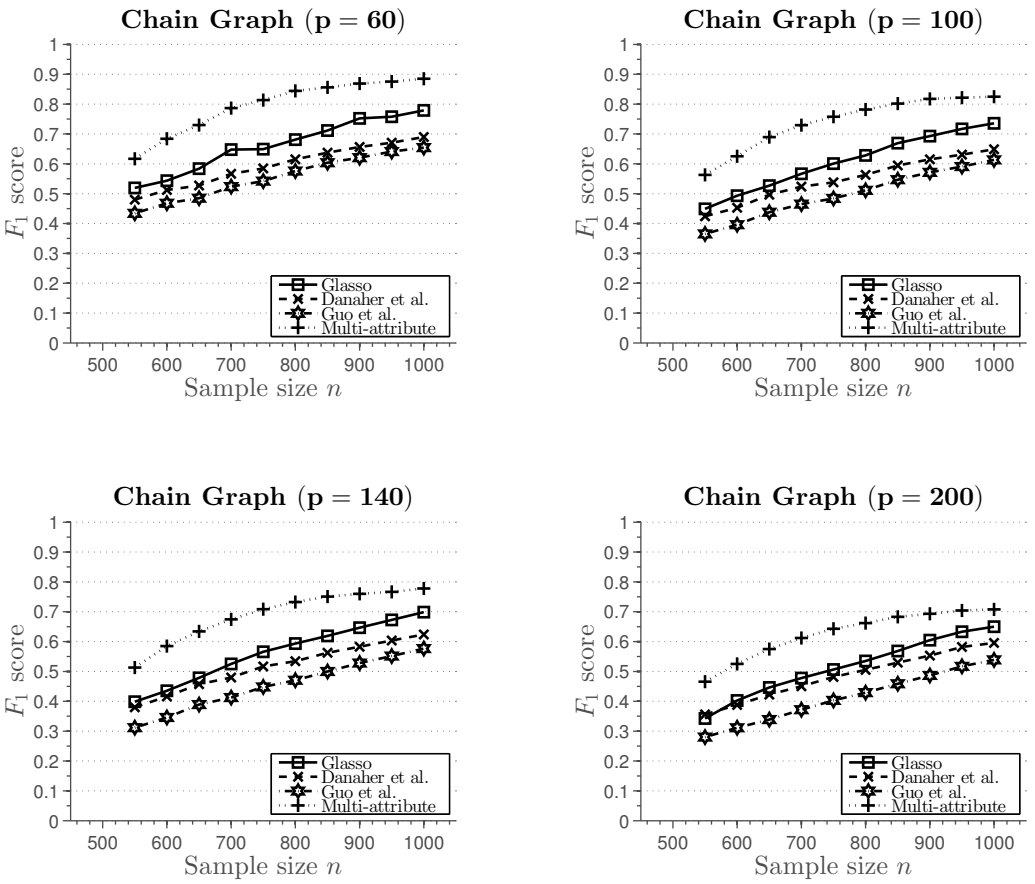


Figure 2: Results of Simulation 1 for smaller sample size. The number of attributes is $k = 2$. Average F_1 score plotted against the sample size.

columns, we set $k = 3$ and vary the total number of nodes in the graph. The third simulation setting sets the total number of nodes $p = 20$ and changes the number of attributes k. In the case of the chain graph, we observe that for small sample sizes the method of Danaher et al. (2014) outperforms all the other methods. We note that the multi-attribute method is estimating many more parameters, which require large sample size in order to achieve high accuracy. However, as the sample size increases, we observe that multi-attribute method starts to outperform the other methods. In particular, for the sample size indexed by $\theta = 13$ all the graph are correctly recovered, while other methods fail to recover the graph consistently at the same sample size. In the case of nearest-neighbor graph, none of the methods recover the graph well for small sample sizes. However, for moderate sample sizes, multi-attribute method outperforms the other methods. Furthermore, as the sample size increases none of the other methods recover the graph exactly. This suggests that the conditions for consistent graph recovery may be weaker in the multi-attribute setting.

From Figure 1 we can observe that for sufficiently large sample size n, multi-attribute method recovers the graph structure exactly. Next, we evaluate performance of the methods

for smaller sample sizes, with the number of attributes $k = 2$. Figure 2 shows average F_1 score plotted against the sample size.[2] This figure shows more precisely performance of the methods for smaller sample sizes that are not sufficient for perfect graph recovery. Again, even though none of the methods perform well, we can observe somewhat better performance of the multi-attribute procedure.

5.1 Alternative Structure of Off-diagonal Blocks

In this section, we investigate performance of different estimation procedures under different assumptions on the elements of the off-diagonal blocks of the precision matrix.

Simulation 2. First, we investigate a situation where the multi-attribute method does not perform as well as the methods that estimate multiple graphical models. One such situation arises when different attributes are conditionally independent. To simulate this situation, we use the data generating approach as before, however, we make each block Ω_{ab} of the precision matrix Ω a diagonal matrix. Figure 3 summarizes results of the simulation. We see that the methods of Danaher et al. (2014) and Guo et al. (2011) perform better, since they are estimating much fewer parameters than the multi-attribute method. glasso does not exploit any structural information underlying the estimation problem and requires larger sample size to correctly estimate the graph than other methods.

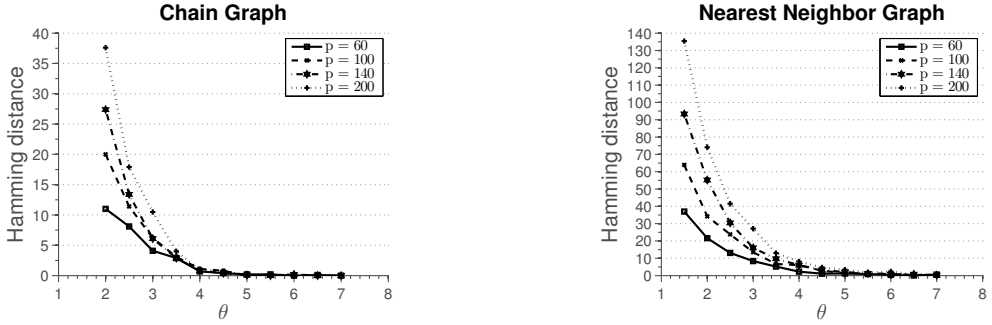
Simulation 3. A completely different situation arises when the edges between nodes can be inferred only based on inter-attribute data, that is, when a graph based on any individual attribute is empty. To generate data under this situation, we follow the procedure as before, but with the diagonal elements of the off-diagonal blocks Ω_{ab} set to zero. Figure 4 summarizes results of the simulation. In this setting, we clearly see the advantage of the multi-attribute method, compared to other three methods. Furthermore, we can see that glasso does better than multi-graph methods of Danaher et al. (2014) and Guo et al. (2011). The reason is that glasso can identify edges based on inter-attribute relationships among nodes, while multi-graph methods rely only on intra-attribute relationships. This simulation illustrates an extreme scenario where inter-attribute relationships are important for identifying edges.

Simulation 4. So far, off-diagonal blocks of the precision matrix were constructed to have constant values. Now, we use the same data generating procedure, but generate off-diagonal blocks of a precision matrix in a different way. Each element of the off-diagonal block Ω_{ab} is generated independently and uniformly from the set $[-0.3, -0.1] \bigcup [0.1, 0.3]$. The results of the simulation are given in Figure 5. Again, qualitatively, the results are similar to those given in Figure 1, except that in this setting more samples are needed to recover the graph correctly.

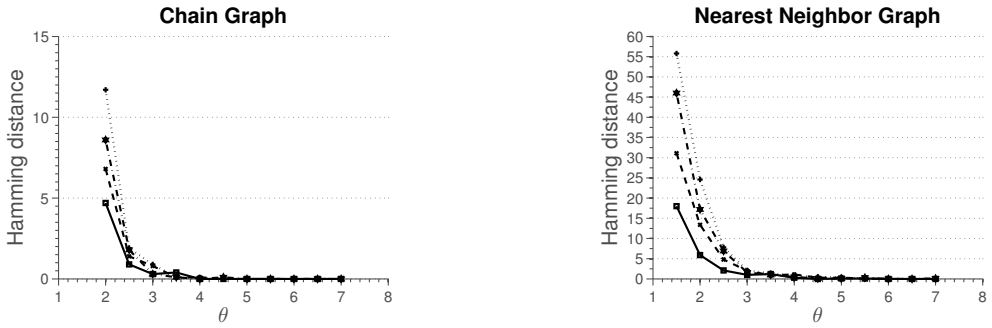
5.2 Different Number of Samples per Attribute

In this section, we show how to deal with a case when different number of samples is available per attribute. As noted in Section 2.2, we can treat non-measured attributes as missing completely at random (see Kolar and Xing, 2012, for more details).

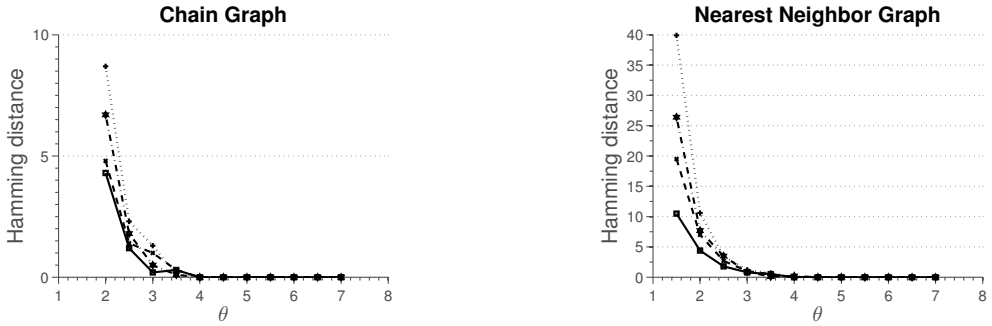
2. The F_1 score is a measure commonly used in information retrieval and is defined as the harmonic mean of precision and recall, that is, $F_1 := 2 * \text{precision} * \text{recall}/(\text{precision} + \text{recall})$. The precision is defined as $\text{precision} := |\widehat{E} \cap E|/|\widehat{E}|$ and the recall is defined as $\text{recall} := |\widehat{E} \cap E|/|E|$.



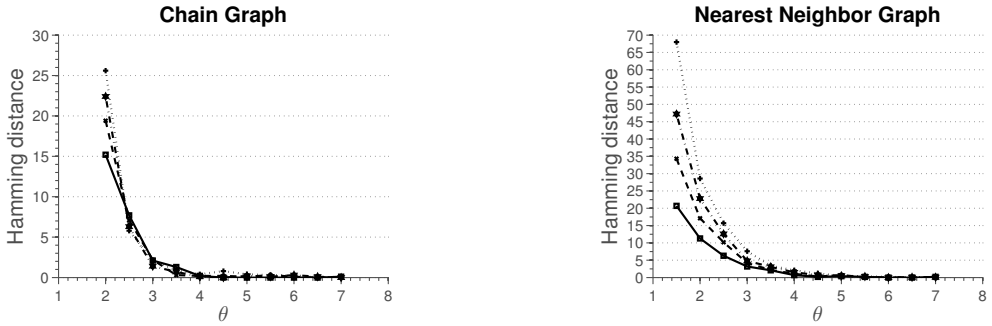
(a) glasso procedure



(b) Procedure of Danaher et al. (2014)

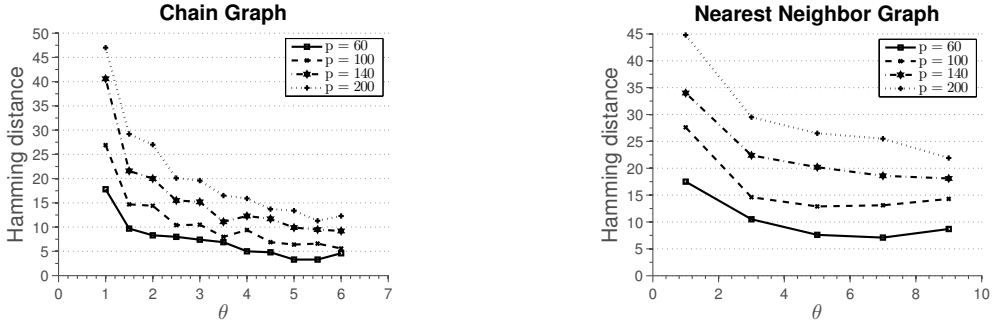


(c) Procedure of Guo et al. (2011)

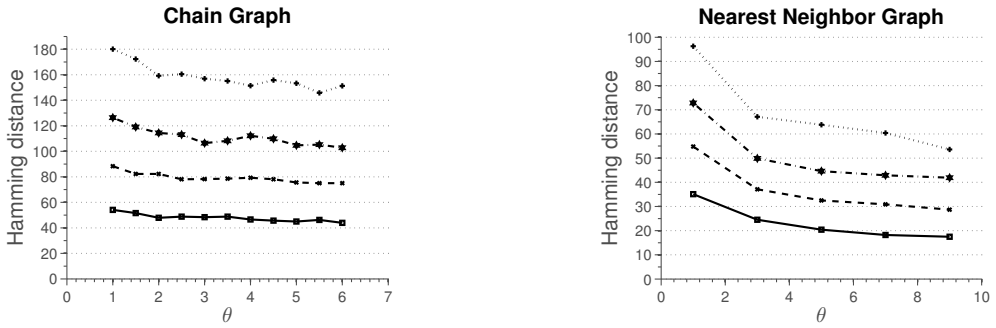


(d) Multi-attribute procedure

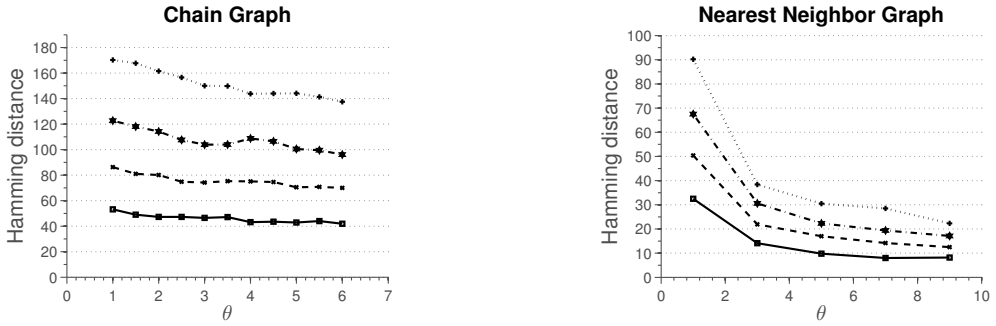
Figure 3: Results of Simulation 2 described in Section 5.1. Average hamming distance plotted against the rescaled sample size. Blocks Ω_{ab} of the precision matrix Ω are diagonal matrices.



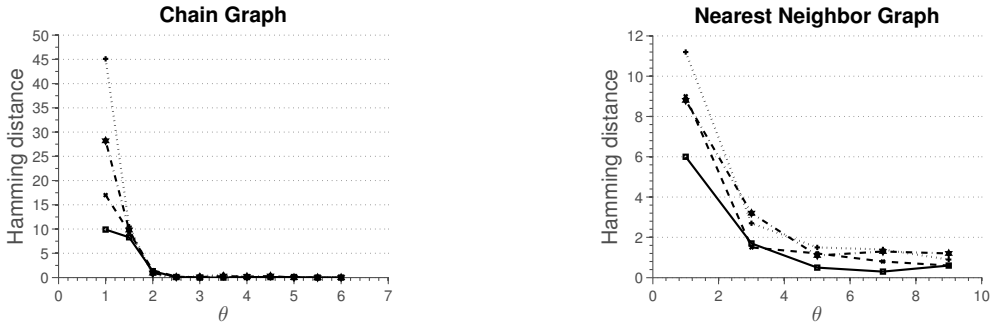
(a) glasso procedure



(b) Procedure of Danaher et al. (2014)

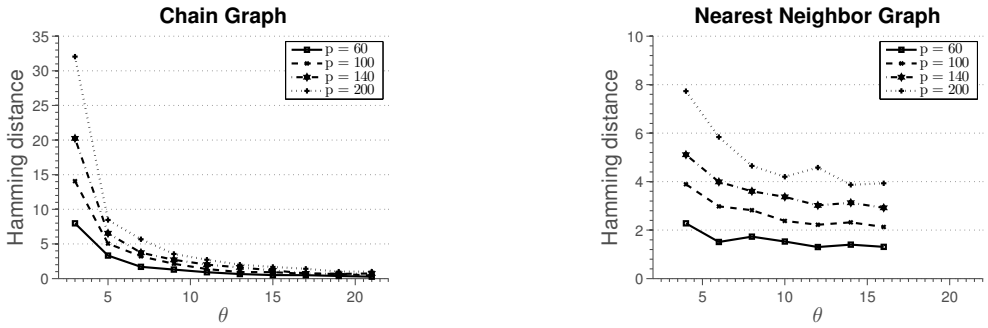


(c) Procedure of Guo et al. (2011)

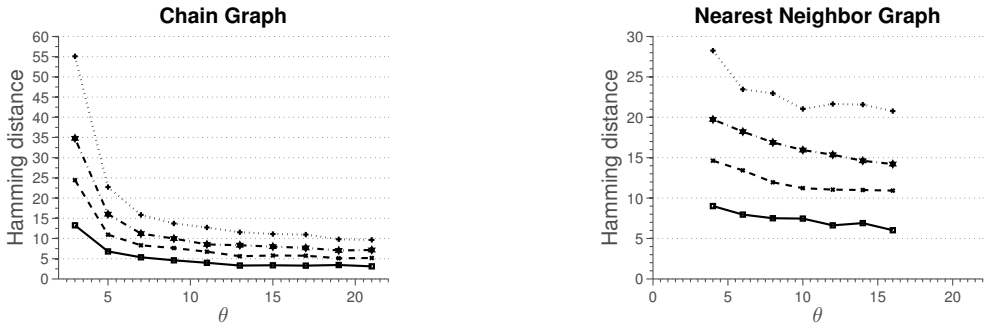


(d) Multi-attribute procedure

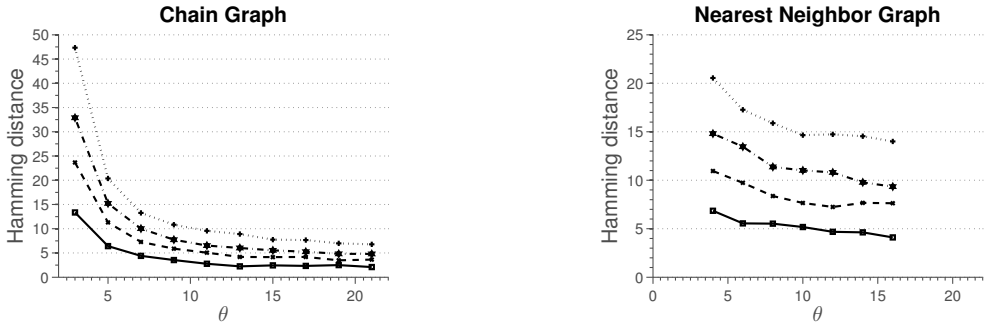
Figure 4: Results of Simulation 3 described in Section 5.1. Average hamming distance plotted against the rescaled sample size. Off-diagonal blocks Ω_{ab} of the precision matrix Ω have zeros as diagonal elements.



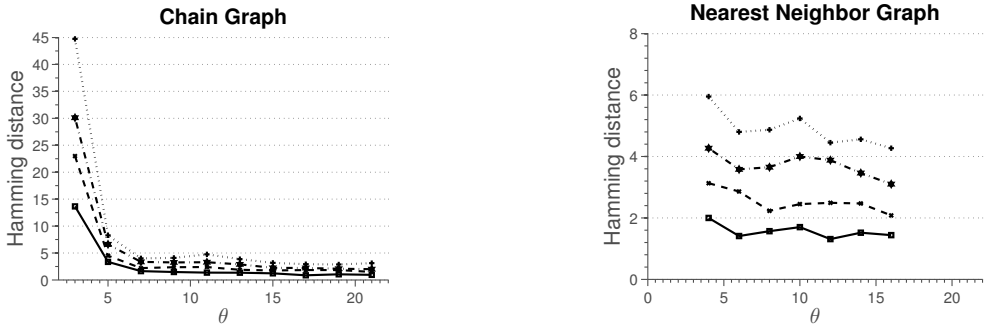
(a) glasso procedure



(b) Procedure of Danaher et al. (2014)



(c) Procedure of Guo et al. (2011)



(d) Multi-attribute procedure

Figure 5: Results of Simulation 4 described in Section 5.1. Average hamming distance plotted against the rescaled sample size. Off-diagonal blocks Ω_{ab} of the precision matrix Ω have elements uniformly sampled from $[-0.3, -0.1] \bigcup [0.1, 0.3]$.

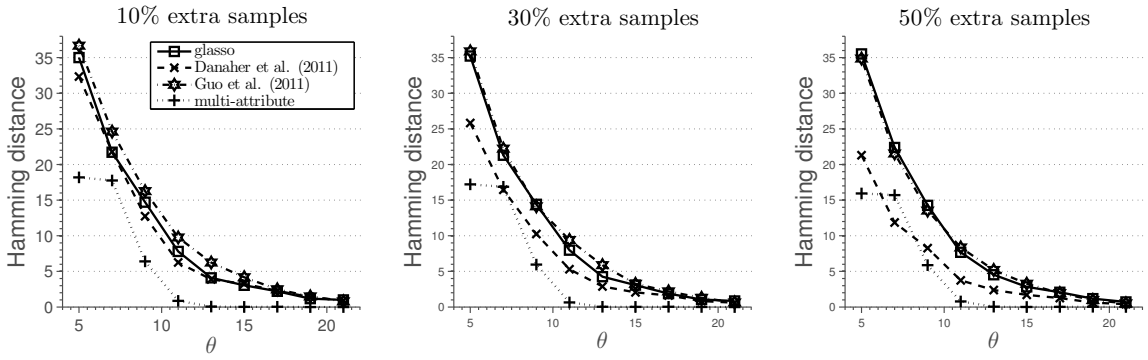


Figure 6: Results of Simulation 5 described in Section 5.2. Average hamming distance plotted against the rescaled sample size. Additional samples available for the first attribute.

Let $R = (r_{il})_{i\in\{1,\ldots,n\},l\in\{1,\ldots,pk\}} \in \mathbb{R}^{n\times pk}$ be an indicator matrix, which denotes for each sample point x_i the components that are observed. Then we can form an estimate of the sample covariance matrix $S = (\sigma_{lk}) \in \mathbb{R}^{pk\times pk}$ as

$$\sigma_{lk} = \frac{\sum_{i=1}^{n} r_{i,l} r_{i,k} x_{i,l} x_{i,k}}{\sum_{i=1}^{n} r_{i,l} r_{i,k}}.$$

This estimate is plugged into the objective in (4).

Simulation 5. We generate a chain graph with $p = 60$ nodes, construct a precision matrix associated with the graph and $k = 3$ attributes, and generate $n = \theta s^2 k^2 \log(pk)$ samples, $\theta > 0$. Next, we generate additional 10%, 30% and 50% samples from the same model, but record only the values for the first attribute. Results of the simulation are given in Figure 6. Qualitatively, the results are similar to those presented in Figure 1.

5.3 Scale-Free Graphs

In this section, we show simulation results when the methods are applied to estimate structure of scale-free graph, that is, graph whose degree distribution follows a power low. A prominent characteristic of these graphs is presence of hub nodes.[3] Such graphs commonly arise in studies of real world systems, such as gene or protein networks (Albert and Barabási, 2002).

Simulation 6. We generate a scale-free graph using the preferential attachment procedure described in Barabási and Albert (1999). The procedure starts with a 4-node cycle. New nodes are added to the graph, one at a time, and connected to nodes currently in the graph with probability proportional to their degree. Once the graph is generated, parameters in the model are set as in Simulation 1, with the number of attributes $k = 2$. Simulation results are summarized in Figure 7. These networks are harder to estimate using the ℓ_1-penalized procedures, due to the presence of high-degree hubs (Peng et al., 2009).

3. Hub nodes are nodes whose degree greatly exceeds average degree in a network.

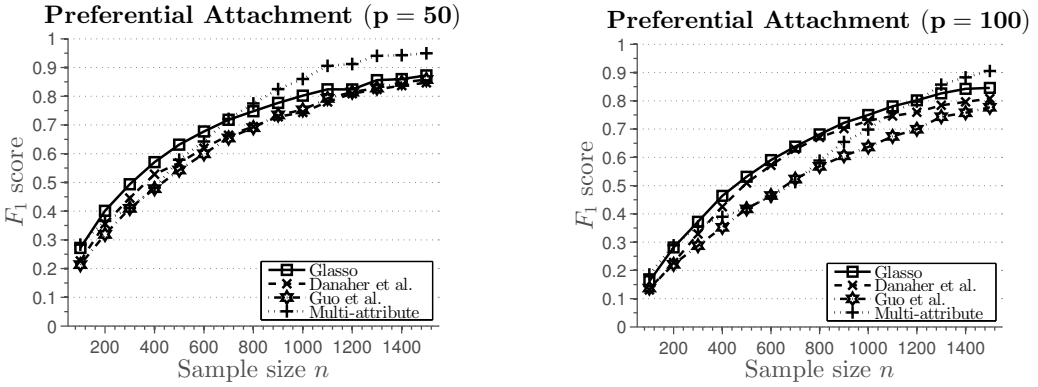


Figure 7: Results of Simulation 6 described in Section 5.3. Average F_1 score plotted against the sample size.

6. Illustrative Applications to Real Data

In this section, we illustrate how to apply our method to data arising in studies of biological regulatory networks and Alzheimer's disease.

6.1 Analysis of a Gene/Protein Regulatory Network

We provide illustrative, exploratory analysis of data from the well-known NCI-60 database, which contains different molecular profiles on a panel of 60 diverse human cancer cell lines. Data set consists of protein profiles (normalized reverse-phase lysate arrays for 92 antibodies) and gene profiles (normalized RNA microarray intensities from Human Genome U95 Affymetrix chip-set for > 9000 genes). We focus our analysis on a subset of 91 genes/proteins for which both types of profiles are available. These profiles are available across the same set of 60 cancer cells. More detailed description of the data set can be found in Katenka and Kolaczyk (2011).

We inferred three types of networks: a network based on protein measurements alone, a network based on gene expression profiles and a single gene/protein network. For protein and gene networks we use the `glasso`, while for the gene/protein network, we use our procedure outlined in Section 2.2. We use the stability selection (Meinshausen and Bühlmann, 2010) procedure to estimate stable networks. In particular, we first select the penalty parameter λ using cross-validation, which over-selects the number of edges in a network. Next, we use the selected λ to estimate 100 networks based on random subsamples containing 80% of the data-points. Final network is composed of stable edges that appear in at least 95 of the estimated networks. Table 1 provides a few summary statistics for the estimated networks. Furthermore, protein and gene/protein networks share 96 edges, while gene and gene/protein networks share 104 edges. Gene and protein network share only 17 edges. Finally, 66 edges are unique to gene/protein network. Figure 8 shows node degree distributions for the three networks. We observe that the estimated networks are much sparser than the association networks in Katenka and Kolaczyk (2011), as expected due to

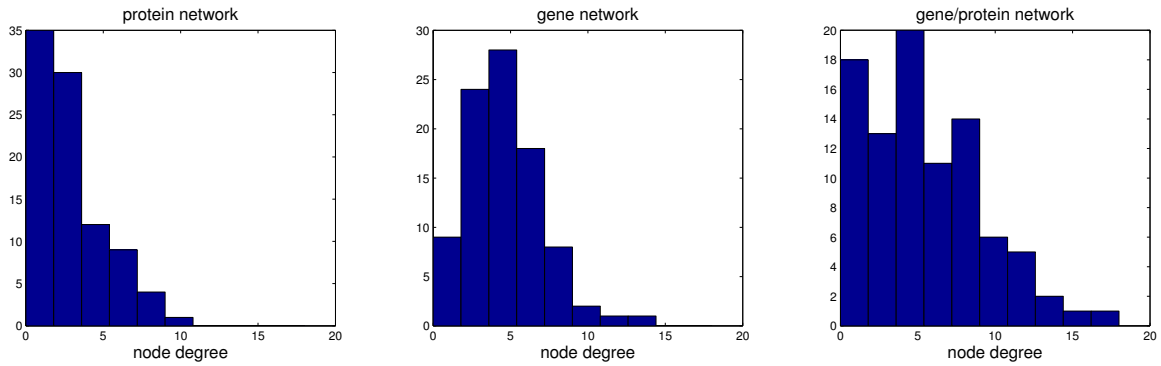


Figure 8: Node degree distributions for protein, gene and gene/protein networks.

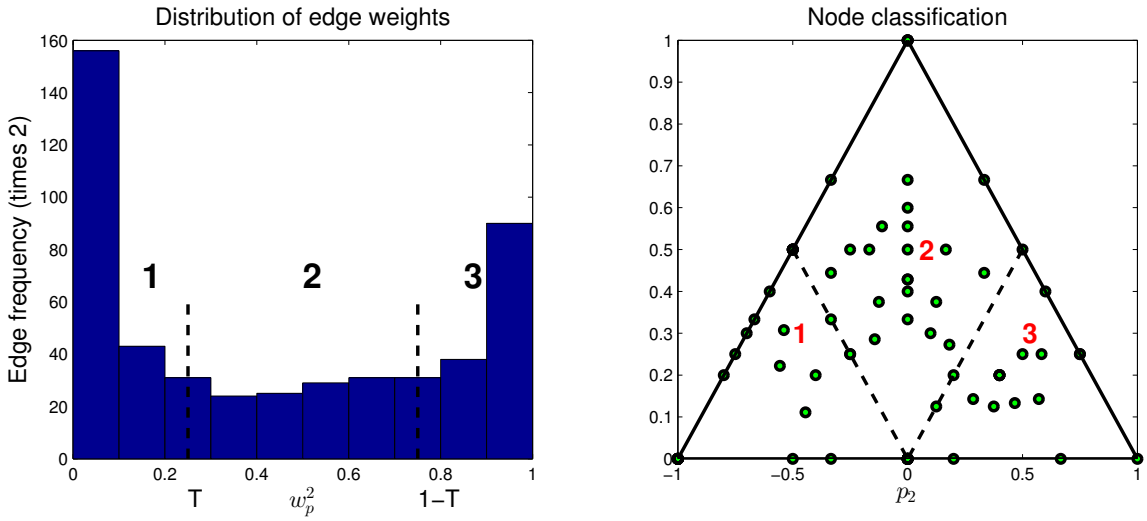


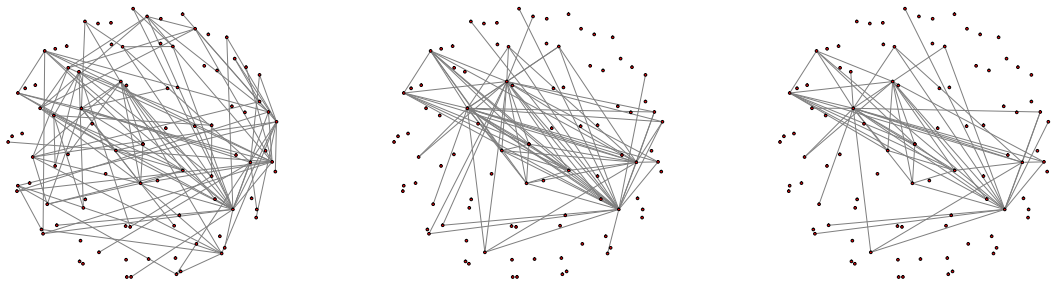
Figure 9: Edge and node classification based on w_p^2.

marginal correlations between a number of nodes. The differences in networks require a closer biological inspection by a domain scientist.

We proceed with a further exploratory analysis of the gene/protein network. We investigate the contribution of two nodal attributes to the existence of an edge between the nodes. Following Katenka and Kolaczyk (2011), we use a simple heuristic based on the

	protein network	gene network	gene/protein network
Number of edges	122	214	249
Density	0.03	0.05	0.06
Largest connected component	62	89	82
Avg Node Degree	2.68	4.70	5.47
Avg Clustering Coefficient	0.0008	0.001	0.003

Table 1: Summary statistics for protein, gene, and gene/protein networks ($p = 91$).



(a) Healthy subjects (b) Mild Cognitive Impairment (c) Alzheimer's & Dementia

Figure 10: Brain connectivity networks

weight vectors to classify the nodes and edges into three classes. For an edge between the nodes a and b, we take one weight vector, say w_a, and normalize it to have unit norm. Denote w_p the component corresponding to the protein attribute. Left plot in Figure 9 shows the values of w_p^2 over all edges. The edges can be classified into three classes based on the value of w_p^2. Given a threshold T, the edges for which $w_p^2 \in (0, T)$ are classified as gene-influenced, the edges for which $w_p^2 \in (1-T, 1)$ are classified as protein influenced, while the remainder of the edges are classified as mixed type. In the left plot of Figure 9, the threshold is set as $T = 0.25$ following Katenka and Kolaczyk (2011). Similar classification can be performed for nodes after computing the proportion of incident edges. Let p_1, p_2 and p_3 denote proportions of gene, protein and mixed edges, respectively, incident with a node. These proportions are represented in a simplex in the right subplot of Figure 9. Nodes with mostly gene edges are located in the lower left corner, while the nodes with mostly protein edges are located in the lower right corner. Mixed nodes are located in the center and towards the top corner of the simplex. Further biological enrichment analysis is possible (see Katenka and Kolaczyk, 2011), however, we do not pursue this here.

6.2 Uncovering Functional Brain Network

We apply our method to the Positron Emission Tomography data set, which contains 259 subjects, of whom 72 are healthy, 132 have mild cognitive Impairment and 55 are diagnosed as Alzheimer's & Dementia. Note that mild cognitive impairment is a transition stage from normal aging to Alzheimer's & Dementia. The data can be obtained from `http://adni.loni.ucla.edu/`. The preprocessing is done in the same way as in Huang et al. (2009).

Each Positron Emission Tomography image contains $91 \times 109 \times 91 = 902,629$ voxels. The effective brain region contains $180,502$ voxels, which are partitioned into 95 regions, ignoring the regions with fewer than 500 voxels. The largest region contains $5,014$ voxels and the smallest region contains 665 voxels. Our preprocessing stage extracts 948 representative voxels from these regions using the K-median clustering algorithm. The parameter K is chosen differently for each region, proportionally to the initial number of voxels in that

	Healthy subjects	Mild Cognitive Impairment	Alzheimer's & Dementia
Number of edges	116	84	59
Density	0.030	0.020	0.014
Largest connected component	48	27	25
Avg Node Degree	2.40	1.73	1.2
Avg Clustering Coefficient	0.001	0.0023	0.0007

Table 2: Summary statistics for brain connectivity networks

region. More specifically, for each category of subjects we have an $n \times (d_1 + \ldots + d_{95})$ matrix, where n is the number of subjects and $d_1 + \ldots + d_{95} = 902,629$ is the number of voxels. Next we set $K_i = \lceil d_i / \sum_j d_j \rceil$, the number of representative voxels in region i, $i = 1, \ldots, 95$. The representative voxels are identified by running the K-median clustering algorithm on a sub-matrix of size $n \times d_i$ with $K = K_i$.

We inferred three networks, one for each subtype of subjects using the procedure outlined in Section 2.2. Note that for different nodes we have different number of attributes, which correspond to medians found by the clustering algorithm. We use the stability selection (Meinshausen and Bühlmann, 2010) approach to estimate stable networks. The stability selection procedure is combined with our estimation procedure as follows. We first select the penalty parameter λ in (4) using cross-validation, which overselects the number of edges in a network. Next, we create 100 subsampled data sets, each of which contain 80% of the data points, and estimate one network for each data set using the selected λ. The final network is composed of stable edges that appear in at least 95 of the estimated networks.

We visualize the estimated networks in Figure 10. Table 2 provides a few summary statistics for the estimated networks. Appendix C contains names of different regions, as well as the adjacency matrices for networks. From the summary statistics, we can observe that in normal subjects there are many more connections between different regions of the brain. Loss of connectivity in Alzheimer's & Dementia has been widely reported in the literature (Greicius et al., 2004; Hedden et al., 2009; Andrews-Hanna et al., 2007; Wu et al., 2011).

Learning functional brain connectivity is potentially valuable for early identification of signs of Alzheimer's disease. Huang et al. (2009) approach this problem using exploratory data analysis. The framework of Gaussian graphical models is used to explore functional brain connectivity. Here we point out that our approach can be used for the same exploratory task, without the need to reduce the information in the whole brain to one number. For example, from our estimates, we observe the loss of connectivity in the cerebellum region of patients with Alzheimer's disease, which has been reported previously in Sjöbeck and Englund (2001). As another example, we note increased connectivity between the frontal lobe and other regions in the patients, which was linked to compensation for the lost connections in other regions (Stern, 2006; Gould et al., 2006).

7. Conclusion and Discussion

This paper extends the classical Gaussian graphical model to handle multi-attribute data. Multi-attribute data appear naturally in social media and scientific data analysis. For example, in a study of social networks, one may use personal information, including demographics, interests, and many other features, as nodal attributes. We proposed a new family of Gaussian graphical models for modeling such multi-attribute data. The main idea is to replace the notion of partial correlation in the existing graphical model literature by *partial canonical correlation*. Such a modification, though simple, has profound impact to both applications and theory. Practically, many challenging data, including brain imaging and gene expression profiles, can be naturally fitted using this model, which has been illustrated in the paper. Theoretically, we proved sufficient conditions that secure the correct recovery of the unknown population network structure.

The methods and theory of this paper can be naturally extended to handle non-Gaussian data by replacing the Gaussian model with the more general nonparanormal model (Liu et al., 2009) or the transelliptical model (Liu et al., 2012). Both of them can be viewed as semiparametric extensions of the Gaussian graphical model. Instead of assuming that data follows a Gaussian distribution, one assumes that there exists a set of strictly increasing univariate functions, so that after marginal transformation the data follows a Gaussian or Elliptical distribution. More details on model interpretation can be found in Liu et al. (2009) and Liu et al. (2012). To handle multi-attribute data in this semiparametric framework, we would replace the sample covariance matrix S in Eq. (4) by a rank-based correlation matrix estimator. We leave the formal analysis of this approach for future work.

Acknowledgments

We thank Eric D. Kolaczyk and Natallia V. Katenka for sharing preprocessed data used in their study with us. Eric P. Xing is partially supported through the grants NIH R01GM087694 and AFOSR FA9550010247. The research of Han Liu is supported by the grants NSF IIS-1116730, NSF III-1332109, NIH R01GM083084, NIH R01HG06841, and FDA HHSF223201000072C. This work was completed in part with resources provided by the University of Chicago Research Computing Center. We are grateful to the editor and three anonymous reviewers for helping us improve the manuscript.

Appendix A. Complexity Analysis of Multi-attribute Estimation

Step 2 of the estimation algorithm updates portions of the precision and covariance matrices corresponding to one node at a time. We observe that the computational complexity of updating the precision matrix is $\mathcal{O}\left(pk^2\right)$. Updating the covariance matrix requires computing $(\widetilde{\Omega}_{\overline{a},\overline{a}})^{-1}$, which can be efficiently done in $\mathcal{O}\left(p^2k^2 + pk^2 + k^3\right) = \mathcal{O}\left(p^2k^2\right)$ operations, assuming that $k \ll p$. With this, the covariance matrix can be updated in $\mathcal{O}\left(p^2k^2\right)$ operations. Therefore the total cost of updating the covariance and precision matrices is $\mathcal{O}\left(p^2k^2\right)$ operations. Since step 2 needs to be performed for each node $a \in V$, the total complexity is $\mathcal{O}\left(p^3k^2\right)$. Let T denote the total number of times step 2 is executed. This leads to the overall complexity of the algorithm as $\mathcal{O}\left(Tp^3k^2\right)$. In practice, we observe that $T \approx 10$ to 20 for

sparse graphs. Furthermore, when the whole solution path is computed, we can use warm starts to further speed up computation, leading to $T < 5$ for each λ.

Appendix B. Technical Proofs

In this appendix, we collect proofs of the results presented in the main part of the paper.

B.1 Proof of Lemma 1

We start the proof by giving to technical results needed later. The following lemma states that the minimizer of (4) is unique and has bounded minimum and maximum eigenvalues, denoted as $\Lambda_{\min}$ and $\Lambda_{\max}$.

Lemma 5 *For every value of $\lambda > 0$, the optimization problem in Eq. (4) has a unique minimizer $\widehat{\Omega}$, which satisfies $\Lambda_{\min}(\widehat{\Omega}) \geqslant (\Lambda_{\max}(S) + \lambda p)^{-1} > 0$ and $\Lambda_{\max}(\widehat{\Omega}) \leqslant \lambda^{-1} \sum_{j \in V} k_j$.*

Proof The optimization objective given in (4) can be written in the equivalent constrained form as

$$\min_{\Omega \succ 0} \ \operatorname{tr} S\Omega - \log |\Omega| \quad \text{subject to} \quad \sum_{a,b} ||\Omega_{ab}||_F \leqslant C(\lambda).$$

The procedure involves minimizing a continuous objective over a compact set, and so by Weierstrass theorem, the minimum is always achieved. Furthermore, the objective is strongly convex and therefore the minimum is unique.

The solution $\widehat{\Omega}$ to the optimization problem (4) satisfies

$$S - \widehat{\Omega}^{-1} + \lambda Z = 0, \tag{14}$$

where $Z \in \partial \sum_{a,b} ||\widehat{\Omega}_{ab}||_F$ is the element of the sub-differential and satisfies $||Z_{ab}||_F \leqslant 1$ for all $(a, b) \in V^2$. Therefore,

$$\Lambda_{\max}(\widehat{\Omega}^{-1}) \leqslant \Lambda_{\max}(S) + \lambda \Lambda_{\max}(Z) \leqslant \Lambda_{\max}(S) + \lambda p.$$

Next, we prove an upper bound on $\Lambda_{\max}(\widehat{\Omega})$. At optimum, the primal-dual gap is zero, which gives that

$$\sum_{a,b} ||\widehat{\Omega}_{ab}||_F \leqslant \lambda^{-1} (\sum_{j \in V} k_j - \operatorname{tr} S\widehat{\Omega}) \leqslant \lambda^{-1} \sum_{j \in V} k_j,$$

as $S \succeq 0$ and $\widehat{\Omega} \succ 0$. Since $\Lambda_{\max}(\widehat{\Omega}) \leqslant \sum_{a,b} ||\widehat{\Omega}_{ab}||_F$, the proof is done. ■

The next results states that the objective function has a Lipschitz continuous gradient, which will be used to show that the generalized gradient descent can be used to find $\widehat{\Omega}$.

Lemma 6 *The function $f(A) = \operatorname{tr} SA - \log |A|$ has a Lipschitz continuous gradient on the set $\{A \in \mathcal{S}^p \ : \ \Lambda_{\min}(A) \geqslant \gamma\}$, with the Lipschitz constant $L = \gamma^{-2}$.*

Proof We have that $\nabla f(A) = S - A^{-1}$. Then

$$\begin{aligned} ||\nabla f(A) - \nabla f(A')||_F &= ||A^{-1} - (A')^{-1}||_F \\ &\leqslant \Lambda_{\max} A^{-1} ||A - A'||_F \Lambda_{\max} A^{-1} \\ &\leqslant \gamma^{-2} ||A - A'||_F, \end{aligned}$$

which completes the proof. ■

Now, we provide the proof of Lemma 1.

By construction, the sequence of estimates $(\widetilde{\Omega}^{(t)})_{t\geqslant 1}$ decrease the objective value and are positive definite.

To prove the convergence, we first introduce some additional notation. Let $f(\Omega) = \operatorname{tr} S\Omega - \log|\Omega|$ and $F(\Omega) = f(\Omega) + \sum_{ab}||\Omega_{ab}||_F$. For any $L > 0$, let

$$Q_L(\Omega;\overline{\Omega}) := f(\overline{\Omega}) + \operatorname{tr}[(\Omega-\overline{\Omega})\nabla f(\overline{\Omega})] + \frac{L}{2}||\Omega-\overline{\Omega}||_F^2 + \sum_{ab}||\Omega_{ab}||_F$$

be a quadratic approximation of $F(\Omega)$ at a given point $\overline{\Omega}$, which has a unique minimizer

$$p_L(\overline{\Omega}) := \arg\min_{\Omega} Q_L(\Omega;\overline{\Omega}).$$

From Lemma 2.3. in Beck and Teboulle (2009), we have that

$$F(\overline{\Omega}) - F(p_L(\overline{\Omega})) \geqslant \frac{L}{2}||p_L(\overline{\Omega})-\overline{\Omega}||_F^2, \tag{15}$$

if $F(p_L(\overline{\Omega})) \leqslant Q_L(p_L(\overline{\Omega});\overline{\Omega})$. Note that $F(p_L(\overline{\Omega})) \leqslant Q_L(p_L(\overline{\Omega});\overline{\Omega})$ always holds if L is as large as the Lipschitz constant of ∇F.

Let $\widetilde{\Omega}^{(t-1)}$ and $\widetilde{\Omega}^{(t)}$ denote two successive iterates obtained by the procedure. Without loss of generality, we can assume that $\widetilde{\Omega}^{(t)}$ is obtained by updating the rows/columns corresponding to the node a. From (15), it follows that

$$\frac{2}{L_k}(F(\widetilde{\Omega}^{(t-1)}) - F(\widetilde{\Omega}^{(t)})) \geqslant ||\widetilde{\Omega}^{(t-1)}_{aa} - \widetilde{\Omega}^{(t)}_{aa}||_F + 2\sum_{b\neq a}||\widetilde{\Omega}^{(t-1)}_{ab} - \widetilde{\Omega}^{(t)}_{ab}||_F, \tag{16}$$

where L_k is a current estimate of the Lipschitz constant. Recall that in our procedure the scalar t serves as a local approximation of $1/L$. Since eigenvalues of $\widehat{\Omega}$ are bounded according to Lemma 5, we can conclude that the eigenvalues of $\widetilde{\Omega}^{(t-1)}$ are bounded as well. Therefore the current Lipschitz constant is bounded away from zero, using Lemma 6. Combining the results, we observe that the right hand side of (16) converges to zero as $t \to \infty$, since the optimization procedure produces iterates that decrease the objective value. This shows that $||\widetilde{\Omega}^{(t-1)}_{aa} - \widetilde{\Omega}^{(t)}_{aa}||_F + 2\sum_{b\neq a}||\widetilde{\Omega}^{(t-1)}_{ab} - \widetilde{\Omega}^{(t)}_{ab}||_F$ converges to zero, for any $a \in V$. Since $(\widetilde{\Omega}^{(t)}$ is a bounded sequence, it has a limit point, which we denote $\widehat{\Omega}$. It is easy to see, from the stationary conditions for the optimization problem given in (6), that the limit point $\widehat{\Omega}$ also satisfies the global KKT conditions to the optimization problem in (4).

B.2 Proof of Lemma 3

Suppose that the solution $\widehat{\Omega}$ to (4) is block diagonal with blocks $P_1, P_2, \ldots, P_l$. For two nodes a, b in different blocks, we have that $(\widehat{\Omega})^{-1}_{ab} = 0$ as the inverse of the block diagonal matrix is block diagonal. From the KKT conditions, it follows that $||S_{ab}||_F \leqslant \lambda$.

Now suppose that $||S_{ab}||_F \leqslant \lambda$ for all $a \in P_j, b \in P_{j'}, j \neq j'$. For every $l' = 1, \ldots, l$ construct

$$\widetilde{\Omega}_{l'} = \arg\min_{\Omega_{l'}>0} \operatorname{tr} S_{l'}\Omega_{l'} - \log|\Omega_{l'}| + \lambda\sum_{a,b}||\Omega_{ab}||_F.$$

Then $\widehat{\Omega} = \text{diag}(\widehat{\Omega}_1, \widehat{\Omega}_2, \ldots, \widehat{\Omega}_l)$ is the solution of (4) as it satisfies the KKT conditions.

B.3 Proof of Eq. (3)

First, we note that

$$\text{var}\left((X_a^T, X_b^T)^T \mid X_{\overline{ab}}\right) = \Sigma_{ab,ab} - \Sigma_{ab,\overline{ab}}\Sigma^{-1}_{\overline{ab},\overline{ab}}\Sigma_{\overline{ab},ab}$$

is the conditional covariance matrix of $(X_a^T, X_b^T)^T$ given the remaining nodes $X_{\overline{ab}}$ (see Proposition C.5 in Lauritzen (1996)). Define $\overline{\Sigma} = \Sigma_{ab,ab} - \Sigma_{ab,\overline{ab}}\Sigma^{-1}_{\overline{ab},\overline{ab}}\Sigma_{\overline{ab},ab}$. Partial canonical correlation between X_a and X_b is equal to zero if and only if $\overline{\Sigma}_{ab} = 0$. On the other hand, the matrix inversion lemma gives that $\Omega_{ab,ab} = \overline{\Sigma}^{-1}$. Now, $\Omega_{ab} = 0$ if and only if $\overline{\Sigma}_{ab} = 0$. This shows the equivalence relationship in Eq. (3).

B.4 Proof of Proposition 4

We provide sufficient conditions for consistent network estimation. Proposition 4 given in Section 3 is then a simple consequence. To provide sufficient conditions, we extend the work of Ravikumar et al. (2011) to our setting, where we observe multiple attributes for each node. In particular, we extend their Theorem 1.

For simplicity of presentation, we assume that $k_a = k$, for all $a \in V$, that is, we assume that the same number of attributes is observed for each node. Our assumptions involve the Hessian of the function $f(A) = \text{tr}\, SA - \log|A|$ evaluated at the true Ω^*,

$$\mathcal{H} = \mathcal{H}(\Omega^*) = (\Omega^*)^{-1} \otimes (\Omega^*)^{-1} \in \mathbb{R}^{(pk)^2 \times (pk)^2}, \tag{17}$$

and the true covariance matrix Σ^*. The Hessian and the covariance matrix can be thought of block matrices with blocks of size $k^2 \times k^2$ and $k \times k$, respectively. We will make use of the operator $\mathcal{C}(\cdot)$ that operates on these block matrices and outputs a smaller matrix with elements that equal to the Frobenius norm of the original blocks,

$$\begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1p} \\ A_{21} & A_{22} & \cdots & A_{2p} \\ \vdots & & \ddots & \vdots \\ A_{p1} & \cdots & & A_{pp} \end{pmatrix} \xrightarrow{\mathcal{C}(\cdot)} \begin{pmatrix} ||A_{11}||_F & ||A_{12}||_F & \cdots & ||A_{1p}||_F \\ ||A_{21}||_F & ||A_{22}||_F & \cdots & ||A_{2p}||_F \\ \vdots & & \ddots & \vdots \\ ||A_{p1}||_F & \cdots & & ||A_{pp}||_F \end{pmatrix}.$$

In particular, $\mathcal{C}(\Sigma^*) \in \mathbb{R}^{p \times p}$ and $\mathcal{C}(\mathcal{H}) \in \mathbb{R}^{p^2 \times p^2}$.

We denote the index set of the non-zero blocks of the precision matrix as

$$\mathcal{T} := \{(a,b) \in V \times V \ : \ ||\Omega^*_{ab}||_2 \neq 0\} \cup \{(a,a) \ : \ a \in V\}$$

and let $\mathcal{N}$ denote its complement in $V \times V$, that is,

$$\mathcal{N} = \{(a,b) \ : \ ||\Omega_{ab}||_F = 0\}.$$

As mentioned earlier, we need to make an assumption on the Hessian matrix, which takes the standard irrepresentable-like form. There exists a constant $\alpha \in [0,1)$ such that

$$|||\mathcal{C}\left(\mathcal{H}_{\mathcal{N}\mathcal{T}}(\mathcal{H}_{\mathcal{T}\mathcal{T}})^{-1}\right)|||_\infty \leqslant 1 - \alpha. \tag{18}$$

These condition extends the irrepresentable condition given in Ravikumar et al. (2011), which was needed for estimation of networks from single attribute observations. It is worth noting, that the condition given in Eq. (18) can be much weaker than the irrepresentable condition of Ravikumar et al. (2011) applied directly to the full Hessian matrix. This can be observed in simulations done in Section 5, where a chain network is not consistently estimated even with a large number of samples.

We will also need the following two quantities to specify the results

$$\kappa_{\Sigma^*} = |||\mathcal{C}(\Sigma^*)|||_\infty, \tag{19}$$

and

$$\kappa_{\mathcal{H}} = |||\mathcal{C}(\mathcal{H}_{\mathcal{T}\mathcal{T}}^{-1})|||_\infty. \tag{20}$$

Finally, the results are going to depend on the tail bounds for the elements of the matrix $\mathcal{C}(S - \Sigma^*)$. We will assume that there is a constant $v_* \in (0, \infty]$ and a function $f : \mathbb{N} \times (0, \infty) \mapsto (0, \infty)$ such that for any $(a, b) \in V \times V$

$$\mathrm{pr}\left(\mathcal{C}(S - \Sigma^*)_{ab} \geqslant \delta\right) \leqslant \frac{1}{f(n, \delta)} \qquad \delta \in (0, v_*^{-1}]. \tag{21}$$

The function $f(n, \delta)$ will be monotonically increasing in both n and δ. Therefore, we define the following two inverse functions

$$\overline{n}_f(\delta; r) = \arg\max\{n \ : \ f(n, \delta) \leqslant r\} \tag{22}$$

and

$$\overline{\delta}_f(r; n) = \arg\max\{\delta \ : \ f(n, \delta) \leqslant r\} \tag{23}$$

for $r \in [1, \infty)$.

With the notation introduced, we have the following result.

Theorem 7 *Assume that the irrepresentable condition in Eq.* (18) *is satisfied and that there exists a constant* $v_* \in (0, \infty]$ *and a function* $f(n, \delta)$ *so that Eq.* (21) *is satisfied for any* $(a, b) \in V \times V$. *Let*

$$\lambda = \frac{8}{\alpha}\overline{\delta}_f(n, p^\tau)$$

for some $\tau > 2$. *If*

$$n > \overline{n}_f\left(\frac{1}{\max(v_*, 6(1 + 8\alpha^{-1})s\max(\kappa_{\Sigma^*}\kappa_{\mathcal{H}}, \kappa_{\Sigma^*}^3\kappa_{\mathcal{H}}^2))}, p^\tau\right), \tag{24}$$

then

$$||\mathcal{C}(\widehat{\Omega} - \Omega)||_\infty \leqslant 2(1 + 8\alpha^{-1})\kappa_{\mathcal{H}}\overline{\delta}_f(n, p^\tau) \tag{25}$$

with probability at least $1 - p^{2-\tau}$.

Theorem 7 is of the same form as Theorem 1 in Ravikumar et al. (2011), but the ℓ_∞ element-wise convergence is established for $\mathcal{C}(\widehat{\Omega} - \Omega)$, which will guarantee successful recovery of non-zero partial canonical correlations if the blocks of the true precision matrix are sufficiently large.

Theorem 7 is proven as Theorem 1 in Ravikumar et al. (2011). We provide technical results in Lemma 8, Lemma 9 and Lemma 10, which can be used to substitute results of Lemma 4, Lemma 5 and Lemma 6 in Ravikumar et al. (2011) under our setting. The rest of the arguments then go through. Below we provide some more details.

First, let $\mathcal{Z} : \mathbb{R}^{pk\times pk} \mapsto \mathbb{R}^{pk\times pk}$ be the mapping defined as

$$\mathcal{Z}(A)_{ab} = \begin{cases} \frac{A_{ab}}{||A_{ab}||_F} & \text{if } ||A_{ab}||_F \neq 0, \\ Z \text{ with } ||Z||_F \leqslant 1 & \text{if } ||A_{ab}||_F = 0, \end{cases} \tag{26}$$

Next, define the function

$$G(\Omega) = \operatorname{tr}\Omega S - \log|\Omega| + \lambda||\mathcal{C}(\Omega)||_1, \quad \forall \Omega \succ 0 \tag{27}$$

and the following system of equations

$$\begin{cases} S_{ab} - (\Omega^{-1})_{ab} = -\lambda\mathcal{Z}(\Omega)_{ab}, & \text{if } \Omega_{ab} \neq 0 \\ ||S_{ab} - (\Omega^{-1})_{ab}||_F \leqslant \lambda, & \text{if } \Omega_{ab} = 0. \end{cases} \tag{28}$$

It is known that $\Omega \in \mathbb{R}^{\widetilde{p}\times\widetilde{p}}$ is the minimizer of optimization problem in Eq. (4) if and only if it satisfies the system of equations given in Eq. (28). We have already shown in Lemma 5 that the minimizer is unique.

Let $\widetilde{\Omega}$ be the solution to the following constrained optimization problem

$$\min_{\Omega\succ 0} \operatorname{tr} S\Omega - \log|\Omega| + \lambda||\mathcal{C}(\Omega)||_1 \text{ subject to } \mathcal{C}(\Omega)_{ab} = 0, \ \forall (a,b) \in \mathcal{N}. \tag{29}$$

Observe that one cannot find $\widetilde{\Omega}$ in practice, as it depends on the unknown set $\mathcal{N}$. However, it is a useful construction in the proof. We will prove that $\widetilde{\Omega}$ is solution to the optimization problem given in Eq. (4), that is, we will show that $\widetilde{\Omega}$ satisfies the system of equations (28).

Using the first-order Taylor expansion we have that

$$\widetilde{\Omega}^{-1} = (\Omega^*)^{-1} - (\Omega^*)^{-1}\Delta(\Omega^*)^{-1} + R(\Delta), \tag{30}$$

where $\Delta = \Omega - \Omega^*$ and $R(\Delta)$ denotes the remainder term. With this, we state and prove Lemma 8, Lemma 9 and Lemma 10. They can be combined as in Ravikumar et al. (2011) to complete the proof of Theorem 7.

Lemma 8 *Assume that*

$$\max_{ab}||\Delta_{ab}||_F \leqslant \frac{\alpha\lambda}{8} \quad \text{and} \quad \max_{ab}||\Sigma^*_{ab} - S_{ab}||_F \leqslant \frac{\alpha\lambda}{8}. \tag{31}$$

Then $\widetilde{\Omega}$ is the solution to the optimization problem in Eq. (4)*.*

Proof We use R to denote $R(\Delta)$. Recall that $\Delta_{\mathcal{N}} = 0$ by construction. Using (30) we can rewrite (28) as

$$\mathcal{H}_{ab,\mathcal{T}}\overline{\Delta}_{\mathcal{T}} - \overline{R}_{ab} + \overline{S}_{ab} - \overline{\Sigma}^*_{ab} + \lambda\overline{\mathcal{Z}}(\widetilde{\Omega})_{ab} = 0 \qquad \text{if } (a,b) \in \mathcal{T} \tag{32}$$

$$||\mathcal{H}_{ab,\mathcal{T}}\overline{\Delta}_{\mathcal{T}} - \overline{R}_{ab} + \overline{S}_{ab} - \overline{\Sigma}^*_{ab}||_2 \leqslant \lambda \qquad \text{if } (a,b) \in \mathcal{N}. \tag{33}$$

By construction, the solution $\widetilde{\Omega}$ satisfy (32). Under the assumptions, we show that (33) is also satisfied with inequality.

From (32), we can solve for $\Delta_{\mathcal{T}}$,

$$\Delta_{\mathcal{T}} = \mathcal{H}^{-1}_{\mathcal{T},\mathcal{T}}[\overline{R}_{\mathcal{T}} - \overline{\Sigma}_{\mathcal{T}} + \overline{S}_{\mathcal{T}} - \lambda\overline{\mathcal{Z}}(\widetilde{\Omega})_{\mathcal{T}}].$$

Then

$$\begin{aligned}
&||\mathcal{H}_{ab,\mathcal{T}}\mathcal{H}^{-1}_{\mathcal{T},\mathcal{T}}[\overline{R}_{\mathcal{T}} - \overline{\Sigma}_{\mathcal{T}} + \overline{S}_{\mathcal{T}} - \lambda\overline{\mathcal{Z}}(\widetilde{\Omega})_{\mathcal{T}}] - \overline{R}_{ab} + \overline{S}_{ab} - \overline{\Sigma}^*_{ab}||_2 \\
&\quad \leqslant \lambda||\mathcal{H}_{ab,\mathcal{T}}\mathcal{H}^{-1}_{\mathcal{T},\mathcal{T}}\overline{\mathcal{Z}}(\widetilde{\Omega})_{\mathcal{T}}||_2 + ||\mathcal{H}_{ab,\mathcal{T}}\mathcal{H}^{-1}_{\mathcal{T},\mathcal{T}}[\overline{R}_{\mathcal{T}} - \overline{\Sigma}_{\mathcal{T}} + \overline{S}_{\mathcal{T}}]||_2 + ||\overline{R}_{ab} + \overline{S}_{ab} - \overline{\Sigma}^*_{ab}||_2 \\
&\quad \leqslant \lambda(1-\alpha) + (2-\alpha)\frac{\alpha\lambda}{4} \\
&\quad < \lambda
\end{aligned}$$

using assumption on $\mathcal{H}$ in (18) and (31). This shows that $\widetilde{\Omega}$ satisfies (28). ■

Lemma 9 *Assume that*

$$||\mathcal{C}(\Delta)||_\infty \leqslant \frac{1}{3\kappa_{\Sigma^*}s}. \tag{34}$$

Then

$$||\mathcal{C}(R(\Delta))||_\infty \leqslant \frac{3s}{2}\kappa^3_{\Sigma^*}||\mathcal{C}(\Delta)||^2_\infty. \tag{35}$$

Proof Remainder term can be written as

$$R(\Delta) = (\Omega^* + \Delta)^{-1} - (\Omega^*)^{-1} + (\Omega^*)^{-1}\Delta(\Omega^*)^{-1}.$$

Using (40), we have that

$$\begin{aligned}
|||\mathcal{C}((\Omega^*)^{-1}\Delta)|||_\infty &\leqslant |||\mathcal{C}((\Omega^*)^{-1})|||_\infty|||\mathcal{C}(\Delta)|||_\infty \\
&\leqslant s|||\mathcal{C}((\Omega^*)^{-1})|||_\infty||\mathcal{C}(\Delta)||_\infty \\
&\leqslant \frac{1}{3},
\end{aligned}$$

which gives us the following expansion

$$(\Omega^* + \Delta)^{-1} = (\Omega^*)^{-1} - (\Omega^*)^{-1}\Delta(\Omega^*)^{-1} + (\Omega^*)^{-1}\Delta(\Omega^*)^{-1}\Delta J(\Omega^*)^{-1},$$

with $J = \sum_{k\geqslant 0}(-1)^k((\Omega^*)^{-1}\Delta)^k$. Using (41) and (40), we have that

$$\begin{aligned}
||\mathcal{C}(R)||_\infty &\leqslant ||\mathcal{C}((\Omega^*)^{-1}\Delta)||_\infty|||\mathcal{C}((\Omega^*)^{-1}\Delta J(\Omega^*)^{-1})^T|||_\infty \\
&\leqslant |||\mathcal{C}((\Omega^*)^{-1})|||^3_\infty||\mathcal{C}(\Delta)||_\infty|||\mathcal{C}(J^T)|||_\infty|||\mathcal{C}(\Delta)|||_\infty \\
&\leqslant s|||\mathcal{C}((\Omega^*)^{-1})|||^3_\infty||\mathcal{C}(\Delta)||^2_\infty|||\mathcal{C}(J^T)|||_\infty.
\end{aligned}$$

Next, we have that

$$\begin{aligned}\|\mathcal{C}(J^T)\|_\infty &\leqslant \sum_{k>0} \|\mathcal{C}(\Delta(\Omega^*)^{-1})\|_\infty^k \\ &\leqslant \frac{1}{1-\|\mathcal{C}(\Delta(\Omega^*)^{-1})\|_\infty} \\ &\leqslant \frac{3}{2},\end{aligned}$$

which gives us

$$||\mathcal{C}(R)||_\infty \leqslant \frac{3s}{2}\kappa_{\Sigma^*}^3 ||\mathcal{C}(\Delta)||_\infty^2$$

as claimed. ■

Lemma 10 *Assume that*

$$r := 2\kappa_{\mathcal{H}}(||\mathcal{C}(S-\Sigma^*)||_\infty + \lambda) \leqslant \min\left(\frac{1}{3\kappa_{\Sigma^*}s}, \frac{1}{3\kappa_{\mathcal{H}}\kappa_{\Sigma^*}^3 s}\right). \tag{36}$$

Then

$$||\mathcal{C}(\Delta)||_\infty \leqslant r. \tag{37}$$

Proof The proof follows the proof of Lemma 6 in Ravikumar et al. (2011). Define the ball

$$\mathcal{B}(r) := \{A \ : \ \mathcal{C}(A)_{ab} \leqslant r, \forall (a,b) \in \mathcal{T}\},$$

the gradient mapping

$$G(\Omega_{\mathcal{T}}) = -(\Omega^{-1})_{\mathcal{T}} + S_{\mathcal{T}} + \lambda \mathcal{Z}(\Omega)_{\mathcal{T}}$$

and

$$F(\overline{\Delta}_{\mathcal{T}}) = -\mathcal{H}_{\mathcal{T}\mathcal{T}}^{-1}\overline{G}(\Omega_{\mathcal{T}}^* + \Delta_{\mathcal{T}}) + \overline{\Delta}_{\mathcal{T}}.$$

We need to show that $F(\mathcal{B}(r)) \subseteq \mathcal{B}(r)$, which implies that $||\mathcal{C}(\Delta_{\mathcal{T}})||_\infty \leqslant r$.

Under the assumptions of the lemma, for any $\Delta_S \in \mathcal{B}(r)$, we have the following decomposition

$$F(\overline{\Delta}_{\mathcal{T}}) = \mathcal{H}_{\mathcal{T}\mathcal{T}}^{-1}\overline{R}(\Delta)_{\mathcal{T}} + \mathcal{H}_{\mathcal{T}\mathcal{T}}^{-1}(\overline{S}_{\mathcal{T}} - \overline{\Sigma}_{\mathcal{T}}^* + \lambda\overline{\mathcal{Z}}(\Omega^* + \Delta)_{\mathcal{T}}).$$

Using Lemma 9, the first term can be bounded as

$$\begin{aligned}||\mathcal{C}(\mathcal{H}_{\mathcal{T}\mathcal{T}}^{-1}\overline{R}(\Delta)_{\mathcal{T}})||_\infty &\leqslant \|\mathcal{C}(\mathcal{H}_{\mathcal{T}\mathcal{T}}^{-1})\|_\infty ||\mathcal{C}(R(\Delta)||_\infty \\ &\leqslant \frac{3s}{2}\kappa_{\mathcal{H}}\kappa_{\Sigma^*}^3 ||\mathcal{C}(\Delta)||_\infty^2 \\ &\leqslant \frac{3s}{2}\kappa_{\mathcal{H}}\kappa_{\Sigma^*}^3 r^2 \\ &\leqslant r/2\end{aligned}$$

where the last inequality follows under the assumptions. Similarly

$$\begin{aligned}&||\mathcal{C}(\mathcal{H}_{\mathcal{T}\mathcal{T}}^{-1}(\overline{S}_{\mathcal{T}} - \overline{\Sigma}_{\mathcal{T}}^* + \lambda\overline{\mathcal{Z}}(\Omega^* + \Delta)_{\mathcal{T}})||_\infty \\ &\quad\leqslant \|\mathcal{C}(\mathcal{H}_{\mathcal{T}\mathcal{T}}^{-1})\|_\infty(||\mathcal{C}(S-\Sigma^*)||_\infty + \lambda||\mathcal{C}(\mathcal{Z}(\Omega^* + \Delta))||_\infty) \\ &\quad\leqslant \kappa_{\mathcal{H}}(||\mathcal{C}(S-\overline{\Sigma}^*)||_\infty + \lambda) \\ &\quad\leqslant r/2.\end{aligned}$$

This shows that $F(\mathcal{B}(r)) \subseteq \mathcal{B}(r)$. ■

The following result is a corollary of Theorem 7, which shows that the graph structure can be estimated consistently under some assumptions.

Corollary 11 *Assume that the conditions of Theorem 7 are satisfied. Furthermore, suppose that*

$$\min_{(a,b)\in\mathcal{T},\ a\neq b} ||\Omega||_F > 2(1+8\alpha^{-1})\kappa_{\mathcal{H}}\overline{\delta}_f(n,p^\tau),$$

then Algorithm 1 estimates a graph $\widehat{G}$ which satisfies

$$\text{pr}\left(\widehat{G} \neq G\right) \geqslant 1 - p^{2-\tau}.$$

Next, we specialize the result of Theorem 7 to a case where X has sub-Gaussian tails. That is, the random vector $X = (X_1, \ldots, X_{pk})^T$ is zero-mean with covariance Σ^*. Each $(\sigma^*_{aa})^{-1/2}X_a$ is sub-Gaussian with parameter γ.

Proposition 12 *Set the penalty parameter in λ in Eq. (4) as*

$$\lambda = 8k\alpha^{-1}\left(128(1+4\gamma^2)^2(\max_a(\sigma^*_{aa})^2)n^{-1}(2\log(2k)+\tau\log(p))\right)^{1/2}.$$

If

$$n > C_1 s^2k^2(1+8\alpha^{-1})^2(\tau\log p + \log 4 + 2\log k),$$

where $C_1 = (48\sqrt{2}(1+4\gamma^2)(\max_a \sigma^*_{aa})\max(\kappa_{\Sigma^*}\kappa_{\mathcal{H}}, \kappa^3_{\Sigma^*}\kappa^2_{\mathcal{H}}))^2$, then

$$||\mathcal{C}(\widehat{\Omega}-\Omega)||_\infty \leqslant 16\sqrt{2}(1+4\gamma^2)\max_i \sigma^*_{ii}(1+8\alpha^{-1})\kappa_{\mathcal{H}}k\left(\frac{\tau\log p+\log 4+2\log k}{n}\right)^{1/2}$$

with probability $1-p^{2-\tau}$.

The proof simply follows by observing that, for any (a,b),

$$\begin{aligned}\text{pr}\left(\mathcal{C}(S-\Sigma^*)_{ab} > \delta\right) &\leqslant \text{pr}\left(\max_{(c,d)\in(a,b)}(\sigma_{cd}-\sigma^*_{cd})^2 > \delta^2/k^2\right)\\ &\leqslant k^2\text{pr}\left(|\sigma_{cd}-\sigma^*_{cd}| > \delta/k\right)\\ &\leqslant 4k^2\exp\left(-\frac{n\delta^2}{c_*k^2}\right)\end{aligned} \tag{38}$$

for all $\delta \in (0, 8(1+4\gamma^2)(\max_a \sigma^*_{aa}))$ with $c_* = 128(1+4\gamma^2)^2(\max_a(\sigma^*_{aa})^2)$. Therefore,

$$\begin{aligned}f(n,\delta) &= \frac{1}{4k^2}\exp(c_*\frac{n\delta^2}{k^2}),\\ \overline{n}_f(\delta;r) &= \frac{k^2\log(4k^2r)}{c_*\delta^2},\\ \overline{\delta}_f(r;n) &= \left(\frac{k^2\log(4k^2r)}{c_*n}\right)^{1/2}.\end{aligned}$$

Theorem 7 and some simple algebra complete the proof.

Proposition 4 is a simple consequence of Proposition 12.

B.5 Some Results on Norms of Block Matrices

Let $\mathcal{T}$ be a partition of V. Throughout this section, we assume that matrices $A, B \in \mathbb{R}^{p\times p}$ and a vector $b \in \mathbb{R}^p$ are partitioned into blocks according to $\mathcal{T}$.

Lemma 13

$$\max_{a\in\mathcal{T}} ||A_{a\cdot}b||_2 \leqslant \max_{a\in\mathcal{T}} \sum_{b\in\mathcal{T}} ||A_{ab}||_F \max_{c\in\mathcal{T}} ||b_c||_2. \tag{39}$$

Proof For any $a \in \mathcal{T}$,

$$\begin{aligned}
||A_{a\cdot}b||_2 &\leqslant \sum_{b\in\mathcal{T}} ||A_{ab}b_b||_2 \\
&= \sum_{b\in\mathcal{T}} \left(\sum_{i\in a}(A_{ib}b_b)^2\right)^{1/2} \\
&\leqslant \sum_{b\in\mathcal{T}} \left(\sum_{i\in a}||A_{ib}||_2^2||b_b||_2^2\right)^{1/2} \\
&\leqslant \sum_{b\in\mathcal{T}} \left(\sum_{i\in a}||A_{ib}||_2^2\right)^{1/2} \max_{c\in\mathcal{T}} ||b_c||_2 \\
&= \sum_{b\in\mathcal{T}} ||A_{ab}||_F \max_{c\in\mathcal{T}} ||b_c||_2.
\end{aligned}$$

■

Lemma 14

$$|||\mathcal{C}(AB)|||_\infty \leqslant |||\mathcal{C}(B)|||_\infty |||\mathcal{C}(A)|||_\infty. \tag{40}$$

Proof Let $\mathbf{C} = AB$ and let $\mathcal{T}$ be a partition of V.

$$\begin{aligned}
|||\mathcal{C}(AB)|||_\infty &= \max_{a\in\mathcal{T}} \sum_{b\in\mathcal{T}} ||\mathbf{C}_{ab}||_F \\
&\leqslant \max_{a\in\mathcal{T}} \sum_b \sum_c ||A_{ac}||_F ||B_{cb}||_F \\
&\leqslant \{\max_{a\in\mathcal{T}} \sum_c ||A_{ac}||_F\}\{\max_{c\in\mathcal{T}} \sum_b ||B_{cb}||_F\} \\
&= |||\mathcal{C}(A)|||_\infty |||\mathcal{C}(B)|||_\infty.
\end{aligned}$$

■

Lemma 15

$$||\mathcal{C}(AB)||_\infty \leqslant ||\mathcal{C}(A)||_\infty |||\mathcal{C}(B)^T|||_\infty. \tag{41}$$

Proof For a fixed a and b,

$$\begin{aligned}\mathcal{C}(AB)_{ab} &= ||\sum_c A_{ac}B_{cb}||_F \\ &\leqslant \sum_c ||A_{ac}||_F||B_c b||_F \\ &\leqslant \max_c ||A_{ac}|| \sum_c ||B_{cb}||_F.\end{aligned}$$

Maximizing over a and b gives the result. ■

Appendix C. Additional Information About Functional Brain Networks

Table 3 contains list of the names of the brain regions. The number before each region is used to index the node in the connectivity models. Figures 11, 12 and 13 contain adjacency matrices for the estimated graph structures.

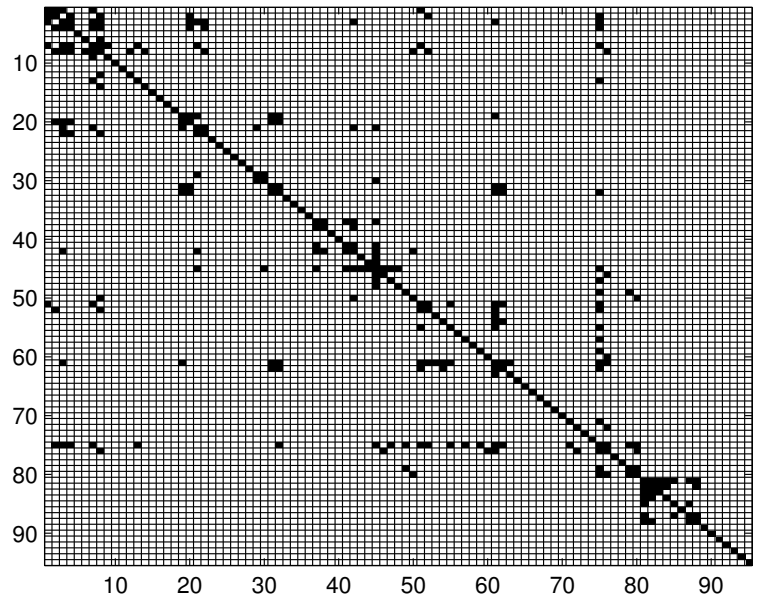


Figure 11: Adjacency matrix for the brain connectivity network: healthy subjects

References

H. Akaike. A new look at the statistical model identification. *IEEE Trans. Automat. Contr.*, 19(6):716–723, Dec 1974.

R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, Jan 2002.

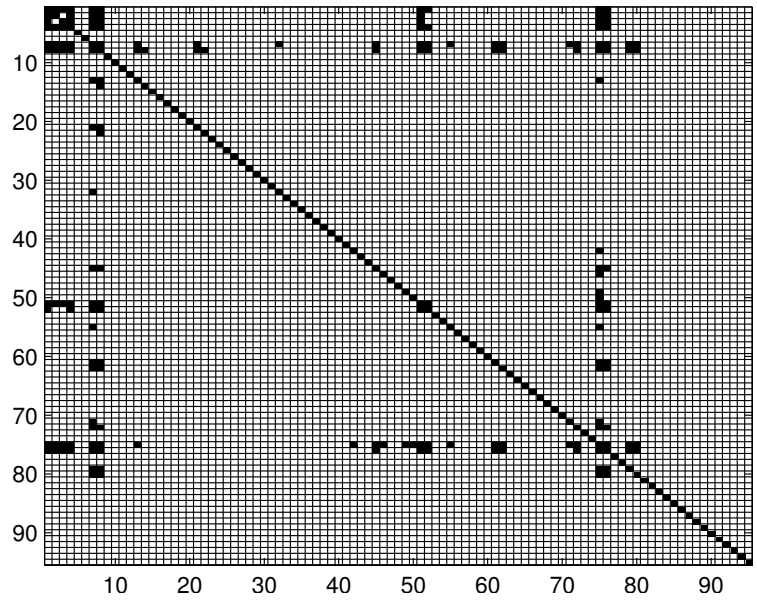


Figure 12: Adjacency matrix for the brain connectivity network: Mild Cognitive Impairment

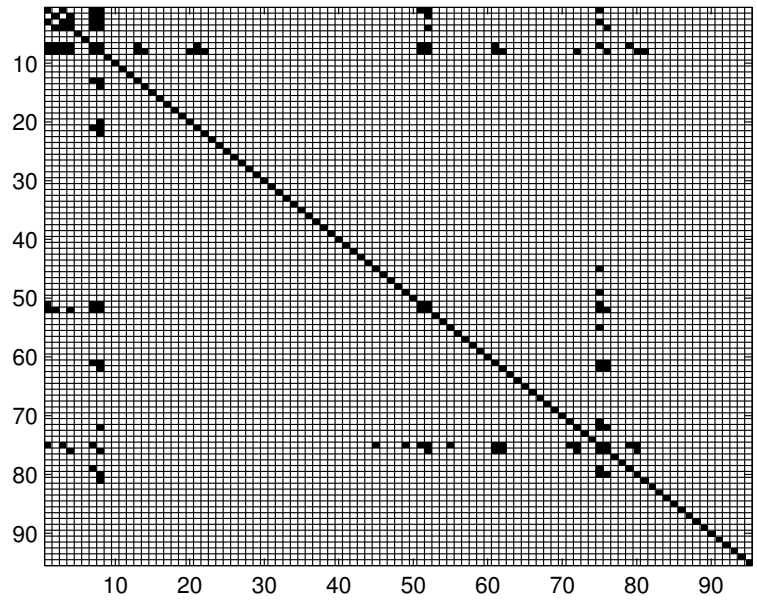


Figure 13: Adjacency matrix for the brain connectivity network: Alzheimer's & Dementia

1	Precentral_L	49	Fusiform_L
2	Precentral_R	50	Fusiform_R
3	Frontal_Sup_L	51	Postcentral_L
4	Frontal_Sup_R	52	Postcentral_R
5	Frontal_Sup_Orb_L	53	Parietal_Sup_L
6	Frontal_Sup_Orb_R	54	Parietal_Sup_R
7	Frontal_Mid_L	55	Parietal_Inf_L
8	Frontal_Mid_R	56	Parietal_Inf_R
9	Frontal_Mid_Orb_L	57	SupraMarginal_L
10	Frontal_Mid_Orb_R	58	SupraMarginal_R
11	Frontal_Inf_Oper_L	59	Angular_L
12	Frontal_Inf_Oper_R	60	Angular_R
13	Frontal_Inf_Tri_L	61	Precuneus_L
14	Frontal_Inf_Tri_R	62	Precuneus_R
15	Frontal_Inf_Orb_L	63	Paracentral_Lobule_L
16	Frontal_Inf_Orb_R	64	Paracentral_Lobule_R
17	Rolandic_Oper_L	65	Caudate_L
18	Rolandic_Oper_R	66	Caudate_R
19	Supp_Motor_Area_L	67	Putamen_L
20	Supp_Motor_Area_R	68	Putamen_R
21	Frontal_Sup_Medial_L	69	Thalamus_L
22	Frontal_Sup_Medial_R	70	Thalamus_R
23	Frontal_Med_Orb_L	71	Temporal_Sup_L
24	Frontal_Med_Orb_R	72	Temporal_Sup_R
25	Rectus_L	73	Temporal_Pole_Sup_L
26	Rectus_R	74	Temporal_Pole_Sup_R
27	Insula_L	75	Temporal_Mid_L
28	Insula_R	76	Temporal_Mid_R
29	Cingulum_Ant_L	77	Temporal_Pole_Mid_L
30	Cingulum_Ant_R	78	Temporal_Pole_Mid_R
31	Cingulum_Mid_L	79	Temporal_Inf_L
32	Cingulum_Mid_R	80	Temporal_Inf_R
33	Hippocampus_L	81	Cerebelum_Crus1_L
34	Hippocampus_R	82	Cerebelum_Crus1_R
35	ParaHippocampal_L	83	Cerebelum_Crus2_L
36	ParaHippocampal_R	84	Cerebelum_Crus2_R
37	Calcarine_L	85	Cerebelum_4_5_L
38	Calcarine_R	86	Cerebelum_4_5_R
39	Cuneus_L	87	Cerebelum_6_L
40	Cuneus_R	88	Cerebelum_6_R
41	Lingual_L	89	Cerebelum_7b_L
42	Lingual_R	90	Cerebelum_7b_R
43	Occipital_Sup_L	91	Cerebelum_8_L
44	Occipital_Sup_R	92	Cerebelum_8_R
45	Occipital_Mid_L	93	Cerebelum_9_L
46	Occipital_Mid_R	94	Cerebelum_9_R
47	Occipital_Inf_L	95	Vermis_4_5
48	Occipital_Inf_R		

Table 3: Names of the brain regions. L means that the brain region is located at the left hemisphere; R means right hemisphere.

J. R. Andrews-Hanna, A. Z. Snyder, J. L. Vincent, C. Lustig, D. Head, M. E. Raichle, and R. L. Buckner. Disruption of large-scale brain systems in advanced aging. *Neuron*, 56(5): 924–935, 2007.

O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation. *J. Mach. Learn. Res.*, 9(3):485–516, 2008.

A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286 (5439):509–512, 1999.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imag. Sci.*, 2:183–202, 2009.

T. T. Cai, W. Liu, and X. Luo. A constrained ℓ_1 minimization approach to sparse precision matrix estimation. *J. Am. Stat. Assoc.*, 106(494):594–607, 2011.

J. Chiquet, Y. Grandvalet, and C. Ambroise. Inferring multiple graphical structures. *Stat. Comput.*, 21(4):537–553, 2011.

P. Danaher, P. Wang, and D. M. Witten. The joint graphical lasso for inverse covariance estimation across multiple classes. *J. R. Stat. Soc. B*, 76(2):373–397, 2014.

A. P. Dempster. Covariance selection. *Biometrics*, 28:157–175, 1972.

J. H. Friedman, T. J. Hastie, and R. J. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

R. L. Gould, B. Arroyo, R. G. Brown, A. M. Owen, E. T. Bullmore, and R. J. Howard. Brain mechanisms of successful compensation during learning in alzheimer disease. *Neurology*, 67(6):1011–1017, 2006.

M. D. Greicius, G. Srivastava, A. L. Reiss, and V. Menon. Default-mode network activity distinguishes alzheimer's disease from healthy aging: Evidence from functional MRI. *Proc. Natl. Acad. Sci. U.S.A.*, 101(13):4637–4642, 2004.

J. Guo, E. Levina, G. Michailidis, and J. Zhu. Joint estimation of multiple graphical models. *Biometrika*, 98(1):1–15, 2011.

T. Hedden, K. R. A. V. Dijk, J. A. Becker, A. Mehta, R. A. Sperling, K. A. Johnson, and R. L. Buckner. Disruption of functional connectivity in clinically normal older adults harboring amyloid burden. *J. Neurosci.*, 29(40):12686–12694, 2009.

J. Honorio and D. Samaras. Multi-task learning of gaussian graphical models. In J. Fürnkranz and T. Joachims, editors, *Proc. of ICML*, pages 447–454, Haifa, Israel, June 2010. Omnipress.

S. Huang, J. Li, L. Sun, J. Liu, T. Wu, K. Chen, A. Fleisher, E. Reiman, and J. Ye. Learning brain connectivity of alzheimer's disease from neuroimaging data. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Proc. of NIPS*, pages 808–816, 2009.

C. Johnson, A. Jalali, and P. Ravikumar. High-dimensional sparse inverse covariance estimation using greedy methods. In N. Lawrence and M. Girolami, editors, *Proc. of AISTATS*, pages 574–582, 2012.

N. Katenka and E. D. Kolaczyk. Multi-attribute networks and the impact of partial information on inference and characterization. *Ann. Appl. Stat.*, 6(3):1068–1094, 2011.

M. Kolar and E. P. Xing. Consistent covariance selection from data with missing values. In J. Langford and J. Pineau, editors, *Proc. of ICML*, pages 551–558, Edinburgh, Scotland, GB, July 2012. Omnipress.

M. Kolar, L. Song, A. Ahmed, and E. P. Xing. Estimating Time-varying networks. *Ann. Appl. Stat.*, 4(1):94–123, 2010.

C. Lam and J. Fan. Sparsistency and rates of convergence in large covariance matrix estimation. *Ann. Stat.*, 37:4254–4278, 2009.

S. L. Lauritzen. *Graphical Models*, volume 17 of *Oxford Statistical Science Series*. The Clarendon Press Oxford University Press, New York, 1996. Oxford Science Publications.

H. Li and J. Gui. Gradient directed regularization for sparse gaussian concentration graphs, with applications to inference of genetic networks. *Biostatistics*, 7(2):302–317, 2006.

H. Liu, J. D. Lafferty, and L. A. Wasserman. The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *J. Mach. Learn. Res.*, 10:2295–2328, 2009.

H. Liu, F. Han, and C.-H. Zhang. Transelliptical graphical models. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Proc. of NIPS*, pages 809–817. 2012.

K. Lounici, M. Pontil, A. B. Tsybakov, and S. A. van de Geer. Oracle inequalities and optimal inference under group sparsity. *Ann. Stat.*, 39:2164–204, 2011.

R. Mazumder and D. K. Agarwal. A flexible, scalable and efficient algorithmic framework for primal graphical lasso. Technical report, Stanford University, 2011.

R. Mazumder and T. J. Hastie. Exact covariance thresholding into connected components for large-scale graphical lasso. *J. Mach. Learn. Res.*, 13:781–794, 2012.

N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the lasso. *Ann. Stat.*, 34(3):1436–1462, 2006.

N. Meinshausen and P. Bühlmann. Stability selection. *J. R. Stat. Soc. B*, 72(4):417–473, 2010.

J. Peng, P. Wang, N. Zhou, and J. Zhu. Partial correlation estimation by joint sparse regression models. *J. Am. Stat. Assoc.*, 104(486):735–746, 2009.

M. Pourahmadi. Covariance estimation: The glm and regularization perspectives. *Stat. Sci.*, 26(3):369–387, 08 2011.

B. Rao. Partial canonical correlations. *Trabajos de Estadstica y de Investigacin Operativa*, 20(2):211–219, 1969.

P. Ravikumar, M. J. Wainwright, G. Raskutti, and B. Yu. High-dimensional covariance estimation by minimizing ℓ_1-penalized log-determinant divergence. *Electron. J. Stat.*, 5: 935–980, 2011.

G. Schwarz. Estimating the dimension of a model. *Ann. Stat.*, 6(2):461–464, 03 1978.

X. Shen, W. Pan, and Y. Zhu. Likelihood-based selection and sharp parameter estimation. *J. Am. Stat. Assoc.*, 107:223–232, 2012.

M. Sjöbeck and E. Englund. Alzheimer's disease and the cerebellum: A morphologic study on neuronal and glial changes. *Dementia and Geriatric Cognitive Disorders*, 12(3):211–218, 2001.

Y. Stern. Cognitive reserve and Alzheimer disease. *Alzheimer Disease & Associated Disorders*, 20(2):112–117, 2006.

P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optim. Theory Appl.*, 109(3):475–494, 2001.

G. Varoquaux, A. Gramfort, J.-B. Poline, and B. Thirion. Brain covariance selection: Better individual functional connectivity models using population prior. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Proc. of NIPS*, pages 2334–2342, 2010.

D. M. Witten, J. H. Friedman, and N. Simon. New insights and faster computations for the graphical lasso. *J. Comput. Graph. Stat.*, 20(4):892–900, 2011.

X. Wu, R. Li, A. S. Fleisher, E. M. Reiman, X. Guan, Y. Zhang, K. Chen, and L. Yao. Altered default mode network connectivity in Alzheimer's diseasea resting functional MRI and Bayesian network study. *Human Brain Mapping*, 32(11):1868–1881, 2011.

M. Yuan. High dimensional inverse covariance matrix estimation via linear programming. *J. Mach. Learn. Res.*, 11:2261–2286, 2010.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. R. Stat. Soc. B*, 68:49–67, 2006.

M. Yuan and Y. Lin. Model selection and estimation in the gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

T. Zhao, H. Liu, K. E. Roeder, J. D. Lafferty, and L. A. Wasserman. The huge package for high-dimensional undirected graph estimation in R. *J. Mach. Learn. Res.*, 13:1059–1062, 2012.

Journal of Machine Learning Research 15 (2014) 1751-1798 Submitted 1/13; Revised 3/14; Published 5/14

Hitting and Commute Times in Large Random Neighborhood Graphs

Ulrike von Luxburg LUXBURG@INFORMATIK.UNI-HAMBURG.DE
Department of Computer Science
University of Hamburg
Vogt-Koelln-Str. 30, 22527 Hamburg, Germany

Agnes Radl AGNES.RADL@MATH.UNI-LEIPZIG.DE
Institute of Mathematics
University of Leipzig
Augustusplatz 10, 04109 Leipzig, Germany

Matthias Hein HEIN@CS.UNI-SAARLAND.DE
Department of Mathematics and Computer Science
Saarland University
Campus E1 1, 66123 Saarbrücken, Germany

Editor: Gabor Lugosi

Abstract

In machine learning, a popular tool to analyze the structure of graphs is the hitting time and the commute distance (resistance distance). For two vertices u and v, the hitting time H_{uv} is the expected time it takes a random walk to travel from u to v. The commute distance is its symmetrized version $C_{uv} = H_{uv} + H_{vu}$. In our paper we study the behavior of hitting times and commute distances when the number n of vertices in the graph tends to infinity. We focus on random geometric graphs (ε-graphs, kNN graphs and Gaussian similarity graphs), but our results also extend to graphs with a given expected degree distribution or Erdős-Rényi graphs with planted partitions. We prove that in these graph families, the suitably rescaled hitting time H_{uv} converges to $1/d_v$ and the rescaled commute time to $1/d_u + 1/d_v$ where d_u and d_v denote the degrees of vertices u and v. In these cases, hitting and commute times do not provide information about the structure of the graph, and their use is discouraged in many machine learning applications.

Keywords: commute distance, resistance, random graph, k-nearest neighbor graph, spectral gap

1. Introduction

Given an undirected, weighted graph $G = (V, E)$ with n vertices, the commute distance between two vertices u and v is defined as the expected time it takes the natural random walk starting in vertex u to travel to vertex v and back to u. It is equivalent (up to a constant) to the resistance distance, which interprets the graph as an electrical network and defines the distance between vertices u and v as the effective resistance between these vertices. See below for exact definitions, for background reading see Doyle and Snell (1984); Klein and Randic (1993); Xiao and Gutman (2003); Fouss et al. (2006), Chapter 2 of Lyons and Peres (2010), Chapter 3 of Aldous and Fill (2001), or Section 9.4 of Levin et al. (2008).

The commute distance is very popular in many different fields of computer science and beyond. As examples consider the fields of graph embedding (Guattery, 1998; Saerens et al., 2004; Qiu and Hancock, 2006; Wittmann et al., 2009), graph sparsification (Spielman and Srivastava, 2008), social network analysis (Liben-Nowell and Kleinberg, 2003), proximity search (Sarkar et al., 2008), collaborative filtering (Fouss et al., 2006), clustering (Yen et al., 2005), semi-supervised learning (Zhou and Schölkopf, 2004), dimensionality reduction (Ham et al., 2004), image processing (Qiu and Hancock, 2005), graph labeling (Herbster and Pontil, 2006; Cesa-Bianchi et al., 2009), theoretical computer science (Aleliunas et al., 1979; Chandra et al., 1989; Avin and Ercal, 2007; Cooper and Frieze, 2003, 2005, 2007, 2011), and various applications in chemometrics and bioinformatics (Klein and Randic, 1993; Ivanciuc, 2000; Fowler, 2002; Roy, 2004; Guillot et al., 2009).

The commute distance has many nice properties, both from a theoretical and a practical point of view. It is a Euclidean distance function and can be computed in closed form. As opposed to the shortest path distance, it takes into account all paths between u and v, not just the shortest one. As a rule of thumb, the more paths connect u with v, the smaller their commute distance becomes. Hence it supposedly satisfies the following, highly desirable property:

> **Property (★):** Vertices in the same "cluster" of the graph have a small commute distance, whereas vertices in different clusters of the graph have a large commute distance to each other.

Consequently, the commute distance is considered a convenient tool to encode the cluster structure of the graph.

In this paper we study how the commute distance behaves when the size of the graph increases. We focus on the case of random geometric graphs (k-nearest neighbor graphs, ε-graphs, and Gaussian similarity graphs). Denote by H_{uv} the expected hitting time and by C_{uv} the commute distance between two vertices u and v, by d_u the degree of vertex u, by $\mathrm{vol}(G)$ the volume of the graph. We show that as the number n of vertices tends to infinity, there exists a scaling term sc such that the hitting times and commute distances in random geometric graphs satisfy

$$sc \cdot \Big|\frac{1}{\mathrm{vol}(G)} H_{uv} - \frac{1}{d_v}\Big| \longrightarrow 0 \qquad \text{and} \qquad sc \cdot \Big|\frac{1}{\mathrm{vol}(G)} C_{uv} - \Big(\frac{1}{d_u} + \frac{1}{d_v}\Big)\Big| \longrightarrow 0,$$

and at the same time $sc \cdot d_u$ and $sc \cdot d_v$ converge to positive constants (precise definitions, assumptions and statements below). Loosely speaking, the convergence result says that the rescaled commute distance can be approximated by the sum of the inverse rescaled degrees.

We present two different strategies to prove these results: one based on flow arguments on electrical networks, and another one based on spectral arguments. While the former approach leads to tighter bounds, the latter is more general. Our proofs heavily rely on prior work by a number of authors: Lovász (1993), who characterized hitting and commute times in terms of their spectral properties and was the first one to observe that the commute distance can be approximated by $1/d_u + 1/d_v$; Boyd et al. (2005), who provided bounds on the

spectral gap in random geometric graphs, and Avin and Ercal (2007), who already studied the growth rates of the commute distance in random geometric graphs. Our main technical contributions are to strengthen the bound provided by Lovász (1993), to extend the results by Boyd et al. (2005) and Avin and Ercal (2007) to more general types of geometric graphs such as k-nearest neighbor graphs with general domain and general density, and to develop the flow-based techniques for geometric graphs.

Loosely speaking, the convergence results say that whenever the graph is reasonably large, the degrees are not too small, and the bottleneck is not too extreme, then the commute distance between two vertices can be approximated by the sum of their inverse degrees. These results have the following important consequences for applications.

Negative implication: Hitting and commute times can be misleading. Our approximation result shows that the commute distance does not take into account any global properties of the data in large geometric graphs. It has been observed before that the commute distance sometimes behaves in an undesired way when high-degree vertices are involved (Liben-Nowell and Kleinberg, 2003; Brand, 2005), but our work now gives a complete theoretical description of this phenomenon: the commute distance just considers the local density (the degree of the vertex) at the two vertices, nothing else. The resulting large sample commute distance $dist(u, v) = 1/d_u + 1/d_v$ is completely meaningless as a distance on a graph. For example, all data points have the same nearest neighbor (namely, the vertex with the largest degree), the same second-nearest neighbor (the vertex with the second-largest degree), and so on. In particular, one of the main motivations to use the commute distance, Property (★), no longer holds when the graph becomes large enough. Even more disappointingly, computer simulations show that n does not even need to be very large before (★) breaks down. Often, n in the order of 1000 is already enough to make the commute distance close to its approximation expression. This effect is even stronger if the dimensionality of the underlying data space is large. Consequently, even on moderate-sized graphs, the use of the raw commute distance should be discouraged.

Positive implication: Efficient computation of approximate commute distances. In some applications the commute distance is not used as a distance function, but as a tool to encode the connectivity properties of a graph, for example in graph sparsification (Spielman and Srivastava, 2008) or when computing bounds on mixing or cover times (Aleliunas et al., 1979; Chandra et al., 1989; Avin and Ercal, 2007; Cooper and Frieze, 2011) or graph labeling (Herbster and Pontil, 2006; Cesa-Bianchi et al., 2009). To obtain the commute distance between all points in a graph one has to compute the pseudo-inverse of the graph Laplacian matrix, an operation of time complexity $O(n^3)$. This is prohibitive in large graphs. To circumvent the matrix inversion, several approximations of the commute distance have been suggested in the literature (Spielman and Srivastava, 2008; Sarkar and Moore, 2007; Brand, 2005). Our results lead to a much simpler and well-justified way of approximating the commute distance on large random geometric graphs.

After introducing general definitions and notation (Section 3), we present our main results in Section 4. This section is divided into two parts (flow-based part and spectral part). In

Section 5 we show in extensive simulations that our approximation results are relevant for many graphs used in machine learning. Relations to previous work is discussed in Section 2. All proofs are deferred to Sections 6 and 7. Parts of this work is built on our conference paper von Luxburg et al. (2010).

2. Related Work

The resistance distance became popular through the work of Doyle and Snell (1984) and Klein and Randic (1993), and the connection between commute and resistance distance was established by Chandra et al. (1989) and Tetali (1991). By now, resistance and commute distances are treated in many text books, for example Chapter IX of Bollobas (1998), Chapter 2 of Lyons and Peres (2010), Chapter 3 of Aldous and Fill (2001), or Section 9.4 of Levin et al. (2008). It is well known that the commute distance is related to the spectrum of the unnormalized and normalized graph Laplacian (Lovász, 1993; Xiao and Gutman, 2003). Bounds on resistance distances in terms of the eigengap and the term $1/d_u + 1/d_v$ have already been presented in Lovász (1993). We present an improved version of this bound that leads to our convergence results in the spectral approach.

Properties of random geometric graphs have been investigated thoroughly in the literature, see for example the monograph of Penrose (2003). Our work concerns the case where the graph connectivity parameter (ε or k, respectively) is so large that the graph is connected with high probability (see Penrose, 1997; Brito et al., 1997; Penrose, 1999, 2003; Xue and Kumar, 2004; Balister et al., 2005). We focus on this case because it is most relevant for machine learning. In applications such as clustering, people construct a neighborhood graph based on given similarity scores between objects, and they choose the graph connectivity parameter so large that the graph is well-connected.

Asymptotic growth rates of commute distances and the spectral gap have already been studied for a particular special case: ε-graphs on a sample from the uniform distribution on the unit cube or unit torus in $\mathbb{R}^d$ (Avin and Ercal, 2007; Boyd et al., 2005; Cooper and Frieze, 2011). However, the most interesting case for machine learning is the case of kNN graphs or Gaussian graphs (as they are the ones used in practice) on spaces with a non-uniform probability distribution (real data is never uniform). A priori, it is unclear whether the commute distances on such graphs behave as the ones on ε-graphs: there are many situations in which these types of graphs behave very different. For example their graph Laplacians converge to different limit objects (Hein et al., 2007). In our paper we now consider the general situation of commute distances in ε, kNN and Gaussian graphs on a sample from a non-uniform distribution on some subset of $\mathbb{R}^d$.

Our main techniques, the canonical path technique for bounding the spectral gap (Diaconis and Stroock, 1991; Sinclair, 1992; Jerrum and Sinclair, 1988; Diaconis and Saloff-Coste, 1993) and the flow-based techniques for bounding the resistance, are text book knowledge (Section 13.5. of Levin et al., 2008; Sec. IX.2 of Bollobas, 1998). Our results on the spectral gap in random geometric graphs, Theorems 6 and 7, build on similar results in Boyd et al.

(2005); Avin and Ercal (2007); Cooper and Frieze (2011). These three papers consider the special case of an ε-graph on the unit cube / unit torus in $\mathbb{R}^d$, endowed with the uniform distribution. For this case, the authors also discuss cover times and mixing times, and Avin and Ercal (2007) also include bounds on the asymptotic growth rate of resistance distances. We extend these results to the case of ε, kNN and Gaussian graphs with general domain and general probability density. The focus in our paper is somewhat different from the related literature, because we care about the exact limit expressions rather than asymptotic growth rates.

3. General Setup, Definitions and Notation

We consider undirected graphs $G = (V, E)$ that are connected and not bipartite. By n we denote the number of vertices. The adjacency matrix is denoted by $W := (w_{ij})_{i,j=1,\dots,n}$. In case the graph is weighted, this matrix is also called the weight matrix. All weights are assumed to be non-negative. The minimal and maximal weights in the graph are denoted by $w_{\min}$ and $w_{\max}$. By $d_i := \sum_{j=1}^n w_{ij}$ we denote the degree of vertex v_i. The diagonal matrix D with diagonal entries $d_1, \dots, d_n$ is called the *degree matrix*, the minimal and maximal degrees are denoted $d_{\min}$ and $d_{\max}$. The *volume* of the graph is given as $\mathrm{vol}(G) = \sum_{j=1,\dots,n} d_j$. The *unnormalized graph Laplacian* is given as $L := D - W$, the normalized one as $L_{\mathrm{sym}} = D^{-1/2} L D^{-1/2}$. Consider the natural random walk on G. Its transition matrix is given as $P = D^{-1}W$. It is well-known that λ is an eigenvalue of L_{sym} if and only if $1 - \lambda$ is an eigenvalue of P. By $1 = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n > -1$ we denote the eigenvalues of P. The quantity $1 - \max\{\lambda_2, |\lambda_n|\}$ is called the *spectral gap* of P.

The *hitting time* H_{uv} is defined as the expected time it takes a random walk starting in vertex u to travel to vertex v (where $H_{uu} = 0$ by definition). The *commute distance* (*commute time*) between u and v is defined as $C_{uv} := H_{uv} + H_{vu}$. Closely related to the commute distance is the *resistance distance*. Here one interprets the graph as an electrical network where the edges represent resistors. The conductance of a resistor is given by the corresponding edge weight. The resistance distance R_{uv} between two vertices u and v is defined as the effective resistance between u and v in the network. It is well known (Chandra et al., 1989) that the resistance distance coincides with the commute distance up to a constant: $C_{uv} = \mathrm{vol}(G) R_{uv}$. For background reading on resistance and commute distances see Doyle and Snell (1984); Klein and Randic (1993); Xiao and Gutman (2003); Fouss et al. (2006).

Recall that for a symmetric, non-invertible matrix A its Moore-Penrose inverse is defined as $A^\dagger := (A+U)^{-1} - U$ where U is the projection on the eigenspace corresponding to eigenvalue 0. It is well known that commute times can be expressed in terms of the Moore-Penrose inverse $L^\dagger$ of the unnormalized graph Laplacian (e.g., Klein and Randic, 1993; Xiao and Gutman, 2003; Fouss et al., 2006):

$$C_{ij} = \mathrm{vol}(G) \left\langle e_i - e_j, L^\dagger (e_i - e_j) \right\rangle,$$

where e_i is the i-th unit vector in $\mathbb{R}^n$. The following representations for commute and hitting times involving the pseudo-inverse $L_{\mathrm{sym}}^\dagger$ of the normalized graph Laplacian are direct consequences of Lovász, 1993:

Proposition 1 (Closed form expression for hitting and commute times) *Let G be a connected, undirected graph with n vertices. The hitting times H_{ij}, $i \neq j$, can be computed by*

$$H_{ij} = \mathrm{vol}(G)\Big\langle \frac{1}{\sqrt{d_j}} e_j \ , \ L_{sym}^{\dagger}\Big(\frac{1}{\sqrt{d_j}} e_j - \frac{1}{\sqrt{d_i}} e_i\Big)\Big\rangle,$$

and the commute times satisfy

$$C_{ij} = \mathrm{vol}(G)\Big\langle \frac{1}{\sqrt{d_i}} e_i - \frac{1}{\sqrt{d_j}} e_j \ , \ L_{sym}^{\dagger}\Big(\frac{1}{\sqrt{d_j}} e_j - \frac{1}{\sqrt{d_i}} e_i\Big)\Big\rangle.$$

Our main focus in this paper is the class of *geometric graphs*. For a *deterministic* geometric graph we consider a fixed set of points $X_1, \ldots, X_n \in \mathbb{R}^d$. These points form the vertices $v_1, \ldots, v_n$ of the graph. In the *ε-graph* we connect two points whenever their Euclidean distance is less than or equal to ε. In the undirected *symmetric k-nearest neighbor graph* we connect v_i to v_j if X_i is among the k nearest neighbors of X_j *or* vice versa. In the undirected *mutual k-nearest neighbor graph* we connect v_i to v_j if X_i is among the k nearest neighbors of X_j *and* vice versa. Note that by default, the terms ε- and kNN-graph refer to unweighted graphs in our paper. When we treat weighted graphs, we always make it explicit. For a general *similarity graph* we build a weight matrix between all points based on a similarity function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$, that is we define the weight matrix W with entries $w_{ij} = k(X_i, X_j)$ and consider the fully connected graph with weight matrix W. The most popular weight function in applications is the Gaussian similarity function $w_{ij} = \exp(-\|X_i - X_j\|^2/\sigma^2)$, where $\sigma > 0$ is a bandwidth parameter.

While these definitions make sense with any fixed set of vertices, we are most interested in the case of *random* geometric graphs. We assume that the underlying set of vertices $X_1, ..., X_n$ has been drawn i.i.d. according to some probability density p on $\mathbb{R}^d$. Once the vertices are known, the edges in the graphs are constructed as described above. In the random setting it is convenient to make regularity assumptions in order to be able to control quantities such as the minimal and maximal degrees.

Definition 2 (Valid region) *Let p be any density on $\mathbb{R}^d$. We call a connected subset $\mathcal{X} \subset \mathbb{R}^d$ a* valid region *if the following properties are satisfied:*

1. *The density on $\mathcal{X}$ is bounded away from 0 and infinity, that is for all $x \in \mathcal{X}$ we have that $0 < p_{\min} \leq p(x) \leq p_{\max} < \infty$ for some constants $p_{\min}, p_{\max}$.*

2. *$\mathcal{X}$ has "bottleneck" larger than some value $h > 0$: the set $\{x \in \mathcal{X} \ : \ dist(x, \partial\mathcal{X}) > h/2\}$ is connected (here $\partial\mathcal{X}$ denotes the topological boundary of $\mathcal{X}$).*

3. *The boundary of $\mathcal{X}$ is regular in the following sense. We assume that there exist positive constants $\alpha > 0$ and $\varepsilon_0 > 0$ such that if $\varepsilon < \varepsilon_0$, then for all points $x \in \partial\mathcal{X}$ we have $\mathrm{vol}(B_\varepsilon(x) \cap \mathcal{X}) \geq \alpha\, \mathrm{vol}(B_\varepsilon(x))$ (where vol denotes the Lebesgue volume). Essentially this condition just excludes the situation where the boundary has arbitrarily thin spikes.*

Sometimes we consider a valid region with respect to two points s, t. *Here we additionally assume that s and t are interior points of $\mathcal{X}$.*

In the spectral part of our paper, we always have to make a couple of assumptions that will be summarized by the term **general assumptions**. They are as follows: First we assume that $\mathcal{X} := \mathrm{supp}(p)$ is a valid region according to Definition 2. Second, we assume that $\mathcal{X}$ does not contain any holes and does not become arbitrarily narrow: there exists a homeomorphism $h : \mathcal{X} \to [0,1]^d$ and constants $0 < L_{\min} < L_{\max} < \infty$ such that for all $x, y \in \mathcal{X}$ we have

$$L_{\min}\|x - y\| \;\leq\; \|h(x) - h(y)\| \;\leq\; L_{\max}\|x - y\|.$$

This condition restricts $\mathcal{X}$ to be topologically equivalent to the cube. In applications this is not a strong assumption, as the occurrence of "holes" with vanishing probability density is unrealistic due to the presence of noise in the data generating process. More generally we believe that our results can be generalized to other homeomorphism classes, but refrain from doing so as it would substantially increase the amount of technicalities.

In the following we denote the volume of the unit ball in $\mathbb{R}^d$ by η_d. For readability reasons, we are going to state our main results using constants $c_i > 0$. These constants are independent of n and the graph connectivity parameter (ε or k or h, respectively) but depend on the dimension, the geometry of $\mathcal{X}$, and p. The values of all constants are determined explicitly in the proofs. They are not the same in different propositions.

4. Main Results

Our paper comprises two different approaches. In the first approach we analyze the resistance distance by flow based arguments. This technique is somewhat restrictive in the sense that it only works for the resistance distance itself (not the hitting times) and we only apply it to random geometric graphs. The advantage is that in this setting we obtain good convergence conditions and rates. The second approach is based on spectral arguments and is more general. It works for various kinds of graphs and can treat hitting times as well. This comes at the price of slightly stronger assumptions and worse convergence rates.

4.1 Results Based on Flow Arguments

Theorem 3 (Commute distance on ε-graphs) *Let $\mathcal{X}$ be a valid region with bottleneck h and minimal density $p_{\min}$. For $\varepsilon \leq h$, consider an unweighted ε-graph built from the sequence $X_1, \ldots, X_n$ that has been drawn i.i.d. from the density p. Fix i and j. Assume that X_i and X_j have distance at least h from the boundary of $\mathcal{X}$, and that the distance between X_i and X_j is at least 8ε. Then there exist constants $c_1, \ldots, c_7 > 0$ (depending on the dimension and geometry of $\mathcal{X}$) such that with probability at least $1 - c_1 n \exp(-c_2 n \varepsilon^d) -$*

$c_3 \exp(-c_4 n\varepsilon^d)/\varepsilon^d$ *the commute distance on the* ε*-graph satisfies*

$$\left| \frac{n\varepsilon^d}{\mathrm{vol}(G)} C_{ij} - \left(\frac{n\varepsilon^d}{d_i} + \frac{n\varepsilon^d}{d_j} \right) \right| \leq \begin{cases} c_5/n\varepsilon^d & \text{if } d > 3 \\ c_6 \cdot \log(1/\varepsilon)/n\varepsilon^3 & \text{if } d = 3 \\ c_7/n\varepsilon^3 & \text{if } d = 2 \end{cases}$$

The probability converges to 1 if $n \to \infty$ *and* $n\varepsilon^d/\log(n) \to \infty$*. The right hand side of the deviation bound converges to 0 as* $n \to \infty$*, if*

$$\begin{cases} n\varepsilon^d \to \infty & \text{if } d > 3 \\ n\varepsilon^3/\log(1/\varepsilon) \to \infty & \text{if } d = 3 \\ n\varepsilon^3 = n\varepsilon^{d+1} \to \infty & \text{if } d = 2. \end{cases}$$

Under these conditions, if the density p *is continuous and if* $\varepsilon \to 0$*, then*

$$\frac{n\varepsilon^d}{\mathrm{vol}(G)} C_{ij} \longrightarrow \frac{1}{\eta_d p(X_i)} + \frac{1}{\eta_d p(X_j)} \quad a.s.$$

Theorem 4 (Commute distance on kNN-graphs) *Let* $\mathcal{X}$ *be a valid region with bottleneck* h *and density bounds* $p_{\min}$ *and* $p_{\max}$*. Consider an unweighted kNN-graph (either symmetric or mutual) such that* $(k/n)^{1/d}/(2p_{\max}) \leq h$*, built from the sequence* $X_1, \ldots, X_n$ *that has been drawn i.i.d. from the density* p*.*
Fix i *and* j*. Assume that* X_i *and* X_j *have distance at least* h *from the boundary of* $\mathcal{X}$*, and that the distance between* X_i *and* X_j *is at least* $4(k/n)^{1/d}/p_{\max}$*. Then there exist constants* $c_1, \ldots, c_5 > 0$ *such that with probability at least* $1 - c_1 n \exp(-c_2 k)$ *the commute distance on both the symmetric and the mutual kNN-graph satisfies*

$$\left| \frac{k}{\mathrm{vol}(G)} C_{ij} - \left(\frac{k}{d_i} + \frac{k}{d_j} \right) \right| \leq \begin{cases} c_4/k & \text{if } d > 3 \\ c_5 \cdot \log(n/k)/k & \text{if } d = 3 \\ c_6 n^{1/2}/k^{3/2} & \text{if } d = 2 \end{cases}$$

The probability converges to 1 if $n \to \infty$ *and* $k/\log(n) \to \infty$*. In case* $d > 3$*, the right hand side of the deviation bound converges to 0 if* $k \to \infty$ *(and under slightly worse conditions in cases* $d = 3$ *and* $d = 2$*). Under these conditions, if the density* p *is continuous and if additionally* $k/n \to 0$*, then* $\frac{k}{\mathrm{vol}(G)} C_{ij} \to 2$ *almost surely.*

Let us make a couple of technical remarks about these theorems.

To achieve the convergence of the commute distance we have to rescale it appropriately (for example, in the ε-graph we scale by a factor of $n\varepsilon^d$). Our rescaling is exactly chosen such that the limit expressions are finite, positive values. Scaling by any other factor in terms of n, ε or k either leads to divergence or to convergence to zero.

In case $d > 3$, all convergence conditions on n and ε (or k, respectively) are the ones to be expected for random geometric graphs. They are satisfied as soon as the degrees grow faster than $\log(n)$. For degrees of order smaller than $\log(n)$, the graphs are not connected

anyway, see for example Penrose (1997, 1999); Xue and Kumar (2004); Balister et al. (2005). In dimensions 3 and 2, our rates are a bit weaker. For example, in dimension 2 we need $n\varepsilon^3 \to \infty$ instead of $n\varepsilon^2 \to \infty$. On the one hand we are not too surprised to get systematic differences between the lowest few dimensions. The same happens in many situations, just consider the example of Polya's theorem about the recurrence/ transience of random walks on grids. On the other hand, these differences might as well be an artifact of our proof methods (and we suspect so at least for the case $d = 3$; but even though we tried, we did not get rid of the log factor in this case). It is a matter of future work to clarify this.

The valid region $\mathcal{X}$ has been introduced for technical reasons. We need to operate in such a region in order to be able to control the behavior of the graph, e.g. the minimal and maximal degrees. The assumptions on $\mathcal{X}$ are the standard assumptions used regularly in the random geometric graph literature. In our setting, we have the freedom of choosing $\mathcal{X} \subset \mathbb{R}^d$ as we want. In order to obtain the tightest bounds one should aim for a valid $\mathcal{X}$ that has a wide bottleneck h and a high minimal density $p_{\min}$. In general this freedom of choosing $\mathcal{X}$ shows that if two points are in the same high-density region of the space, the convergence of the commute distance is fast, while it gets slower if the two points are in different regions of high density separated by a bottleneck.

We stated the theorems above for a fixed pair i, j. However, they also hold uniformly over all pairs i, j that satisfy the conditions in the theorem (with exactly the same statement). The reason is that the main probabilistic quantities that enter the proofs are bound on the minimal and maximal degrees, which of course hold uniformly.

4.2 Results Based on Spectral Arguments

The representation of the hitting and commute times in terms of the Moore-Penrose inverse of the normalized graph Laplacian (Proposition 1) can be used to derive the following key proposition that is the basis for all further results in this section.

Proposition 5 (Absolute and relative bounds in any fixed graph) *Let G be a finite, connected, undirected, possibly weighted graph that is not bipartite.*

1. *For $i \neq j$*

$$\left| \frac{1}{\mathrm{vol}(G)} H_{ij} - \frac{1}{d_j} \right| \leq 2 \left(\frac{1}{1-\lambda_2} + 1 \right) \frac{w_{\max}}{d_{\min}^2}.$$

2. *For $i \neq j$*

$$\begin{aligned} \left| \frac{1}{\mathrm{vol}(G)} C_{ij} - \left(\frac{1}{d_i} + \frac{1}{d_j} \right) \right| &\leq \frac{w_{\max}}{d_{\min}} \left(\frac{1}{1-\lambda_2} + 2 \right) \left(\frac{1}{d_i} + \frac{1}{d_j} \right) \\ &\leq 2 \left(\frac{1}{1-\lambda_2} + 2 \right) \frac{w_{\max}}{d_{\min}^2}. \end{aligned} \quad (1)$$

We would like to point out that even though the bound in Part 2 of the proposition is reminiscent to statements in the literature, it is much tighter. Consider the following formula from Lovász (1993)

$$\frac{1}{2}\left(\frac{1}{d_i}+\frac{1}{d_j}\right) \;\leq\; \frac{1}{\mathrm{vol}(G)}C_{ij} \;\leq\; \frac{1}{1-\lambda_2}\left(\frac{1}{d_i}+\frac{1}{d_j}\right)$$

that can easily be rearranged to the following bound:

$$\left|\frac{1}{\mathrm{vol}(G)}C_{ij}-\left(\frac{1}{d_i}+\frac{1}{d_j}\right)\right| \leq \frac{1}{1-\lambda_2}\frac{2}{d_{\min}}. \tag{2}$$

The major difference between our bound (1) and Lovasz' bound (2) is that while the latter has the term $d_{\min}$ in the denominator, our bound has the term $d_{\min}^2$ in the denominator. This makes all of a difference: in the graphs under considerations our bound converges to 0 whereas Lovasz' bound diverges. In particular, our convergence results are not a trivial consequence of Lovász (1993).

4.2.1 Application to Unweighted Random Geometric Graphs

In the following we are going to apply Proposition 5 to various random geometric graphs. Next to some standard results about the degrees and number of edges in random geometric graphs, the main ingredients are the following bounds on the spectral gap in random geometric graphs. These bounds are of independent interest because the spectral gap governs many important properties and processes on graphs.

Theorem 6 (Spectral gap of the ε-graph) *Suppose that the general assumptions hold. Then there exist constants $c_1,\ldots,c_6>0$ such that with probability at least $1-c_1 n\exp(-c_2 n\varepsilon^d)-c_3\exp(-c_4 n\varepsilon^d)/\varepsilon^d$,*

$$1-\lambda_2 \;\geq\; c_5\cdot\varepsilon^2 \qquad \textit{and} \qquad 1-|\lambda_n| \;\geq\; c_6\cdot\varepsilon^{d+1}/n.$$

If $n\varepsilon^d/\log n\to\infty$, then this probability converges to 1.

Theorem 7 (Spectral gap of the kNN-graph) *Suppose that the general assumptions hold. Then for both the symmetric and the mutual kNN-graph there exist constants $c_1,\ldots,c_4>0$ such that with probability at least $1-c_1 n\exp(-c_2 k)$,*

$$1-\lambda_2 \;\geq\; c_3\cdot(k/n)^{2/d} \qquad \textit{and} \qquad 1-|\lambda_n| \;\geq\; c_4\cdot k^{2/d}/n^{(d+2)/d}.$$

If $k/\log n\to\infty$, then the probability converges to 1.

The following theorems characterize the hitting and commute times for ε-and kNN-graphs. They are direct consequences of plugging the results about the spectral gap into Proposition 5. In the corollaries, the reader should keep in mind that the degrees also depend on n and ε (or k, respectively).

Corollary 8 (Hitting and commute times on ε-graphs) *Assume that the general assumptions hold. Consider an unweighted ε-graph built from the sequence $X_1, \ldots, X_n$ drawn i.i.d. from the density p. Then there exist constants $c_1, \ldots, c_5 > 0$ such that with probability at least $1 - c_1 n \exp(-c_2 n\varepsilon^d) - c_3 \exp(-c_4 n\varepsilon^d)/\varepsilon^d$, we have uniformly for all $i \neq j$ that*

$$\left| \frac{n\varepsilon^d}{\mathrm{vol}(G)} H_{ij} - \frac{n\varepsilon^d}{d_j} \right| \;\leq\; \frac{c_5}{n\varepsilon^{d+2}}.$$

If the density p is continuous and $n \to \infty, \varepsilon \to 0$ and $n\varepsilon^{d+2} \to \infty$, then

$$\frac{n\varepsilon^d}{\mathrm{vol}(G)} H_{ij} \longrightarrow \frac{1}{\eta_d \cdot p(X_j)} \quad \textit{almost surely.}$$

For the commute times, the analogous results hold due to $C_{ij} = H_{ij} + H_{ji}$.

Corollary 9 (Hitting and commute times on kNN-graphs) *Assume that the general assumptions hold. Consider an unweighted kNN-graph built from the sequence $X_1, \ldots, X_n$ drawn i.i.d. from the density p. Then for both the symmetric and mutual kNN-graph there exist constants $c_1, c_2, c_3 > 0$ such that with probability at least $1 - c_1 \cdot n \cdot \exp(-kc_2)$, we have uniformly for all $i \neq j$ that*

$$\left| \frac{k}{\mathrm{vol}(G)} H_{ij} - \frac{k}{d_j} \right| \;\leq\; c_3 \cdot \frac{n^{2/d}}{k^{1+2/d}}.$$

If the density p is continuous and $n \to \infty$, $k/n \to 0$ and $k\big(k/n\big)^{2/d} \to \infty$, then

$$\frac{k}{\mathrm{vol}(G)} H_{ij} \longrightarrow 1 \quad \textit{almost surely.}$$

For the commute times, the analogous results hold due to $C_{ij} = H_{ij} + H_{ji}$.

Note that the density shows up in the limit for the ε-graph, but not for the kNN graph. The explanation is that in the former, the density is encoded in the degrees of the graph, while in the latter it is only encoded in the k-nearest neighbor distance, but not the degrees themselves. As a rule of thumb, it is possible to convert the last two corollaries into each other by substituting ε by $(k/(np(x)\eta^d))^{1/d}$ or vice versa.

4.2.2 Application to Weighted Graphs

In several applications, ε-graphs or kNN graphs are endowed with edge weights. For example, in the field of machine learning it is common to use Gaussian weights $w_{ij} = \exp(-\|X_i - X_j\|^2/\sigma^2)$, where $\sigma > 0$ is a bandwidth parameter. We can use standard spectral results to prove approximation theorems in such cases.

Theorem 10 (Results on fully connected weighted graphs) *Consider a fixed, fully connected weighted graph with weight matrix W. Assume that its entries are upper and*

lower bounded by some constants $w_{\min}, w_{\max}$*, that is* $0 < w_{\min} \leq w_{ij} \leq w_{\max}$ *for all* i, j*. Then, uniformly for all* $i, j \in \{1, ..., n\}$*,* $i \neq j$*,*

$$\left| \frac{n}{\mathrm{vol}(G)} H_{ij} - \frac{n}{d_j} \right| \;\leq\; 4n \left(\frac{w_{\max}}{w_{\min}} \right) \frac{w_{\max}}{d_{\min}^2} \;\leq\; 4 \frac{w_{\max}^2}{w_{\min}^3} \frac{1}{n}.$$

For example, this result can be applied directly to a Gaussian similarity graph (for fixed bandwidth σ).

The next theorem treats the case of Gaussian similarity graphs with adapted bandwidth σ. The technique we use to prove this theorem is rather general. Using the Rayleigh principle, we reduce the case of the fully connected Gaussian graph to a truncated graph where edges beyond a certain length are removed. Bounds for this truncated graph, in turn, can be reduced to bounds of the unweighted ε-graph. With this technique it is possible to treat very general classes of graphs.

Theorem 11 (Results on Gaussian graphs with adapted bandwidth) *Let* $\mathcal{X} \subseteq \mathbb{R}^d$ *be a compact set and* p *a continuous, strictly positive density on* $\mathcal{X}$*. Consider a fully connected, weighted similarity graph built from the points* $X_1, \ldots, X_n$ *drawn i.i.d. from density* p*. As weight function use the Gaussian similarity function* $k_\sigma(x, y) = \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$*. If the density* p *is continuous and* $n \to \infty, \sigma \to 0$ *and* $n\sigma^{d+2}/\log(n) \to \infty$*, then*

$$\frac{n}{\mathrm{vol}(G)} C_{ij} \longrightarrow \frac{1}{p(X_i)} + \frac{1}{p(X_j)} \quad \textit{almost surely.}$$

Note that in this theorem, we introduced the scaling factor $1/\sigma^d$ already in the definition of the Gaussian similarity function to obtain the correct density estimate $p(X_j)$ in the limit. For this reason, the resistance results are rescaled with factor n instead of $n\sigma^d$.

4.2.3 Application to Random Graphs with Given Expected Degrees and Erdős-Rényi Graphs

Consider the general random graph model where the edge between vertices i and j is chosen independently with a certain probability p_{ij} that is allowed to depend on i and j. This model contains popular random graph models such as the Erdős-Rényi random graph, planted partition graphs, and random graphs with given expected degrees. For this class of random graphs, bounds on the spectral gap have been proved by Chung and Radcliffe (2011). These bounds can directly be applied to derive bounds on the resistance distances. It is not surprising to see that hitting times are meaningless, because these graphs are expander graphs and the random walk mixes fast. The model of Erdős-Rényi graphs with planted partitions is more interesting because it gives insight to the question how strongly clustered the graph can be before our results break down.

Theorem 12 (Chung and Radcliffe, 2011) *Let* G *be a random graph where edges between vertices* i *and* j *are put independently with probabilities* p_{ij}*. Consider the normalized Laplacian* L_{sym}*, and define the expected normalized Laplacian as the matrix* $\overline{L_{sym}} :=$

$I - \overline{D^{-1/2}}\overline{A}\overline{D^{-1/2}}$ where $\overline{A}_{ij} = E(A_{ij}) = p_{ij}$ and $\overline{D} = E(D)$. Let $\overline{d_{\min}}$ be the minimal expected degree. Denote the eigenvalues of L_{sym} by μ, the ones of $\overline{L_{sym}}$ by $\overline{\mu}$. Choose $\varepsilon > 0$. Then there exists a constant $k = k(\varepsilon)$ such that if $\overline{d_{\min}} > k \log(n)$, then with probability at least $1 - \varepsilon$,

$$\forall j = 1, ..., n: \quad |\mu_j - \overline{\mu_j}| \leq 2\sqrt{\frac{3\log(4n/\varepsilon)}{\overline{d_{\min}}}}.$$

Random graphs with given expected degrees. For a graph of n vertices we have n parameters $\bar{d}_1, ..., \bar{d}_n > 0$. For each pair of vertices v_i and v_j, we independently place an edge between these two vertices with probability $\bar{d}_i\bar{d}_j / \sum_{k=1}^{n} \bar{d}_k$. It is easy to see that in this model, vertex v_i has expected degree $\bar{d}_i$ (cf. Section 5.3. in Chung and Lu, 2006 for background reading).

Corollary 13 (Random graphs with given expected degrees) *Consider any sequence of random graphs with expected degrees such that $\overline{d_{\min}} = \omega(\log n)$. Then the commute distances satisfy for all $i \neq j$,*

$$\left|\frac{1}{\mathrm{vol}(G)} C_{ij} - \left(\frac{1}{d_i} + \frac{1}{d_j}\right)\right| \Big/ \left(\frac{1}{d_i} + \frac{1}{d_j}\right) = O\left(\frac{1}{\log(2n)}\right) \longrightarrow 0, \text{ almost surely.}$$

Planted partition graphs. Assume that the n vertices are split into two "clusters" of equal size. We put an edge between two vertices u and v with probability p_{within} if they are in the same cluster and with probability $p_{between} < p_{within}$ if they are in different clusters. For simplicity we allow self-loops.

Corollary 14 (Random graph with planted partitions) *Consider an Erdős-Rényi graph with planted bisection. Assume that $p_{within} = \omega(\log(n)/n)$ and $p_{between}$ such that $np_{between} \to \infty$ (arbitrarily slow). Then, for all vertices u, v in the graph*

$$\left|\frac{1}{n} \cdot H_{ij} - 1\right| = O\left(\frac{1}{n\, p_{between}}\right) \to 0 \quad \text{in probability.}$$

This result is a nice example to show that even though there is a strong cluster structure in the graph, hitting times and commute distances cannot see this cluster structure any more, once the graph gets too large. Note that the corollary even holds if $np_{between}$ grows much slower than np_{within}. That is, the larger our graph, the more pronounced is the cluster structure. Nevertheless, the commute distance converges to a trivial result. On the other hand, we also see that the speed of convergence is $O(np_{\text{between}})$, that is, if $p_{\text{between}} = g(n)/n$ with a slowly growing function g, then convergence can be slow. We might need very large graphs before the degeneracy of the commute time will be visible.

5. Experiments

In this section we examine the convergence behavior of the commute distance in practice. We ran a large number of simulations, both on artificial and real world data sets, in order to evaluate whether the rescaled commute distance in a graph is close to its predicted limit expression $1/d_u + 1/d_v$ or not.We conducted simulations for artificial graphs (random geometric graphs, planted partition graphs, preferential attachment graphs) and real world data sets of various types and sizes (social networks, biological networks, traffic networks; up to 1.6 million vertices and 22 million edges). The general setup is as follows. Given a graph, we compute the pairwise resistance distance R_{ij} between all points. The relative deviation between resistance distance and predicted result is then given by

$$RelDev(i,j) := \frac{|R_{ij} - 1/d_i - 1/d_j|}{R_{ij}}.$$

Note that we report relative rather than absolute deviations because this is more meaningful if R_{ij} is strongly fluctuating on the graph. It also allows to compare the behavior of different graphs with each other. In all figures, we then report the maximum, mean and median relative deviations. In small and moderate sized graphs, these operations are taken over all pairs of points. In some of the larger graphs, computing all pairwise distances is prohibitive. In these cases, we compute mean, median and maximum based on a random subsample of vertices (see below for details).

The bottom line of our experiments is that in nearly all graphs, the deviations are small. In particular, this also holds for many moderate sized graphs, even though our results are statements about $n \to \infty$. This shows that the limit results for the commute distance are indeed relevant for practice.

5.1 Random Geometric Graphs

We start with the class of random geometric graphs, which is very important for machine learning. We use a mixture of two Gaussian distributions on $\mathbb{R}^d$. The first two dimensions contain a two-dimensional mixture of two Gaussians with varying separation (centers $(-sep/2, 0)$ and $(+sep/2, 0)$, covariance matrix $0.2 \cdot Id$, mixing weights 0.5 for both Gaussians). The remaining $d-2$ dimensions contain Gaussian noise with variance 0.2 as well. From this distribution we draw n sample points. Based on this sample, we either compute the unweighted symmetric kNN graph, the unweighted ε-graphs or the Gaussian similarity graph. In order to be able to compare the results between these three types of graphs we match the parameters of the different graphs: given some value k for the kNN-graph we choose the values of ε for the ε-graph and σ for the Gaussian graph as the maximal k-nearest neighbor distance in the data set.

Figure 1 shows the results of the simulations. We can see that the deviations decrease fast with the sample size. In particular, already for small sample sizes reported, the maximal deviations get very small. The more clustered the data is (separation is larger), the larger the deviations get. This is the case as the deviation bound scales inversely with the spectral gap, which gets larger the more clustered the data is. The deviations also decrease with

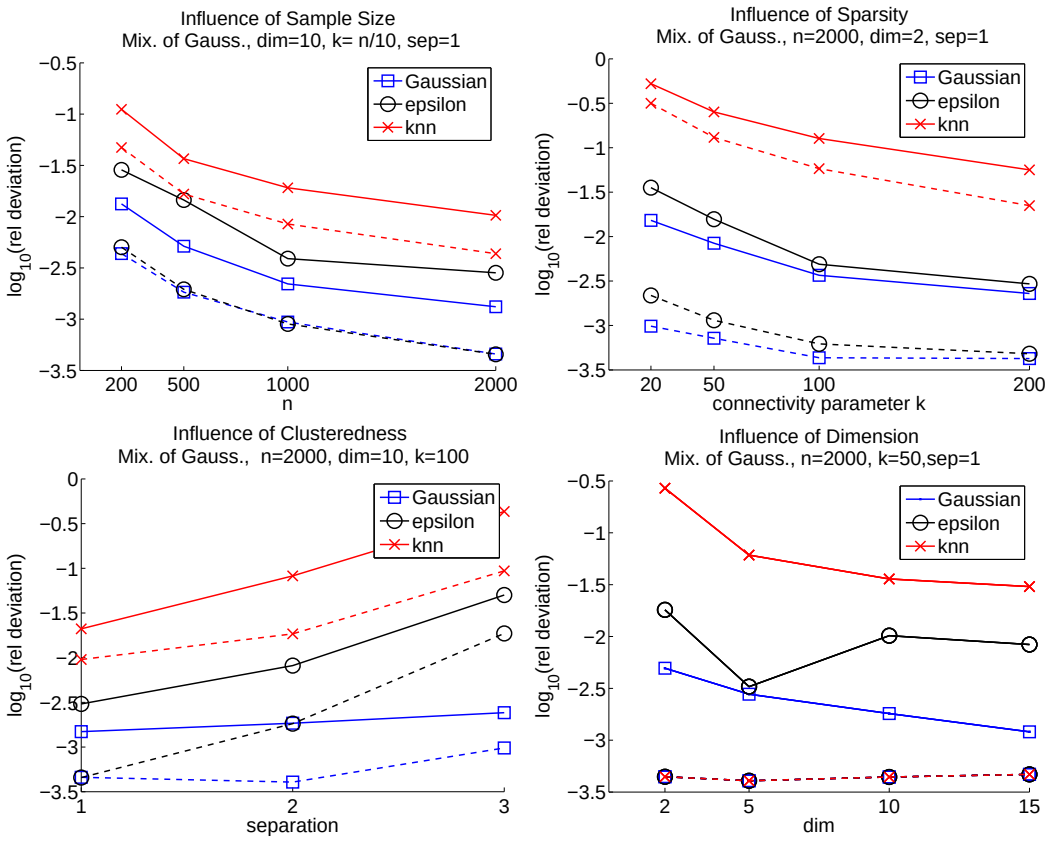


Figure 1: Deviations in random geometric graphs. Solid lines show the maximum relative deviation, dashed lines the mean relative deviation. See text for more details.

increasing dimension, as predicted by our bounds. The intuitive explanation is that in higher dimensions, geometric graphs mix faster as there exist more "shortcuts" between the two sides of the point cloud. For a similar reason, the deviation bounds also decrease with increasing connectivity in the graph. All in all, the deviations are very small even though our sample sizes in these experiments are modest.

5.2 Planted Partition Graphs

Next we consider graphs according to the planted partition model. We modeled a graph with n vertices and two equal sized clusters with connectivity parameters p_{within} and $p_{between}$. As the results in Figure 2 show, the deviation decreases rapidly when n increases, and it decreases when the cluster structure becomes less pronounced.

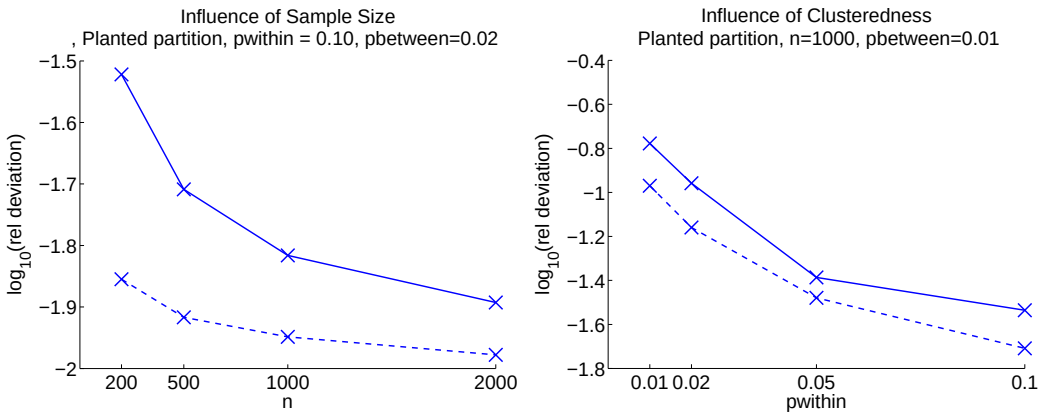


Figure 2: Deviations in planted partition graphs. Solid lines show the maximum relative deviation, dashed lines the mean relative deviation. See text for more details.

5.3 Preferential Attachment Graphs

An important class of random graphs is the preferential attachment model (Barabási and Albert, 1999), because it can be used to model graphs with power law behavior. Our current convergence proofs cannot be carried over to preferential attachment graphs: the minimal degree in preferential attachment graphs is constant, so our proofs break down. However, our simulation results show that approximating commute distances by the limit expression $1/d_u + 1/d_v$ gives accurate approximations as well. This finding indicates that our convergence results seem to hold even more generally than our theoretical findings suggest.

In our simulation we generated preferential attachment graphs according to the following standard procedure: Starting with a graph that consists of two vertices connected by an edge, in each time step we add a new vertex to the graph. The new vertex is connected by a fixed number of edges to existing vertices (this number is called NumLinks in the figures below). The target vertices of these edges are chosen randomly among all existing vertices, where the probability to connect to a particular vertex is proportional to its degree. All edges in the graph are undirected. As an example, we show the adjacency matrix and the degree histogram for such a graph in Figure 3. The next plots in this figure shows the relative deviations in such a preferential attachment graph, plotted against the sparsity of the graph (number of outlinks). Overall we can see that the deviations are very small, they are on the same scale as the ones in all the other simulations above. The mean deviations simply decrease as the connectivity of the graph increases. The maximum deviations show an effect that is different from what we have seen in the other graphs: while it decreases in the sparse regime, it starts to increase again when the graph becomes denser. However, investigating this effect more closely reveals that it is just generated by the three vertices in the graph which have the largest degrees. If we exclude these three vertices when computing the maximum deviation, then the maximum deviation decreases as gracefully as the mean deviation. All in all we can say that in the sparse regime, the commute distance can be well

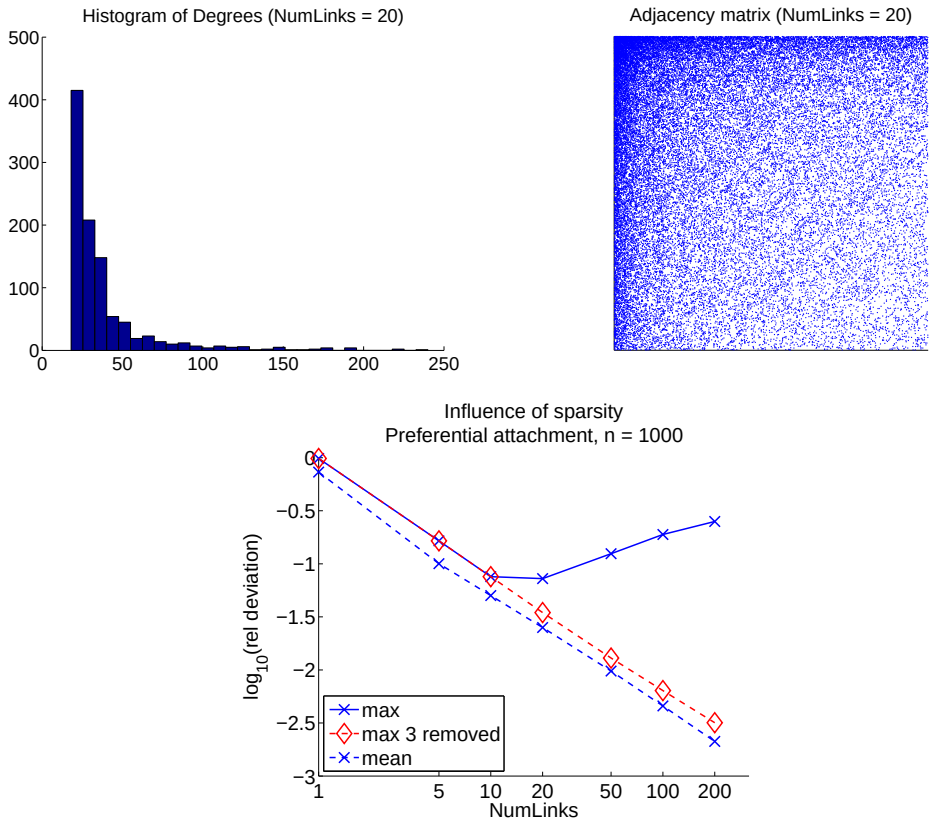


Figure 3: Relative deviations for a preferential attachment graph. First two figures: illustration of such a graph in terms of histogram of degrees and heat plot of the adjacency matrix. Third figure: Relative deviations. As before the solid line shows the maximum relative deviation and the dashed line the mean relative deviation. The line "max 3 removed" refers to the maximum deviation when we exclude the three highest-degree vertices from computing the maximum. See text for details.

approximated by the expression $1/d_u + 1/d_v$. In the dense regime, the same is true unless u or v is among the very top degree vertices.

5.4 Real World Data

Now we consider the deviations for a couple of real world data sets. We start with similarity graphs on real data, as they are often used in machine learning. As example we use the full USPS data set of handwritten digits (9298 points in 256 dimensions) and consider the different forms of similarity graphs (kNN, ε, Gaussian) with varying connectivity parameter. In Figure 4 we can see that overall, the relative errors are pretty small.

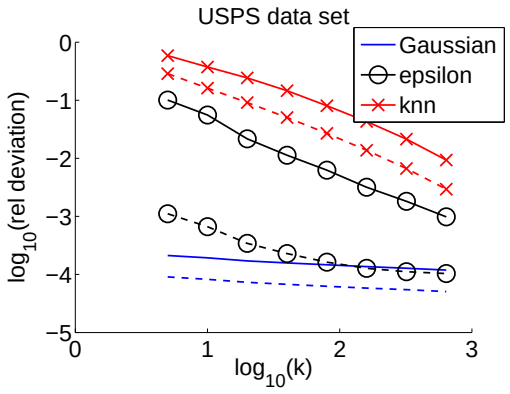


Figure 4: Relative deviations on different similarity graphs on the USPS data. As before the solid line shows the maximum relative deviation and the dashed line the mean relative deviation. See text for details.

Name	Type of Network	Number Vertices	Number Edges	mean rel dev	median rel dev	max rel dev
mousegene[2]	gene network	42923	28921954	0.12	0.01	0.70
jazz[1]	social network	198	5484	0.09	0.06	0.51
soc-Slashdot0902[2]	social network	82168	1008460	0.10	0.06	0.33
cit-HepPh[2]	citation graph	34401	841568	0.09	0.07	0.25
soc-sign-epinions[2]	social network	119130	1408534	0.14	0.09	0.53
pdb1HYS[2]	protein databank	36417	4308348	0.12	0.10	0.28
as-Skitter[2]	internet topology	1694616	22188418	0.16	0.12	0.54
celegans[1]	neural network	453	4065	0.18	0.12	0.77
email[1]	email traffic	1133	10903	0.15	0.13	0.77
Amazon0505[2]	co-purchase	410236	4878874	0.28	0.24	0.80
CoPapersCiteseer[2]	coauthor graph	434102	32073440	0.33	0.30	0.76
PGP[1]	user network	10680	48632	0.54	0.53	0.97

Figure 5: Relative Error of the approximation in real world networks. Downloaded from: 1, `http://deim.urv.cat/~aarenas/data/welcome.htm` and 2, `http://www.cise.ufl.edu/research/sparse/matrices/`. See text for details.

Furthermore, we consider a number of network data sets that are available online, see Figure 5 for a complete list. Some of these networks are directed, but we use their undirected versions. In cases the graphs were not connected, we ran the analysis on the largest connected component. On the smaller data sets, we computed all pairwise commute distances and used all these values to compute mean, median and maximum relative deviations. On the larger data sets we just draw 20 vertices at random, then compute all pairwise commute distances between these 20 vertices, and finally evaluate mean, median and maximum

based on this subset. Figure 5 shows the mean, median and maximum relative commute distances in all these networks. We can see that in many of these data sets, the deviations are reasonable small. Even though they are not as small as in the artificial data sets, they are small enough to acknowledge that our approximation results tend to hold in real world graphs.

6. Proofs for the Flow-Based Approach

For notational convenience, in this section we work with the resistance distance $R_{uv} = C_{uv}/\operatorname{vol}(G)$ instead of the commute distance C_{uv}, then we do not have to carry the factor $1/\operatorname{vol}(G)$ everywhere.

6.1 Lower Bound

It is easy to prove that the resistance distance between two points is lower bounded by the sum of the inverse degrees.

Proposition 15 (Lower bound) *Let G be a weighted, undirected, connected graph and consider two vertices s and t, $s \neq t$. Assume that G remains connected if we remove s and t. Then the effective resistance between s and t is bounded by*

$$R_{st} \geq \frac{Q_{st}}{1 + w_{st} Q_{st}},$$

where $Q_{st} = 1/(d_s - w_{st}) + 1/(d_t - w_{st})$. Note that if s and t are not connected by a direct edge (that is, $w_{st} = 0$), then the right hand side simplifies to $1/d_s + 1/d_t$.

Proof. The proof is based on Rayleigh's monotonicity principle that states that increasing edge weights (conductances) in the graph can never increase the effective resistance between two vertices (cf. Corollary 7 in Section IX.2 of Bollobas, 1998). Given our original graph G, we build a new graph G' by setting the weight of all edges to infinity, except the edges that are adjacent to s or t (setting the weight of an edge to infinity means that this edge has infinite conductance and no resistance any more). This can also be interpreted as taking all vertices except s and t and merging them to one super-node a. Now our graph G' consists of three vertices s, a, t with several parallel edges from s to a, several parallel edges from a to t, and potentially the original edge between s and t (if it existed in G). Exploiting the laws in electrical networks (resistances add along edges in series, conductances add along edges in parallel; see Section 2.3 in Lyons and Peres (2010) for detailed instructions and examples) leads to the desired result. ■

6.2 Upper Bound

This is the part that requires the hard work. Our proof is based on a theorem that shows how the resistance between two points in the graph can be computed in terms of flows on the graph. The following result is taken from Corollary 6 in Section IX.2 of Bollobas (1998).

Theorem 16 (Resistance in terms of flows, cf. Bollobas, 1998) *Let $G = (V, E)$ be a weighted graph with edge weights w_e ($e \in E$). The effective resistance R_{st} between two fixed vertices s and t can be expressed as*

$$R_{st} = \inf \left\{ \sum_{e \in E} \frac{u_e^2}{w_e} \;\Big|\; u = (u_e)_{e \in E} \text{ unit flow from } s \text{ to } t \right\}.$$

Note that evaluating the formula in the above theorem for any fixed flow leads to an upper bound on the effective resistance. The key to obtaining a tight bound is to distribute the flow as widely and uniformly over the graph as possible.

For the case of geometric graphs we are going to use a grid on the underlying space to construct an efficient flow between two vertices. Let $X_1, ..., X_n$ be a fixed set of points in $\mathbb{R}^d$ and consider a geometric graph G with vertices $X_1, ..., X_n$. Fix any two of them, say $s := X_1$ and $t := X_2$. Let $\mathcal{X} \subset \mathbb{R}^d$ be a connected set that contains both s and t. Consider a regular grid with grid width g on $\mathcal{X}$. We say that grid cells are neighbors of each other if they touch each other in at least one edge.

Definition 17 (Valid grid) *We call the grid* valid *if the following properties are satisfied:*

1. *The grid width is not too small: Each cell of the grid contains at least one of the points $X_1, ..., X_n$.*
2. *The grid width g is not too large: Points in the same or neighboring cells of the grid are always connected in the graph G.*
3. *Relation between grid width and geometry of $\mathcal{X}$: Define the bottleneck h of the region $\mathcal{X}$ as the largest u such that the set $\{x \in \mathcal{X} \;|\; dist(x, \partial\mathcal{X}) > u/2\}$ is connected.*

   We require that $\sqrt{d}\, g \leq h$ (a cube of side length g should fit in the bottleneck).

Under the assumption that a valid grid exists, we can prove the following general proposition that gives an upper bound on the resistance distance between vertices in a fixed geometric graph. Note that proving the existence of the valid grid will be an important part in the proofs of Theorems 3 and 4.

Proposition 18 (Resistance on a fixed geometric graph) *Consider a fixed set of points $X_1, ..., X_n$ in some connected region $\mathcal{X} \subset \mathbb{R}^d$ and a geometric graph on $X_1, ..., X_n$. Assume that $\mathcal{X}$ has bottleneck not smaller than h (where the bottleneck is defined as in the definition of a valid grid). Denote $s = X_1$ and $t = X_2$. Assume that s and t can be connected by a straight line that stays inside $\mathcal{X}$ and has distance at least $h/2$ to $\partial\mathcal{X}$. Denote the distance between s and t by $d(s, t)$. Let g be the width of a valid grid on $\mathcal{X}$ and assume that $d(s, t) > 4\sqrt{d}\, g$. By $N_{\min}$ denote the minimal number of points in each grid cell, and define a as*

$$a \;:=\; \left\lfloor h/(2g\sqrt{d-1}) \right\rfloor.$$

Assume that points that are connected in the graph are at most Q grid cells apart from each other (for example, two points in the two grey cells in Figure 6b are 5 cells apart from each other). Then the effective resistance between s and t can be bounded as follows:

$$\begin{aligned}
\textit{In case } d>3: \quad & R_{st} \le \frac{1}{d_s}+\frac{1}{d_t}+\left(\frac{1}{d_s}+\frac{1}{d_t}\right)\frac{1}{N_{\min}}+\frac{1}{N_{\min}^2}\left(6+\frac{d(s,t)}{g(2a+1)^3}+2Q\right)\\
\textit{In case } d=3: \quad & R_{st} \le \frac{1}{d_s}+\frac{1}{d_t}+\left(\frac{1}{d_s}+\frac{1}{d_t}\right)\frac{1}{N_{\min}}\\
& \qquad +\frac{1}{N_{\min}^2}\left(4\log(a)+8+\frac{d(s,t)}{g(2a+1)^2}+2Q\right)\\
\textit{In case } d=2: \quad & R_{st} \le \frac{1}{d_s}+\frac{1}{d_t}+\left(\frac{1}{d_s}+\frac{1}{d_t}\right)\frac{1}{N_{\min}}+\frac{1}{N_{\min}^2}\left(4a+2+\frac{d(s,t)}{g(2a+1)}+2Q\right)
\end{aligned}$$

Proof. The general idea of the proof is to construct a flow from s to t with the help of the underlying grid. On a high level, the construction of the proof is not so difficult, but the details are lengthy and a bit tedious. The rest of this section is devoted to it.

Construction of the flow — overview. Without loss of generality we assume that there exists a straight line connecting s and t which is along the first dimension of the space. By $C(s)$ we denote the grid cell in which s sits.

Step 0. We start a unit flow in vertex s.

Step 1. We make a step to all neighbors $\mathrm{Neigh}(s)$ of s and distribute the flow uniformly over all edges. That is, we traverse d_s edges and send flow $1/d_s$ over each edge (see Figure 6a). This is the crucial step which, ultimately, leads to the desired limit result.

Step 2. Some of the flow now sits inside $C(s)$, but some of it might sit outside of $C(s)$. In this step, we bring back all flow to $C(s)$ in order to control it later on (see Figure 6b).

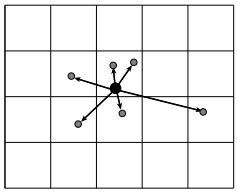
Step 3. We now distribute the flow from $C(s)$ to a larger region, namely to a hypercube $H(s)$ of side length h that is perpendicular to the linear path from s to t and centered at $C(s)$ (see the hypercubes in Figure 6c). This can be achieved in several substeps that will be defined below.

Step 4. We now traverse from $H(s)$ to an analogous hypercube $H(t)$ located at t using parallel paths, see Figure 6c.

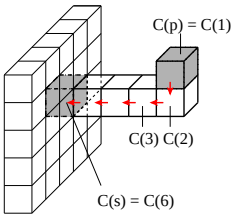
Step 5. From the hypercube $H(t)$ we send the flow to the neighborhood $\mathrm{Neigh}(t)$ (this is the "reverse" of steps 2 and 3).

Step 6. From $\mathrm{Neigh}(t)$ we finally send the flow to the destination t ("reverse" of step 1).

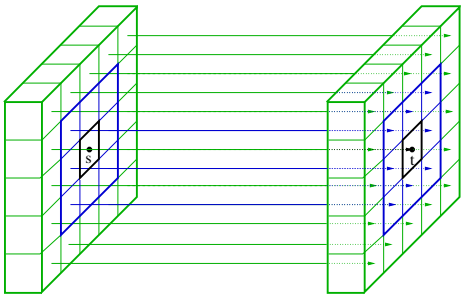
Details of the flow construction and computation of the resistance between s and t in the general case $d>3$. We now describe the individual steps and their contribution to the



(a) Step 1. Distributing the flow from s (black dot) to all its neighbors (grey dots).

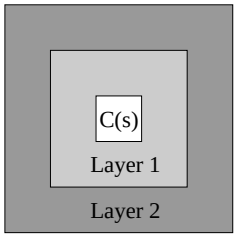


(b) Step 2. We bring back all flow from p to $C(s)$. Also shown in the figure is the hypercube to which the flow will be expanded in Step 3.

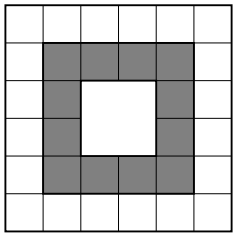


(c) Steps 3 and 4 of the flow construction: distribute the flow from $C(s)$ to a "hypercube" $H(s)$, then transmit it to a similar hypercube $H(t)$ and guide it to $C(t)$.

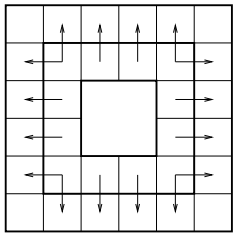
Figure 6: The flow construction — overview.



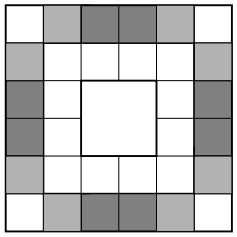
(a) Definition of layers.



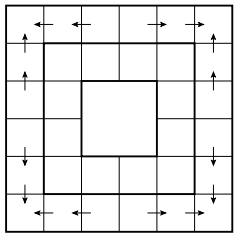
(b) Before Step 3A starts, all flow is uniformly distributed in Layer $i-1$ (dark area).



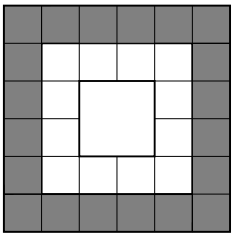
(c) Step 3A then distributes the flow from Layer $i-1$ to the adjacent cells in Layer i



(d) After Step 3A: all flow is in Layer i, but not yet uniformly distributed



(e) Step 3B redistributes the flow in Layer i.



(f) After Step 3B, the flow is uniformly distributed in Layer i.

Figure 7: Details of Step 3 between Layers $i-1$ and i. The first row corresponds to the expansion phase, the second row to the redistribution phase. The figure is shown for the case of $d = 3$.

bound on the resistance. We start with the general case $d > 3$. We will discuss the special cases $d = 2$ and $d = 3$ below.

In the computations below, by the "contribution of a step" we mean the part of the sum in Theorem 16 that goes over the edges considered in the current step.

Step 1: We start with a unit flow at s that we send over all d_s adjacent edges. This leads to flow $1/d_s$ over d_s edges. According to the formula in Theorem 16 this contributes

$$r_1 = d_s \cdot \frac{1}{d_s^2} = \frac{1}{d_s}$$

to the overall resistance R_{st}.

Step 2: After Step 1, the flow sits on all neighbors of s, and these neighbors are not necessarily all contained in $C(s)$. To proceed we want to re-concentrate all flow in $C(s)$. For each neighbor p of s, we thus carry the flow along a Hamming path of cells from p back to $C(s)$, see Figure 6b for an illustration.

To compute an upper bound for Step 2 we exploit that each neighbor p of s has to traverse at most Q cells to reach $C(s)$ (recall the definition of Q from the proposition). Let us fix p. After Step 1, we have flow of size $1/d_s$ in p. We now move this flow from p to all points in the neighboring cell $C(2)$ (cf. Figure 6b). For this we can use at least $N_{\min}$ edges. Thus we send flow of size $1/d_s$ over $N_{\min}$ edges, that is each edge receives flow $1/(d_s N_{\min})$. Summing the flow from $C(p)$ to $C(2)$, for all points p, gives

$$d_s N_{\min} \left(\frac{1}{d_s N_{\min}} \right)^2 = \frac{1}{d_s N_{\min}},$$

Then we transport the flow from $C(2)$ along to $C(s)$. Between each two cells on the way we can use $N_{\min}^2$ edges. Note, however, that we need to take into account that some of these edges might be used several times (for different points p). In the worst case, $C(2)$ is the same for all points p, in which case we send the whole unit flow over these edges. This amounts to flow of size $1/(N_{\min}^2)$ over $(Q-1)N_{\min}^2$ edges, that is a contribution of

$$\frac{Q-1}{N_{\min}^2}.$$

Altogether we obtain

$$r_2 \leq \frac{1}{d_s N_{\min}} + \frac{Q}{N_{\min}^2}.$$

Step 3: At the beginning of this step, the complete unit flow resides in the cube $C(s)$. We now want to distribute this flow to a "hypercube" of three dimensions (no matter what d

is, as long as $d > 3$) that is perpendicular to the line that connects s and t (see Figure 6c, where the case of $d = 3$ and a 2-dimensional "hypercube" are shown). To distribute the flow to this cube we divide it into layers (see Figure 7a). Layer 0 consists of the cell $C(s)$ itself, the first layer consists of all cells adjacent to $C(s)$, and so on. Each side of Layer i consists of

$$l_i = (2i + 1)$$

cells. For the 3-dimensional cube, the number z_i of grid cells in Layer i, $i \geq 1$, is given as

$$z_i = \underbrace{6 \cdot (2i-1)^2}_{\text{interior cells of the faces}} + \underbrace{12 \cdot (2i-1)}_{\text{cells along the edges (excluding corners)}} + \underbrace{8}_{\text{corner cells}} = 24i^2 + 2.$$

All in all we consider

$$a = \left\lfloor h/(2g\sqrt{d-1}) \right\rfloor \leq \left\lfloor h/(2(g-1)\sqrt{d-1}) \right\rfloor$$

layers, so that the final layer has diameter just a bit smaller than the bottleneck h. We now distribute the flow stepwise through all layers, starting with unit flow in Layer 0. To send the flow from Layer $i-1$ to Layer i we use two phases, see Figure 7 for details. In the "expansion phase" 3A(i) we transmit the flow from Layer $i-1$ to all adjacent cells in Layer i. In the "redistribution phase" 3B(i) we then redistribute the flow in Layer i to achieve that it is uniformly distributed in Layer i. In all phases, the aim is to use as many edges as possible.

Expansion phase 3A(i). We can lower bound the number of edges between Layer $i-1$ and Layer i by $z_{i-1}N_{\min}^2$: each of the z_{i-1} cells in Layer $i-1$ is adjacent to at least one of the cells in Layer i, and each cell contains at least $N_{\min}$ points. Consequently, we can upper bound the contribution of the edges in the expansion phase 3A(i) to the resistance by

$$r_{3A(i)} \leq z_{i-1}N_{\min}^2 \cdot \left(\frac{1}{z_{i-1}N_{\min}^2}\right)^2 = \frac{1}{z_{i-1}N_{\min}^2}.$$

Redistribution phase 3B(i). We make a crude upper bound for the redistribution phase. In this phase we have to move some part of the flow from each cell to its neighboring cells. For simplicity we bound this by assuming that for each cell, we had to move *all* its flow to neighboring cells. By a similar argument as for Step 3A(i), the contribution of the redistribution step can be bounded by

$$r_{3B(i)} \leq z_i N_{\min}^2 \cdot \left(\frac{1}{z_i N_{\min}^2}\right)^2 = \frac{1}{z_i N_{\min}^2}.$$

All of Step 3. All in all we have a layers. Thus the overall contribution of Step 3 to the resistance can be bounded by

$$r_3 \;=\; \sum_{i=1}^{a} r_{3A(i)} + r_{3B(i)} \;\leq\; \frac{2}{N_{\min}^2} \sum_{i=1}^{a} \frac{1}{z_{i-1}} \;\leq\; \frac{2}{N_{\min}^2} \left(1 + \frac{1}{24} \sum_{i=1}^{a-1} \frac{1}{i^2}\right) \;\leq\; \frac{3}{N_{\min}^2}.$$

To see the last inequality, note that the sum $\sum_{i=1}^{a-1} 1/i^2$ is a partial sum of the over-harmonic series that converges to a constant smaller than 2.

Step 4: Now we transfer all flow in "parallel cell paths" from $H(s)$ to $H(t)$. We have $(2a+1)^3$ parallel rows of cells going from $H(s)$ to $H(t)$, each of them contains $d(s,t)/g$ cells. Thus all in all we traverse $(2a+1)^3 N_{\min}^2 d(s,t)/g$ edges, and each edge carries flow $1/((2a+1)^3 N_{\min}^2)$. Thus step 4 contributes

$$r_4 \leq (2a+1)^3 N_{\min}^2 \frac{d(s,t)}{g} \cdot \left(\frac{1}{(2a+1)^3 N_{\min}^2}\right)^2 \;=\; \frac{d(s,t)}{g(2a+1)^3 N_{\min}^2}.$$

Step 5 is completely analogous to steps 2 and 3, with the analogous contribution $r_5 = \frac{1}{d_t N_{\min}} + \frac{Q}{N_{\min}^2} + r_3$.

Step 6 is completely analogous to step 1 with overall contribution of $r_6 = 1/d_t$.

Summing up the general case $d > 3$. All these contributions lead to the following overall bound on the resistance in case $d > 3$:

$$R_{st} \leq \frac{1}{d_s} + \frac{1}{d_t} + \left(\frac{1}{d_s} + \frac{1}{d_t}\right) \frac{1}{N_{\min}} + \frac{1}{N_{\min}^2} \left(6 + \frac{d(s,t)}{g(2a+1)^3} + 2Q\right)$$

with a and Q as defined in Proposition 18. This is the result stated in the proposition for case $d > 3$.

Note that as spelled out above, the proof works whenever the dimension of the space satisfies $d > 3$. In particular, note that even if d is large, we only use a 3-dimensional "hypercube" in Step 3. It is sufficient to give the rate we need, and carrying out the construction for higher-dimensional hypercube (in particular Step 3B) is a pain that we wanted to avoid.

The special case $d = 3$. In this case, everything works similar to above, except that we we only use a 2-dimensional "hypercube" (this is what we always show in the figures). The only place in the proof where this really makes a difference is in Step 3. The number z_i of grid cells in Layer i is given as $z_i = 8i$. Consequently, instead of obtaining an over-harmonic sum in r_3 we obtain a harmonic sum. Using the well-known fact that $\sum_{i=1}^{a} 1/i \leq \log(a) + 1$ we obtain

$$r_3 \leq \frac{2}{N_{\min}^2}\left(1 + \frac{1}{8}\sum_{i=1}^{a-1}\frac{1}{i}\right) \;\leq\; \frac{2}{N_{\min}^2}\left(2 + \log(a)\right).$$

In Step 4 we just have to replace the terms $(2a+1)^3$ by $(2a+1)^2$. This leads to the result in Proposition 18.

The special case $d = 2$. Here our "hypercube" only consists of a "pillar" of $2a+1$ cells. The fundamental difference to higher dimensions is that in Step 3, the flow does not have so much "space" to be distributed. Essentially, we have to distribute all unit flow through a "pillar", which results in contributions

$$r_3 \leq \frac{2a+1}{N_{\min}^2}$$
$$r_4 \leq \frac{d(s,t)}{g}\frac{1}{(2a+1)N_{\min}^2}.$$

This concludes the proof of Proposition 18. ■

Let us make a couple of technical remarks about this proof. For the ease of presentation we simplified the proof in a couple of respects.

Strictly speaking, we do not need to distribute the whole unit flow to the outmost Layer a. The reason is that in each layer, a fraction of the flow already "branches off" in direction of t. We simply ignore this leaving flow when bounding the flow in Step 3, our construction leads to an upper bound. It is not difficult to take the outbound flow into account, but it does not change the order of magnitude of the final result. So for the ease of presentation we drop this additional complication and stick to our rough upper bound.

When we consider Steps 2 and 3 together, it turns out that we might have introduced some loops in the flow. To construct a proper flow, we can simply remove these loops. This would then just reduce the contribution of Steps 2 and 3, so that our current estimate is an overestimation of the whole resistance.

The proof as it is spelled out above considers the case where s and t are connected by a straight line. It can be generalized to the case where they are connected by a piecewise linear path. This does not change the result by more than constants, but adds some technicality at the corners of the paths.

The construction of the flow only works if the bottleneck of $\mathcal{X}$ is not smaller than the diameter of one grid cell, if s and t are at least a couple of grid cells apart from each other, and if s and t are not too close to the boundary of $\mathcal{X}$. We took care of these conditions in Part 3 of the definition of a valid grid.

6.3 Proof of the Theorems 3 and 4

First of all, note that by Rayleigh's principle (cf. Corollary 7 in Section IX.2 of Bollobas, 1998) the effective resistance between vertices cannot decrease if we delete edges from the graph. Given a sample from the underlying density p, a random geometric graph based on this sample, and some valid region $\mathcal{X}$, we first delete all points that are not in $\mathcal{X}$. Then we consider the remaining geometric graph. The effective resistances on this graph are upper bounds on the resistances of the original graph. Then we conclude the proofs with the following arguments:

Proof of Theorem 3. The lower bound on the deviation follows immediately from Proposition 15. The upper bound is a consequence of Proposition 18 and well known properties of random geometric graphs (summarized in the appendix). In particular, note that we can choose the grid width $g := \varepsilon/(2\sqrt{d-1})$ to obtain a valid grid. The quantity $N_{\min}$ can be bounded as stated in Proposition 28 and is of order $n\varepsilon^d$, the degrees behave as described in Proposition 29 and are also of order $n\varepsilon^d$ (we use $\delta = 1/2$ in these results for simplicity). The quantity a in Proposition 18 is of the order $1/\varepsilon$, and Q can be bounded by $Q = \varepsilon/g$ and by the choice of g is indeed a constant. Plugging all these results together leads to the final statement of the theorem. ■

Proof of Theorem 4. This proof is analogous to the ε-graph. As grid width g we choose $g = R_{k,\min}/(2\sqrt{d-1})$ where $R_{k,\min}$ is the minimal k-nearest neighbor distance (note that this works for both the symmetric and the mutual kNN-graph). Exploiting Propositions 28 and 30 we can see that $R_{k,\min}$ and $R_{k,\max}$ are of order $(k/n)^{1/d}$, the degrees and $N_{\min}$ are of order k, a is of the order $(n/k)^{1/d}$ and Q a constant. Now the statements of the theorem follow from Proposition 18. ■

7. Proofs for the Spectral Approach

In this section we present the proofs of the results obtained by spectral arguments.

7.1 Proof of Propositions 1 and 5

First we prove the general formulas to compute and approximate the hitting times.

Proof of Proposition 1. For the hitting time formula, let $u_1, \ldots, u_n$ be an orthonormal set of eigenvectors of the matrix $D^{-1/2}WD^{-1/2}$ corresponding to the eigenvalues $\mu_1, \ldots, \mu_n$. Let u_{ij} denote the j-th entry of u_i. According to Lovász (1993) the hitting time is given by

$$H_{ij} = \mathrm{vol}(G) \sum_{k=2}^{n} \frac{1}{1-\mu_k} \left(\frac{u_{kj}^2}{d_j} - \frac{u_{ki}u_{kj}}{\sqrt{d_i d_j}} \right).$$

A straightforward calculation using the spectral representation of L_{sym} yields

$$H_{ij} = \mathrm{vol}(G) \left\langle \tfrac{1}{\sqrt{d_j}} e_j, \sum_{k=2}^{n} \frac{1}{1-\mu_k} \left\langle u_k, \tfrac{1}{\sqrt{d_j}} e_j - \tfrac{1}{\sqrt{d_i}} e_i \right\rangle u_k \right\rangle$$

$$= \mathrm{vol}(G) \left\langle \tfrac{1}{\sqrt{d_j}} e_j, L_{\mathrm{sym}}^{\dagger} \left(\tfrac{1}{\sqrt{d_j}} e_j - \tfrac{1}{\sqrt{d_i}} e_i \right) \right\rangle .$$

The result for the commute time follows from the one for the hitting times. ■

In order to prove Proposition 5 we first state a small lemma. For convenience, we set $A = D^{-1/2} W D^{-1/2}$ and $u_i = e_i / \sqrt{d_i}$. Furthermore, we are going to denote the projection on the eigenspace of the j-the eigenvalue λ_j of A by P_j.

Lemma 19 (Pseudo-inverse $L_{\mathrm{sym}}^{\dagger}$) *The pseudo-inverse of the symmetric Laplacian satisfies*

$$L_{sym}^{\dagger} = I - P_1 + M,$$

where I denotes the identity matrix and M is given as follows:

$$M \;=\; \sum_{k=1}^{\infty} (A - P_1)^k \;=\; \sum_{r=2}^{n} \frac{\lambda_r}{1-\lambda_r} P_r. \tag{3}$$

Furthermore, for all $u, v \in \mathbb{R}^n$ we have

$$|\, \langle u, Mv \rangle \,| \;\leq\; \frac{1}{1-\lambda_2} \cdot \|(A-P_1)u\| \;\cdot\; \|(A-P_1)v\| + |\, \langle u \,,\; (A-P_1)v \rangle \,|. \tag{4}$$

Proof. The projection onto the null space of L_{sym} is given by $P_1 = \sqrt{d}\sqrt{d}^T / \sum_{i=1} d_i$ where $\sqrt{d} = (\sqrt{d_1}, \ldots, \sqrt{d_n})^T$. As the graph is not bipartite, $\lambda_n > -1$. Thus the pseudoinverse of L_{sym} can be computed as

$$L_{\mathrm{sym}}^{\dagger} = (I - A)^{\dagger} = (I - A + P_1)^{-1} - P_1 = \sum_{k=0}^{\infty} (A - P_1)^k - P_1.$$

Thus

$$\begin{aligned} M := \sum_{k=1}^{\infty} (A-P_1)^k \;&=\; \sum_{k=0}^{\infty} (A-P_1)^k (A-P_1) \\ = (\sum_{k=0}^{\infty} \sum_{r=2}^{n} \lambda_r^k P_r)(\sum_{r=2}^{n} \lambda_r P_r) \;&=\; (\sum_{r=2}^{n} \frac{1}{1-\lambda_r} P_r)(\sum_{r=2}^{n} \lambda_r P_r) \\ = \sum_{r=2}^{n} \frac{\lambda_r}{1-\lambda_r} P_r \end{aligned}$$

which proves Equation (3). By a little detour, we can also see

$$M \;=\; \sum_{k=0}^{\infty} (A-P_1)^k (A-P_1)^2 + (A-P_1) \;=\; (\sum_{r=2}^{n} \frac{1}{1-\lambda_r} P_r)(A-P_1)^2 + (A-P_1).$$

Exploiting that $(A - P_1)$ commutes with all P_r gives

$$\langle u, Mv\rangle = \left\langle (A-P_1)u \ , \ (\sum_{r=2}^{n} \frac{1}{1-\lambda_r} P_r)(A-P_1)v \right\rangle + \langle u, (A-P_1)v\rangle \, .$$

Applying the Cauchy-Schwarz inequality and the fact $\| \sum_{r=2}^{n} \frac{1}{1-\lambda_r} P_r \|_2 = 1/(1-\lambda_2)$ leads to the desired statement. ∎

Proof of Proposition 5. This proposition now follows easily from the Lemma above. Observe that

$$\begin{aligned}
\langle u_i, Au_j\rangle &= \frac{w_{ij}}{d_i d_j} \leq \frac{w_{\max}}{d_{\min}^2} \\
\|Au_i\|^2 &= \sum_{k=1}^{n} \frac{w_{ik}^2}{d_i^2 d_k} \leq \frac{w_{\max}}{d_{\min} d_i^2} \sum_k w_{ik} = \frac{w_{\max}}{d_{\min}} \frac{1}{d_i} \leq \frac{w_{\max}}{d_{\min}^2} \\
\|A(u_i - u_j)\|^2 &\leq \frac{w_{\max}}{d_{\min}} \left(\frac{1}{d_i} + \frac{1}{d_j} \right) \leq \frac{2 w_{\max}}{d_{\min}^2}.
\end{aligned}$$

Exploiting that $P_1(u_i - u_j) = 0$ we get for the hitting time

$$\begin{aligned}
\left| \frac{1}{\mathrm{vol}(G)} H_{ij} - \frac{1}{d_j} \right| &= |\, \langle u_j, M(u_j - u_i)\rangle \,| \\
&\leq \frac{1}{1-\lambda_2} \|Au_j\| \cdot \|A(u_j - u_i)\| + |\langle u_j, A(u_j - u_i)\rangle| \\
&\leq \frac{1}{1-\lambda_2} \frac{w_{\max}}{d_{\min}} \left(\frac{1}{\sqrt{d_j}} \sqrt{\frac{1}{d_i} + \frac{1}{d_j}} \right) + \frac{w_{ij}}{d_i d_j} + \frac{w_{jj}}{d_j^2} \\
&\leq \ 2 \frac{w_{\max}}{d_{\min}^2} \left(\frac{1}{1-\lambda_2} + 1 \right).
\end{aligned}$$

For the commute time, we note that

$$\begin{aligned}
\left| \frac{1}{\mathrm{vol}(G)} C_{ij} - \left(\frac{1}{d_i} + \frac{1}{d_j} \right) \right| &= \left| \, \langle u_i - u_j, M(u_i - u_j)\rangle \, \right| \\
&\leq \frac{1}{1-\lambda_2} \|A(u_i - u_j)\|^2 + \left| \, \langle u_i - u_j, A(u_i - u_j)\rangle \, \right| \\
&\leq \frac{w_{\max}}{d_{\min}} \left(\frac{1}{1-\lambda_2} + 2 \right) \left(\frac{1}{d_i} + \frac{1}{d_j} \right).
\end{aligned}$$

∎

We would like to point out that the key to achieving this bound is not to give in to the temptation to manipulate Eq. (3) directly, but to bound Eq. (4). The reason is that we can compute terms of the form $\langle u_i, Au_j\rangle$ and related terms explicitly, whereas we do not have any explicit formulas for the eigenvalues and eigenvectors in (3).

7.2 The Spectral Gap in Random Geometric Graphs

As we have seen above, a key ingredient in the approximation result for hitting times and commute distances is the spectral gap. In this section we show how the spectral gap can be lower bounded for random geometric graphs. We first consider the case of a fixed geometric graph. From this general result we then derive the results for the special cases of the ε-graph and the kNN-graphs. All graphs considered in this section are unweighted and undirected. We follow the strategy in Boyd et al. (2005) where the spectral gap is bounded by means of the Poincaré inequality (see Diaconis and Stroock, 1991, for a general introduction to this technique; see Cooper and Frieze, 2011, for a related approach in simpler settings). The outline of this technique is as follows: for each pair (X, Y) of vertices in the graph we need to select a path γ_{XY} in the graph that connects these two vertices. In our case, this selection is made in a random manner. Then we need to consider all edges in the graph and investigate how many of the paths γ_{XY}, on average, traverse this edge. We need to control the maximum of this "load" over all edges. The higher this load is, the more pronounced is the bottleneck in the graph, and the smaller the spectral gap is. Formally, the spectral gap is related to the maximum average load b as follows.

Proposition 20 (Spectral gap, Diaconis and Stroock, 1991) *Consider a finite, connected, undirected, unweighted graph that is not bipartite. For each pair of vertices $X \neq Y$ let P_{XY} be a probability distribution over all paths that connect X and Y and have uneven length. Let $(\gamma_{XY})_{X,Y}$ be a family of paths independently drawn from the respective P_{XY}. Define $b := \max_{\{e \text{ edge}\}} \mathbb{E}|\{\gamma_{XY} \mid e \in \gamma_{XY}\}|$. Denote by $|\gamma_{\max}|$ the maximum path length (where the length of the path is the number of edges in the path). Then the spectral gap in the graph is bounded as follows:*

$$1 - \lambda_2 \geq \frac{\mathrm{vol}(G)}{d_{\max}^2 |\gamma_{\max}| b} \qquad \text{and} \qquad 1 - |\lambda_n| \geq \frac{2}{d_{\max} |\gamma_{\max}| b}.$$

For deterministic sets Γ, this proposition has been derived as Corollary 1 and 2 in Diaconis and Stroock (1991). The adaptation for random selection of paths is straightforward, see Boyd et al. (2005).

The key to tight bounds based on Proposition 20 is a clever choice of the paths. We need to make sure that we distribute the paths as "uniformly" as possible over the whole graph. This is relatively easy to achieve in the special situation where $\mathcal{X}$ is a torus with uniform distribution (as studied in Boyd et al., 2005; Cooper and Frieze, 2011) because of symmetry arguments and the absence of boundary effects. However, in our setting with general $\mathcal{X}$ and p we have to invest quite some work.

7.2.1 Fixed Geometric Graph on the Unit Cube in $\mathbb{R}^d$

We first treat the special case of a fixed geometric graph with vertices in the unit cube $[0, 1]^d$ in $\mathbb{R}^d$. Consider a grid on the cube with grid width g. For now we assume that the grid cells are so small that points in neighboring cells are always connected in the geometric graph, and so large that each cell contains a minimal number of data points. We will specify the exact value of g later. In the following, cells of the grid are identified with their center points.

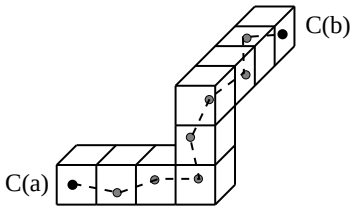


Figure 8: Canonical path between a and b. We first consider a "Hamming path of cells" between a and b. In all intermediate cells, we randomly pick a point.

Construction of the paths. Assume we want to construct a path between two vertices a and b that correspond to the points $a = (a_1, \ldots, a_d),\ b = (b_1, \ldots, b_d) \in [0,1]^d$. Let $C(a)$ and $C(b)$ denote the grid cells containing a and b, denote the centers of these cells by $c(a) = (c(a)_1, \ldots, c(a)_d)$ and $c(b) = (c(b)_1, \ldots, c(b)_d)$. We first construct a deterministic "cell path" between the cells $C(a)$ and $C(b)$ (see Figure 8). This path simply follows a Hamming path: starting at cell $C(a)$ we change the first coordinate until we have reached $c(b)_1$. For example, if $c(a)_1 < c(b)_1$ we traverse the cells

$$\big(c(a)_1, c(a)_2, \ldots, c(a)_d\big) \rightsquigarrow \big(c(a)_1 + g, c(a)_2, \ldots, c(a)_d\big) \rightsquigarrow \ldots \rightsquigarrow \big(c(b)_1, c(a)_2, \ldots, c(a)_d\big).$$

Then we move along the second coordinate from $c(a)_2$ until we have reached $c(b)_2$, that is we traverse the cells $(c(b)_1, *, c(a)_3, \ldots, c(a)_d)$. And so on. This gives a deterministic way of traversing adjacent cells from $C(a)$ to $C(b)$. Now we transform this deterministic "cell path" to a random path on the graph. In the special cases where a and b are in the same cell or in neighboring cells, we directly connect a and b by an edge. In the general case, we select one data point uniformly at random in each of the interior cells on the cell path. Then we connect the selected points to form a path. Note that we can always force the paths to have uneven lengths by adding one more point somewhere in between.

Proposition 21 (Path construction is valid) *Assume that (1) Each cell of the grid contains at least one data point. (2) Data points in the same and in neighboring cells are always connected in the graph. Then the graph is connected, and the paths constructed above are paths in the graph.*

Proof. Obvious, by construction of the paths. ■

In order to apply Proposition 20 we now need to compute the maximal average load of all paths.

Proposition 22 (Maximum average load for fixed graph on cube) *Consider a geometric graph on $[0,1]^d$ and the grid of width g on $[0,1]^d$. Denote by $N_{\min}$ and $N_{\max}$ the minimal and maximal number of points per grid cell. Construct a random set of paths as described above.*

1. *Let C be any fixed cell in the grid. Then there exist at most d/g^{d+1} pairs of cells (A, B) such that cell paths starting in cell A and ending in cell B pass through C.*

2. If the path construction is valid, then the maximal average load is upper bounded by

$$b \leq 1 + \left(\frac{N_{\max}^2}{N_{\min}^2} + 2\frac{N_{\max}}{N_{\min}} \right) \frac{d}{g^{d+1}}.$$

Proof. *Part 1.* We identify cells with their centers. Consider two different grid cells A and B with centers a and b. By construction, the Hamming path between A and B has the corners

$$\begin{aligned} a =& (a_1, a_2, a_3, \ldots, a_d) \rightsquigarrow (b_1, a_2, a_3, \ldots, a_d) \rightsquigarrow (b_1, b_2, a_3, \ldots, a_d) \\ & \rightsquigarrow \ldots \rightsquigarrow (b_1, b_2, b_3, \ldots, b_{d-1}, a_d) \rightsquigarrow (b_1, b_2, b_3, \ldots, b_{d-1}, b_d) = b. \end{aligned}$$

All cells on the path have the form $(b_1, b_2, \ldots, b_{l-1}, *, a_{l+1}, \ldots, a_d)$ where $*$ can take any value between a_l and b_l. A path can only pass through the fixed cell with center c if there exists some $l \in \{1, \ldots, d\}$ such that

$$(c_1, \ldots, c_d) = (b_1, b_2, \ldots, b_{l-1}, *, a_{l+1}, \ldots, a_d).$$

That is, there exists some $l \in \{1, \ldots, d\}$ such that

$$(I)\ \ b_i = c_i \text{ for all } i = 1, \ldots, l-1 \qquad \text{and} \qquad (II)\ \ a_i = c_i \text{ for all } i = l+1, \ldots, d.$$

For the given grid size g there are $1/g$ different cell centers per dimension. For fixed l there thus exist $1/g^{d-l+1}$ cell centers that satisfy (I) and $1/g^l$ cell centers that satisfy (II). So all in all there are $1/g^{d+1}$ pairs of cells A and B such that both (I) and (II) are satisfied for a fixed value of l. Adding up the possibilities for all choices of $l \in \{1, \ldots, d\}$ leads to the factor d.

Part 2. Fix an edge e in the graph and consider its two adjacent vertices v_1 and v_2. If v_1 and v_2 are in two different cells that are not neighbors to each other, then by construction none of the paths traverses the edge. If they are in the same cell, by construction at most one of the paths can traverse this edge, namely the one directly connecting the two points. The interesting case is the one where v_1 and v_2 lie in two neighboring grid cells C and $\tilde{C}$. If both cells are "interior" cells of the path, then by construction each edge connecting the two cells has equal probability of being selected. As there are at least $N_{\min}$ points in each cell, there are at least $N_{\min}^2$ different edges between these cells. Thus each of the edges between the cells is selected with probability at most $1/N_{\min}^2$. We know by Part 1 that there are at most d/g^{d+1} pairs of start/end cells. As each cell contains at most $N_{\max}$ points, this leads to $N_{\max}^2 d/g^{d+1}$ different paths passing through C. This is also an upper bound on the number of paths passing through both C and $\tilde{C}$. Thus, the number of paths using each edge is at most $dN_{\max}^2/(g^{d+1}N_{\min}^2)$.
If at least one of the cells is the start cell of the path, then the corresponding vertex, say v_1, is the start point of the path. If v_2 is an intermediate point, then it is selected with probability at most $1/N_{\min}$ (the case where v_2 is an end point has already been treated at the beginning). Similarly to the last case, there are at most $N_{\max} d/g^{d+1}$ paths that start in v_1 and pass through $\tilde{C}$. This leads to an average load of $dN_{\max}/(g^{d+1}N_{\min})$ on edge e. The same holds with the roles of v_1 and v_2 exchanged, leading to a factor 2.
The overall average load is now the sum of the average loads in the different cases. ■

7.2.2 Fixed Geometric Graph on a Domain $\mathcal{X}$ That Is Homeomorphic to a Cube

Now assume that $\mathcal{X} \subset \mathbb{R}^d$ is a compact subset that is homeomorphic to the cube $[0,1]^d$ in the following sense: we assume that there exists a homeomorphism $h : \mathcal{X} \to [0,1]^d$ and constants $0 < L_{\min} < L_{\max} < \infty$ such that for all $x, y \in \mathcal{X}$ we have

$$L_{\min}\|x - y\| \;\leq\; \|h(x) - h(y)\| \;\leq\; L_{\max}\|x - y\|. \tag{5}$$

The general idea is now as follows. Assume we are given a geometric graph on $X_1, \ldots, X_n \in \mathcal{X}$. In order to construct the paths we first map the points in the cube using h. Then we construct the paths on $h(X_1), \ldots, h(X_n) \in [0,1]^d$ as in the last section. Finally, we map the path back to $\mathcal{X}$.

Proposition 23 (Maximum average load for fixed graph on general domain) *Let G be a geometric graph based on $X_1, \ldots, X_n \in \mathcal{X}$. Assume that there exists some $\tilde{g} > 0$ such that points of distance smaller than $\tilde{g}$ are always connected in the graph. Consider a mapping $h : \mathcal{X} \to [0,1]^d$ as in Equation (5) and a grid of width g on $[0,1]^d$. Let $(C_i)_i$ be the cells of the g-grid on $[0,1]^d$, denote their centers by c_i. Let B_i and B_i' be balls in $\mathcal{X}$ with radius $r = g/(2L_{\max})$ and $R = \sqrt{d}\, g/L_{\min}$ centered at $h^{-1}(c_i)$.*

1. *These balls satisfy $B_i \subseteq h^{-1}(C_i) \subseteq B_i'$.*
2. *Denote by $\tilde{N}_{\min}$ the minimal number of points in B_i and $\tilde{N}_{\max}$ the maximal number of points in B_i'. Construct paths between the points $h(X_i) \in [0,1]^d$ as described in the previous subsection. If $\tilde{N}_{\min} \geq 1$ and $g \leq L_{\min}\tilde{g}/\sqrt{d+3}$, then these paths are valid.*
3. *In this case, the maximal average load can be upper bounded by*

$$1 + \left(\frac{\tilde{N}_{\max}^2}{\tilde{N}_{\min}^2} + 2\frac{\tilde{N}_{\max}}{\tilde{N}_{\min}} \right) \frac{d}{(\tilde{g}L_{\min}/\sqrt{d+3})^{d+1}}.$$

Proof. *Part (1).* To see the first inclusion, consider any point $x \in B_i$. By the Lipschitz continuity and the definition of B_i we get

$$\|h(x) - c_i\| \leq L_{max}\|x - h^{-1}(c_i)\| \leq L_{max} \cdot r = g/2.$$

Hence, $h(x) \in C_i$. To show the other inclusion let $x \in h^{-1}(C_i)$. Then by the Lipschitz continuity and the definition of C_i we get

$$\|x - h^{-1}(c_i)\| \leq \|h(x) - c_i\|/L_{\min} \leq \sqrt{d}g/L_{\min} = R,$$

hence $x \in B_i'$.
Part (2). By the definition of $\tilde{N}_{\min}$ and Part (1) it is clear that each cell of the grid contains at least one point. Consider two points $X_i, X_j \in \mathcal{X}$ such that $h(X_i)$ and $h(X_j)$ are in neighboring cells of the g-grid. Then $\|h(X_i) - h(X_j)\| \leq g\sqrt{d+3}$. By the properties of h,

$$\|h^{-1}(X_i) - h^{-1}(X_j)\| \leq \frac{1}{L_{\min}}\|X_i - X_j\| \leq \frac{1}{L_{\min}}\sqrt{d+3}\, g \;\leq\; \tilde{g}.$$

Thus, by the definition of $\tilde{g}$ the points X_i and X_j are connected in G.

Part (3). Follows directly from Proposition 22 and Parts (1) and (2). ∎

7.2.3 Spectral Gap for the ε-Graph

Now we are going to apply Proposition 23 to ε-graphs. We will use the general results on ε-graphs summarized in the appendix.

Proposition 24 (Maximal average load for ε-graph) *Assume that $\mathcal{X}$ is homeomorphic to the cube with a mapping h as described in Equation (5). Then there exist constants $c_1, c_2, c_3 > 0$ such that with probability at least $1 - c_1 \exp(-c_2 n\varepsilon^d)/\varepsilon^d$, the maximum average load is upper bounded by c_3/ε^{d+1}. If $n\varepsilon^d/\log n \to \infty$, then this probability tends to 1 as $n \to \infty$.*

Proof. The proof is based on Proposition 23. By construction we know that points with distance at most $\tilde{g} = \varepsilon$ are always connected in the ε-graph. By Part 2 of Proposition 23, to ensure that points in neighboring grid cells are always connected in the graph we thus need to choose the grid width $g = \varepsilon \cdot L_{\min}/\sqrt{d+3}$. The radius r defined in Proposition 23 is then given as

$$r = \frac{g}{2L_{\max}} = \varepsilon \cdot \frac{L_{\min}}{2\sqrt{d+3}L_{\max}}.$$

The probability mass of the balls B_i is thus bounded by

$$b_{\min} \;\geq\; r^d \eta_d p_{\min} \alpha \;=\; \varepsilon^d \;\cdot\; \left(\frac{L_{\min}}{L_{\max}}\right)^d \frac{\eta_d}{2^d (d+3)^{d/2}}\, p_{\min}\alpha \;\;=:\;\; \varepsilon^d \cdot c_{\min}$$

(recall that the constant α takes care of boundary points, see Definition 2 of the valid region).
We have

$$K = 1/g^d = \sqrt{d+3}/L_{\min}^d \;\cdot\; (1/\varepsilon^d) =: \kappa \;\cdot\; (1/\varepsilon^d)$$

grid cells and thus the same number of balls B_i. We can now apply Proposition 28 (with $\delta := 1/2$) to deduce the bound for the quantity $\tilde{N}_{\min}$ used in Proposition 23:

$$P\Big(\tilde{N}_{\min} \;\leq\; n\varepsilon^d c_{\min}/2\Big) \;\;\leq\;\; \frac{\kappa}{\varepsilon^d} \exp(-n\varepsilon^d c_{\min}/12).$$

Analogously, for $\tilde{N}_{\max}$ we have $R = \varepsilon > \varepsilon\sqrt{d}/\sqrt{d+3}$ and $b_{\max} = R^d \eta_d p_{\max} = \varepsilon^d \eta_d p_{\max} := \varepsilon^d \cdot c_{\max}$. With $\delta = 0.5$ we then obtain

$$P\Big(\tilde{N}_{\max} \geq n\varepsilon^d c_{\max} 3/2\Big) \leq \frac{\kappa}{\varepsilon^d} \exp(-n\varepsilon^d c_{\max}/12).$$

Plugging these values into Proposition 23 leads to the result. ∎

Proof of Theorem 6. With probability at least $1 - c_1 n \exp(-c_2 n\varepsilon^d)$, both the minimal and maximal degrees in the graph are of the order $\Theta(n\varepsilon^d)$ (cf. Proposition 29), and the volume of G is of order $\Theta(n^2\varepsilon^d)$. To compute the maximal number $|\gamma_{\max}|$ of edges in each of the paths constructed above, observe that each path can traverse at most $d \cdot 1/g = (d\sqrt{d+3}/L_{\min}) \cdot (1/\varepsilon)$ cubes, and a path contains just one edge per cube. Thus $|\gamma_{\max}|$ is of the order $\Theta(1/\varepsilon)$. In Proposition 24 we have seen that with probability at least $c_4 \exp(-c_5 n\varepsilon^d)/\varepsilon^d$ the maximum average load b is of the order $\Omega(1/\varepsilon^{d+1})$. Plugging all these quantities in Proposition 20 leads to the result. ∎

7.2.4 Spectral Gap for the kNN-Graph

As in the case of the flow proofs, the techniques in the case of the kNN-graphs are identical to the ones for the ε-graph, we just have to replace the deterministic radius ε by the minimal kNN-radius. As before we exploit that if two sample points have distance less than $R_{k,\min}$ from each other, then they are always connected both in the symmetric and mutual kNN-graph.

Proposition 25 (Maximal average load in the kNN-graph) *Under the general assumptions, with probability at least* $1-c_1 \cdot n \cdot \exp(-c_2 k)$ *the maximal average load in both the symmetric and mutual kNN-graph is bounded from above by* $c_3(n/k)^{1+1/d}$. *If* $k/\log n \to \infty$, *then this probability converges to 1.*

Proof. This proof is analogous to the one of Proposition 24, the role of ε is now taken over by $R_{k,\min}$. ■

Proof of Theorem 7. With probabilities at least $1-n\exp(-c_1 k)$ the following statements hold: the minimal and maximal degree are of order $\Theta(k)$, thus the number of edges in the graph is of order $\Theta(nk)$. Analogously to the proof for the ε-graph, the maximal path length $|\gamma_{\max}|$ is of the order $1/R_{k,\min} = \Theta((k/n)^{1/d})$. The maximal average load is of the order $O((n/k)^{d+1/d})$. Plugging all these quantities in Proposition 20 leads to the result. ■

7.3 Proofs of Corollaries 8 and 9

Now we collected all ingredients to finally present the following proofs.

Proof of Corollary 8. This is a direct consequence of the results on the minimal degree (Proposition 29) and the spectral gap (Theorem 6). Plugging these results into Proposition 5 leads to the first result. The last statement in the theorem follows by a standard density estimation argument, as the degree of a vertex in the ε-graph is a consistent density estimator (see Proposition 29). ■

Proof of Corollary 9. Follows similarly as Theorem 8 by applying Proposition 5. The results on the minimal degree and the spectral gap can be found in Proposition 30 and Theorem 7. The last statement follows from the convergence of the degrees, see Proposition 30. ■

7.4 Weighted Graphs

For weighted graphs, we use the following results from the literature.

Proposition 26 (Spectral gap in weighted graphs) *1. For any row-stochastic matrix* P,

$$\lambda_2 \;\le\; \frac{1}{2}\max_{i,j}\sum_{k=1}^{n}\Big|\frac{w_{ik}}{d_i}-\frac{w_{jk}}{d_j}\Big| \;\le\; 1-n\min_{i,j}\frac{w_{ij}}{d_i} \;\le\; 1-\frac{w_{\min}}{w_{\max}}.$$

2. Consider a weighted graph G *with edge weights* $0 < w_{\min} \le w_{ij} \le w_{\max}$ *and denote its second eigenvalue by* $\lambda_{2,weighted}$. *Consider the corresponding unweighted graph where*

all edge weights are replaced by 1*, and denote its second eigenvalue by* $\lambda_{2,unweighted}$*. Then we have*

$$(1-\lambda_{2,unweighted})\cdot\frac{w_{\min}}{w_{\max}} \;\leq\; (1-\lambda_{2,weighted}) \;\leq\; (1-\lambda_{2,unweighted})\cdot\frac{w_{\max}}{w_{\min}}.$$

Proof.

1. This bound was obtained by Zenger (1972), see also Section 2.5 of Seneta (2006) for a discussion. Note that the second inequality is far from being tight. But in our application, both bounds lead to similar results.

2. This statement follows directly from the well-known representation of the second eigenvalue μ_2 of the normalized graph Laplacian L_{sym} (see Sec. 1.2 in Chung, 1997),

$$\mu_2 = \inf_{f\in\mathbb{R}^n} \frac{\sum_{i,j=1}^n w_{ij}(f_i-f_j)^2}{\min_{c\in\mathbb{R}}\sum_{i=1}^n d_i(f_i-c)^2}.$$

   Note that the eigenvalue μ_2 of the normalized Laplacian and the eigenvalue λ_2 of the random walk matrix P are in relation $1-\lambda_2=\mu_2$. ■

Proof of Theorem 10. Follows directly from plugging in the first part of Proposition 26 in Proposition 5. ■

Proof of Theorem 11. We split

$$\left|nR_{ij}-\frac{1}{p(X_i)}-\frac{1}{p(X_j)}\right| \leq \left|nR_{ij}-\frac{n}{d_i}-\frac{n}{d_j}\right| + \left|\frac{n}{d_i}+\frac{n}{d_j}-\frac{1}{p(X_i)}-\frac{1}{p(X_j)}\right|.$$

Under the given assumption, the second term on the right hand side converges to 0 a.s. by a standard kernel density estimation argument (e.g., Section 9 of Devroye and Lugosi, 2001). The main work is the first term on the right hand side. We treat upper and lower bounds of $R_{ij}-1/d_i-1/d_j$ separately.

To get a lower bound, recall that by Proposition 15 we have

$$R_{ij} \geq \frac{Q_{ij}}{1+w_{ij}Q_{ij}},$$

where $Q_{ij}=1/(d_i-w_{ij})+1/(d_j-w_{ij})$ and w_{ij} is the weight of the edge between i and j. Observe that under the given conditions, for any two fixed points X_i, X_j the Gaussian edge weight w_{ij} converges to 0 (as $\sigma\to 0$). Thus

$$n\left(R_{ij}-\frac{1}{d_i}-\frac{1}{d_j}\right) \geq n\left(\frac{Q_{ij}}{1+w_{ij}Q_{ij}}-\frac{1}{d_i}-\frac{1}{d_j}\right) \to 0 \;\; a.s.$$

To treat the upper bound, we define the ε-truncated Gauss graph G^ε as the graph with edge weights

$$w_{ij}^\varepsilon := \begin{cases} w_{ij} & \text{if } \|X_i-X_j\|\leq\varepsilon, \\ 0 & \text{else.} \end{cases}$$

Let $d_i^\varepsilon = \sum_{j=1}^n w_{ij}^\varepsilon$. Because of $w_{ij}^\varepsilon \le w_{ij}$ and Rayleigh's principle, we have $R_{ij} \le R_{ij}^\varepsilon$, where R^ε denotes the resistance of the ε-truncated Gauss graph. Obviously,

$$\begin{aligned} nR_{ij} - \left(\frac{n}{d_i} + \frac{n}{d_j}\right) &\le \left| nR_{ij}^\varepsilon - \left(\frac{n}{d_i} + \frac{n}{d_j}\right)\right| \\ &\le \underbrace{\left| nR_{ij}^\varepsilon - \left(\frac{n}{d_i^\varepsilon} + \frac{n}{d_j^\varepsilon}\right)\right|}_{(*)} + \underbrace{\left|\left(\frac{n}{d_i^\varepsilon} + \frac{n}{d_j^\varepsilon}\right) - \left(\frac{n}{d_i} + \frac{n}{d_j}\right)\right|}_{(**)}. \end{aligned}$$

To bound term $(**)$ we show that the degrees in the truncated graph converge to the ones in the non-truncated graph. To see this, note that

$$\begin{aligned} \mathbb{E}\Big(\frac{d_i^\varepsilon}{n} \;\Big|\; X_i\Big) &= \frac{1}{(2\pi)^{\frac{d}{2}}}\frac{1}{\sigma^d}\int_{B(X_i,\varepsilon)} e^{\frac{-\|X_i-y\|^2}{2\sigma^2}}\, p(y)\, dy \\ &= \frac{1}{(2\pi)^{\frac{d}{2}}}\int_{B(0,\frac{\varepsilon}{\sigma})} e^{-\frac{\|z\|^2}{2}}\, p(X_i+\sigma z)\, dz \\ &= \mathbb{E}\Big(\frac{d_i}{n} \;\Big|\; X_i\Big) - \frac{1}{(2\pi)^{\frac{d}{2}}}\int_{\mathbb{R}^d\setminus B(0,\frac{\varepsilon}{\sigma})} e^{-\frac{\|z\|^2}{2}}\, p(X_i+\sigma z)\, dz. \end{aligned}$$

Exploiting that

$$\begin{aligned} \frac{1}{(2\pi)^{\frac{d}{2}}}\int_{\mathbb{R}^d\setminus B(0,\frac{\varepsilon}{\sigma})} e^{-\frac{\|z\|^2}{2}} &\le \frac{1}{(2\pi)^{\frac{d}{2}}} e^{-\frac{\varepsilon^2}{4\sigma^2}}\int_{\mathbb{R}^d} e^{-\frac{\|z\|^2}{4}} \\ &\le 2^{\frac{d}{2}} e^{-\frac{\varepsilon^2}{4\sigma^2}} = 2^{\frac{d}{2}}\frac{1}{(n\varepsilon^{d+2})^{\frac{1}{4}}} \end{aligned}$$

we obtain the convergence of the expectations: under the assumptions on n and σ from the theorem,

$$\Big|\, \mathbb{E}\Big(\frac{d_i^\varepsilon}{n} \;\Big|\; X_i\Big) - \mathbb{E}\Big(\frac{d_i}{n} \;\Big|\; X_i\Big)\,\Big| \;\to\; 0.$$

Now, a probabilistic bound for term $(**)$ can be obtained by standard concentration arguments.

We now bound term $(*)$. Up to now, our argument holds for arbitrary ε. For this last step, we now require that ε satisfies $\sigma^2 = \omega(\varepsilon^2/\log(n\varepsilon^{d+2}))$. Note that for this choice of ε, the truncated Gaussian graph "converges" to the non-truncated graph, as we truncate less and less weight.

Denote by $\lambda^{\varepsilon,\text{weighted}}$ the eigenvalues of the ε-truncated Gauss graph, and by $w_{\min}^\varepsilon$, $w_{\max}^\varepsilon$ its minimal and maximal edge weights. Also consider the graph G'' that is the unweighted version of the ε-truncated Gauss graph G^ε . Note that G'' coincides with the standard ε-graph. We denote its eigenvalues by $\lambda^{\varepsilon,\text{unweighted}}$. By applying Proposition 5, Corollary 8

and Proposition 26 we get

$$\left| nR_{ij}^{\varepsilon} - \left(\frac{n}{d_i^{\varepsilon}} + \frac{n}{d_j^{\varepsilon}} \right) \right| \leq \frac{w_{\max}^{\varepsilon}}{d_{\min}^{\varepsilon}} \left(\frac{1}{1 - \lambda_2^{\varepsilon,\text{weighted}}} + 2 \right) \left(\frac{n}{d_i^{\varepsilon}} + \frac{n}{d_j^{\varepsilon}} \right)$$
$$\leq \frac{w_{\max}^{\varepsilon}}{d_{\min}^{\varepsilon}} \left(\frac{w_{\max}^{\varepsilon}}{w_{\min}^{\varepsilon}} \frac{1}{1 - \lambda_2^{\varepsilon,\text{unweighted}}} + 2 \right) \left(\frac{n}{d_i^{\varepsilon}} + \frac{n}{d_j^{\varepsilon}} \right) \tag{6}$$

where the first inequality holds with probability at least $1 - c_1 n \exp(-c_2 n \sigma^d) - c_3 \exp(-c_4 n \sigma^d)/\sigma^d$. By $(**)$ we already know that the last factor in (6) converges to a constant:

$$\left(\frac{n}{d_i^{\varepsilon}} + \frac{n}{d_j^{\varepsilon}} \right) \to 1/p(X_i) + 1/p(X_j).$$

For the other factors of Term (6) we use the following quantities:

$$w_{\min}^{\varepsilon} \geq \frac{1}{\sigma^d} \exp(-\frac{\varepsilon^2}{2\sigma^2})$$
$$w_{\max}^{\varepsilon} \leq \frac{1}{\sigma^d}$$
$$d_{\min}^{\varepsilon} \geq n\varepsilon^d w_{\min}^{\varepsilon}$$
$$1 - \lambda_2^{\varepsilon,\text{unweighted}} \geq \varepsilon^2$$

Plugging these quantities in (6) and exploiting $\sigma^2 = \omega(\varepsilon^2/\log(n\varepsilon^{d+2}))$ leads to the desired convergence of $(*)$ to 0. ■

Proof of Corollary 13. We use the result from Theorem 4 in Chung and Radcliffe (2011) which states that under the assumption that the minimal expected degree $\overline{d_{\min}}$ satisfies $\overline{d_{\min}}/\log(n) \to \infty$, then with probability at least $1 - 1/n$ the spectral gap is bounded by a term of the order $O(\log(2n)/\overline{d_{\min}})$. Plugging this in Part (2) of Proposition 5 shows that with high probability,

$$\frac{\left| \frac{1}{\text{vol}(G)} C_{ij} - \frac{1}{d_i} - \frac{1}{d_j} \right|}{\frac{1}{d_i} + \frac{1}{d_j}} s \leq \left(\frac{\overline{d_{\min}}}{\log(2n)} + 2 \right) \frac{1}{d_{\min}} = O\left(\frac{1}{\log(2n)} \right).$$

■

Proof of Corollary 14. The expected degree of each vertex is $n(p_{\text{within}} + p_{\text{between}})/2$, the expected volume of the graph is $n^2(p_{\text{within}} + p_{\text{between}})/2$. The matrix $\overline{A}$ has the form $\begin{pmatrix} pJ & qJ \\ qJ & pJ \end{pmatrix}$ where J is the $(n/2 \times n/2)$-matrix of all ones and $p = p_{\text{within}}$ and $q = p_{\text{between}}$. The expected degree of all vertices is $n(p+q)/2$. Hence, $\overline{D^{-1/2}}\overline{A}\overline{D^{-1/2}} = \frac{2}{n(p+q)} \cdot \overline{A}$. This matrix has rank 2, its largest eigenvalue is 1 (with eigenvector the constant 1 vector), the other eigenvalue is $(p-q)/(p+q)$ with eigenvector $(1, ..., 1, -1, ..., -1)$. Hence, the spectral gap in this model is $2q/(p+q)$.

Under the assumption that $p = \omega(\log(n)/n)$, the deviations in Theorem 12 converge to 0. Plugging the spectral gap in our bound in Proposition 5 shows that with high probability,

$$\frac{n(p_{\text{within}} + p_{\text{between}})}{2} \cdot \left| \frac{1}{\text{vol}(G)} H_{ij} - \frac{1}{d_i} \right| \leq \frac{4}{np_{\text{between}}} + \frac{4}{n(p_{\text{within}} + p_{\text{between}})} = O\left(\frac{1}{np_{\text{between}}}\right).$$

■

8. Discussion

We have presented different strategies to prove that in many large graphs the commute distance can be approximated by $1/d_i + 1/d_j$. Both our approaches tell a similar story. Our result holds as soon as there are "enough disjoint paths" between i and j, compared to the size of the graph, and the minimal degree is "large enough" compared to n. Most relevant for machine learning, our results hold for all kinds of random geometric graphs (ε-graphs, kNN graphs, Gaussian similarity graphs). Here, the limit distance function $dist(i, j) = 1/d_i + 1/d_j$ is meaningless: It just considers the local density (the degree) at the two vertices, but does not take into account any global property such as the cluster structure of the graph. As the speed of convergence is very fast (for example, of the order $1/n$ in the case of Gaussian similarity graphs), the use of the raw commute distance should be discouraged even on moderate sized graphs.

An important point to note is that the results on the degeneracy of the hitting and commute times are not due to pathologies such as a "misconstruction" of the graphs or "wrong scaling constants". For example, in the random geometric graph setting the graph Laplacian can be proved to converge to the Laplace-Beltrami operator on the underlying space under similar assumptions as the ones above (Hein et al., 2007). But even though the Laplacian itself converges to a meaningful limit, the resistance distance, which is computed based on point evaluations of the inverse of this Laplacian, does not converge to a useful limit.

There are two important classes of graphs that are not covered in our approach. In power law graphs as well as in grid-like graphs, the minimal degree is constant, thus our results do not lead to tight bounds. The resistance distances on grid-like graphs has been studied in some particular cases. For example, Cserti (2000) and Wu (2004) prove explicit formulas for the resistance on regular one-and two-dimensional grids, and Benjamini and Rossignol (2008) characterize the variance of the resistance on random Bernoulli grids.

Beyond the commute distance, there exists a large variety of distance functions on graphs, and it is an interesting question to study their convergence behavior as $n \to \infty$. In particular, several authors constructed parametric families of distances that include both the shortest path distance and commute distance as special cases (e.g., Yen et al., 2008, Chebotarev, 2011, Alamgir and von Luxburg, 2011). For such families, the limit behavior is particularly interesting. On the one hand, it is well-known that shortest path distances

on random geometric graphs converge to the Euclidean or geodesic distances in the underlying space (Bernstein et al., 2000). On the other hand, as we have just seen in this paper, the commute distance converges to a meaningless distance function. Hence it is an interesting problem to characterize those distances that are degenerate. For the family of p-resistance distances this question has been solved in Alamgir and von Luxburg (2011). If $p > p^*$ for some critical threshold p^*, we are in a similar regime as the commute distance and the p-resistance is degenerate. For $p < p^*$ the commute distance is in a similar regime as the shortest path distance and is not degenerate. We believe that a solution to this question would be particularly interesting for the family of logarithmic forest distances by Chebotarev (2011). This family has many nice properties and, in our experience, tends to perform nicely in practice. So what are the parameters α for which we can guarantee that the distance is not degenerate as $n \to \infty$?

Acknowledgements

We would like to thank Bernhard Schölkopf for helpful discussions. Moreover, we would like to thank one of the anonymous reviewers encountered in an earlier submission for his constructive and detailed comments. Ulrike von Luxburg is grateful for funding of the German Research Foundation (grant LU1718/1-1 and Research Unit 1735 "Structural Inference in Statistics: Adaptation and Efficiency").

Appendix A. General Properties of Random Geometric Graphs

In this appendix we collect some basic results on random geometric graphs. These results are well-known, cf. Penrose (2003), but we did not find any reference where the material is presented in the way we need it (often the results are used implicitly or are tailored towards particular applications).

In the following, assume that $\mathcal{X} := \mathrm{supp}(p)$ is a valid region according to Definition 1. Recall the definition of the boundary constant α in the valid region.

A convenient tool for dealing with random geometric graphs is the following well-known concentration inequality for binomial random variables with small p (originally found by Chernoff, 1952 and Hoeffding, 1963, we use the version as published in Angluin and Valiant, 1977).

Proposition 27 (Concentration inequalities) *Let N be a $Bin(n,p)$-distributed random variable. Then, for all $\delta \in]0,1]$,*

$$\begin{aligned} P\Big(N \;\leq\; (1-\delta)np\Big) \;&\leq\; \exp(-\frac{1}{3}\delta^2 np) \\ P\Big(N \;\geq\; (1+\delta)np\Big) \;&\leq\; \exp(-\frac{1}{3}\delta^2 np). \end{aligned}$$

We will see below that computing expected, minimum and maximum degrees in random geometric graphs always boils down to counting the number of data points in certain balls

in the space. The following proposition is a straightforward application of the concentration inequality above and serves as "template" for all later proofs.

Proposition 28 (Counting sample points) *Consider a sample $X_1, \ldots, X_n$ drawn i.i.d. according to density p on $\mathcal{X}$. Let $B_1, \ldots, B_K$ be a fixed collection of subsets of $\mathcal{X}$ (the B_i do not need to be disjoint). Denote by $b_{\min} := \min_{i=1,\ldots,K} \int_{B_i} p(x)dx$ the minimal probability mass of the sets B_i (similarly by $b_{\max}$ the maximal probability mass), and by $N_{\min}$ and $N_{\max}$ the minimal (resp. maximal) number of sample points in the sets B_i. Then for all $\delta \in]0,1]$*

$$P\Big(N_{\max} \geq (1+\delta)nb_{\max}\Big) \;\leq\; K \cdot \exp(-\delta^2 nb_{\max}/3)$$
$$P\Big(N_{\min} \leq (1-\delta)nb_{\min}\Big) \;\leq\; K \cdot \exp(-\delta^2 nb_{\min}/3).$$

Proof. This is a straightforward application of Proposition 27 using the union bound. ■

When working with ε-graphs or kNN-graphs, we often need to know the degrees of the vertices. As a rule of thumb, the expected degree of a vertex in the ε-graph is of the order $\Theta(n\varepsilon^d)$, the expected degree of a vertex in both the symmetric and mutual kNN-graph is of the order $\Theta(k)$. The expected kNN-distance is of the order $\Theta((k/n)^{1/d})$. Provided the graph is "sufficiently connected" , all these rules of thumb also apply to the minimal and maximal values of these quantities. The following propositions make these rules of thumb explicit.

Proposition 29 (Degrees in the ε-graph) *Consider an ε-graph on a valid region $\mathcal{X} \subset \mathbb{R}^d$.*

1. *Then, for all $\delta \in]0,1]$, the minimal and maximal degrees in the ε-graph satisfy*

$$P\Big(d_{\max} \geq (1+\delta)n\varepsilon^d p_{\max}\eta_d\Big) \;\leq\; n \cdot \exp(-\delta^2 n\varepsilon^d p_{\max}\eta_d/3)$$
$$P\Big(d_{\min} \leq (1-\delta)n\varepsilon^d p_{\min}\eta_d\alpha\Big) \;\leq\; n \cdot \exp(-\delta^2 n\varepsilon^d p_{\min}\eta_d\alpha/3).$$

   In particular, if $n\varepsilon^d/\log n \to \infty$, then these probabilities converge to 0 as $n \to \infty$.

2. *If $n \to \infty, \varepsilon \to 0$ and $n\varepsilon^d/\log n \to \infty$, and the density p is continuous, then for each interior point $X_i \in \mathcal{X}$ the degree is a consistent density estimate: $d_i/(n\varepsilon^d\eta_d) \longrightarrow p(X_i)$ a.s.*

Proof. *Part 1* follows by applying Proposition 28 to the balls of radius ε centered at the data points. Note that for the bound on $d_{\min}$, we need to take into account boundary effects as only a part of the ε-ball around a boundary point is contained in $\mathcal{X}$. This is where the constant α comes in (recall the definition of α from the definition of a valid region). *Part 2* is a standard density estimation argument: the expected degree of X_i is the expected number of points in the ε-ball around X_i. For ε small enough, the ε-ball around X_i is completely contained in $\mathcal{X}$ and the density is approximately constant on this ball because we assumed the density to be continuous. The expected number of points is approximately $n\varepsilon^d\eta_d p(X_i)$ where η_d denotes the volume of a d-dimensional unit ball. The

result now follows from Part 1. ■

Recall the definitions of the k-nearest neighbor radii: $R_k(x)$ denotes the distance of x to its k-nearest neighbor among the X_i, and the maximum and minimum values are denoted $R_{k,\max} := \max_{i=1,\ldots,n} R_k(X_i)$ and $R_{k,\min} := \max_{i=1,\ldots,n} R_k(X_i)$. Also recall the definition of the boundary constant α from the definition of a valid region.

Proposition 30 (Degrees in the kNN-graph) *Consider a valid region $\mathcal{X} \subset \mathbb{R}^d$.*

1. *Define the constants $a = 1/(2p_{\max}\eta_d)^{1/d}$ and $\tilde{a} := 2/(p_{\min}\eta_d\alpha)^{1/d}$. Then*

$$\begin{aligned} P\Big(R_{k,\min} \leq a\left(\frac{k}{n}\right)^{1/d}\Big) &\leq n\exp(-k/3) \\ P\Big(R_{k,\max} \geq \tilde{a}\left(\frac{k}{n}\right)^{1/d}\Big) &\leq n\exp(-k/12). \end{aligned}$$

   If $n \to \infty$ and $k/\log n \to \infty$, then these probabilities converge to 0.

2. *Moreover, with probability at least $1 - n\exp(-c_4 k)$ the minimal and maximal degree in both the symmetric and mutual kNN-graph are of the order $\Theta(k)$ (the constants differ).*

3. *If the density is continuous, $n \to \infty$, $k/\log n \to \infty$ and additionally $k/n \to 0$, then in both the symmetric and the mutual kNN-graph, the degree of any fixed vertex v_i in the interior of $\mathcal{X}$ satisfies $k/d_i \to 1$ a.s.*

Proof. *Part 1.* Define the constant $a = 1/(2p_{\max}\eta_d)^{1/d}$ and the radius $r := a\,(k/n)^{1/d}$, fix a sample point x, and denote by $\mu(x)$ the probability mass of the ball around x with radius r. Set $\mu_{\max} := r^d\eta_d p_{\max} \geq \max_{x\in\mathcal{X}} \mu(x)$. Note that $\mu_{\max} < 1$. Observe that $R_k(x) \leq r$ if and only if there are at least k data points in the ball of radius r around x. Let $M \sim Bin(n, \mu)$ and $V \sim Bin(n, \mu_{\max})$. Note that by the choices of a and r we have $E(V) = k/2$. All this leads to

$$P\Big(R_k(x) \leq r\Big) \;\leq\; P\Big(M \geq k\Big) \;\leq\; P\Big(V \geq k\Big) \;=\; P\Big(V \geq 2E(V)\Big).$$

Applying the concentration inequality of Proposition 27 (with $\delta := 1$)) and using a union bound leads to the following result for the minimal kNN-radius:

$$\begin{aligned} P\Big(R_{k,\min} \leq a\left(\frac{k}{n}\right)^{1/d}\Big) &\leq P\Big(\exists i \;:\; R_k(X_i) \leq a\left(\frac{k}{n}\right)^{1/d}\Big) \\ &\leq n \max_{i=1,\ldots,n} \Big(R_k(X_i) \leq r\Big) \\ &\leq n\exp(-k/3). \end{aligned}$$

By a similar approach we can prove the analogous statement for the maximal kNN-radius. Note that for the bound on $R_{k,\max}$ we additionally need to take into account boundary

effects: at the boundary of $\mathcal{X}$, only a part of the ball around a point is contained in $\mathcal{X}$, which affects the value of $\mu_{\min}$. We thus define $\tilde{a} := 2/(p_{\min}\eta_d\alpha)^{1/d}$, $r := \tilde{a}(k/n)^{1/d}$, $\mu_{\min} := r^d\eta_d p_{\min}\alpha$ where $\alpha \in]0,1]$ is the constant defined in the valid region. With $V = Bin(n, \mu_{\min})$ with $EV = 2k$ we continue similarly to above and get (using $\delta = 1/2$)

$$P\Big(R_{k,\max} \geq \tilde{a}\left(\frac{k}{n}\right)^{1/d}\Big) \quad \leq \quad n\exp(-k/12).$$

Part 2. In the directed kNN-graph, the degree of each vertex is exactly k. Thus, in the mutual kNN-graph, the maximum degree over all vertices is upper bounded by k, in the symmetric kNN-graph the minimum degree over all vertices is lower bounded by k.
For the symmetric graph, observe that the maximal degree in the graph is bounded by the maximal number of points in the balls of radius $R_{k,\max}$ centered at the data points. We know that with high probability, a ball of radius $R_{k,\max}$ contains of the order $\Theta(nR_{k,\max}^d)$ points. Using Part 1 we know that with high probability, $R_{k,\max}$ is of the order $(k/n)^{1/d}$. Thus the maximal degree in the symmetric kNN-graph is of the order $\Theta(k)$, with high probability.
In the mutual graph, observe that the minimal degree in the graph is bounded by the minimal number of points in the balls of radius $R_{k,\min}$ centered at the data points. Then the statement follows analogously to the last one.
Part 3, proof sketch. Consider a fixed point x in the interior of $\mathcal{X}$. We know that both in the symmetric and mutual kNN-graph, two points cannot be connected if their distance is larger than $R_{k,\max}$. As we know that $R_{k,\max}$ is of the order $(k/n)^{1/d}$, under the growth conditions on n and k this radius becomes arbitrarily small. Thus, because of the continuity of the density, if n is large enough we can assume that the density in the ball $B(x, R_{k,\max})$ of radius $R_{k,\max}$ around x is approximately constant. Thus, all points $y \in B(x, R_{k,\max})$ have approximately the same k-nearest neighbor radius $R := (k/(n \cdot p(x)\eta_d))^{1/d}$. Moreover, by concentration arguments it is easy to see that the actual kNN-radii only deviate by a factor $1 \pm \delta$ from their expected values.
Then, with high probability, all points inside of $B(x, R(1-\delta))$ are among the k nearest neighbors of x, and all k nearest neighbors of x are inside $B(x, R(1+\delta))$. On the other hand, with high probability x is among the k nearest neighbors of all points $y \in B(x, R(1-\delta))$, and not among the k nearest neighbors of any point outside of $B(x, R(1+\delta))$. Hence, in the mutual kNN-graph, with high probability x is connected exactly to all points $y \in B(x, R(1-\delta))$. In the symmetric kNN-graph, x might additionally be connected to the points in $B(x, R(1+\delta)) \setminus B(x, R(1-\delta))$. By construction, with high probability the number of sample points in these balls is $(1+\delta)k$ and $(1-\delta)k$. Driving δ to 0 leads to the result. ■

References

M. Alamgir and U. von Luxburg. Phase transition in the familiy of p-resistances. In *Neural Information Processing Systems (NIPS)*, 2011.

D. Aldous and J. Fill. Reversible markov chains and random walks on graphs. *Preprint, available at http://www.stat.berkeley.edu/users/aldous/RWG/book.html*, 2001.

R. Aleliunas, R. Karp, R. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *20th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 218–223, 1979.

D. Angluin and L.G. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. In *Symposium on the Theory of Computing (STOC)*, 1977.

C. Avin and G. Ercal. On the cover time and mixing time of random geometric graphs. *Theor. Comput. Sci*, 380(1-2):2–22, 2007.

P. Balister, B. Bollobás, A. Sarkar, and M. Walters. Connectivity of random k-nearest-neighbour graphs. *Advances in Applied Probability*, 37(1):1–24, 2005.

A. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439): 509–512, 1999.

I. Benjamini and R. Rossignol. Submean variance bound for effective resistance of random electric networks. *Communications in Mathematical Physics*, 280(2):445–462, 2008.

M. Bernstein, V. De Silva, J. Langford, and J. Tenenbaum. Graph approximations to geodesics on embedded manifolds. Technical report, Department of Psychology, Stanford University, 2000.

B. Bollobas. *Modern Graph Theory*. Springer, 1998.

S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Mixing times for random walks on geometric random graphs. In C. Demetrescu, R. Sedgewick, and R.Tamassia, editors, *Proceedings of the Second Workshop on Analytic Algorithmics and Combinatorics, ANALCO*, pages 240–249, 2005.

M. Brand. A random walks perspective on maximizing satisfaction and profit. In *SIAM International Conference on Data Mining, April 21-23 (SDM)*, 2005.

M. Brito, E. Chavez, A. Quiroz, and J. Yukich. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Statistics and Probability Letters*, 35: 33 – 42, 1997.

N. Cesa-Bianchi, C. Gentile, and F. Vitale. Fast and optimal prediction on a labeled tree. In *Conference on Learning Theory (COLT)*, 2009.

A. Chandra, P. Raghavan, W. Ruzzo, R. Smolensky, and P. Tiwari. The electrical resistance of a graph captures its commute and cover times. In *Proceedings of the 21st Annual Symposium on Theory of Computing (STOC)*, pages 574–586, 1989.

P. Chebotarev. A class of graph-geodetic distances generalizing the shortest path and the resistance distances. *Discrete Applied Mathematics*, 159(295 – 302), 2011.

H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

F. Chung. *Spectral graph theory*, volume 92 of the CBMS Regional Conference Series in Mathematics. Conference Board of the Mathematical Sciences, Washington, 1997.

F. Chung and L. Lu. *Complex Graphs and Networks*. American Mathematical Society, Providence, 2006.

F. Chung and M. Radcliffe. On the spectra of general random graphs. *Electronic Journal of Combinatorics*, 18(P215):1, 2011.

C. Cooper and A. Frieze. The cover time of sparse random graphs. In *Proceedings of the fourteenth Annual Symposium on Discrete Algorithms (SODA)*, pages 140–147, 2003.

C. Cooper and A. Frieze. The cover time of random regular graphs. *SIAM Journal on Discrete Mathematics*, 18(4):728–740, 2005.

C. Cooper and A. Frieze. The cover time of the preferential attachment graph. *J. Comb. Theory, Ser. B*, 97(2):269–290, 2007.

C. Cooper and A. Frieze. The cover time of random geometric graphs. *Random Structures & Algorithms*, 38(3):324–349, 2011.

J. Cserti. Application of the lattice Green's function for calculating the resistance of an infinite network of resistors. *American Journal of Physics*, 68:896, 2000.

L. Devroye and G. Lugosi. *Combinatorial Methods in Density Estimation*. Springer, New York, 2001.

P. Diaconis and L. Saloff-Coste. Comparison theorems for reversible Markov chains. *The Annals of Applied Probability*, 3(3):696–730, 1993.

P. Diaconis and D. Stroock. Geometric bounds for eigenvalues of Markov chains. *The Annals of Applied Probability*, pages 36–61, 1991.

P. G. Doyle and J. L. Snell. *Random walks and electric networks*, volume 22 of *Carus Mathematical Monographs*. Mathematical Association of America, Washington, DC, 1984.

F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. A novel way of computing dissimilarities between nodes of a graph, with application to collaborative filtering and subspace projection of the graph nodes. Technical Report IAG WP 06/08, Université catholique de Louvain, 2006.

P.W. Fowler. Resistance distances in fullerene graphs. *Croatica Chemica Acta*, 75(2):401–408, 2002.

S. Guattery. Graph embeddings, symmetric real matrices, and generalized inverses. Technical report, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, 1998.

G. Guillot, R. Leblois, A. Coulon, and A.C. Frantz. Statistical methods in spatial genetics. *Molecular Ecology*, 18(23):4734–4756, 2009.

J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *International Conference on Machine Learning (ICML)*, pages 369–376. ACM, 2004.

M. Hein, J.-Y. Audibert, and U. von Luxburg. Graph Laplacians and their convergence on random neighborhood graphs. *Journal of Machine Learning Research*, 8:1325 – 1370, 2007.

M. Herbster and M. Pontil. Prediction on a graph with a perceptron. In Bernhard Schölkopf, John C. Platt, and Thomas Hoffman, editors, *Neural Information Processing Systems (NIPS)*, pages 577–584, 2006.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.*, 58:13 – 30, 1963.

O. Ivanciuc. QSAR and QSPR molecular descriptors computed from the resistance distance and electrical conductance matrices. *ACH Models in Chemistry*, 137(5/6):607–632, 2000.

M. Jerrum and A. Sinclair. Conductance and the rapid mixing property for markov chains: the approximation of permanent resolved. In *Proceedings of the twentieth annual ACM symposium on Theory of computing (STOC)*, pages 235–244, 1988.

D. Klein and M. Randic. Resistance distance. *Journal of Mathematical Chemistry*, 12: 81 – 95, 1993.

D. Levin, Y. Peres, and E. Wilmer. *Markov Chains and Mixing Times*. Americal Mathematical Society, 2008.

D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *International Conference on Information and Knowledge Management (CIKM)*, pages 556–559, 2003.

L. Lovász. Random walks on graphs: a survey. In *Combinatorics, Paul Erdös is eighty*, Bolyai Soc. Math. Stud., pages 353 – 397. János Bolyai Math. Soc., Budapest, 1993.

R. Lyons and Y. Peres. Probability on trees and networks. *Book in preparation, available online on the webpage of Yuval Peres*, 2010.

M. Penrose. The longest edge of the random minimal spanning tree. *The Annals of Applied Probability*, pages 340–361, 1997.

M. Penrose. A strong law for the longest edge of the minimal spanning tree. *Ann. of Prob.*, 27(1):246 – 260, 1999.

M. Penrose. *Random Geometric Graphs*. Oxford University Press, 2003.

H. Qiu and E. Hancock. Graph embedding using commute time. *Structural, Syntactic, and Statistical Pattern Recognition*, pages 441–449, 2006.

H. Qiu and E.R. Hancock. Image segmentation using commute times. In *Proceedings of the 16th British Machine Vision Conference (BMVC)*, pages 929–938, 2005.

K. Roy. Topological descriptors in drug design and modeling studies. *Molecular Diversity*, 8(4):321–323, 2004. ISSN 1381-1991.

M. Saerens, F. Fouss, L. Yen, and P. Dupont. The principal components analysis of a graph, and its relationships to spectral clustering. In *Proceedings of the 15th European Conference on Machine Learning (ECML)*, pages 371 – 383, 2004.

P. Sarkar and A. Moore. A tractable approach to finding closest truncated-commute-time neighbors in large graphs. In *23rd Conference on Uncertainty in Artificial Intelligence(UAI)*, 2007.

P. Sarkar, A. Moore, and A. Prakash. Fast incremental proximity search in large graphs. In W. Cohen, A. McCallum, and S. Roweis, editors, *Proceedings of the Twenty-Fifth International Conference of Machine Learning (ICML)*, pages 896–903, 2008.

E. Seneta. *Non-negative Matrices and Markov Chains*. Springer, 2006.

A. Sinclair. Improved bounds for mixing rates of Markov chains and multicommodity flow. *Combinatorics, Probability and Computing*, 1(04):351–370, 1992.

D. Spielman and N. Srivastava. Graph sparsification by effective resistances. In R. Ladner and C. Dwork, editors, *Proceedings of the 40th Annual Symposium on Theory of Computing (STOC)*, pages 563–568, 2008.

P. Tetali. Random walks and the effective resistance of networks. *Journal of Theoretical Probability*, 4(1):101–109, 1991.

U. von Luxburg, A. Radl, and M. Hein. Getting lost in space: Large sample analysis of the commute distance. In *Neural Information Processing Systems (NIPS)*, 2010.

D. Wittmann, D. Schmidl, F. Blöchl, and F. Theis. Reconstruction of graphs based on random walks. *Theoretical Computer Science*, 410(38-40):3826–3838, 2009.

F. Y. Wu. Theory of resistor networks: The two-point resistance. *Journal of Physics A: Mathematical and General*, 37:6653–6673, 2004.

W. Xiao and I. Gutman. Resistance distance and Laplacian spectrum. *Theoretical Chemistry Accounts*, 110:284 – 298, 2003.

F. Xue and P. Kumar. The number of neighbors needed for connectivity of wireless networks. *Wireless networks*, 10(2):169–181, 2004.

L. Yen, D. Vanvyve, F. Wouters, F. Fouss, M. Verleysen, and M. Saerens. Clustering using a random walk based distance measure. In *Proceedings of the 13th Symposium on Artificial Neural Networks (ESANN)*, pages 317–324, 2005.

L. Yen, M. Saerens, A. Mantrach, and M. Shimbo. A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–793, 2008.

C. Zenger. A comparison of some bounds for the nontrivial eigenvalues of stochastic matrices. *Numer. Math.*, 9:209 – 22, 1972.

D. Zhou and B. Schölkopf. Learning from labeled and unlabeled data using random walks. In *Pattern Recognition, Proceedings of the 26th DAGM Symposium*, pages 237 – 244, 2004.

Journal of Machine Learning Research 15 (2014) 1799-1847 Submitted 1/12; Revised 2/14; Published 5/14

Bayesian Inference with Posterior Regularization and Applications to Infinite Latent SVMs

Jun Zhu DCSZJ@MAIL.TSINGHUA.EDU.CN
Ning Chen NINGCHEN@MAIL.TSINGHUA.EDU.CN
Department of Computer Science and Technology
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Tsinghua University
Beijing, 100084 China

Eric P. Xing EPXING@CS.CMU.EDU
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Editor: Tony Jebara

Abstract

Existing Bayesian models, especially nonparametric Bayesian methods, rely on specially conceived priors to incorporate domain knowledge for discovering improved latent representations. While priors affect posterior distributions through Bayes' rule, imposing posterior regularization is arguably more direct and in some cases more natural and general. In this paper, we present *regularized Bayesian inference* (RegBayes), a novel computational framework that performs posterior inference with a regularization term on the desired post-data posterior distribution under an information theoretical formulation. RegBayes is more flexible than the procedure that elicits expert knowledge via priors, and it covers both directed Bayesian networks and undirected Markov networks. When the regularization is induced from a linear operator on the posterior distributions, such as the expectation operator, we present a general convex-analysis theorem to characterize the solution of RegBayes. Furthermore, we present two concrete examples of RegBayes, *infinite latent support vector machines* (iLSVM) and *multi-task infinite latent support vector machines* (MT-iLSVM), which explore the large-margin idea in combination with a nonparametric Bayesian model for discovering predictive latent features for classification and multi-task learning, respectively. We present efficient inference methods and report empirical studies on several benchmark data sets, which appear to demonstrate the merits inherited from both large-margin learning and Bayesian nonparametrics. Such results contribute to push forward the interface between these two important subfields, which have been largely treated as isolated in the community.

Keywords: Bayesian inference, posterior regularization, Bayesian nonparametrics, large-margin learning, classification, multi-task learning

1. Introduction

Over the past decade, nonparametric Bayesian models have gained remarkable popularity in machine learning and other fields, partly owing to their desirable utility as a "nonparamet-

ric" prior distribution for a wide variety of probabilistic models, thereby turning the largely heuristic model selection practice, such as determining the unknown number of components in a mixture model (Antoniak, 1974) or the unknown dimensionality of latent features in a factor analysis model (Griffiths and Ghahramani, 2005), as a Bayesian inference problem in an unbounded model space. Popular examples include Gaussian process (GP) (Rasmussen and Ghahramani, 2002), Dirichlet process (DP) (Ferguson, 1973; Antoniak, 1974), and Beta process (BP) (Thibaux and Jordan, 2007). DP is often described with a Chinese restaurant process (CRP) metaphor, and similarly BP is often described with an Indian buffet process (IBP) metaphor (Griffiths and Ghahramani, 2005). Such nonparametric Bayesian approaches allow the model complexity to grow as more data are observed, which is a key factor differing them from other traditional "parametric" Bayesian models.

One recent development in practicing Bayesian nonparametrics is to relax some unrealistic assumptions on data, such as homogeneity and exchangeability. For example, to handle heterogenous observations, predictor-dependent processes (MacEachern, 1999; Williamson et al., 2010) have been proposed; and to relax the exchangeability assumption, stochastic processes with various correlation structures, such as hierarchical structures (Teh et al., 2006), temporal or spatial dependencies (Beal et al., 2002; Blei and Frazier, 2010), and stochastic ordering dependencies (Hoff, 2003; Dunson and Peddada, 2007), have been introduced. A common principle shared by these approaches is that they rely on defining, or in some unusual cases learning (Welling et al., 2012) a nonparametric Bayesian prior[1] encoding some special structures, which *indirectly*[2] influences the posterior distribution of interest through an interplay with a likelihood model according to the Bayes' rule (also known as Bayes' theorem). In this paper, we explore a different principle known as *posterior regularization*, which offers an additional and arguably richer and more flexible set of means to augment a posterior distribution under rich side information, such as predictive margin, structural bias, etc., which can be harder, if possible, to be captured by a Bayesian prior.

Let Θ denote model parameters and H denote hidden variables. Then given a set of observed data $\mathcal{D}$, posterior regularization (Ganchev et al., 2010) is generally defined as solving a regularized maximum likelihood estimation (MLE) problem:

$$\textbf{Posterior Regularization}: \quad \max_{\Theta} \mathcal{L}(\Theta; \mathcal{D}) + \Omega(p(H|\mathcal{D}, \Theta)), \tag{1}$$

where $\mathcal{L}(\Theta; \mathcal{D})$ is the marginal likelihood of $\mathcal{D}$, and $\Omega(\cdot)$ is a regularization function of the model posterior over latent variables. Note that here we view posterior as a generic post-data distribution on hidden variables in the sense of Ghosh and Ramamoorthi (2003, pp.15), not necessarily corresponding to a Bayesian posterior that must be induced by the Bayes' rule. The regularizer can be defined as a KL-divergence between a desired distribution with certain properties over latent variables and the model posterior in question, or other constraints on the model posterior, such as those used in generalized expectation (Mann and McCallum, 2010) or constraint-driven semi-supervised learning (Chang

1. Although likelihood is another dimension that can incorporate domain knowledge, existing work on Bayesian nonparametrics has been mainly focusing on the priors. Following this convention, this paper assumes that a common likelihood model (e.g., Gaussian likelihood for continuous data) is given.
2. A hard constraint on the prior (e.g., a truncated Gaussian) can directly affect the support of the posterior. RegBayes covers this as a special case as shown in Remark 7.

et al., 2007). An EM-type procedure can be applied to solve (1) approximately, and obtain an augmented MLE of the hidden variable model: $p(H|\mathcal{D}, \Theta_{\text{MLE}})$. When a distribution over the model parameter is of interest, going beyond the classical Bayesian theory, recent attempts toward learning a regularized posterior distribution of model parameters (and latent variables as well if present) include the "learning from measurements" (Liang et al., 2009), maximum entropy discrimination (MED) (Jaakkola et al., 1999; Zhu and Xing, 2009) and maximum entropy discrimination latent Dirichlet allocation (MedLDA) (Zhu et al., 2009). All these methods are parametric in that they give rise to distributions over a fixed and finite-dimensional parameter space. To the best of our knowledge, very few attempts have been made to impose posterior regularization in a nonparametric setting where model complexity depends on data, such as the case for nonparametric Bayesian latent variable models. A general formalism for (parametric and nonparametric) Bayesian inference with posterior regularization seems to be not yet available or apparent. In this paper, we present such a formalism, which we call *regularized Bayesian inference*, or RegBayes, built on the convex duality theory over distribution function spaces; and we apply this formalism to learn regularized posteriors under the Indian buffet process (IBP), conjoining two powerful machine learning paradigms, nonparametric Bayesian inference and SVM-style max-margin constrained optimization.

Unlike the regularized MLE formulation in (1), under the traditional formulation of Bayesian inference one is not directly optimizing an objective with respect to the posterior. To enable a regularized optimization formulation of RegBayes, we begin with a variational reformulation of the Bayes' theorem, and define $\mathcal{L}(q(\mathbf{M}|\mathcal{D}))$ as the KL-divergence between a desired post-data posterior $q(\mathbf{M}|\mathcal{D})$ over model $\mathbf{M}$ and the standard Bayesian posterior $p(\mathbf{M}|\mathcal{D})$ (see Section 3.1 for a recapitulation of the connection between KL-minimization and Bayes' theorem). RegBayes solves the following optimization problem:

$$\textbf{RegBayes}: \quad \inf_{q(\mathbf{M}|\mathcal{D})\in\mathcal{P}_{\text{prob}}} \mathcal{L}(q(\mathbf{M}|\mathcal{D})) + \Omega(q(\mathbf{M}|\mathcal{D})), \tag{2}$$

where the regularization $\Omega(\cdot)$ is a function of the post-data posterior $q(\mathbf{M}|\mathcal{D})$, and $\mathcal{P}_{\text{prob}}$ is the feasible space of well-defined distributions. By appropriately defining the model and its prior distribution, RegBayes can be instantiated to perform either parametric and nonparametric regularized Bayesian inference.

One particularly interesting way to derive the posterior regularization is to impose posterior constraints. Let $\boldsymbol{\xi}$ denote slack variables and $\mathcal{P}_{\text{post}}(\boldsymbol{\xi})$ denote the general soft posterior constraints (see Section 3.2 for a formal description), then, we can express the regularization term variationally:

$$\Omega(q(\mathbf{M}|\mathcal{D})) = \inf_{\boldsymbol{\xi}} \; U(\boldsymbol{\xi}), \quad \text{s.t.: } q(\mathbf{M}|\mathcal{D}) \in \mathcal{P}_{\text{post}}(\boldsymbol{\xi}), \tag{3}$$

where $U(\boldsymbol{\xi})$ is normally defined as a convex penalty function. The RegBayes formalism defined in (2) applies to a wide spectrum of models, including directed graphical models (i.e., Bayesian networks) and undirected Markov networks. For undirected models, when performing Bayesian inference the resulting posterior takes the form of a hybrid chain graphical model (Frydenberg, 1990; Murray and Ghahramani, 2004; Qi et al., 2005; Welling and Parise, 2006), which is usually much more challenging to regularize than for Bayesian

inference with directed graphical models. When the regularization term is convex and induced from a linear operator (e.g., expectation) of the posterior distributions, RegBayes can be solved with convex analysis theory.

By allowing direct regularization over posterior distributions, RegBayes provides a significant source of extra flexibility for post-data posterior inference, which applies to both parametric and nonparametric Bayesian learning (see the remarks after the main Theorem 6). In this paper, we focus on applying this technique to the later case, and illustrate how to use RegBayes to facilitate integration of Bayesian nonparametrics and large-margin learning, which have complementary advantages but have been largely treated as two disjoint subfields. Previously, it has been shown that, the core ideas of support vector machines (Vapnik, 1995) and maximum entropy discrimination (Jaakkola et al., 1999), as well as their structured extensions to the max-margin Markov networks (Taskar et al., 2003) and maximum entropy discrimination Markov networks (Zhu and Xing, 2009), have led to successful outcomes in many scenarios. But a large-margin model rarely has the flexibility of nonparametric Bayesian models to automatically handle model complexity from data, especially when latent variables are present (Jebara, 2001; Zhu et al., 2009). In this paper, we intend to bridge this gap using the RegBayes principle.

Specifically, we develop the *infinite latent support vector machines* (iLSVM) and *multi-task infinite latent support vector machines* (MT-iLSVM), which explore the discriminative large-margin idea to learn infinite latent feature models for classification and multi-task learning (Argyriou et al., 2007; Bakker and Heskes, 2003), respectively. We show that both models can be readily instantiated from the RegBayes master equation (2) by defining appropriate posterior regularization using the large-margin principle, and by employing an appropriate prior. For iLSVM, we use the IBP prior to allow the model to have an unbounded number of latent features *a priori*. For MT-iLSVM, we use a similar IBP prior to infer a latent projection matrix to capture the correlations among multiple predictive tasks while avoiding pre-specifying the dimensionality of the projection matrix. The regularized inference problems can be efficiently solved with an iterative procedure, which leverages existing high-performance convex optimization techniques.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents regularized Bayesian inference (RegBayes), together with the convex duality results that will be needed in latter sections. Section 4 concretizes the ideas of RegBayes and presents two infinite latent feature models with large-margin constraints for both classification and multi-task learning. Section 5 presents some preliminary experimental results. Finally, Section 6 concludes and discusses future research directions.

2. Related Work

Expectation regularization or expectation constraints have been considered to regularize model parameter estimation in the context of semi-supervised learning or learning with weakly labeled data. Mann and McCallum (2010) summarized the recent developments of the generalized expectation (GE) criteria for training a discriminative probabilistic model with unlabeled data, e.g., maximum entropy models or conditional random fields (Lafferty et al., 2001). By providing appropriate side information, such as labeled features or estimates of label distributions, a GE-based penalty function is defined to regularize the model

distribution, e.g., the distribution of class labels. One commonly used GE function is the KL-divergence between empirical expectation and model expectation of some feature functions if the expectations are normalized or the general Bregman divergence for unnormalized expectations. Although the GE criteria can be used alone as a scoring function to estimate the unknown parameters of a discriminative model, it is more usually used as a regularization term to an estimation method, such as maximum (conditional) likelihood estimation. Bellare et al. (2009) presented a different formulation of using expectation constraints in semi-supervised learning by introducing an auxiliary distribution to GE, together with an alternating projection algorithm, which can be more efficient. Liang et al. (2009) proposed to use the general notion of "measurements" to encapsulate the variety of weakly labeled data for learning exponential family models. The measurements can be labels, partial labels or other constraints on model predictions. Under the EM framework, posterior constraints were used in Graca et al. (2007) to modify the E-step of an EM algorithm to project model posterior distributions onto the subspace of distributions that satisfy a set of auxiliary constraints.

Dudík et al. (2007) studied the generalized maximum entropy principle with a rich form of expectation constraints using convex duality theory, where the standard moment matching constraints of maximum entropy are relaxed to inequality constraints. But their analysis was restricted to KL-divergence minimization (maximum entropy is a special case) and the finite dimensional space of observations. Later on, Altun and Smola (2006) presented a more general duality theory for a family of divergence functions on Banach spaces. We have drawn inspiration from both papers to develop the regularized Bayesian inference framework using convex duality theory.

When using large-margin posterior regularization, RegBayes generalizes the previous work on maximum entropy discrimination (Jaakkola et al., 1999; Zhu and Xing, 2009). The present paper provides a full extension of our preliminary work on max-margin nonparametric Bayesian models (Zhu et al., 2011b,a). For example, the infinite SVM (iSVM) (Zhu et al., 2011b) is a latent class model, where each data example is assigned to a single mixture component (i.e., an 1-dimensional space), and both iLSVM and MT-iLSVM extend the ideas to infinite latent feature models. For multi-task learning, nonparametric Bayesian models have been developed by Xue et al. (2007) and Rai and Daume III (2010) for learning features shared by multiple tasks. However, these methods are based on standard Bayesian inference without a posterior regularization using, for example, the large-margin constraints. Finally, MT-iLSVM can be also regarded as a nonparametric Bayesian formulation of the popular multi-task learning methods (Ando and Zhang, 2005; Jebara, 2011).

3. Regularized Bayesian Inference

We begin by laying out a general formulation of regularized Bayesian inference, using an optimization framework built on convex duality theory.

3.1 Variational formulation of Bayes' theorem

We first derive an optimization-theoretic reformulation of the Bayes' theorem. Let $\mathcal{M}$ denote the space of feasible models, and $\mathbf{M} \in \mathcal{M}$ represents an atom in this space. We assume that $\mathcal{M}$ is a complete separable metric space endowed with its Borel σ-algebra $\mathcal{B}(\mathcal{M})$. Let

Π be a distribution (i.e., a probability measure) on the measurable space $(\mathcal{M}, \mathcal{B}(\mathcal{M}))$. We assume that Π is absolutely continuous with respect to some background measure μ, so that there exists a density π such that $\mathrm{d}\Pi = \pi \mathrm{d}\mu$. Let $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^N$ be a collection of observed data, which we assume to be i.i.d. given a model. Let $P(\cdot|\mathbf{M})$ be the likelihood distribution, which is assumed to be dominated by a σ-finite measure λ for all $\mathbf{M}$ with positive density, so that there exists a density $p(\cdot|\mathbf{M})$ such that $\mathrm{d}P(\cdot|\mathbf{M}) = p(\cdot|\mathbf{M})\mathrm{d}\lambda$. Then, the Bayes' conditionalization rule gives a posterior distribution with the density (Ghosh and Ramamoorthi, 2003, Chap.1.3):

$$p(\mathbf{M}|\mathcal{D}) = \frac{\pi(\mathbf{M})p(\mathcal{D}|\mathbf{M})}{p(\mathcal{D})} = \frac{\pi(\mathbf{M})\prod_{n=1}^N p(\mathbf{x}_n|\mathbf{M})}{p(\mathbf{x}_1, \cdots, \mathbf{x}_N)},$$

a density over $\mathbf{M}$ with respect to the base measure μ, where $p(\mathcal{D})$ is the marginal likelihood of the observed data.

For reasons to be clear shortly, we now introduce a variational formulation of the Bayes' theorem. Let Q are an arbitrary distribution on the measurable space $(\mathcal{M}, \mathcal{B}(\mathcal{M}))$. We assume that Q is absolutely continuous with respect to Π and denote by q its density with respect to the background measure μ.[3] It can be shown that the posterior distribution of $\mathbf{M}$ due to the Bayes' theorem is equivalent to the optimum solution of the following convex optimization problem:

$$\begin{aligned} &\inf_{q(\mathbf{M})} \ \mathrm{KL}(q(\mathbf{M})\|\pi(\mathbf{M})) - \int_{\mathcal{M}} \log p(\mathcal{D}|\mathbf{M}) q(\mathbf{M}) \mathrm{d}\mu(\mathbf{M}) \\ &\text{s.t.} : q(\mathbf{M}) \in \mathcal{P}_{\text{prob}}, \end{aligned} \tag{4}$$

where $\mathrm{KL}(q(\mathbf{M})\|\pi(\mathbf{M})) = \int_{\mathcal{M}} q(\mathbf{M}) \log(q(\mathbf{M})/\pi(\mathbf{M}))\mathrm{d}\mu(\mathbf{M})$ is the Kullback-Leibler (KL) divergence from $q(\cdot)$ to $\pi(\cdot)$, and $\mathcal{P}_{\text{prob}}$ represents the feasible space of all density functions over $\mathbf{M}$ with respect to the measure μ. The proof is straightforward by noticing that the objective will become $\mathrm{KL}(q(\mathbf{M})\|p(\mathbf{M}|\mathcal{D}))$ by adding the constant $\log p(\mathcal{D})$. It is noteworthy that $q(\mathbf{M})$ here represents the density of a general post-data posterior distribution in the sense of Ghosh and Ramamoorthi (2003, pp.15), not necessarily corresponding to a Bayesian posterior that is induced by the Bayes' rule. As we shall see soon later, when we introduce additional constraints, the post-data posterior $q(\mathbf{M})$ is different from the Bayesian posterior $p(\mathbf{M}|\mathcal{D})$, and moreover, it could even not be obtainable from any Bayesian conditionalization in a different model. In the sequel, in order to distinguish $q(\cdot)$ from the Bayesian posterior, we will call it post-data distribution in short or post-data posterior distribution in full.[4] For notation simplicity, we have omitted the condition $\mathcal{D}$ in the post-data posterior distribution $q(\mathbf{M})$.

Remark 1 *The optimization formulation in (4) implies that Bayes' rule is an information projection procedure that projects a prior density to a post-data posterior by taking account of the observed data. In general, Bayes's rule is a special case of the principle of minimum information (Williams, 1980).*

3. This assumption is necessary to make the KL-divergence between the two distributions Q and Π well-defined. This assumption (or constraint) will be implicitly included in $\mathcal{P}_{\text{prob}}$ for clarity.
4. Rigorously, $q(\cdot)$ is the density of the post-data posterior distribution $Q(\cdot)$. We simply call q a distribution if no confusion arises.

3.2 Regularized Bayesian Inference with Expectation Constraints

In the variational formulation of Bayes' rule in (4), the constraints on $q(\mathbf{M})$ ensure that q is well-normalized and the objective is well-defined, i.e., $q(\mathbf{M}) \in \mathcal{P}_{\text{prob}}$, which do not capture any domain knowledge or structures of the model or data. Some previous efforts have been devoted to eliciting domain knowledge by constraining the prior or the base measure μ (Robert, 1995; Garthwaite et al., 2005). As we shall see, such constraints without considering data are special cases of RegBayes to be presented.

Specifically, the optimization-based formulation of Bayes' rule makes it straightforward to generalize Bayesian inference to a richer type of posterior inference, by replacing the standard normality constraint on q with a wide spectrum of knowledge-driven and/or data-driven constraints or regularization. To contrast, we will refer to problem (4) as "unconstrained" or "unregularized". Formally, we define *regularized Bayesian inference* (RegBayes) as a generalized posterior inference procedure that solves a constrained optimization problem due to such additional regularization imposed on q:

$$\begin{aligned} \inf_{q(\mathbf{M}),\boldsymbol{\xi}} \ & \mathrm{KL}(q(\mathbf{M})\|\pi(\mathbf{M})) - \int_{\mathcal{M}} \log p(\mathcal{D}|\mathbf{M}) q(\mathbf{M}) \mathrm{d}\mu(\mathbf{M}) + U(\boldsymbol{\xi}) \\ \text{s.t.} : \ & q(\mathbf{M}) \in \mathcal{P}_{\text{post}}(\boldsymbol{\xi}), \end{aligned} \tag{5}$$

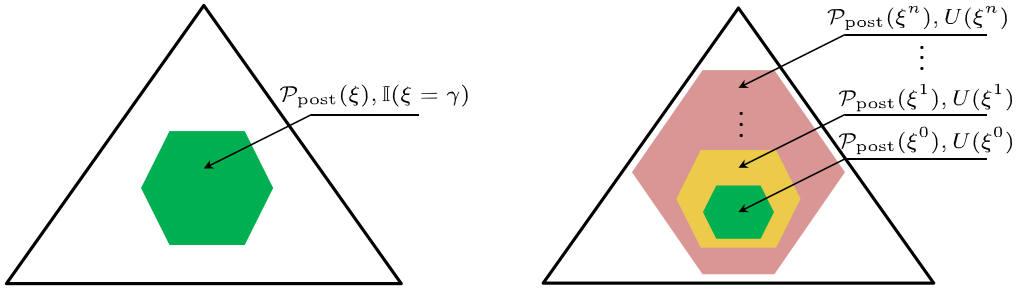
where $\mathcal{P}_{\text{post}}(\boldsymbol{\xi})$ is a subspace of distributions that satisfy a set of additional constraints besides the standard normality constraint of a probability distribution. Using the variational formulation in (3), problem (5) can be rewritten in the form of the master equation (2), of which the objective is: $\mathcal{L}(q(\mathbf{M})) = \mathrm{KL}(q(\mathbf{M})\|\pi(\mathbf{M})) - \int_{\mathcal{M}} \log p(\mathcal{D}|\mathbf{M}) q(\mathbf{M}) \mathrm{d}\mu(\mathbf{M}) = \mathrm{KL}(q(\mathbf{M})\|p(\mathbf{M},\mathcal{D}))$ and the posterior regularization is $\Omega(q(\mathbf{M})) = \inf_{\boldsymbol{\xi}} U(\boldsymbol{\xi})$, s.t.: $q(\mathbf{M}|\mathcal{D}) \in \mathcal{P}_{\text{post}}(\boldsymbol{\xi})$. Note that when $\mathcal{D}$ is given, the distribution $p(\mathbf{M},\mathcal{D})$ is unnormalized for $\mathbf{M}$; and we have abused the KL notation for unnormalized distributions in $\mathrm{KL}(q(\mathbf{M})\|p(\mathbf{M},\mathcal{D}))$, but with the same formula.

Obviously this formulation enables different types of constraints to be employed in practice. In this paper, we focus on the *expectation constraints*, of which each one is a function of $q(\mathbf{M})$ through an expectation operator. For instance, let $\boldsymbol{\psi} = (\psi_1, \cdots, \psi_T)$ be a vector of feature functions, each of which is $\psi_t(\mathbf{M};\mathcal{D})$ defined on $\mathbf{M}$ and possibly data dependent. Then a subspace of feasible post-data distributions can be defined in the following form:

$$\mathcal{P}_{\text{post}}(\boldsymbol{\xi}) \stackrel{\text{def}}{=} \Big\{ q(\mathbf{M}) | \ \forall t = 1, \cdots, T, \ h\big(Eq(\psi_t;\mathcal{D})\big) \le \xi_t \Big\}, \tag{6}$$

where E is the expectation operator that maps $q(\mathbf{M})$ to a point in the space $\mathbb{R}^T$, and for each feature function ψ_t: $Eq(\psi_t;\mathcal{D}) \stackrel{\text{def}}{=} \mathbb{E}_{q(\mathbf{M})}[\psi_t(\mathbf{M};\mathcal{D})]$. The function h can be of any form in theory, though a simple h function will make the optimization problem easy to solve. The auxiliary parameters $\boldsymbol{\xi}$ are usually nonnegative and interpreted as slack variables. The constraints with non-trivial $\boldsymbol{\xi}$ are soft constraints as illustrated in Figure 1(b). But we emphasize that by defining U as an indicator function, the formulation (5) covers the case where hard constraints are imposed. For instance, if we define

$$U(\boldsymbol{\xi}) = \sum_{t=1}^{T} \mathbb{I}(\xi_t = \gamma_t) = \mathbb{I}(\boldsymbol{\xi} = \boldsymbol{\gamma}),$$



(a) (b)

Figure 1: Illustration for the (a) hard and (b) soft constraints in the simple setting which has only three possible models. For hard constraints, we have only one feasible subspace. In contrast, we have many (normally infinite for continuous $\boldsymbol{\xi}$) feasible subspaces for soft constraints and each of them is associated with a different complexity or penalty, measured by the U function.

where $\mathbb{I}(c)$ is an indicator function that equals to 0 if the condition c is satisfied; otherwise ∞, then all the expectation constraints in (6) are hard constraints. As illustrated in Figure 1(a), hard constraints define one single feasible subspace (assuming to be non-empty). In general, we assume that $U(\boldsymbol{\xi})$ is a convex function, which represents a penalty on the size of the feasible subspaces, as illustrated in Figure 1(b). A larger subspace typically leads to models with a higher complexity. In the classification models to be presented, U corresponds to a surrogate loss, e.g., hinge loss of a prediction rule, as we shall see.

Similarly, the formulation of RegBayes with expectation constraints in (6) can be equivalently written in an "unconstrained" form by using the rule in (3). Specifically, let $g(Eq(\boldsymbol{\psi}; \mathcal{D})) \stackrel{\text{def}}{=} \inf_{\boldsymbol{\xi}} U(\boldsymbol{\xi})$, s.t. : $h(Eq(\psi_t; \mathcal{D})) \leq \xi_t$, $\forall t$, we have the equivalent optimization problem:

$$\inf_{q(\mathbf{M}) \in \mathcal{P}_{\text{prob}}} \text{KL}(q(\mathbf{M}) \| \pi(\mathbf{M})) - \int_{\mathcal{M}} \log p(\mathcal{D}|\mathbf{M}) q(\mathbf{M}) \text{d}\mu(\mathbf{M}) + g(Eq(\boldsymbol{\psi}; \mathcal{D})), \tag{7}$$

where $Eq(\boldsymbol{\psi}; \mathcal{D})$ is a point in $\mathbb{R}^T$ and the t-th coordinate is $Eq(\psi_t; \mathcal{D})$, a function of $q(\mathbf{M})$ as defined before. We assume that the real-valued function $g : \mathbb{R}^T \to \mathbb{R}$ is convex and lower semi-continuous. For each U, we can induce a g function by taking the infimum of $U(\boldsymbol{\xi})$ over $\boldsymbol{\xi}$ with the posterior constraints; vice versa. If we use hard constraints, similar as in regularized maximum entropy density estimation (Altun and Smola, 2006; Dudík et al., 2007), we have

$$g(Eq) = \sum_{t=1}^{T} \mathbb{I}(h(Eq(\psi_t; \mathcal{D})) \leq \gamma_t).$$

For the regularization function g, as well as U, we can have many choices, besides the above mentioned indicator function. For example, if the feature function ψ_t is an

indicator function and we could obtain 'prior' expectations $\mathbb{E}_{\tilde{p}}[\psi_t]$ from domain/expert knowledge about $\mathbf{M}$. If we further normalize the empirical expectations of T functions and denote the discrete distribution by $\tilde{p}(\mathbf{M})$, one natural regularization function would be the KL-divergence between prior expectations and the expectations computed from the normalized model posterior $q(\mathbf{M})$, i.e., $g(Eq) = \sum_t s(\mathbb{E}_{\tilde{p}}[\psi_t], Eq(\psi_t)) = \mathrm{KL}(\tilde{p}(\mathbf{M})\|q(\mathbf{M}))$, where $s(x, y) = x \log(x/y)$ for $x, y \in (0, 1)$. The general Bregman divergence can be used for unnormalized expectations. This kind of regularization function has been used in Mann and McCallum (2010) for label regularization, in the context of semi-supervised learning. Other choices of the regularization function include the ℓ_2^2 penalty or indicator function with equality constraints; see Table 1 in Dudík et al. (2007) for a summary.

Remark 2 *So far, we have focused on RegBayes in the context of full Bayesian inference. Indeed, RegBayes can be generalized to apply to empirical Bayesian inference, where some model parameters need to be estimated. More generally, RegBayes applies to both directed Bayesian networks (of which the hierarchical Bayesian models we have discussed are an example) and undirected Markov random fields. But for undirected models, a RegBayes treatment will have to deal with a chain graph resultant from Bayesian inference, which is more challenging due to existence of normalization factors. We will discuss some details and examples in Appendix A.*

3.3 Optimization with Convex Duality Theory

Depending on several factors, including the size of the model space, the data likelihood model, the prior distribution, and the regularization function, a RegBayes problem in general can be highly non-trivial to solve, either in the constrained or unconstrained form, as can be seen from several concrete examples of RegBayes models we will present in the next section and in the Appendix B. In this section, we present a representation theorem to characterize the solution the convex RegBayes problem (7) with expectation regularization. These theoretical results will be used later in developing concrete RegBayes models.

To make the subsequent statements general, we consider the following problem:

$$\inf_{x \in \mathcal{X}} f(x) + g(Ax),$$

where $f : \mathcal{X} \to \mathbb{R}$ is a convex function; $A : \mathcal{X} \to \mathcal{B}$ is a bounded linear operator; and $g : \mathcal{B} \to \mathbb{R}$ is also convex. Below we introduce some tools in convex analysis theory to study this problem. We begin by formulating the primal-dual space relationships of convex optimization problems in the general settings, where we assume both $\mathcal{X}$ and $\mathcal{B}$ are Banach spaces.[5] An important result we build on is the Fenchel duality theorem.

Definition 3 (Convex Conjugate) *Let $\mathcal{X}$ be a Banach space and $\mathcal{X}^*$ be its dual space. The convex conjugate or the Legendre-Frenchel transformation of a function $f : \mathcal{X} \to [-\infty, +\infty]$ is $f^* : \mathcal{X}^* \to [-\infty, +\infty]$, where*

$$f^*(x^*) = \sup_{x \in \mathcal{X}} \{\langle x^*, x\rangle - f(x)\}.$$

5. A Banach space is a vector space with a metric that allows the computation of vector length and distance between vectors. Moreover, a Cauchy sequence of vectors always converges to a well defined limit in the space.

Theorem 4 (Fenchel Duality (Borwein and Zhu, 2005)) *Let $\mathcal{X}$ and $\mathcal{B}$ be Banach spaces, $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathcal{B} \to \mathbb{R} \cup \{+\infty\}$ be convex functions and $A : \mathcal{X} \to \mathcal{B}$ be a bounded linear map. Define the primal and dual values t, d by the Fenchel problems*

$$t = \inf_{x \in \mathcal{X}} \{f(x) + g(Ax)\} \text{ and } d = \sup_{x^* \in \mathcal{B}^*} \{-f^*(A^*x^*) - g^*(-x^*)\}.$$

Then these values satisfy the weak duality inequality $t \geq d$. If f, g and A satisfy either

$$0 \in \text{core}(\text{dom}g - A\text{dom}f) \text{ and both } f \text{ and } g \text{ are lower semicontinuous } (lsc),$$

or

$$A\text{dom}f \cap \text{cont}g \neq \emptyset,$$

then $t = d$ and the supremum to the dual problem is attainable if finite.

Let $\mathcal{S}$ be a subset of a Banach space $\mathcal{B}$. In the above theorem, we say s is in the *core* of $\mathcal{S}$, denoted by $s \in \text{core}(\mathcal{S})$, provided that $\cup_{\lambda > 0} \lambda(\mathcal{S} - s) = \mathcal{B}$.

The Fenchel duality theorem has been applied to solve divergence minimization problems for density estimation (Altun and Smola, 2006; Dudík et al., 2007). Let $\boldsymbol{\psi} \stackrel{\text{def}}{=} (\psi_1, \cdots, \psi_T)$ be a vector of feature functions. Each feature function is a mapping, $\psi_t : \mathcal{M} \to \mathbb{R}$. Therefore, $\mathcal{B}$ is the product space $\mathbb{R}^T$, a simple Banach space. Let $\mathcal{X}$ be the Banach space of finite signed measures (with total variation as the norm) that are absolutely continuous with respect to the measure μ, and let A be the expectation operator of the feature functions with respect to the distribution q on $\mathcal{M}$, that is, $Aq \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{M} \sim q}[\boldsymbol{\psi}(\mathbf{M})]$, where $\boldsymbol{\psi}(\mathbf{M}) = (\psi_1(\mathbf{M}), \cdots, \psi_T(\mathbf{M}))$. Let $\tilde{\boldsymbol{\psi}}$ be a reference point in $\mathbb{R}^T$. As for density estimation, we have some observations of $\mathbf{M}$ here, and $\tilde{\boldsymbol{\psi}} = Ap_{\text{emp}}[\boldsymbol{\psi}(\mathbf{M})]$, where p_{emp} is the empirical distribution. Then, when the f function is a KL-divergence and the constraints are relaxed moment matching constraints, the following result can be proven.

Lemma 5 (KL-divergence with Constraints (Altun and Smola, 2006))

$$\begin{aligned} &\inf_q \left\{ \text{KL}(q\|p) \text{ s.t. : } \|\mathbb{E}_q[\boldsymbol{\psi}] - \tilde{\boldsymbol{\psi}}\|_{\mathcal{B}} \leq \epsilon \text{ and } q \in \mathcal{P}_{\text{prob}} \right\} \\ = \; &\sup_{\boldsymbol{\phi}} \left\{ \langle \boldsymbol{\phi}, \tilde{\boldsymbol{\psi}} \rangle - \log \int_{\mathcal{M}} p(\mathbf{M}) \exp(\langle \boldsymbol{\phi}, \boldsymbol{\psi}(\mathbf{M}) \rangle) \mathrm{d}\mu(\mathbf{M}) - \epsilon \|\boldsymbol{\phi}\|_{\mathcal{B}^*} \right\}, \end{aligned}$$

where the unique solution is given by $\hat{q}_{\hat{\boldsymbol{\phi}}}(\mathbf{M}) = p(\mathbf{M}) \exp(\langle \hat{\boldsymbol{\phi}}, \boldsymbol{\psi}(\mathbf{M}) \rangle - \Lambda_{\hat{\boldsymbol{\phi}}})$; $\hat{\boldsymbol{\phi}}$ is the solution of the dual problem; and $\Lambda_{\hat{\boldsymbol{\phi}}}$ is the log-partition function.

Note that for this lemma and the ones to be presented below to hold, the problems need to meet some regularity conditions (or constraint qualifications), such as those in Theorem 4. In practice it can be difficult to check whether the constraint qualifications hold. One solution is to solve the dual optimization problem and examine if the conditions hold depending on whether the solution diverge or not (Altun and Smola, 2006).

The problem in the above lemma is subject to hard constraints, therefore the corresponding g is the indicator function $\mathbb{I}(\|\mathbb{E}_q[\boldsymbol{\psi}] - \tilde{\boldsymbol{\psi}}\|_{\mathcal{B}} \leq \epsilon)$ when applying the Fenchel duality

theorem. Other examples of the posterior constraints can be found in Dudík et al. (2007); Mann and McCallum (2010); Ganchev et al. (2010), as we have discussed in Section 3.2. In this paper, we consider the general soft constraints as defined in the RegBayes problem (5). Furthermore, we do not assume the existence of a fully observed data set to compute the empirical expectation $\tilde{\boldsymbol{\phi}}$. Specifically, following a similar line of reasoning as in Altun and Smola (2006), though this time with an un-normalized p in $\mathrm{KL}(q\|p)$, we have the following result. The detailed proof is deferred to Appendix C.1.

Theorem 6 (Representation theorem of RegBayes) *Let E be the expectation operator with feature functions $\boldsymbol{\psi}(\mathbf{M};\mathcal{D})$, and assume g is convex and lower semicontinuous (lsc). We have*

$$\begin{aligned}&\inf_{q(\mathbf{M})}\Big\{\mathrm{KL}(q(\mathbf{M})\|p(\mathbf{M},\mathcal{D}))+g(Eq)\ \text{s.t.}:q(\mathbf{M})\in\mathcal{P}_{\text{prob}}\Big\}\\ =\ &\sup_{\boldsymbol{\phi}}\Big\{-\log\int_{\mathcal{M}}p(\mathbf{M},\mathcal{D})\exp(\langle\boldsymbol{\phi},\boldsymbol{\psi}(\mathbf{M};\mathcal{D})\rangle)\mathrm{d}\mu(\mathbf{M})-g^*(-\boldsymbol{\phi})\Big\},\end{aligned}$$

where the unique solution is given by $\hat{q}_{\hat{\boldsymbol{\phi}}}(\mathbf{M})=p(\mathbf{M},\mathcal{D})\exp(\langle\hat{\boldsymbol{\phi}},\boldsymbol{\psi}(\mathbf{M};\mathcal{D})\rangle-\Lambda_{\hat{\boldsymbol{\phi}}})$; and $\hat{\boldsymbol{\phi}}$ is the solution of the dual problem; and $\Lambda_{\hat{\boldsymbol{\phi}}}$ is the log-partition function.

From the optimum solution $\hat{q}_{\hat{\boldsymbol{\phi}}}(\mathbf{M})$, we can see that the form of the RegBayes posterior is symbolically similar to that of the Bayesian posterior; but instead of multiplying the likelihood term with a prior distribution, RegBayes introduces an extra term, $\exp(\langle\hat{\boldsymbol{\phi}},\boldsymbol{\psi}(\mathbf{M};\mathcal{D})\rangle-\Lambda_{\hat{\boldsymbol{\phi}}})$, whose coefficients are derived from an constrained optimization problem resultant from the constraints on the posterior. We make the following remarks.

Remark 7 (Putting constraints on priors is a special case of RegBayes) *If both the feature function $\boldsymbol{\psi}(\mathbf{M};\mathcal{D})$ and $\hat{\phi}$ depend on the model $\mathbf{M}$ only, this extra term contributes to define a new prior $\pi'(\mathbf{M})\propto\pi(\mathbf{M})\exp(\langle\hat{\boldsymbol{\phi}},\boldsymbol{\psi}(\mathbf{M};\mathcal{D})\rangle-\Lambda_{\hat{\boldsymbol{\phi}}})$. For example, if we constrain the model space to a subset $\mathcal{M}_0\subset\mathcal{M}$ a priori, this constraint can be incorporated in RegBayes by defining the expectation constraint on $\mathbf{M}$ only. Specifically, define the single feature function $\psi(\mathbf{M})$: $\psi(\mathbf{M})=0$ if $\mathbf{M}\in\mathcal{M}_0$, otherwise 1; and define the simple posterior regularization $g(Eq)=\mathbb{I}(\mathbb{E}_q[\psi(\mathbf{M})]=0)$. Then, by Theorem 6,*[6] *we have $\hat{\phi}=-\infty$ and $\hat{q}_{\hat{\phi}}(\mathbf{M})\propto\pi'(\mathbf{M})p(\mathcal{D}|\mathbf{M})$, where $\pi'(\mathbf{M})\propto\pi(\mathbf{M})\mathbb{I}(\mathbf{M}\in\mathcal{M}_0)$ is the constrained prior. Therefore, such a constraint lets RegBayes cover the widely used truncated priors, such as truncated Gaussian (Robert, 1995).*

Remark 8 (RegBayes is more flexible than Bayes' rule) *For the more general case where $\boldsymbol{\psi}(\mathbf{M};\mathcal{D})$ depends on both $\mathbf{M}$ and $\mathcal{D}$, the term $p(\mathbf{M},\mathcal{D})\exp(\langle\hat{\boldsymbol{\phi}},\boldsymbol{\psi}(\mathbf{M};\mathcal{D})\rangle)$ implicitly defines a joint distribution on $(\mathbf{M},\mathcal{D})$ if it has a finite measure. In this case, RegBayes is doing implicit Bayesian conditionalization, that is, the posterior $\hat{q}_{\hat{\boldsymbol{\phi}}}(\mathbf{M})$ can be obtained through Bayes' rule with some well-defined prior and likelihood. However, it could be that the integral of $p(\mathbf{M},\mathcal{D})\exp(\langle\hat{\boldsymbol{\phi}},\boldsymbol{\psi}(\mathbf{M};\mathcal{D})\rangle)$ with respect to $(\mathbf{M},\mathcal{D})$ is not finite because of the way $\hat{\boldsymbol{\phi}}$ varies with $\mathcal{D}$,*[7] *in which case there is no implicit prior and likelihood that give back*

6. We also used the fact that if $f(x)=\mathbb{I}(x=c)$ is an indicator function, its conjugate is $f^*(\mu)=c\cdot\mu$.
7. Note: this does not affect the well-normalization of the posterior $\hat{q}_{\hat{\boldsymbol{\phi}}}(\mathbf{M})$ because its integral is taken over $\mathbf{M}$ only, with $\mathcal{D}$ fixed.

$\hat{q}_{\hat{\phi}}(\mathbf{M})$ *through Bayesian conditionalization. Therefore, RegBayes is more flexible than the standard Bayesian inference, where the prior and likelihood model are explicitly defined, but no additional constraints or regularization can be systematically incorporated. The recent work (Mei et al., 2014) presents an example. Specifically, we show that incorporating domain knowledge via posterior regularization can lead to a flexible framework that automatically learns the importance of each piece of knowledge, thereby allowing for a robust incorporation, which is important in the scenarios where noisy knowledge is collected from crowds. In contrast, eliciting expert knowledge via fitting some priors is generally hard, especially in high-dimensional spaces, as experts are normally good at perceiving low-dimensional and well-behaved distributions but can be very bad in perceiving high-dimensional or skewed distributions (Garthwaite et al., 2005).*

It is worth mentioning that although the above theorem provides a generic representation of the solution to RegBayes, in practice we usually need to make additional assumptions in order to make either the primal or dual problem tractable to solve. Since such assumptions could make the feasible space non-convex, additional cautions need to be paid. For instance, the mean-field assumptions will lead to a non-convex feasible space (Wainright and Jordan, 2008), and we can only apply the convex analysis theory to deal with convex sub-problems within an EM-type procedure. More concrete examples will be provided later along the developments of various models. We should also note that the modeling flexibility of RegBayes comes with risks. For example, it might lead to inconsistent posteriors (Barron et al., 1999; Choi and Ramamoorthi, 2008). This paper focuses on presenting several practical instances of RegBayes and we leave a systematic analysis of the Bayesian asymptotic properties (e.g., posterior consistency and convergence rates) for future work.

Now, we derive the conjugate functions of three examples which will be used shortly for developing the infinite latent SVM models we have intended. We defer the proof to Appendix C. Specifically, the first one is the conjugate of a simple function, which will be used in a binary latent SVM classification model.

Lemma 9 *Let $g_0 : \mathbb{R} \to \mathbb{R}$ be defined as $g_0(x) = C \max(0, x)$. Then, we have*

$$g_0^*(\mu) = \mathbb{I}(0 \leq \mu \leq C).$$

The second function is slightly more complex, which will be used for defining a multi-way latent SVM classifier. Specifically, we define the function $g_1 : \mathbb{R}^L \to \mathbb{R}$ as

$$g_1(\mathbf{x}) = C \max(\mathbf{x}), \tag{8}$$

where $\max(\mathbf{x}) \stackrel{\text{def}}{=} \max(x_1, \cdots, x_L)$. Apparently, g_1 is convex because it is a point-wise maximum (Boyd and Vandenberghe, 2004) of the simple linear functions $\phi_i(\mathbf{x}) = x_i$. Then, we have the following results.

Lemma 10 *The convex conjugate of $g_1(\mathbf{x})$ as defined above is*

$$g_1^*(\boldsymbol{\mu}) = \mathbb{I}\Big(\forall i, \mu_i \geq 0; \text{ and } \sum_i \mu_i = C\Big).$$

Let $y \in \mathbb{R}$ and $\epsilon \in \mathbb{R}_+$ are fixed parameters. The last function that we are interested in is $g_2 : \mathbb{R} \to \mathbb{R}$, where

$$g_2(x; y, \epsilon) = C \max(0, |x - y| - \epsilon).$$

Finally, we have the following lemma, which will be used in developing large-margin regression models.

Lemma 11 *The convex conjugate of $g_2(x)$ as defined above is*

$$g_2^*(\mu; y, \epsilon) = \mu y + \epsilon|\mu| + \mathbb{I}\big(|\mu| \leq C\big).$$

4. Infinite Latent Support Vector Machines

Given the general theoretical framework of RegBayes introduced in Section 3, now we are ready to present its application to the development of two interesting nonparametric RegBayes models. In these two models we conjoin the ideas behind the nonparametric Bayesian infinite feature model known as the Indian buffet process (IBP), and the large margin classifier known as support vector machines (SVM) to build a new class of models for simultaneous single-task (or multi-task) classification and feature learning. A parametric Bayesian model is presented in Appendix B.

Specifically, to illustrate how to develop latent large-margin classifiers and automatically resolve the unknown dimensionality of latent features from data, we demonstrate how to choose/define the three key elements of RegBayes, that is, *prior distribution*, *likelihood model*, and *posterior regularization*. We first present the single-task classification model. The basic setup is that we project each data example $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^D$ to a latent feature vector $\mathbf{z}$. Here, we consider binary features. Real-valued features can be easily considered by elementwise multiplication of $\mathbf{z}$ by a Guassian vector (Griffiths and Ghahramani, 2005). Given a set of N data examples, let $\mathbf{Z}$ be the matrix, of which each row is a binary vector $\mathbf{z}_n$ associated with data sample n. Instead of pre-specifying a fixed dimension of $\mathbf{z}$, we resort to the nonparametric Bayesian methods and let $\mathbf{z}$ have an infinite number of dimensions. To make the expected number of active latent features finite, we employ an IBP as prior for the binary feature matrix $\mathbf{Z}$, as reviewed below.

4.1 Indian Buffet Process

Indian buffet process (IBP) was proposed in Griffiths and Ghahramani (2005) and has been successfully applied in various fields, such as link prediction (Miller et al., 2009) and multi-task learning (Rai and Daume III, 2010). We will make use of its stick-breaking construction (Teh et al., 2007), which is good for developing efficient inference methods. Let $\pi_k \in (0, 1)$ be a parameter associated with each column of the binary matrix $\mathbf{Z}$. Given π_k, each z_{nk} in column k is sampled independently from Bernoulli(π_k). The parameter $\boldsymbol{\pi}$ are generated by a stick-breaking process

$$\pi_1 = \nu_1, \text{ and } \pi_k = \nu_k \pi_{k-1} = \prod_{i=1}^{k} \nu_i,$$

where $\nu_i \sim \text{Beta}(\alpha, 1)$. Since each ν_i is less than 1, this process generates a decreasing sequence of π_k. Specifically, given a finite data set, the probability of seeing feature k decreases exponentially with k.

IBP has several properties. For a finite number of rows, N, the prior of the IBP gives zero mass on matrices with an infinite number of ones, as the total number of columns with non-zero entries is Poisson(αH_N), where H_N is the Nth harmonic number, $H_N = \sum_{j=1}^{N} \frac{1}{j}$. Thus, $\mathbf{Z}$ has almost surely only a finite number of non-zero entries, though this number is unbounded. A second property of IBP is that the number of features possessed by each data point follows a Poisson(α) distribution. Therefore, the expected number of non-zero entries in $\mathbf{Z}$ is $N\alpha$.

4.2 Infinite Latent Support Vector Machines

Consider a single-task, but multi-way classification, where each training data is provided with a categorical label $y \in \mathcal{Y} \overset{\text{def}}{=} \{1, \cdots, L\}$. Suppose that the latent features $\mathbf{z}_n$ for document n are given, then we can define the *latent discriminant function* as linear

$$f(y, \mathbf{x}_n, \mathbf{z}_n; \boldsymbol{\eta}) \overset{\text{def}}{=} \boldsymbol{\eta}^\top \mathbf{g}(y, \mathbf{x}_n, \mathbf{z}_n), \tag{9}$$

where $\mathbf{g}(y, \mathbf{x}_n, \mathbf{z}_n)$ is a vector stacking L subvectors of which the yth is $\mathbf{z}_n^\top$ and all the others are zero;[8] $\boldsymbol{\eta}$ is the corresponding infinite-dimensional vector of feature weights. Since we are doing Bayesian inference, we need to maintain the entire distribution profile of the latent feature matrix $\mathbf{Z}$. However, in order to make a prediction on the observed data $\mathbf{x}$, we need to remove the uncertainty of $\mathbf{Z}$. Here, we define the *effective discriminant function* as an expectation[9] (i.e., a weighted average considering all possible values of $\mathbf{Z}$) of the latent discriminant function. To fully explore the flexibility offered by Bayesian inference, we also treat $\boldsymbol{\eta}$ as random and aim to infer its posterior distribution from given data. For the prior, we assume all the dimensions of $\boldsymbol{\eta}$ are independent and each dimension η_k follows the standard normal distribution. This is in fact a Gaussian process (GP) prior as $\boldsymbol{\eta}$ is infinite dimensional. More formally, the effective discriminant function $f : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ is

$$\begin{aligned} f\big(y, \mathbf{x}_n; q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})\big) &\overset{\text{def}}{=} \mathbb{E}_{q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})}\big[f(y, \mathbf{x}_n, \mathbf{z}_n; \boldsymbol{\eta})\big] \\ &= \mathbb{E}_{q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})}\big[\boldsymbol{\eta}^\top \mathbf{g}(y, \mathbf{x}_n, \mathbf{z}_n)\big], \end{aligned} \tag{10}$$

where $q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})$ is the post-data posterior distribution we want to infer. We have included $\mathbf{W}$ as a place holder for any other variables we may define, e.g., the variables arising from a data likelihood model. Since we are taking the expectation, the variables which do not appear in the feature map $\mathbf{g}$ (i.e., $\mathbf{W}$) will be marginalized out.

Before moving on, we should note that since we require q to be absolutely continuous with respect to the prior to make the KL-divergence term well defined in the RegBayes

8. We can consider the input features $\mathbf{x}_n$ or its certain statistics in combination with the latent features $\mathbf{z}_n$ to define a classifier boundary, by simply concatenating them in the subvectors.
9. Although other choices such as taking the mode are possible, our choice could lead to a computationally easy problem because expectation is a linear functional of the distribution under which the expectation is taken. Moreover, expectation can be more robust than taking the mode (Khan et al., 2010), and it has been widely used in previous work (Zhu et al., 2009, 2011b).

problem, $q(\mathbf{Z})$ will also put zero mass on $\mathbf{Z}$'s with an infinite number of non-zero entries, because of the properties of the IBP prior. The sparsity of $\mathbf{Z}$ is essential to ensure that the dot-product in (9) and the expectation in (10) are well defined, i.e., with finite values.[10] Moreover, in practice, to make the problem computationally feasible, we usually set a finite upper bound K to the number of possible features, where K is sufficiently large and known as the truncation level (See Section 4.4 and Appendix D.2 for details). As shown in Doshi-Velez (2009), the ℓ_1-distance truncation error of marginal distributions decreases exponentially as K increases. For a finite truncation level, all the expectations are definitely finite.

Let $\mathcal{I}_{\text{tr}}$ denote the set of training data. Then, with the above definitions, we define the $\mathcal{P}_{\text{post}}(\boldsymbol{\xi})$ in problem (5) using soft large-margin constraints as[11]

$$\mathcal{P}^c_{\text{post}}(\boldsymbol{\xi}) \stackrel{\text{def}}{=} \left\{ q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W}) \,\middle|\, \begin{array}{c} \forall n \in \mathcal{I}_{\text{tr}} : \Delta f(y, \mathbf{x}_n; q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})) \geq \ell^{\Delta}_n(y) - \xi_n, \forall y \\ \xi_n \geq 0 \end{array} \right\},$$

where $\Delta f(y, \mathbf{x}_n; q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})) \stackrel{\text{def}}{=} f(y_n, \mathbf{x}_n; q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})) - f(y, \mathbf{x}_n; q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W}))$ is the margin favored by the true label y_n over an arbitrary label y and the superscript is used to distinguish from the posterior constraints for multi-task iLSVM to be presented. We define the penalty function for classification as

$$U^c(\boldsymbol{\xi}) \stackrel{\text{def}}{=} C \sum_{n \in \mathcal{I}_{\text{tr}}} \xi^{\kappa}_n,$$

where $\kappa \geq 1$. If κ is 1, minimizing $U^c(\boldsymbol{\xi})$ is equivalent to minimizing the hinge-loss (or ℓ_1-loss) $\mathcal{R}^c_h$ of the averaging prediction rule (13), where

$$\mathcal{R}^c_h(q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})) = C \sum_{n \in \mathcal{I}_{\text{tr}}} \max_y \big(\ell^{\Delta}_n(y) - \Delta f(y_n, \mathbf{x}_n; q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})) \big);$$

if κ is 2, the surrogate loss is the squared ℓ_2-loss. For clarity, we consider the hinge loss. The non-negative cost function $\ell^{\Delta}_n(y)$ (e.g., 0/1-cost) measures the cost of predicting $\mathbf{x}_n$ to be y when its true label is y_n. $\mathcal{I}_{\text{tr}}$ is the index set of training data.

Besides performing the prediction task, we may also be interested in explaining observed data $\mathbf{x}$ using the latent factors $\mathbf{Z}$. This can be done by defining a likelihood model $p(\mathbf{x}|\mathbf{Z})$. Here, we define the most common linear-Gaussian likelihood model for real-valued data

$$p\big(\mathbf{x}_n | \mathbf{z}_n, \mathbf{W}, \sigma^2_{n0}\big) = \mathcal{N}\big(\mathbf{x}_n | \mathbf{W}\mathbf{z}^{\top}_n, \sigma^2_{n0} I\big),$$

where $\mathbf{W}$ is a $D \times \infty$ random loading matrix. We assume $\mathbf{W}$ follows an independent Gaussian prior and each entry has the prior distribution $\pi(w_{dk}) = \mathcal{N}(w_{dk}|0, \sigma^2_0)$. The hyperparameters σ^2_0 and σ^2_{n0} can be set a priori or estimated from observed data (See Appendix D.2 for details). Figure 2 (a) shows the graphical structure of iLSVM as defined above, where the plate means N replicates.

10. A more rigorous derivation of finiteness of these quantities is beyond the scope of this work and could require additional technical conditions (Orbanz, 2012). We refer the readers to Stummer and Vajda (2012) for a generic definition of Bregman divergence (or KL divergence in particular) on Banach spaces and in the case where the second measure is unnormalized.

11. Hard constraints for the separable cases are covered by simply setting $\boldsymbol{\xi} = 0$.

Training: Putting the above definitions together, we get the RegBayes problem for iLSVM in the following two equivalent forms

$$\inf_{q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W}),\boldsymbol{\xi}} \mathrm{KL}(q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})\|p(\mathbf{Z},\boldsymbol{\eta},\mathbf{W},\mathcal{D})) + U^c(\boldsymbol{\xi}) \quad (11)$$
$$\text{s.t.}: \ q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W}) \in \mathcal{P}^c_{\text{post}}(\boldsymbol{\xi})$$

$$\Longleftrightarrow \quad \inf_{q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})\in\mathcal{P}_{\text{prob}}} \mathrm{KL}(q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})\|p(\mathbf{Z},\boldsymbol{\eta},\mathbf{W},\mathcal{D})) + \mathcal{R}^c_h(q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})), \quad (12)$$

where $p(\mathbf{Z},\boldsymbol{\eta},\mathbf{W},\mathcal{D}) = \pi(\boldsymbol{\eta})\pi(\mathbf{Z})\pi(\mathbf{W})\prod_{n=1}^N p(\mathbf{x}_n|\mathbf{z}_n,\mathbf{W},\sigma_{n0}^2)$ is the joint distribution of the model; $\pi(\mathbf{Z})$ is an IBP prior; and $\pi(\boldsymbol{\eta})$ and $\pi(\mathbf{W})$ are Gaussian process priors with identity covariance functions.

Directly solving the iLSVM problems is not easy because either the posterior constraints or the non-smooth regularization function $\mathcal{R}^c$ is hard to deal with. Thus, we resort to convex duality theory, which will be useful for developing approximate inference algorithms. We can either solve the constrained form (11) using Lagrangian duality theory (Ito and Kunisch, 2008) or solve the unconstrained form (12) using Fenchel duality theory. Here, we take the second approach. In this case, the linear operator is the expectation operator, denoted by $E: \ \mathcal{P}_{\text{prob}} \to \mathbb{R}^{|\mathcal{I}_{\text{tr}}|\times L}$ and the element of Eq evaluated at y for the nth example is

$$Eq(n,y) \overset{\text{def}}{=} \Delta f\big(y,\mathbf{x}_n;q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})\big) = \mathbb{E}_{q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})}\left[\boldsymbol{\eta}^\top\Delta\mathbf{g}_n(y,\mathbf{Z})\right],$$

where $\Delta\mathbf{g}_n(y,\mathbf{Z}) \overset{\text{def}}{=} \mathbf{g}(y_n,\mathbf{x}_n,\mathbf{z}) - \mathbf{g}(y,\mathbf{x}_n,\mathbf{z})$. Then, let $g_1: \ \mathbb{R}^L \to \mathbb{R}$ be a function defined in the same form as in (8). We have

$$\mathcal{R}^c_h\big(q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})\big) = \sum_{n\in\mathcal{I}_{\text{tr}}} g_1\left(\ell_n^\Delta - Eq(n)\right),$$

where $Eq(n) \overset{\text{def}}{=} (Eq(n,1),\cdots,Eq(n,L))$ and $\ell_n^\Delta \overset{\text{def}}{=} (\ell_n^\Delta(1),\cdots,\ell_n^\Delta(L))$ are the vectors of elements evaluated for nth data. By the Fenchel's duality theorem and the results in Lemma 10, we can derive the conjugate of the problem (12). The proof is deferred to Appendix C.4.

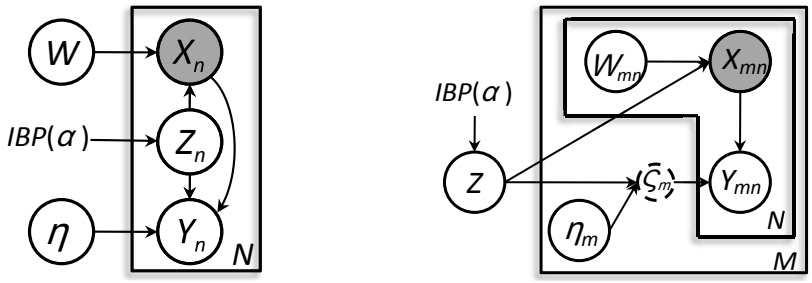
Lemma 12 (Conjugate of iLSVM) *For the iLSVM problem, we have that*

$$\inf_{q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})\in\mathcal{P}_{\text{prob}}} \mathrm{KL}\big(q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})\|p(\mathbf{Z},\boldsymbol{\eta},\mathbf{W},\mathcal{D})\big) + \mathcal{R}^c_h\big(q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})\big)$$
$$= \sup_{\boldsymbol{\omega}} \quad -\log Z(\boldsymbol{\omega}|\mathcal{D}) + \sum_{n\in\mathcal{I}_{tr}}\sum_y \omega_n^y\ell_n^\Delta(y) - \sum_n g_1^*(\boldsymbol{\omega}_n),$$

where $\boldsymbol{\omega}_n = (\omega_n^1,\cdots,\omega_n^L)$ is the subvector associated with data n. Moreover, The optimum distribution is the posterior distribution

$$\hat{q}(\mathbf{Z},\boldsymbol{\eta},\mathbf{W}) = \frac{1}{Z(\hat{\boldsymbol{\omega}}|\mathcal{D})}p(\mathbf{Z},\boldsymbol{\eta},\mathbf{W},\mathcal{D})\exp\left\{\sum_{n\in\mathcal{I}_{tr}}\sum_y \hat{\omega}_n^y\boldsymbol{\eta}^\top\Delta\mathbf{g}_n(y,Z)\right\},$$

where $Z(\hat{\boldsymbol{\omega}}|\mathcal{D})$ is the normalization factor and $\hat{\boldsymbol{\omega}}$ is the solution of the dual problem.



(a) (b)

Figure 2: Graphical structures of (a) infinite latent SVM (iLSVM); and (b) multi-task infinite latent SVM (MT-iLSVM). For MT-iLSVM, the dashed nodes (i.e., ς_m) illustrate the task relatedness but do not exist.

Testing: to make prediction on test examples, we put both training and test data together to do regularized Bayesian inference. For training data, we impose the above large-margin constraints because of the awareness of their true labels, while for test data, we do the inference without the large-margin constraints since we do not know their true labels. Therefore, the classifier $q(\boldsymbol{\eta})$ is learned from the training data only, while both training and testing data influence the posterior distributions of the likelihood model $\mathbf{W}$. After inference, we make the prediction via the rule

$$y^* \stackrel{\text{def}}{=} \operatorname*{argmax}_y f\big(y, \mathbf{x}; q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})\big). \tag{13}$$

Note that the ability to generalize to test data relies on the fact that all the data examples share $\boldsymbol{\eta}$ and the IBP prior. We can also cast the problem as a transductive inference problem by imposing additional large-margin constraints on test data (Joachims, 1999). However, the resulting problem will be generally harder to solve because it needs to resolve the unknown labels of testing examples. We also note that the testing is different from the standard inductive setting (Zhu et al., 2011b), where the latent features of a new data example can be approximately inferred given the training data. Our empirical study shows little difference on performance between our setting and the standard inductive setting.

4.3 Multi-Task Infinite Latent Support Vector Machines

Different from classification, which is typically formulated as a single learning task, multi-task learning aims to improve a set of related tasks through sharing statistical strength among these tasks, which are performed jointly. Many different approaches have been developed for multi-task learning; see Jebara (2011) for a review. In particular, learning a common latent representation shared by all the related tasks has proven to be an effective way to capture task relationships (Ando and Zhang, 2005; Argyriou et al., 2007; Rai and Daume III, 2010). Below, we present the multi-task infinite latent SVM (MT-iLSVM) for learning a common binary projection matrix $\mathbf{Z}$ to capture the relationships among multiple

tasks. Similar as in iLSVM, we also put the IBP prior on $\mathbf{Z}$ to allow it to have an unbounded number of columns.

Suppose we have M related tasks. Let $\mathcal{D}_m = \{(\mathbf{x}_{mn}, y_{mn})\}_{n \in \mathcal{I}_{\text{tr}}^m}$ be the training data for task m. We consider binary classification tasks, where $\mathcal{Y}_m = \{+1, -1\}$. Extension to multi-way classification or regression can be easily done. A naïve way to solve this learning problem with multiple tasks is to perform the multiple tasks independently. In order to make the multiple tasks coupled and share statistical strength, MT-iLSVM introduces a latent projection matrix $\mathbf{Z}$. If the latent matrix $\mathbf{Z}$ is given, we define the *latent discriminant function* for task m as

$$f_m(\mathbf{x}_{mn}, \mathbf{Z}; \boldsymbol{\eta}_m) \stackrel{\text{def}}{=} (\mathbf{Z}\boldsymbol{\eta}_m)^\top \mathbf{x}_{mn} = \boldsymbol{\eta}_m^\top (\mathbf{Z}^\top \mathbf{x}_{mn}),$$

where $\mathbf{x}_{mn}$ is one data example in $\mathcal{D}_m$ and $\boldsymbol{\eta}_m$ is the vector of parameters for task m. The dimension of $\boldsymbol{\eta}_m$ is the number of columns of the latent projection matrix $\mathbf{Z}$, which is unbounded in the nonparametric setting. This definition provides two views of how the M tasks get related.

(1) If we let $\varsigma_m = \mathbf{Z}\boldsymbol{\eta}_m$, then ς_m is the actual parameter of task m and all ς_m in different tasks are coupled by sharing the same latent matrix $\mathbf{Z}$;

(2) Another view is that each task m has its own parameters $\boldsymbol{\eta}_m$, but all the tasks share the same latent projection matrix $\mathbf{Z}$ to extract latent features $\mathbf{Z}^\top \mathbf{x}_{mn}$, which is a projection of the input features $\mathbf{x}_{mn}$.

As such, our method can be viewed as a nonparametric Bayesian treatment of alternating structure optimization (ASO) (Ando and Zhang, 2005), which learns a single projection matrix with a pre-specified latent dimension. Moreover, different from Jebara (2011), which learns a binary vector with known dimensionality to select features or kernels on $\mathbf{x}$, we learn an unbounded projection matrix $\mathbf{Z}$ using nonparametric Bayesian techniques.

As in iLSVM, we employ a Bayesian treatment of $\boldsymbol{\eta}_m$, and view it as random variables. We assume that $\boldsymbol{\eta}_m$ has a fully-factorized Gaussian prior, i.e., $\eta_{mk} \sim \mathcal{N}(0, 1)$. Then, we define the effective discriminant function for task m as the expectation

$$f_m\big(\mathbf{x}; q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})\big) \stackrel{\text{def}}{=} \mathbb{E}_{q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})}\big[f_m(\mathbf{x}, \mathbf{Z}; \boldsymbol{\eta}_m)\big] = \mathbb{E}_{q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})}[\mathbf{Z}\boldsymbol{\eta}_m]^\top \mathbf{x},$$

where $\mathbf{W}$ is a place holder for the variables that possibly arise from other parts of the model. As in iLSVM, since we are taking expectation, the variables which do not appear in the feature map (i.e., $\mathbf{W}$) will be marginalized out. Then, the prediction rule for task m is naturally $y_m^* \stackrel{\text{def}}{=} \text{sign} f_m(\mathbf{x})$. Similarly, we perform regularized Bayesian inference by defining:

$$U^{MT}(\boldsymbol{\xi}) \stackrel{\text{def}}{=} C \sum_{m, n \in \mathcal{I}_{\text{tr}}^m} \xi_{mn}$$

and imposing the following constraints:

$$\mathcal{P}_{\text{post}}^{MT}(\boldsymbol{\xi}) \stackrel{\text{def}}{=} \left\{ q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W}) \,\middle|\, \begin{array}{l} \forall m,\ \forall n \in \mathcal{I}_{\text{tr}}^m :\ y_{mn} \mathbb{E}_{q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})}[\mathbf{Z}\boldsymbol{\eta}_m]^\top \mathbf{x}_{mn} \geq 1 - \xi_{mn} \\ \qquad\qquad\qquad\qquad\quad \xi_{mn} \geq 0 \end{array} \right\}. \quad (14)$$

Finally, as in iLSVM we may also be interested in explaining observed data $\mathbf{x}$. Therefore, we relate $\mathbf{Z}$ to the observed data $\mathbf{x}$ by defining a likelihood model:

$$p\big(\mathbf{x}_{mn}|\mathbf{w}_{mn},\mathbf{Z},\lambda_{mn}^2\big)=\mathcal{N}\big(\mathbf{x}_{mn}|\mathbf{Z}\mathbf{w}_{mn},\lambda_{mn}^2 I\big),$$

where $\mathbf{w}_{mn}$ is a vector. We assume that $\mathbf{W}$ (the collection of $\mathbf{w}_{mn}$) has an independent prior $\pi(\mathbf{W})=\prod_{mn}\mathcal{N}(\mathbf{w}_{mn}|0,\sigma_{m0}^2 I)$. Fig. 2 (b) illustrates the graphical structure of MT-iLSVM.

For training, we can derive the similar convex conjugate as in the case of iLSVM. Similar as in iLSVM, minimizing $U^{MT}(\boldsymbol{\xi})$ is equivalent to minimizing the hinge-loss $\mathcal{R}_h^{MT}$ of the multiple binary prediction rules, where

$$\mathcal{R}_h^{MT}\big(q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})\big)=C\sum_{m,n\in\mathcal{I}_{\text{tr}}^m}\max\Big(0,1-y_{mn}\mathbb{E}_{q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})}[\mathbf{Z}\boldsymbol{\eta}_m]^\top\mathbf{x}_{mn}\Big).$$

Thus, the RegBayes problem of MT-iLSVM can be equivalently written as

$$\inf_{q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})}\mathrm{KL}\big(q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})\|p(\mathbf{Z},\boldsymbol{\eta},\mathbf{W},\mathcal{D})\big)+\mathcal{R}_h^{MT}\big(q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})\big). \tag{15}$$

Then, by the Fenchel's duality theorem and Lemma 9, we can derive the conjugate of MT-iLSVM. The proof is deferred to Appendix C.5.

Lemma 13 (Conjugate of MT-iLSVM) *For the MT-iLSVM problem, we have that*

$$\begin{aligned}&\inf_{q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})\in\mathcal{P}_{\text{prob}}}\ \mathrm{KL}(q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})\|p(\mathbf{Z},\boldsymbol{\eta},\mathbf{W},\mathcal{D}))+\mathcal{R}_h^{MT}(q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W}))\\ =\quad&\sup_{\boldsymbol{\omega}}\quad -\log Z'(\boldsymbol{\omega}|\mathcal{D})+\sum_{m,n}\omega_{mn}-\sum_{m,n}g_0^*(\omega_{mn}).\end{aligned}$$

Moreover, The optimum distribution is the posterior distribution

$$\hat{q}(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})=\frac{1}{Z'(\hat{\boldsymbol{\omega}}|\mathcal{D})}p(\mathbf{Z},\boldsymbol{\eta},\mathbf{W},\mathcal{D})\exp\left\{\sum_{m,n}y_{mn}\hat{\omega}_{mn}(\mathbf{Z}\boldsymbol{\eta}_m)^\top\mathbf{x}_{mn}\right\},$$

where $Z'(\hat{\boldsymbol{\omega}}|\mathcal{D})$ is the normalization factor and $\hat{\boldsymbol{\omega}}$ is the solution of the dual problem.

For testing, we use the same strategy as in iLSVM to do Bayesian inference on both training and test data. The difference is that training data are subject to large-margin constraints, while test data are not. Similarly, the hyper-parameters σ_{m0}^2 and λ_{mn}^2 can be set a priori or estimated from data (See Appendix D.1 for details).

4.4 Inference with Truncated Mean-Field Constraints

Now we discuss how to perform regularized Bayesian inference with the large-margin constraints for both iLSVM and MT-iLSVM. From the primal-dual formulations, it is obvious that there are basically two methods to perform the regularized Bayesian inference. One is to directly solve the primal problem for the posterior distribution $q(\mathbf{Z},\boldsymbol{\eta},\mathbf{W})$, and the other is to first solve the dual problem for the optimum $\hat{\boldsymbol{\omega}}$ and then infer the posterior distribution. However, both the primal and dual problems are intractable for iLSVM and

Algorithm 1 Inference Algorithm for Infinite Latent SVMs

1: **Input:** corpus $\mathcal{D}$ and constants (α, C).
2: **Output:** posterior distribution $q(\boldsymbol{\nu}, \mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})$.
3: **repeat**
4: infer $q(\boldsymbol{\nu}), q(\mathbf{W})$ and $q(\mathbf{Z})$ with $q(\boldsymbol{\eta})$ and $\boldsymbol{\omega}$ given;
5: infer $q(\boldsymbol{\eta})$ and solve for $\boldsymbol{\omega}$ with $q(\mathbf{Z})$ given.
6: **until** convergence

MT-iLSVM. The intrinsic hardness is due to the mutual dependency among the latent variables in the desired posterior distribution. Therefore, a natural approximation method is the mean field (Jordan et al., 1999), which breaks the mutual dependency by assuming that q is of some factorization form. This method approximates the original problems by imposing additional constraints. An alternative method is to apply approximate methods (e.g., MCMC sampling) to infer the true posterior distributions derived via convex conjugates as above, and iteratively estimate the dual parameters using approximate statistics (e.g., feature expectations estimated using samples) (Schofield, 2006). Below, we use MT-iLSVM as an example to illustrate the idea of the first strategy. A full discussion on the second strategy is beyond the scope of this paper. For iLSVM, the similar procedure applies and we defer its details to Appendix D.2.

To make the problem easier to solve, we use the stick-breaking representation of IBP, which includes the auxiliary variable $\boldsymbol{\nu}$, and infer the augmented posterior $q(\boldsymbol{\nu}, \mathbf{W}, \mathbf{Z}, \boldsymbol{\eta})$. The joint model distribution is now $q(\boldsymbol{\nu}, \mathbf{W}, \mathbf{Z}, \boldsymbol{\eta}, \mathcal{D})$. Furthermore, we impose the truncated mean-field constraint that

$$q(\boldsymbol{\nu}, \mathbf{W}, \mathbf{Z}, \boldsymbol{\eta}) = q(\boldsymbol{\eta}) \prod_{k=1}^{K} \Big(q(\nu_k|\boldsymbol{\gamma}_k) \prod_{d=1}^{D} q(z_{dk}|\psi_{dk}) \Big) \prod_{mn} q\Big(\mathbf{w}_{mn}|\Phi_{mn}, \sigma_{mn}^2 I\Big), \tag{16}$$

where K is the truncation level, and we assume that

$$q(\nu_k|\boldsymbol{\gamma}_k) = \text{Beta}(\gamma_{k1}, \gamma_{k2}),$$

$$q(z_{dk}|\psi_{dk}) = \text{Bernoulli}(\psi_{dk}),$$

$$q(\mathbf{w}_{mn}|\Phi_{mn}, \sigma_{mn}^2 I) = \mathcal{N}(\mathbf{w}_{mn}|\Phi_{mn}, \sigma_{mn}^2 I).$$

Then, we can use the duality theory[12] to solve the RegBayes problem by alternating between two substeps, as outlined in Algorithm 1 and detailed below.

Infer $q(\boldsymbol{\nu})$, $q(\mathbf{W})$ and $q(\mathbf{Z})$: Since $q(\boldsymbol{\nu})$ and $q(\mathbf{W})$ are not directly involved in the posterior constraints, we can solve for them by using standard Bayesian inference, i.e., minimizing a KL-divergence. Specifically, for $q(\mathbf{W})$, since the prior is also normal, we can easily derive the update rules for Φ_{mn} and σ_{mn}^2. For $q(\boldsymbol{\nu})$, we have the same update rules as in Doshi-Velez (2009). We defer the details to Appendix D.1.

12. Lagrangian duality (Ito and Kunisch, 2008) was used in Zhu et al. (2011a) to solve the constrained variational formulations, which is closely related to Fenchel duality (Magnanti, 1974) and leads to the same solutions for iLSVM and MT-iLSVM.

For $q(\mathbf{Z})$, it is directly involved in the posterior constraints. So, we need to solve it together with $q(\boldsymbol{\eta})$ using conjugate theory. However, this is intractable. Here, we adopt an alternating strategy that first infers $q(\mathbf{Z})$ with $q(\boldsymbol{\eta})$ and dual parameters $\boldsymbol{\omega}$ fixed, and then infers $q(\boldsymbol{\eta})$ and solves for $\boldsymbol{\omega}$. Specifically, since the large-margin constraints are linear of $q(\mathbf{Z})$, we can get the mean-field update equation as

$$\psi_{dk} = \frac{1}{1 + e^{-\vartheta_{dk}}},$$

where

$$\vartheta_{dk} = \sum_{j=1}^{k} \mathbb{E}_q[\log v_j] - \mathcal{L}_k^{\nu} - \sum_{mn} \frac{1}{2\lambda_{mn}^2} \Big((K\sigma_{mn}^2 + (\phi_{mn}^k)^2) \\ -2x_{mn}^d \phi_{mn}^k + 2\sum_{j \neq k} \phi_{mn}^j \phi_{mn}^k \psi_{dj} \Big) + \sum_{m,n \in \mathcal{I}_{\text{tr}}^m} y_{mn} \mathbb{E}_q[\eta_{mk}] x_{mn}^d,$$

and $\mathcal{L}_k^{\nu}$ is an lower bound of $\mathbb{E}_q[\log(1 - \prod_{j=1}^{k} v_j)]$ (See Appendix D.1 for details). The last term of ϑ_{dk} is due to the large-margin posterior constraints as defined in (14). Therefore, from this equation we can see how the large-margin constraints regularize the procedure of inferring the latent matrix $\mathbf{Z}$.

Infer $q(\boldsymbol{\eta})$ and solve for $\boldsymbol{\omega}$: Now, we can apply the convex conjugate theory and show that the optimum posterior distribution of $\boldsymbol{\eta}$ is

$$q(\boldsymbol{\eta}) = \prod_m q(\boldsymbol{\eta}_m), \text{ where } q(\boldsymbol{\eta}_m) \propto \pi(\boldsymbol{\eta}_m) \exp\left\{ \boldsymbol{\eta}_m^{\top} \boldsymbol{\mu}_m \right\},$$

and $\boldsymbol{\mu}_m = \sum_{n \in \mathcal{I}_{\text{tr}}^m} y_{mn} \omega_{mn} (\boldsymbol{\psi}^{\top} \mathbf{x}_{mn})$. Here, we assume $\pi(\boldsymbol{\eta}_m)$ is standard normal. Then, we have $q(\boldsymbol{\eta}_m) = \mathcal{N}(\boldsymbol{\eta}_m | \boldsymbol{\mu}_m, I)$ and the optimum dual parameters can be obtained by solving the following M independent dual problems

$$\sup_{\boldsymbol{\omega}_m} \; -\frac{1}{2} \boldsymbol{\mu}_m^{\top} \boldsymbol{\mu}_m + \sum_{n \in \mathcal{I}_{\text{tr}}^m} \omega_{mn}$$
$$\forall n \in \mathcal{I}_{\text{tr}}^m, \text{ s.t. : } \; 0 \leq \omega_{mn} \leq C,$$

where the constraints are from the conjugate function g_0^* in Lemma 13. These dual problems (or their primal forms) can be efficiently solved with a binary SVM solver, such as SVM-light or LibSVM.

5. Experiments

We present empirical results for both classification and multi-task learning. Our results appear to demonstrate the merits inherited from both Bayesian nonparametrics and large-margin learning.

5.1 Multi-way Classification

We evaluate the infinite latent SVM (iLSVM) for classification on the real TRECVID2003 and Flickr image data sets, which have been extensively evaluated in the context of learning

Model	TRECVID2003		Flickr	
	Accuracy	F1 score	Accuracy	F1 score
EFH+SVM	0.565 ± 0.0	0.427 ± 0.0	0.476 ± 0.0	0.461 ± 0.0
MMH	**0.566** ± 0.0	0.430 ± 0.0	**0.538** ± 0.0	**0.512** ± 0.0
IBP+SVM	0.553 ± 0.013	0.397 ± 0.030	0.500 ± 0.004	0.477 ± 0.009
iLSVM	0.563 ± 0.010	**0.448** ± 0.011	0.533 ± 0.005	0.510 ± 0.010

Table 1: Classification accuracy and F1 scores on the TRECVID2003 and Flickr image data sets (Note: MMH and EFH have zero std because of their deterministic initialization).

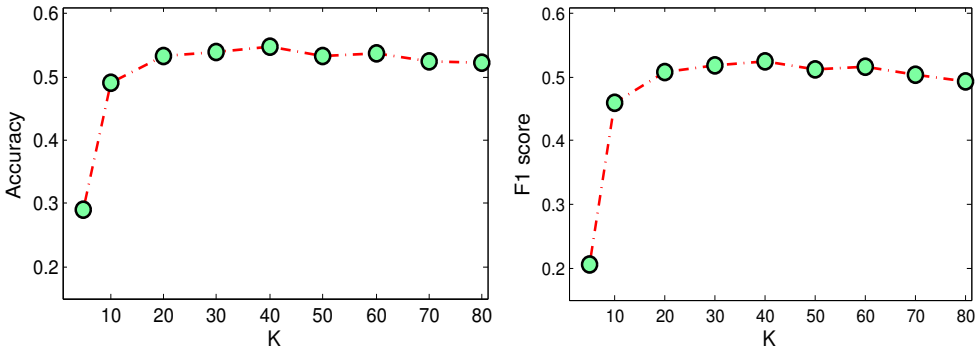


Figure 3: Accuracy and F1 score of MMH on the Flickr data set with different numbers of latent features.

finite latent feature models (Chen et al., 2012). TRECVID2003 consists of 1078 video keyframes that belong to 5 categories, including *Airplane scene*, *Basketball scene*, *Weather news*, *Baseball scene*, and *Hockey scene*. Each data example has two types of features, including a 1894-dimension binary vector of text features and a 165-dimension HSV color histogram. The Flickr image data set consists of 3411 natural scene images about 13 types of animals, including *squirrel, cow, cat, zebra, tiger, lion, elephant, whales, rabbit, snake, antlers, hawk and wolf*, downloaded from the Flickr website.[13] Also, each example has two types of features, including 500-dimension SIFT bag-of-words and 634-dimension real-valued features (e.g., color histogram, edge direction histogram, and block-wise color moments). Here, we consider the real-valued features only by defining Gaussian likelihood distributions for $\mathbf{x}$; and we define the discriminant function using latent features only as in (9). We follow the same training/testing splits as in Chen et al. (2012).

We compare iLSVM with the large-margin Harmonium (MMH) (Chen et al., 2012), which was shown to outperform many other latent feature models, and two decoupled approaches of *EFH+SVM* and *IBP+SVM*. EFH+SVM uses the exponential family Harmo-

13. The website is available at: `http://www.flickr.com/`.

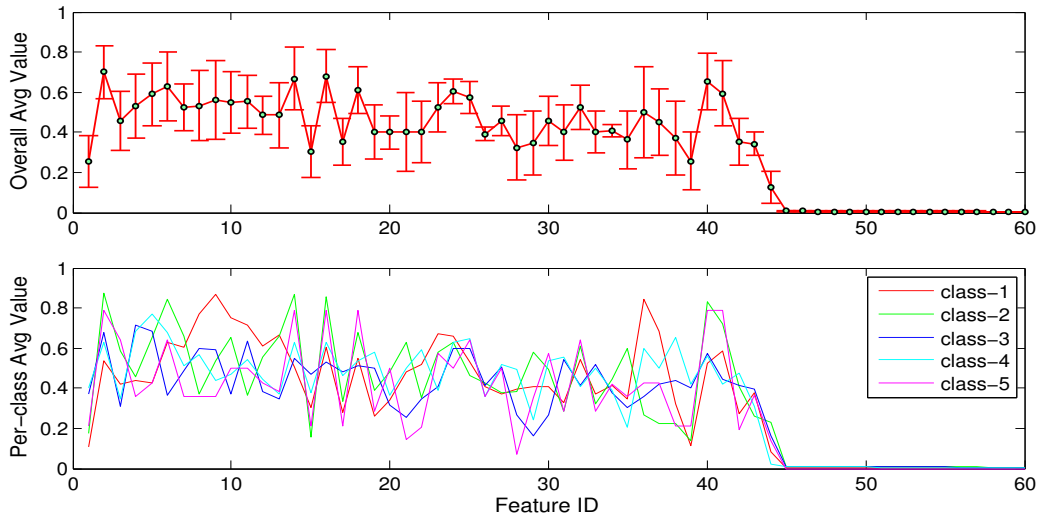


Figure 4: (Up) the overall average values of the latent features with standard deviation over different classes; and (Bottom) the per-class average values of latent features learned by iLSVM on the TRECVID data set.

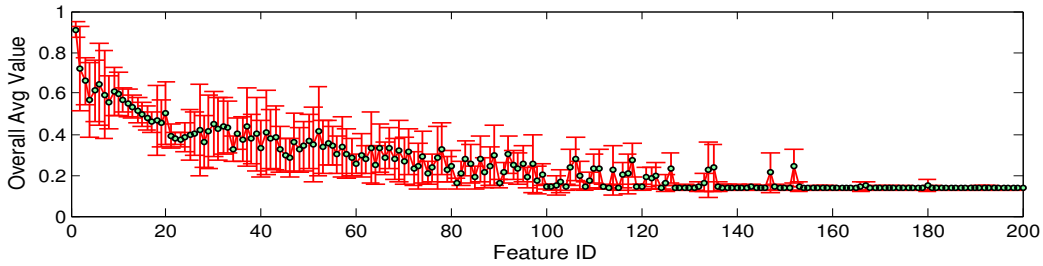


Figure 5: The overall average values of the latent features with standard deviation over different classes on the Flickr data set.

nium (EFH) (Welling et al., 2004) to discover latent features and then learns a multi-way SVM classifier. IBP+SVM is similar, but uses an IBP factor analysis model to discover latent features (Griffiths and Ghahramani, 2005). To initialize the learning algorithms for these models, we found that using the SVD factors of the input feature matrix as the initial weights for MMH and EFH can produce better results. Here, we also use the SVD factors as the initial mean of weights in the likelihood models for iLSVM. Both MMH and EFH+SVM are finite models and they need to pre-specify the dimensionality of latent features. We report their results on classification accuracy and F1 score (i.e., the average F1 score over all possible classes) (Zhu et al., 2011b) achieved with the best dimensionality in Table 1. Figure 3 illustrates the performance change of MMH when using different number of latent features, from which we can see that $K = 40$ produces the best performance and either increasing or decreasing K could make the performance worse. For

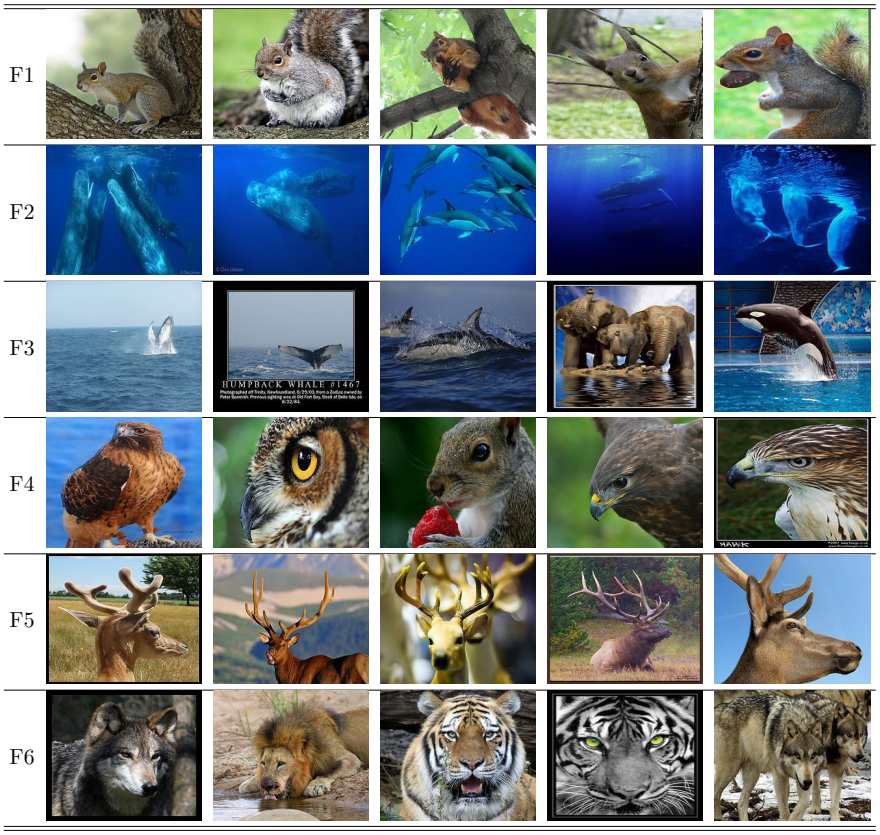


Figure 6: Six example features discovered iLSVM on the Flickr animal data set. For each feature, we show 5 top-ranked images.

iLSVM and IBP+SVM, we use the mean-field inference method and present the average performance with 5 randomly initialized runs (Please see Appendix D.2 for the algorithm and initialization details). We perform 5-fold cross-validation on training data to select hyperparameters, e.g., α and C (we use the same procedure for MT-iLSVM). We can see that iLSVM can achieve comparable performance with the nearly optimal MMH, without needing to pre-specify the latent feature dimension,[14] and is much better than the decoupled approaches (i.e., IBP+SVM and EFH+SVM). For the two stage methods, we don't have a clear winner—IBP+SVM performs a bit worse than EFH+SVM on the TRECVID data set, while it outperforms EFH+SVM on the flickr data set. The reason for the difference may be due to the initialization or different properties of the data.

It is also interesting to examine the discovered latent features. Figure 4 shows the overall average values of latent features and the per-class average feature values of iLSVM in one run on the TRECVID data set. We can see that on average only about 45 features are active for the TRECVID data set. For the overall average, we also present the standard deviation over the 5 categories. A larger deviation means that the corresponding feature is more discriminative when predicting different categories. For example, feature 26 and feature 34 are generally less discriminative than many other features, such as feature 1

14. We set the truncation level to 300, which is sufficiently large.

and feature 30. Figure 5 shows the overall average feature values together with standard deviation on the Flickr data set. We omitted the per-class average because that figure is too crowded with 13 categories. We can that as k increases, the probability that feature k is active decreases. The reason for the features with stable values (i.e., standard deviations are extremely small) is due to our initialization strategy (each feature has 0.5 probability to be active). Initializing ψ_{dk} as being exponentially decreasing (e.g., like the constructing process of $\boldsymbol{\pi}$) leads to a faster decay and many features will be inactive. To examine the semantics of each feature,[15] Figure 6 presents some example features discovered on the Flickr animal data set. For each feature, we present 5 top-ranked images which have large values on this particular feature. We can see that most of the features are semantically interpretable. For instance, feature F1 is about squirrel; feature F2 is about ocean animal, which is whales in the Flickr data set; and feature F4 is about hawk. We can also see that some features are about different aspects of the same category. For example, feature F2 and feature F3 are both about whales, but with different background.

5.2 Multi-task Learning

Now, we evaluate the multi-task infinite latent SVM (MT-iLSVM) on several well-studied real data sets.

5.2.1 Description of the Data

Scene and Yeast Data: These data sets are from the UCI repository, and each data example has multiple labels. As in Rai and Daume III (2010), we treat the multi-label classification as a multi-task learning problem, where each label assignment is treated as a binary classification task. The Yeast data set consists of 1500 training and 917 test examples, each having 103 features, and the number of labels (or tasks) per example is 14. The Scene data set consists 1211 training and 1196 test examples, each having 294 features, and the number of labels (or tasks) per example for this data set is 6.

School Data: This data set comes from the Inner London Education Authority and has been used to study the effectiveness of schools. It consists of examination records of 15,362 students from 139 secondary schools in years 1985, 1986 and 1987. The data set is publicly available and has been extensively evaluated in various multi-task learning methods (Bakker and Heskes, 2003; Bonilla et al., 2008; Zhang and Yeung, 2010), where each task is defined as predicting the exam scores of students belonging to a specific school based on four student-dependent features (year of the exam, gender, VR band and ethnic group) and four school-dependent features (percentage of students eligible for free school meals, percentage of students in VR band 1, school gender and school denomination). In order to compare with the above methods, we follow the same setup described by Argyriou et al. (2007) and Bakker and Heskes (2003) and similarly we create dummy variables for those features that are categorical forming a total of 19 student-dependent features and 8 school-dependent features. We use the same 10 random splits[16] of the data, so that 75% of the examples from each school (task) belong to the training set and 25% to the test set.

15. The interpretation of latent features depends heavily on the input data.
16. The splits are available at: `http://ttic.uchicago.edu/~argyriou/code/index.html`.

Data set	Model	Acc	F1-Micro	F1-Macro
Yeast	YaXue	0.5106	0.3897	0.4022
	Piyushrai-1	0.5212	0.3631	0.3901
	Piyushrai-2	0.5424	0.3946	0.4112
	MT-IBP+SVM	0.5475 ± 0.005	0.3910 ± 0.006	0.4345 ± 0.007
	MT-iLSVM	**0.5792** ± 0.003	**0.4258** ± 0.005	**0.4742** ± 0.008
Scene	YaXue	0.7765	0.2669	0.2816
	Piyushrai-1	0.7756	0.3153	0.3242
	Piyushrai-2	0.7911	0.3214	0.3226
	MT-IBP+SVM	0.8590 ± 0.002	0.4880 ± 0.012	0.5147 ± 0.018
	MT-iLSVM	**0.8752** ± 0.004	**0.5834** ± 0.026	**0.6148** ± 0.020

Table 2: Multi-label classification performance on Scene and Yeast data sets.

On average, the training set includes about 80 students per school and the test set about 30 students per school.

5.2.2 Results

Scene and Yeast Data: We compare with the closely related nonparametric Bayesian methods, including kernel stick-breaking (YaXue) (Xue et al., 2007) and the basic and augmented infinite predictor subspace models (i.e., Piyushrai-1 and Piyushrai-2) proposed by Rai and Daume III (2010). These nonparametric Bayesian models were shown to outperform the independent Bayesian logistic regression and a single-task pooling approach in previous work (Rai and Daume III, 2010). We also compare with a decoupled method *MT-IBP+SVM* that uses an IBP factor analysis model to find shared latent features among multiple tasks and then builds separate SVM classifiers for different tasks.[17] For MT-iLSVM and MT-IBP+SVM, we use the mean-field inference method in Sec 4.4 and report the average performance with 5 randomly initialized runs (See Appendix D.1 for initialization details). For comparison with Rai and Daume III (2010) and Xue et al. (2007), we use the overall classification accuracy, F1-Macro and F1-Micro as performance measures. Table 2 shows the results. On both data sets, MT-iLSVM needs less than 50 latent features on average. We can see that the large-margin MT-iLSVM performs much better than other nonparametric Bayesian methods and MT-IBP+SVM, which separates the inference of latent features from learning the classifiers.

School Data: We use the percentage of explained variance (Bakker and Heskes, 2003) as the measure of the regression performance, which is defined as the total variance of the data minus the sum-squared error on the test set as a percentage of the total variance. Since we use the same settings, we can compare with the state-of-the-art results of

(1) Bayesian multi-task learning (BMTL) (Bakker and Heskes, 2003);

(2) Multi-task Gaussian processes (MTGP) (Bonilla et al., 2008);

17. This decoupled approach is in fact an one-iteration MT-iLSVM, where we first infer the shared latent matrix $\mathbf{Z}$ and then learn an SVM classifier for each task.

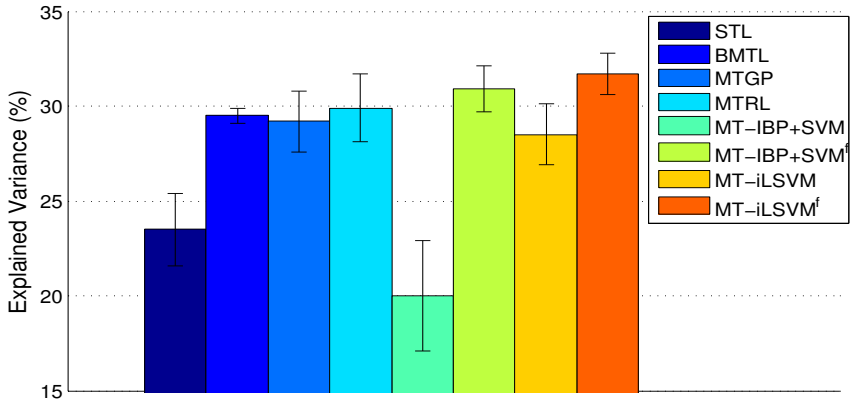


Figure 7: Percentage of explained variance by various models on the School data set.

(3) Convex multi-task relationship learning (MTRL) (Zhang and Yeung, 2010);

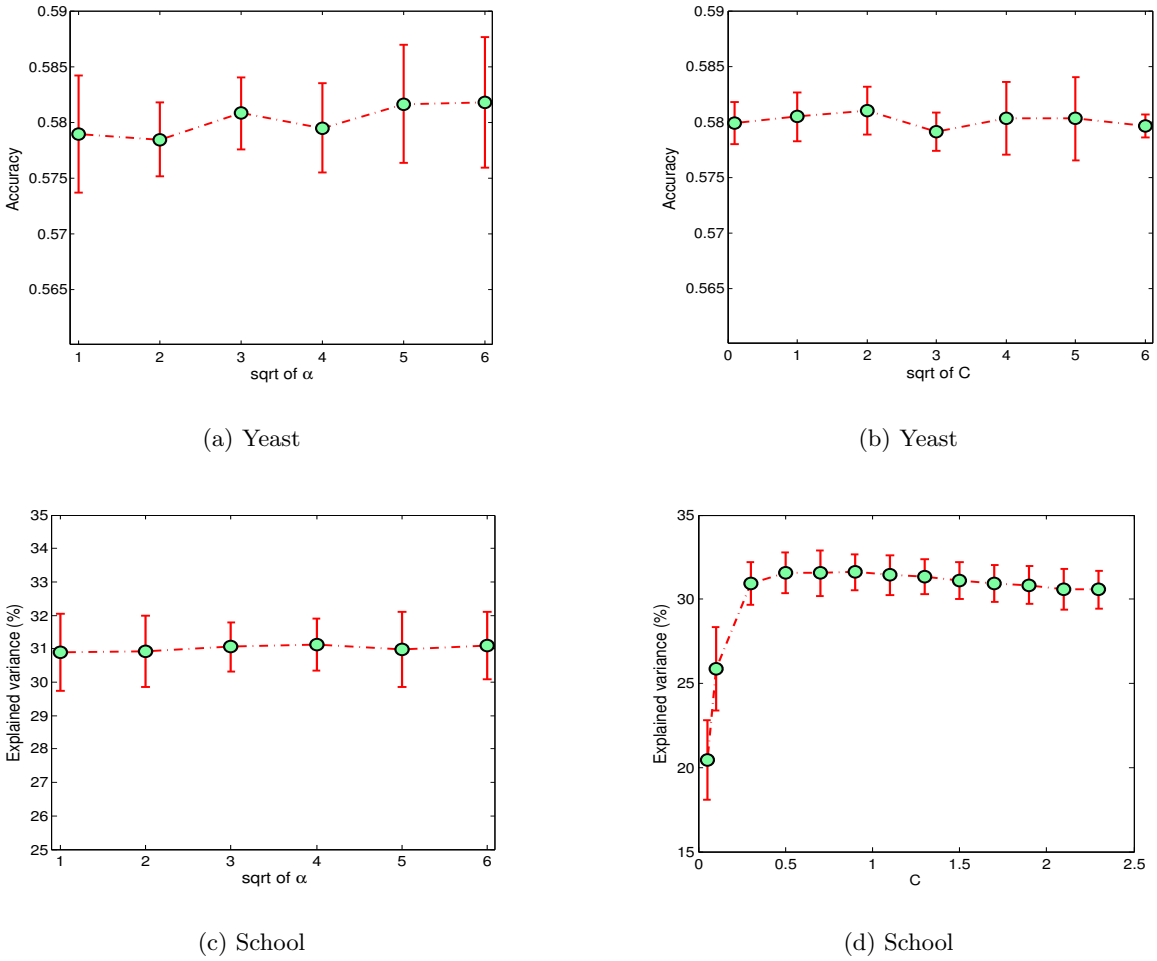
and single-task learning (STL) as reported in Bonilla et al. (2008) and Zhang and Yeung (2010). For MT-iLSVM and MT-IBP+SVM, we also report the results achieved by using both the latent features (i.e., $\mathbf{Z}^\top \mathbf{x}$) and the original input features $\mathbf{x}$ through vector concatenation, and we denote the corresponding methods by *MT-iLSVM*^{f} and *MT-IBP+SVM*^{f}, respectively. On average the multi-task latent SVM (i.e., MT-iLSVM) needs about 50 latent features to get sufficiently good and robust performance. From the results in Figure 7, we can see that the MT-iLSVM achieves better results than the existing methods that have been tested in previous studies. Again, the joint MT-iLSVM performs much better than the decoupled method MT-IBP+SVM, which separates the latent feature inference from the training of large-margin classifiers. Finally, using both latent features and the original input features can boost the performance slightly for MT-iLSVM, while much more significantly for the decoupled MT-IBP+SVM.

5.3 Sensitivity Analysis

Figure 8 shows how the performance of MT-iLSVM changes against the hyper-parameter α and regularization constant C on the Yeast and School data sets. We can see that on the Yeast data set, MT-iLSVM is insensitive to both α and C. For the School data set, MT-iLSVM is very insensitive the α, and it is stable when C is set between 0.3 and 1.

Figure 9 shows how the training size affects the performance and running time of MT-iLSVM on the School data set. We use the first $b\%$ ($b = 50, 60, 70, 80, 90, 100$) of the training data in each of the 10 random splits as training set and use the corresponding test data as test set. We can see that as training size increases, the performance and running time generally increase; and MT-iLSVM achieves the state-of-art performance when using about 70% training data. From the running time, we can also see that MT-iLSVM is generally quite efficient by using mean-field inference.

Finally, we investigate how the performance of MT-iLSVM changes against the hyper-parameters σ_{m0}^2 and λ_{mn}^2. We initially set $\sigma_{m0}^2 = 1$ and compute λ_{mn}^2 from observed data. If we further estimate them by maximizing the objective function, the performance does



(a) Yeast

(b) Yeast

(c) School

(d) School

Figure 8: Sensitivity study of MT-iLSVM: (a) classification accuracy with different α on Yeast data; (b) classification accuracy with different C on Yeast data; (c) percentage of explained variance with different α on School data; and (d) percentage of explained variance with different C on School data.

not change much ($\pm 0.3\%$ for average explained variance on the School data set). We have similar observations for iLSVM.

6. Conclusions and Discussions

We present regularized Bayesian inference (RegBayes), a computational framework to perform post-data posterior inference with a rich set of regularization/constraints on the desired post-data posterior distributions. RegBayes is formulated as a information-theoretical optimization problem, and it is applicable to both directed and undirected graphical models. We present a general theorem to characterize the solution of RegBayes, when the posterior

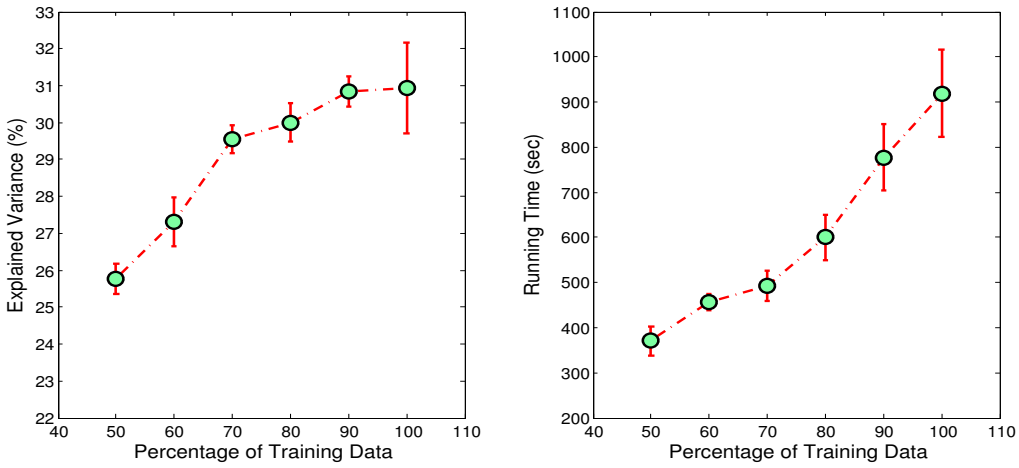


Figure 9: Percentage of explained variance and running time by MT-iLSVM with various training sizes.

regularization is induced from a linear operator (e.g., expectation). Furthermore, we particularly concentrate on developing two large-margin nonparametric Bayesian models under the RegBayes framework to learn predictive latent features for classification and multi-task learning, by exploring the large-margin principle to define posterior constraints. Both models allow the latent dimension to be automatically resolved from the data. The empirical results on several real data sets appear to demonstrate that our methods inherit the merits from both Bayesian nonparametrics and large-margin learning.

RegBayes offers a flexible framework for considering posterior regularization in performing parametric or nonparametric Bayesian inference. For future work, we plan to study other posterior regularization beyond the large-margin constraints, such as posterior constraints defined on manifold structures (Huh and Fienberg, 2012) and those represented in the form of first-order logic, and investigate how posterior regularization can be used in other interesting nonparametric Bayesian models (Beal et al., 2002; Teh et al., 2006; Blei and Frazier, 2010) in different contexts, such as link prediction (Miller et al., 2009) for social network analysis and low-rank matrix factorization for collaborative prediction. Some of our preliminary results (Xu et al., 2012; Zhu, 2012; Mei et al., 2014) have shown great promise. It is interesting to investigate more carefully along this direction. Moreover, as we have stated, RegBayes can be developed for undirected MRFs. But the inference would be even harder. We plan to do a systematic investigation along this direction too. We have some preliminary results presented in Chen et al. (2014), but there is a lot of room to further improve. Finally, regularized Bayesian inference in general leads to a highly nontrivial inference problem. Although the general solution can be derived with convex analysis theory, it is normally intractable to infer them directly. Therefore, approximate inference techniques such as the truncated mean-field approximation have to be used. For the current truncated inference methods, one key limit is to pre-specify the truncation level. A too conservative truncation level could lead to a waste of computing resources. So, it is important to develop

inference algorithms that could adaptively determine the number of latent features, such as Monte Carlo methods. We have some preliminary progress along this direction as reported in Jiang et al. (2012) and Zhu et al. (2013). It is interesting to extend these techniques to deal with other challenging nonparametric Bayesian models.

Acknowledgements

We thank the anonymous reviewers and editors for many helpful comments to improve the manuscript. NC and JZ are supported by National Key Foundation R&D Projects (No.s 2013CB329403, 2012CB316301), National Natural Science Foundation of China (Nos. 61322308, 61332007, 61305066), Tsinghua University Initiative Scientific Research Program (No. 20121088071), a Microsoft Research Asia Research Fund (No. 20123000007), and the China Postdoctoral Science Foundation Grant (No. 2013T60117) to NC. EX is supported by AFOSR FA95501010247, ONR N000140910758, NSF Career DBI-0546594 and an Alfred P. Sloan Research Fellowship.

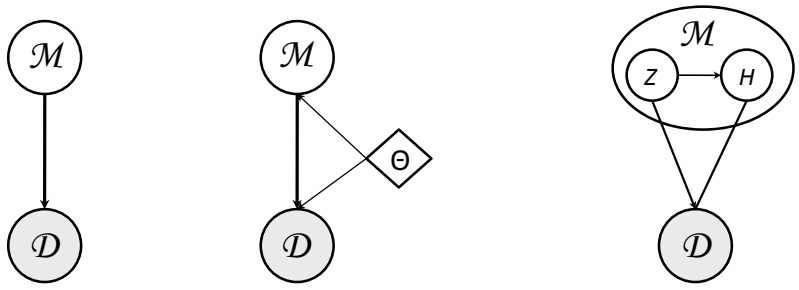
Appendix A. Generalization Beyond Bayesian Networks

Standard Bayesian inference and the proposed regularized Bayesian inference implicitly make the assumption that the model can be graphically drawn as a Bayesian network as illustrated in Figure 10(a).[18] Here, we consider a more general formulation which could cover both directed and undirected latent variable models, such as the well-studied Boltzmann machines (Murray and Ghahramani, 2004; Welling et al., 2004), as well as the case where a model could have some unknown parameters (e.g., hyper-parameters) and need an estimation procedure, such as maximum likelihood estimation (MLE), besides posterior inference. The latter is also known as empirical Bayesian methods, which are frequently employed by practitioners.

Extension 1: Empirical Bayesian Inference with Unknown Parameters: As illustrated in Figure 10(b), in some cases we need to perform the empirical Bayesian inference in the presence of unknown parameters. For instance, in a linear-Gaussian Bayesian model, we may choose to estimate its covariance matrix using MLE; and in a latent Dirichlet allocation (LDA) (Blei et al., 2003) model, we may choose to estimate the unknown topical dictionary, although in principle we can treat these parameters as random variables and perform full Bayesian inference. In such cases, we need some mechanisms to estimate the unknown parameters when doing Bayesian inference. Let Θ be model parameters. We can formulate empirical Bayesian inference as solving[19]

$$\begin{aligned} &\inf_{\Theta, q(\mathbf{M})} \mathrm{KL}(q(\mathbf{M})\|\pi(\mathbf{M})) - \int_{\mathcal{M}} \log p(\mathcal{D}|\mathbf{M}, \Theta) q(\mathbf{M}) d\mathbf{M} \\ &\text{s.t.} : q(\mathbf{M}) \in \mathcal{P}_{\text{prob}}. \end{aligned} \tag{17}$$

18. The structure within $\mathbf{M}$ can be arbitrary, either a directed, undirected or hybrid chain graph.
19. The objective can be derived using variational techniques. It is in fact a variational upper bound of the negative log-likelihood.



(a) (b) (c)

Figure 10: Illustration graphs for three different types of models that involve Bayesian inference: (a) a Bayesian generative model; (b) a Bayesian generative model with unknown parameters Θ; and (c) a chain graph model.

Although the problem is convex over $q(\mathbf{M})$ for any fixed Θ, it is not jointly convex in general. A natural algorithm to solve this problem is the well-known EM procedure (Dempster et al., 1977), which converges to a local optimum. Specifically, we have the following result.

Lemma 14 *For problem (17), the optimum solution of $q(\mathbf{M})$ is equivalent to the posterior distribution by Bayes' theorem for any Θ; and the optimum Θ^* is the MLE*

$$\Theta^* = \operatorname*{argmax}_{\Theta} \log p(\mathcal{D}|\Theta).$$

Proof According to the variational formulation of Bayes' rule in (4), we get that the optimum solution is $q(\mathbf{M}) = p(\mathbf{M}|\mathcal{D}, \Theta)$ for any Θ. Substituting the optimum solution of q into the objective, we get the optimization problem of Θ. ■

Extension 2: Chain Graph: In the above cases, we have assumed that the observed data are generated by some model in a directed causal sense. This assumption holds in directed latent variable models. However, in many cases, we may choose alternative formulations to define the joint distribution of a model and the observed data. Figure 10(c) illustrates one such scenario, where the model $\mathbf{M}$ consists of two subsets of random variables. One subset H is connected to the observed data via an undirected graph and the other subset Z is connected to the observed data and H using directed edges. This graph is known as a chain graph. Due to the Markov properties of chain graph (Frydenberg, 1990), we know that the joint distribution has the factorization form as

$$p(\mathbf{M}, \mathcal{D}) = p(Z)p(H, \mathcal{D}|Z), \tag{18}$$

where $p(H, \mathcal{D}|Z)$ is a Markov random field (MRF). One concrete example of such a hybrid chain model is the Bayesian Boltzman machines (Murray and Ghahramani, 2004), which

treat the parameters of a Boltzmann machine as random variables and perform Bayesian inference with MCMC sampling methods.

The insights that RegBayes covers undirected or chain graph latent variable models come from the observation that the objective $\mathcal{L}(q(\mathbf{M}))$ of problem (4) is in fact an KL-divergence, namely, we can show that

$$\mathcal{L}(q(\mathbf{M})) = \mathrm{KL}(q(\mathbf{M})\|p(\mathbf{M}, \mathcal{D})), \tag{19}$$

where $p(\mathbf{M}, \mathcal{D})$ is the joint distribution. Note that when $\mathcal{D}$ is given, the distribution $p(\mathbf{M}, \mathcal{D})$ is non-normalized for $\mathbf{M}$; and we have abused the KL notation for non-normalized distributions in (19), but with the same formula. For directed Bayesian networks (Zhu et al., 2011a), we naturally have $p(\mathbf{M}, \mathcal{D}) = \pi(\mathbf{M})p(\mathcal{D}|\mathbf{M})$. For the undirected MRF models, we have $\mathbf{M} = \{Z, H\}$ and again we can define the joint distribution as in (18).

Putting the above two extensions of Bayesian inference together, the regularized Bayesian inference with estimating unknown model parameters can be generally formulated as

$$\begin{aligned} &\inf_{\Theta, q(\mathbf{M}), \boldsymbol{\xi}} \mathcal{L}(\Theta, q(\mathbf{M})) + U(\boldsymbol{\xi}) \quad \text{or} \quad \inf_{\Theta, q(\mathbf{M})} \mathcal{L}(\Theta, q(\mathbf{M})) + g(Eq(\mathbf{M})) \\ &\text{s.t.}: q(\mathbf{M}) \in \mathcal{P}_{\text{post}}(\boldsymbol{\xi}) \qquad\qquad \text{s.t.}: \ q(\mathbf{M}) \in \mathcal{P}_{\text{prob}}, \end{aligned} \tag{20}$$

where $\mathcal{L}(\Theta, q(\mathbf{M}))$ is the objective function of problem (17). These two formulations are equivalent. We will call the former a *constrained* formulation and call the latter an *unconstrained* formulation by ignoring the standard normalization constraints, which are easy to deal with.

Appendix B. MedLDA—A RegBayes Model with Finite Latent Features

This section presents a new interpretation of MedLDA (maximum entropy discrimination latent Dirichlet allocation) (Zhu et al., 2009) under the framework of regularized Bayesian inference. MedLDA is a max-margin supervised topic model, an extension of latent Dirichlet allocation (LDA) (Blei et al., 2003) for supervised learning tasks. In MedLDA, each data example is projected to a point in a finite dimensional latent space, of which each feature corresponds to a topic, i.e., a unigram distribution over the terms in a vocabulary. MedLDA represents each data as a probability distribution over the features, which results in a conservation constraint (i.e., the more a data expresses on one feature, the less it can express others) (Griffiths and Ghahramani, 2005). The infinite latent feature models discussed in Section 4 do not have such a constraint.

Without loss of generality, we consider the MedLDA regression model as an example (classification model is similar), whose graphical structure is shown in Figure 11. We assume that all data examples have the same length V for notation simplicity. Each document is associated with a response variable Y, which is observed in the training phase but unobserved in testing. We will use y to denote an instance value of Y. Let K be the number of topics or the dimensionality of the latent topic space. MedLDA builds an LDA model to describe the observed words. The generating process of LDA is that each document n has a mixing proportion $\boldsymbol{\theta}_n \sim \text{Dirichlet}(\boldsymbol{\alpha})$; each word w_{nm} is associated with a topic $z_{nm} \sim \boldsymbol{\theta}_n$, which indexes the topic that generates the word, i.e., $w_{nm} \sim \boldsymbol{\beta}_{z_{nm}}$. Define $\bar{Z}_n = \frac{1}{V}\sum_{m=1}^{V} Z_{nm}$ as the average topic assignment for document n. Let $\Theta = \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \delta^2\}$ denote the unknown

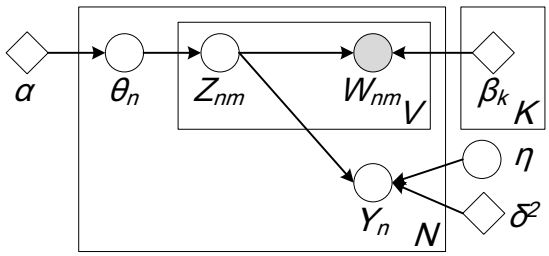


Figure 11: Graphical structure of MedLDA.

model parameters and $\mathcal{D} = \{y_n, w_{nm}\}$ be the training set. MedLDA was defined as solving a regularized MLE problem with expectation constraints

$$\begin{aligned} \inf_{\Theta, \boldsymbol{\xi}, \boldsymbol{\xi}^*} \quad & -\log p(\{y_n, w_{nm}\}|\Theta) + C \sum_{n=1}^{N} (\xi_n + \xi_n^*) \\ \text{s.t. } \forall n: \quad & \begin{cases} y_n - \mathbb{E}_p[\boldsymbol{\eta}^\top \bar{Z}_n] \leq \epsilon + \xi_n \\ -y_n + \mathbb{E}_p[\boldsymbol{\eta}^\top \bar{Z}_n] \leq \epsilon + \xi_n^* \\ \xi_n,\ \xi_n^* \geq 0 \end{cases} \end{aligned}$$

The posterior constraints are imposed following the large-margin principle and they correspond to a quality measure of the prediction results on training data. In fact, it is easy to show that minimizing $U(\boldsymbol{\xi}, \boldsymbol{\xi}^*) = C \sum_{n=1}^{N} (\xi_n + \xi_n^*)$ under the above constraints is equivalent to minimizing an ϵ-insensitive loss (Smola and Schölkopf, 2003)

$$\mathcal{R}_\epsilon\Big(p(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}|\mathcal{D}, \Theta)\Big) = C \sum_{n=1}^{N} \max\left(0, \left|y_n - \mathbb{E}_p\left[\boldsymbol{\eta}^\top \bar{Z}_n\right]\right| - \epsilon\right).$$

of the expected linear prediction rule $\hat{y}_n = \mathbb{E}_p[\boldsymbol{\eta}^\top \bar{Z}_n]$.

To practically learn an MedLDA model, since the above problem is intractable, variational methods were used by introducing an auxiliary distribution $q(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}|\Theta)$ to approximate the true posterior $p(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}|\mathcal{D}, \Theta)$,[20] replacing the negative data likelihood with its upper bound $\mathcal{L}\big(q(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}|\Theta)\big)$, and replacing p by q in the constraints. The variational MedLDA regression model is

$$\begin{aligned} \inf_{q, \Theta, \boldsymbol{\xi}, \boldsymbol{\xi}^*} \quad & \mathcal{L}\Big(q(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}|\Theta)\Big) + C \sum_{n=1}^{N} (\xi_n + \xi_n^*) \\ \text{s.t. } \forall n: \quad & \begin{cases} y_n - \mathbb{E}_q[\boldsymbol{\eta}^\top \bar{Z}_n] \leq \epsilon + \xi_n \\ -y_n + \mathbb{E}_q[\boldsymbol{\eta}^\top \bar{Z}_n] \leq \epsilon + \xi_n^* \\ \xi_n,\ \xi_n^* \geq 0 \end{cases} \end{aligned}$$

where $\mathcal{L}\big(q(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}|\Theta)\big) = -\mathbb{E}_q\big[\log p(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}, \mathcal{D}|\Theta)\big] - \mathcal{H}\big(q(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}|\Theta)\big)$ is a variational upper-bound of the negative data log-likelihood. The upper bound is tight if no

20. We have explicitly written the condition on model parameters.

restricting constraints are made on the variational distribution q. In practice, additional assumptions (e.g., mean-field) can be made on q to derive a practical approximate algorithm.

Based on the previous discussions on the extensions of RegBayes and the duality in Lemma 14, we can reformulate the MedLDA regression model as an example of RegBayes. Specifically, for the MedLDA regression model, we have $\mathbf{M} = \{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}$. According to (19), we can easily show that

$$\begin{aligned}\mathcal{L}\Big(q(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}|\Theta)\Big) &= \mathrm{KL}\Big(q(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}|\Theta) \| p(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}, \{w_{nm}, y_n\}|\Theta)\Big) \\ &= \mathcal{L}_B\Big(\Theta, q(\mathbf{M}|\Theta)\Big).\end{aligned}$$

Then, the MedLDA problem is a RegBayes model in (20) with

$$\mathcal{P}_{\text{post}}^{\text{MedLDA}}(\Theta, \boldsymbol{\xi}, \boldsymbol{\xi}^*) \stackrel{\text{def}}{=} \left\{ q(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}|\Theta) \left| \begin{array}{rc} \forall n: & y_n - \mathbb{E}_q[\boldsymbol{\eta}^\top \bar{Z}_n] \le \epsilon + \xi_n \\ & -y_n + \mathbb{E}_q[\boldsymbol{\eta}^\top \bar{Z}_n] \le \epsilon + \xi_n^* \\ & \xi_n,\ \xi_n^* \ge 0 \end{array} \right. \right\}. \tag{21}$$

For the MedLDA problem, we can use Lagrangian methods to solve the constrained formulation. Alternatively, we can use the convex duality theorem to solve an equivalent unconstrained form. For the variational MedLDA, the ϵ-insensitive loss is $\mathcal{R}_\epsilon(q(\{\theta_n, z_{nm}, \boldsymbol{\eta}\}|\Theta))$. Its conjugate can be derived using the results of Lemma 11. Specifically, we have the following result, whose proof is deferred to Appendix C.6.

Lemma 15 (Conjugate of MedLDA) *For the variational MedLDA problem, we have*

$$\begin{aligned}&\inf_{\Theta, q(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}|\Theta) \in \mathcal{P}_{\text{prob}}} \mathcal{L}\left(q(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}|\Theta), \Theta\right) + \mathcal{R}_\epsilon\left(q(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}|\Theta)\right) \\ =& \sup_{\boldsymbol{\omega}} -\log Z'(\boldsymbol{\omega}, \Theta^*) - \sum_n g_2^*(\boldsymbol{\omega}_n; -y_n + \epsilon, y_n + \epsilon),\end{aligned}$$

where $\boldsymbol{\omega}_n = (\omega_n, \omega_n')$. Moreover, The optimum distribution is the posterior distribution

$$\hat{q}(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}|\Theta^*) = \frac{1}{Z'(\hat{\boldsymbol{\omega}}, \Theta^*|\mathcal{D})} p\left(\{\boldsymbol{\theta}_n, z_{nm}, \boldsymbol{\eta}\}, \mathcal{D}|\Theta^*\right) \exp\left\{\sum_n (\hat{\omega}_n - \hat{\omega}_n') \boldsymbol{\eta}^\top \bar{z}_n\right\}, \tag{22}$$

where $Z'(\hat{\boldsymbol{\omega}}, \Theta|\mathcal{D})$ is the normalization factor and the optimum parameters are

$$\Theta^* = \operatorname*{argmax}_{\Theta} \log p(\mathcal{D}|\Theta).$$

Note that although in general, either the primal or the dual problem is hard to solve exactly, the above conjugate results are still useful when developing approximate inference algorithms. For instance, we can impose additional mean-field assumptions on q in the primal formulation and iteratively solve for each factor; and in this process convex conjugates are useful to deal with the large-margin constraints (Zhu et al., 2009). Alternatively, we can apply approximate methods (e.g., MCMC sampling) to infer q based on its solution in (22), and iteratively solves for the dual parameters $\boldsymbol{\omega}$ using approximate statistics (Schofield, 2006). We will discuss more on this when presenting the inference algorithms for iLSVM and MT-iLSVM.

In the above discussions, we have treated the topics $\boldsymbol{\beta}$ as fixed unknown parameters. A fully Bayesian formulation would treat $\boldsymbol{\beta}$ as random variables, e.g., with a Dirichlet prior (Blei et al., 2003; Griffiths and Steyvers, 2004). Under the RegBayes interpretation, we can easily do such an extension of MedLDA, simply by moving $\boldsymbol{\beta}$ from Θ to $\mathbf{M}$.

Appendix C. Proof of the Theorems and Lemmas

This section provides the proof of the theorems and lemmas.

C.1 Proof of Theorem 6

Proof The adjoint of the linear operator E is given by $\langle Ex, \phi\rangle = \langle E^*\phi, x\rangle$. In this theorem, E is the expectation with respect to q. Thus, we have

$$\begin{aligned}\langle Eq, \phi\rangle &= \left\langle \int q(\mathbf{M})\boldsymbol{\psi}(\mathbf{M}, \mathcal{D})\mathrm{d}\mu(\mathbf{M}), \phi \right\rangle \\ &= \int q(\mathbf{M}) \left\langle \boldsymbol{\psi}(\mathbf{M}, \mathcal{D}), \phi\right\rangle \mathrm{d}\mu(\mathbf{M}) \\ &= (E^*\phi)(q),\end{aligned}$$

where $E^*\phi = \langle \phi, \psi(.)\rangle$.

By definition, we have $\mathrm{KL}(q(\mathbf{M})\|p(\mathbf{M}, \mathcal{D})) = \mathrm{KL}(q(\mathbf{M})\|p(\mathbf{M}|\mathcal{D})) + c$, where c is a constant, $c = -\log p(\mathcal{D})$. Let $f(q(\mathbf{M}))$ denote the KL-divergence $\mathrm{KL}(q(\mathbf{M})\|p(\mathbf{M}|\mathcal{D}))$. The following proof is similar to the proof of the Fenchel duality theorem (Borwein and Zhu, 2005). Let t and d denote the primal value and the dual value, respectively. By Lemma 4.3.1 (Borwein and Zhu, 2005), under appropriate regularity conditions, there is a $\hat{\phi}$ such that

$$t \leq \left[f(q) - \left\langle \hat{\phi}, Eq\right\rangle\right] + \left[g(\phi) + \left\langle \hat{\phi}, \phi\right\rangle\right] + c.$$

For any μ, setting $\phi = Eq + \mu$ in the above inequality, we have

$$\begin{aligned}t &\leq f(q) + g(Eq + \mu) + \left\langle \hat{\phi}, \mu\right\rangle + c \\ &= \left\{f(q) - \left\langle E^*\hat{\phi}, q\right\rangle\right\} + \left\{g(Eq + \mu) - \left\langle -\hat{\phi}, Eq + \mu\right\rangle\right\} + c.\end{aligned}$$

Taking the infimum over all points μ, we have

$$t \leq \left\{f(q) - \left\langle E^*\hat{\phi}, q\right\rangle\right\} - g^*(-\hat{\phi}) + c.$$

Then, taking the infimum over all points $q \in \mathcal{P}_{\mathrm{prob}}$, we have

$$\begin{aligned}t &\leq \inf_{q\in\mathcal{P}_{\mathrm{prob}}} \left\{f(q) - \left\langle E^*\hat{\phi}, q\right\rangle\right\} - g^*(-\hat{\phi}) + c \\ &= -f^*(E^*\hat{\phi}) - g^*(-\hat{\phi}) + c \\ &\leq d,\end{aligned} \tag{23}$$

where

$$f^*(E^*\phi) = \log \int p(\mathbf{M}|\mathcal{D}) \exp\left(\langle \phi, \psi(\mathbf{M}, \mathcal{D})\rangle\right) \mathrm{d}\mu(\mathbf{M})$$

is the convex conjugate of the KL-divergence.

Since $d \leq t$ due to the Fenchel weak duality theorem (Borwein and Zhu, 2005) (Theorem 4.4.2), we have the strong duality that $t = d$, and $\hat{\phi}$ attains the supremum in the dual problem. During the deviation of the infimum in (23), we get the optimum solution of q:

$$\begin{aligned}\hat{q}_{\hat{\phi}}(\mathbf{M}) &\propto p(\mathbf{M}|\mathcal{D}) \exp\left(\left\langle \hat{\phi}, \boldsymbol{\psi}(\mathbf{M}; \mathcal{D})\right\rangle\right) \\ &= p(\mathbf{M}, \mathcal{D}) \exp\left(\left\langle \hat{\phi}, \boldsymbol{\psi}(\mathbf{M}; \mathcal{D})\right\rangle - \Lambda_{\hat{\phi}}\right).\end{aligned}$$

Absorbing the constant c into f^*, we get the dual objective of Theorem 6. ■

C.2 Proof of Lemma 9

Proof By definition, $g_0^*(\mu) = \sup_{x\in\mathbb{R}}(x\mu - C\max(0, x))$. We consider two cases. First, if $\mu < 0$, we have

$$g_0^*(\mu) \geq \sup_{x<0}(x\mu - C\max(0, x)) = \sup_{x<0} x\mu = \infty.$$

Therefore, we have $g_0^*(\mu) = \infty$ if $\mu < 0$. Second, if $\mu \geq 0$, we have

$$g_0^*(\mu) = \sup_{x\geq 0}(x\mu - Cx) = \mathbb{I}(\mu \leq C).$$

Putting the above results together, we prove the claim. ■

C.3 Proof of Lemma 10

Proof The proof has a similar structure as the proof of Lemma 9. By definition, we have

$$g_1^*(\boldsymbol{\mu}) = \sup_{\mathbf{x}}\left\{\boldsymbol{\mu}^\top \mathbf{x} - g_1(\mathbf{x})\right\} = \sup_{\mathbf{x}}\left\{\sum_j \mu_j x_j - \max(x_1, \cdots, x_L)\right\}.$$

We first show that $\forall i,\ \mu_i \geq 0$ in order to have finite g_1^* values. Suppose that $\exists j,\ \mu_j < 0$. Then, we define

$$\mathcal{G}_j = \left\{\mathbf{x} \in \mathbb{R}^L : x_j < 0\right\}, \text{ and } \mathcal{G}_j^o = \left\{\mathbf{x} \in \mathcal{G}_j : x_i = 0, \text{ if } i \neq j\right\}.$$

Since $\mathcal{G}_j^o \subset \mathcal{G}_j \subset \mathbb{R}^L$, we have

$$g_1^*(\boldsymbol{\mu}) \geq \sup_{\mathbf{x}\in\mathcal{G}_j}\left\{\boldsymbol{\mu}^\top \mathbf{x} - g_1(\mathbf{x})\right\} \geq \sup_{\mathbf{x}\in\mathcal{G}_j^o}\left\{\boldsymbol{\mu}^\top \mathbf{x} - g_1(\mathbf{x})\right\} = \sup_{x_j\in\mathbb{R}_-}\{x_j\mu_j - 0\} = \infty.$$

Therefore, $g_1^*(\boldsymbol{\mu}) = \infty$ if $\exists j,\ \mu_j < 0$.

Now, we consider the second case, where $\forall i, \mu_i \geq 0$. We can easily show that

$$\forall \mathbf{x} \in \mathbb{R}^L,\ \boldsymbol{\mu}^\top \mathbf{x} - g_1(\mathbf{x}) \leq \sum_i \mu_i \max(\mathbf{x}) - g_1(\mathbf{x}).$$

Therefore

$$g_1^*(\boldsymbol{\mu}) \leq \sup_{\mathbf{x}\in\mathbb{R}^L} \left\{ (\sum_i \mu_i - C)\max(\mathbf{x}) \right\} = \mathbb{I}\left(\sum_i \mu_i = C \right).$$

Moreover, let $\mathcal{G}^1 = \{\mathbf{x} \in \mathbb{R}^L :\ \mathbf{x} = x\mathbf{e},\ x \in \mathbb{R}\}$, where $\mathbf{e}$ is a vector with every element being 1. Then, we have

$$g_1^*(\boldsymbol{\mu}) \geq \sup_{\mathbf{x}\in\mathcal{G}^1} \left\{ \boldsymbol{\mu}^\top \mathbf{x} - g_1(\mathbf{x}) \right\} = \sup_{x\in\mathbb{R}} \left\{ (\sum_i \mu_i - C)x \right\} = \mathbb{I}\left(\sum_i \mu_i = C \right).$$

Putting the above results together proves the claim. ■

C.4 Proof of Lemma 11

Proof By definition, the conjugate is

$$\begin{aligned}
g_2^*(\mu) &= \sup_{x\in\mathbb{R}} \{\mu x - C\max(0, |x-y| - \epsilon)\}. \\
&= -\inf_{x\in\mathbb{R}} \{-\mu x + C\max(0, |x-y| - \epsilon)\}. \\
&= -\inf_{x\in\mathbb{R}; t\geq 0; t\geq |x-y|-\epsilon} \{-\mu x + Ct\} \\
&= -\sup_{\alpha,\beta\geq 0} \left\{ \inf_{x,t\in\mathbb{R}} \{-\mu x + Ct - \alpha(t - |x-y| + \epsilon) - \beta t\} \right\} \\
&= -\sup_{\alpha,\beta\geq 0} \left\{ \inf_{x\in\mathbb{R}} \{-\mu x + \alpha|x-y|\} + \inf_{t\in\mathbb{R}} \{Ct - \alpha t - \beta t\} - \alpha\epsilon \right\}
\end{aligned}$$

For the second infimum, it is easy to show that

$$\inf_{t\in\mathbb{R}} \left\{ Ct - \alpha t - \beta t \right\} = -\mathbb{I}(\alpha + \beta = C).$$

For the first infimum, we can show that

$$\inf_{x\in\mathbb{R}} \left\{ -\mu x + \alpha|x-y| \right\} = -\mu y + \inf_{x'\in\mathbb{R}} \left\{ -\mu x' + \alpha|x'| \right\} = -\mu y - \mathbb{I}(|\mu| \leq \alpha).$$

Thus, we have

$$\begin{aligned}
g_2^*(\mu) &= -\sup_{\alpha,\beta\geq 0} \left\{ -\mu y - \alpha\epsilon - \mathbb{I}(|\mu| \leq \alpha) - \mathbb{I}(\alpha + \beta = C) \right\} \\
&= -(-\mu y - \epsilon|\mu| - \mathbb{I}(|\mu| \leq C)) \\
&= \mu y + \epsilon|\mu| + \mathbb{I}(|\mu| \leq C),
\end{aligned}$$

where the second equality holds by setting $\alpha = |\mu|$, under the condition that ϵ is positive; the condition $|\mu| \leq C$ is induced from the conditions $\alpha + \beta = C$ and $\beta \geq 0$. ■

C.5 Proof of Lemma 12

Proof By definition, we have $g(Eq) \stackrel{\text{def}}{=} \mathcal{R}_h^c\big(q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})\big) = \sum_n g_1(\ell_n^\Delta - Eq(n))$. Let $\boldsymbol{\mu}_n = Eq(n)$. We have the conjugate

$$\begin{aligned} g^*(\boldsymbol{\omega}) &= \sup_{\boldsymbol{\mu}} \Big\{ \boldsymbol{\omega}^\top \boldsymbol{\mu} - \sum_n g_1(\ell_n^\Delta - \boldsymbol{\mu}_n) \Big\} \\ &= \sum_n \sup_{\boldsymbol{\mu}_n} \Big\{ \boldsymbol{\omega}_n^\top \boldsymbol{\mu}_n - g_1(\ell_n^\Delta - \boldsymbol{\mu}_n) \Big\} \\ &= \sum_n \sup_{\boldsymbol{\nu}_n} \Big\{ \boldsymbol{\omega}_n^\top (\ell_n^\Delta - \boldsymbol{\nu}_n) - g_1(\boldsymbol{\nu}_n) \Big\} \\ &= \sum_n \Big(\boldsymbol{\omega}_n^\top \ell_n^\Delta + g_1^*(-\boldsymbol{\omega}_n) \Big). \end{aligned}$$

Thus,

$$g^*(-\boldsymbol{\omega}) = \sum_n \Big(-\boldsymbol{\omega}_n^\top \ell_n^\Delta + g_1^*(\boldsymbol{\omega}_n) \Big).$$

Using the results of Theorem 6 proves the claim. ■

C.6 Proof of Lemma 13

Proof Similar structure as the proof of Lemma 12. In this case, the linear expectation operator is $E: \mathcal{P}_{\text{prob}} \to \mathbb{R}^{\sum_m |\mathcal{I}_{\text{tr}}^m|}$ and the element of Eq evaluated at the nth example for task m is

$$Eq(n,m) \stackrel{\text{def}}{=} y_{mn} \mathbb{E}_{q(\mathbf{Z},\boldsymbol{\eta})}[\mathbf{Z}\boldsymbol{\eta}_m]^\top \mathbf{x}_{mn} = \mathbb{E}_{q(\mathbf{Z},\boldsymbol{\eta})} \Big[y_{mn} (\mathbf{Z}\boldsymbol{\eta}_m)^\top \mathbf{x}_{mn} \Big].$$

Then, let $g_0: \mathbb{R} \to \mathbb{R}$ be a function defined in Lemma 9. We have

$$g(Eq) \stackrel{\text{def}}{=} \mathcal{R}_h^{MT}\Big(q(\mathbf{Z}, \boldsymbol{\eta}, \mathbf{W})\Big) = \sum_{m,n \in \mathcal{I}_{\text{tr}}^m} g_0\Big(1 - Eq(n,m)\Big).$$

Let $\boldsymbol{\mu} = Eq$. By definition, the conjugate is

$$\begin{aligned} g^*(\boldsymbol{\omega}) &= \sup_{\boldsymbol{\mu}} \Big\{ \boldsymbol{\omega}^\top \boldsymbol{\mu} - \sum_{m,n \in \mathcal{I}_{\text{tr}}^m} g_0(1 - \mu_{mn}) \Big\} \\ &= \sum_{m,n \in \mathcal{I}_{\text{tr}}^m} \sup_{\mu_{mn}} \Big\{ \omega_{mn} \mu_{mn} - g_0(1 - \mu_{mn}) \Big\} \\ &= \sum_{m,n \in \mathcal{I}_{\text{tr}}^m} \sup_{\nu_n^m} \Big\{ \omega_{mn}(1 - \nu_{mn}) - g_0(\nu_{mn}) \Big\} \\ &= \sum_{m,n \in \mathcal{I}_{\text{tr}}^m} \Big(\omega_{mn} + g_0^*(-\omega_{mn}) \Big). \end{aligned}$$

Thus,

$$g^*(-\boldsymbol{\omega}) = \sum_{m,n \in \mathcal{I}_{\text{tr}}^m} \Big(-\omega_{mn} + g_0^*(\omega_{mn}) \Big).$$

By the results in Theorem 6 and Lemma 9, we can derive the conjugate of the problem (15). ■

C.7 Proof of Lemma 15

Proof Similar structure as the proof of Lemma 12. In this case, the linear expectation operator is $E: \mathcal{P}_{\text{prob}} \to \mathbb{R}^N$ and the elements of Eq evaluated at the nth example is

$$\mu_n = \mathbb{E}_{q(\{\theta_n, z_{nm}, \boldsymbol{\eta}\}|\Theta)} \left[\boldsymbol{\eta}^\top \bar{z}_n \right].$$

Then, using the g_2 function defined in Lemma 11, we have

$$g(Eq) \stackrel{\text{def}}{=} \mathcal{R}_\epsilon(q(\{\theta_n, z_{nm}, \boldsymbol{\eta}\}|\Theta)) = \sum_n g_2\Big(\mu_n; y_n, \epsilon\Big).$$

Therefore $g^*(\boldsymbol{\omega}) = \sum_n g_2^*(\omega_n; y_n, \epsilon)$ and $g^*(-\boldsymbol{\omega}) = \sum_n g_2^*(-\omega_n; y_n, \epsilon)$. By the results in Theorem 6 and Lemma 9, we can derive the conjugate and the optimum solution of $\hat{q}$. The optimum solution of Θ is due to Lemma 14. Note that the constraints are not directly dependent on Θ. ■

Appendix D. Inference Algorithms for Infinite Latent SVMs

D.1 Inference for MT-iLSVM

In this section, we provide the derivation of the inference algorithm for MT-iLSVM, which is outlined in Algorithm 2 and detailed below.

For MT-iLSVM, the model $\mathbf{M}$ consists of all the latent variables $(\boldsymbol{\nu}, \mathbf{W}, \mathbf{Z}, \boldsymbol{\eta})$. Let $L_{mn}(q) \stackrel{\text{def}}{=} \mathbb{E}_q[\log p(\mathbf{x}_{mn}|\mathbf{Z}, \mathbf{w}_{mn}, \lambda_{mn}^2)]$ be the expected data likelihood. Then, under the truncated mean-field assumption (16), we have

$$L_{mn}(q) = -\frac{\mathbf{x}_{mn}^\top \mathbf{x}_{mn} - 2\mathbf{x}_{mn}^\top \mathbb{E}_q[\mathbf{Z}\mathbf{w}_{mn}] + \mathbb{E}_q[\mathbf{w}_{mn}^\top \mathbf{U} \mathbf{w}_{mn}]}{2\lambda_{mn}^2} - \frac{D \log(2\pi\lambda_{mn}^2)}{2},$$

where $\mathbf{x}_{mn}^\top \mathbb{E}_q[\mathbf{Z}\mathbf{w}_{mn}] = \sum_k \mathbf{x}_{mn}^\top \boldsymbol{\psi}_{\cdot k}$; $\boldsymbol{\psi}_{\cdot k} \stackrel{\text{def}}{=} (\psi_{1k} \cdots \psi_{Dk})^\top$ is the kth column of $\boldsymbol{\psi} = \mathbb{E}_q[\mathbf{Z}]$;

$$\mathbb{E}_q[\mathbf{w}_{mn}^\top \mathbf{U} \mathbf{w}_{mn}] = 2 \sum_{j<k} \phi_{mn}^j \phi_{mn}^k \mathbf{U}_{jk} + \sum_k \mathbf{U}_{kk} \left(K\sigma_{mn}^2 + \Phi_{mn}^\top \Phi_{mn} \right);$$

and $\mathbf{U} \stackrel{\text{def}}{=} \mathbb{E}_q[\mathbf{Z}^\top \mathbf{Z}]$ is a $K \times K$ matrix, whose element is

$$\mathbf{U}_{ij} = \begin{cases} \sum_d \psi_{di}, & \text{if } i = j \\ \sum_d \psi_{di}\psi_{dj}, & \text{otherwise.} \end{cases}$$

For the KL-divergence term, we have $\mathrm{KL}(q(\mathbf{M})\|\pi(\mathbf{M})) = \mathrm{KL}(q(\boldsymbol{\nu})\|\pi(\boldsymbol{\nu}))+\mathrm{KL}(q(\mathbf{W})\|\pi(\mathbf{W}))+ \mathbb{E}_{q(\boldsymbol{\nu})}[\mathrm{KL}(q(\mathbf{Z})\|\pi(\mathbf{Z}|\boldsymbol{\nu}))] + \mathrm{KL}(q(\boldsymbol{\eta})\|\pi(\boldsymbol{\eta}))$, where the individual terms are

$$\begin{aligned}
\mathrm{KL}(q(\boldsymbol{\nu})\|\pi(\boldsymbol{\nu})) &= \sum_{k=1}^{K}\Big((\gamma_{k1}-\alpha)(\varphi(\gamma_{k1})-\varphi(\gamma_{k1}+\gamma_{k2})) - \log\frac{\Gamma(\gamma_{k1})\Gamma(\gamma_{k2})}{\Gamma(\gamma_{k1}+\gamma_{k2})} \\
&\qquad +(\gamma_{k2}-1)(\varphi(\gamma_{k2})-\varphi(\gamma_{k1}+\gamma_{k2}))\Big) - K\log\alpha, \\
\mathbb{E}_{q(\boldsymbol{\nu})}[\mathrm{KL}(q(\mathbf{Z})\|\pi(\mathbf{Z}|\boldsymbol{\nu}))] &= \sum_{dk}\Big(-\psi_{dk}\sum_{j=1}^{k}\mathbb{E}_q[\log\nu_j] - (1-\psi_{dk})\mathbb{E}_q[\log(1-\prod_{j=1}^{k}\nu_j)] \\
&\qquad +\psi_{dk}\log\psi_{dk} + (1-\psi_{dk})\log(1-\psi_{dk})\Big) \\
\mathrm{KL}(q(\mathbf{W})\|\pi(\mathbf{W})) &= \sum_{mn}\left(\frac{K\sigma_{mn}^2+\Phi_{mn}^\top\Phi_{mn}}{2\sigma_{m0}^2} - \frac{K(1+\log\frac{\sigma_{mn}^2}{\sigma_{m0}^2})}{2}\right).
\end{aligned}$$

where $\varphi(\cdot)$ is the digamma function and $\mathbb{E}_q[\log v_j] = \varphi(\gamma_{j1})-\varphi(\gamma_{j1}+\gamma_{j2})$. For $\mathrm{KL}(q(\boldsymbol{\eta})\|\pi(\boldsymbol{\eta}))$, we do not need to write it explicitly, as we shall see. Finally, the effective discriminant function is

$$f_m(\mathbf{x}_{mn}; q(\mathbf{Z},\boldsymbol{\eta})) = \mathbb{E}_q[\boldsymbol{\eta}_m]^\top\boldsymbol{\psi}^\top\mathbf{x}_{mn} = \sum_{k=1}^{K}\mathbb{E}_q[\eta_{mk}]\boldsymbol{\psi}_{\cdot k}^\top\mathbf{x}_{mn}.$$

All the above terms can be easily computed, except the term $\mathbb{E}_q[\log(1-\prod_{j=1}^{k}\nu_j)]$. Here, we adopt the multivariate lower bound (Doshi-Velez, 2009)

$$\begin{aligned}
\mathbb{E}_q\left[\log\left(1-\prod_{j=1}^{k}\nu_j\right)\right] &\geq \sum_{m=1}^{k}q_{km}\varphi(\gamma_{m2}) + \sum_{m=1}^{k-1}\left(\sum_{n=m+1}^{k}q_{kn}\right)\varphi(\gamma_{m1}) \\
&\quad - \sum_{m=1}^{k}\left(\sum_{n=m}^{k}q_{kn}\right)\varphi(\gamma_{m1}+\gamma_{m2}) + \mathcal{H}(q_{k\cdot}),
\end{aligned}$$

where the variational parameters $q_{k\cdot} = (q_{k1}\cdots q_{kk})^\top$ belong to the k-simplex, and $\mathcal{H}(q_{k\cdot})$ is the entropy of $q_{k\cdot}$. The tightest lower bound is achieved by setting $q_{k\cdot}$ to be the optimum value

$$q_{km} = \frac{1}{Z_k}\exp\left(\varphi(\gamma_{m2}) + \sum_{n=1}^{m-1}\varphi(\gamma_{n1}) - \sum_{n=1}^{m}\varphi(\gamma_{n1}+\gamma_{n2})\right), \tag{24}$$

where Z_k is a normalization factor to make $q_{k\cdot}$ be a distribution. We denote the tightest lower bound by $\mathcal{L}_k^\nu$. Replacing the term $\mathbb{E}_q[\log(1-\prod_{j=1}^{k}\nu_j)]$ with its lower bound $\mathcal{L}_k^\nu$, we can have an upper bound of $\mathrm{KL}(q(\mathbf{M})\|\pi(\mathbf{M}))$ and we denote this upper bound by $\mathcal{L}(q)$.

With the above terms and the upper bound $\mathcal{L}(q)$, we can implement the general procedure outlined in Algorithm 1 to solve the MT-iLSVM problem. Specifically, the inference procedure iteratively solves the following steps, as summarized in Algorithm 2:

Algorithm 2 Inference Algorithm of MT-iLSVM

Input: data $\mathcal{D} = \{(\mathbf{x}_{mn}, y_{mn})\}_{m,n\in\mathcal{I}^m_{\text{tr}}} \cup \{\mathbf{x}_{mn}\}_{m,n\in\mathcal{I}^m_{\text{tst}}}$, constants α and C
Output: distributions $q(\boldsymbol{\nu})$, $q(\mathbf{Z})$, $q(\mathbf{W})$, $q(\boldsymbol{\eta})$ and hyper-parameters σ^2_{m0} and λ^2_{mn}
Initialize $\gamma_{k1} = \alpha$, $\gamma_{k2} = 1$, $\psi_{dk} = 0.5 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.001)$, $\Phi_{mn} = 0$, $\sigma^2_{mn} = \sigma^2_{m0} = 1$, $\boldsymbol{\mu}_m = 0$, λ^2_{mn} is computed from $\mathcal{D}$.
repeat
  repeat
    update $(\gamma_{k1}, \gamma_{k2})$ using (26), $\forall 1 \le k \le K$;
    update ϕ^k_{mn} and σ^2_{mn} using (25), $\forall m, \forall n, \forall 1 \le k \le K$;
    update ψ_{dk} using (27), $\forall 1 \le d \le D, \forall 1 \le k \le K$;
  until relative change of L is less than τ (e.g., $1e^{-3}$) or iteration number is T (e.g., 10)
  for $m = 1$ **to** M **do**
    solve the dual problem (28) using a binary SVM learner.
  end for
  update the hyper-parameters σ^2_{m0} using (29) and λ^2_{mn} using (30). (*Optional*)
until relative change of L is less than τ' (e.g., $1e^{-4}$) or iteration number is T' (e.g., 20)

Infer $q(\boldsymbol{\nu})$, $q(\mathbf{Z})$ and $q(\mathbf{W})$: For $q(\mathbf{W})$, since both the prior $\pi(\mathbf{W})$ and $q(\mathbf{W})$ are Gaussian, we can easily derive the update rules, similar as in Gaussian mixture models

$$\begin{aligned}\phi^k_{mn} &= \frac{\sum_d x^d_{mn}\psi_{dk} - \sum_{j\neq k}\phi^j_{mn}\mathbf{U}_{kj}}{\lambda^2_{mn}}\left(\frac{1}{\sigma^2_{m0}} + \frac{\sum_d \psi_{dk}}{\lambda^2_{mn}}\right)^{-1} \\ \sigma^2_{mn} &= \left(\frac{1}{\sigma^2_{m0}} + \frac{1}{K}\sum_k \frac{\mathbf{U}_{kk}}{\lambda^2_{mn}}\right)^{-1}\end{aligned} \tag{25}$$

For $q(\boldsymbol{\nu})$, we have the update rules similar as in (Doshi-Velez, 2009), that is,

$$\begin{aligned}\gamma_{k1} &= \alpha + \sum_{m=k}^{K}\sum_{d=1}^{D}\psi_{dm} + \sum_{m=k+1}^{K}\left(D - \sum_{d=1}^{D}\psi_{dm}\right)\left(\sum_{i=k+1}^{m} q_{mi}\right) \\ \gamma_{k2} &= 1 + \sum_{m=k}^{K}\left(D - \sum_{d=1}^{D}\psi_{dm}\right)q_{mk}.\end{aligned} \tag{26}$$

For $q(\mathbf{Z})$, we have the mean-field update equation as

$$\psi_{dk} = \frac{1}{1 + e^{-\vartheta_{dk}}}, \tag{27}$$

where

$$\begin{aligned}\vartheta_{dk} = \sum_{j=1}^{k}\mathbb{E}_q[\log v_j] - \mathcal{L}^\nu_k - \sum_{mn}\frac{1}{2\lambda^2_{mn}}\Big((K\sigma^2_{mn} + (\phi^k_{mn})^2) \\ -2x^d_{mn}\phi^k_{mn} + 2\sum_{j\neq k}\phi^j_{mn}\phi^k_{mn}\psi_{dj}\Big) + \sum_{m,n\in\mathcal{I}^m_{\text{tr}}} y_{mn}\mathbb{E}_q[\eta_{mk}]x^d_{mn}.\end{aligned}$$

Infer $q(\boldsymbol{\eta})$ and solve for $\boldsymbol{\omega}$: By the convex duality theory, we have the solution

$$q(\boldsymbol{\eta}) \propto \pi(\boldsymbol{\eta}) \exp\left\{\sum_{m,n\in\mathcal{I}_{\text{tr}}^m} y_{mn}\omega_{mn}\boldsymbol{\eta}_m^\top \boldsymbol{\psi}^\top \mathbf{x}_{mn}\right\}$$
$$= \prod_{m=1}^M \pi(\boldsymbol{\eta}_m) \exp\left\{\boldsymbol{\eta}_m^\top \left(\sum_{n\in\mathcal{I}_{\text{tr}}^m} y_{mn}\omega_{mn}\boldsymbol{\psi}^\top \mathbf{x}_{mn}\right)\right\}.$$

Therefore, we can see that although we did not assume $q(\boldsymbol{\eta})$ is factorized, we can get the induced factorization form $q(\boldsymbol{\eta}) = \prod_m q(\boldsymbol{\eta}_m)$, where

$$q(\eta_m) \propto \pi(\boldsymbol{\eta}_m) \exp\left\{\boldsymbol{\eta}_m^\top \left(\sum_{n\in\mathcal{I}_{\text{tr}}^m} y_{mn}\omega_{mn}\boldsymbol{\psi}^\top \mathbf{x}_{mn}\right)\right\}.$$

Here, we assume $\pi(\boldsymbol{\eta}_m)$ is standard normal. Then, we have $q(\boldsymbol{\eta}_m) = \mathcal{N}(\boldsymbol{\eta}_m|\boldsymbol{\mu}_m, I)$, where

$$\boldsymbol{\mu}_m = \sum_{n\in\mathcal{I}_{\text{tr}}^m} y_{mn}\omega_{mn}\boldsymbol{\psi}^\top \mathbf{x}_{mn}.$$

The optimum dual parameters can be obtained by solving the following M independent dual problems

$$\sup_{\boldsymbol{\omega}_m} -\frac{1}{2}\boldsymbol{\mu}_m^\top\boldsymbol{\mu}_m + \sum_{n\in\mathcal{I}_{\text{tr}}^m}\omega_{mn} \qquad \text{s.t.. : } 0 \le \omega_{mn} \le C, \forall n \in \mathcal{I}_{\text{tr}}^m, \tag{28}$$

which (and its primal form) can be efficiently solved with a binary SVM solver, such as SVM-light.

As we have stated, the hyperparameters σ_0^2 and λ_{mn}^2 can be set a priori or estimated from the data. The empirical estimation can be easily done with closed form solutions by optimizing the RegBayes objective with all the variational terms fixed. For MT-iLSVM, we have

$$\sigma_{m0}^2 = \frac{\sum_{n=1}^{N_m}(K\sigma_{mn}^2 + \Phi_{mn}^\top\Phi_{mn})}{KN_m} \tag{29}$$

$$\lambda_{mn}^2 = \frac{\mathbf{x}_{mn}^\top\mathbf{x}_{mn} - 2\mathbf{x}_{mn}^\top\mathbb{E}_q[\mathbf{Z}\mathbf{w}_{mn}] + \mathbb{E}_q[\mathbf{w}_{mn}^\top\mathbf{U}\mathbf{w}_{mn}]}{D}. \tag{30}$$

D.2 Inference for Infinite Latent SVM

In this section, we develop the inference algorithm for iLSVM based on the stick-breaking construction of the IBP prior. The algorithm is outlined in Algorithm 3.

Similar as in the inference for MT-iLSVM, we make the additional constraint about the feasible distribution

$$q(\boldsymbol{\nu}, \mathbf{W}, \mathbf{Z}, \boldsymbol{\eta}) = q(\boldsymbol{\eta})q(\mathbf{W}|\Phi,\Sigma)\prod_n\left(\prod_{k=1}^K q(z_{nk}|\psi_{nk})\right)\prod_{k=1}^K q(\nu_k|\boldsymbol{\gamma}_k),$$

where $q(\mathbf{W}|\Phi,\Sigma) = \prod_k \mathcal{N}(\mathbf{W}_{\cdot k}|\Phi_{\cdot k}, \sigma_k^2 I)$; $q(z_{nk}|\phi_{nk}) = \text{Bernoulli}(\phi_{nk})$; and $q(\nu_k|\boldsymbol{\gamma}_k) = \text{Beta}(\gamma_{k1}, \gamma_{k2})$; and K is the truncation level. Then, we solve the unconstrained problem using convex duality with dual parameters being $\boldsymbol{\omega}$. Let $L_n(q) \stackrel{\text{def}}{=} \mathbb{E}_q[\log p(\mathbf{x}_n|\mathbf{z}_n, \mathbf{W})]$. We have

$$L_n(q) = -\frac{\mathbf{x}_n^\top \mathbf{x}_n - 2\mathbf{x}_n^\top \Phi \mathbb{E}_q[\mathbf{z}_n]^\top + \mathbb{E}_q[\mathbf{z}_n \mathbf{A} \mathbf{z}_n^\top]}{2\sigma_{n0}^2} - \frac{D\log(2\pi\sigma_{n0}^2)}{2},$$

where $\mathbf{A} \stackrel{\text{def}}{=} \mathbb{E}_q[\mathbf{W}^\top \mathbf{W}]$ is a $K \times K$ matrix; $\mathbf{x}_n^\top \Phi \mathbb{E}_q[\mathbf{z}_n]^\top = 2\sum_k \psi_{nk}(\mathbf{x}_n^\top \Phi_{\cdot k})$; and

$$\mathbb{E}_q[\mathbf{z}_n \mathbf{A} \mathbf{z}_n^\top] = 2\sum_{j<k} \psi_{nj}\psi_{nk}\mathbf{A}_{jk} + \sum_k \psi_{nk}(D\sigma_k^2 + \mathbf{A}_{kk}).$$

The effective discriminant function is $f(y, \mathbf{x}_n) = \sum_k \mathbb{E}_q[\eta_y^k]\psi_{nk}$. Again, for computational tractability, we need the lower bound $\mathcal{L}_k^\nu$ of the term $\mathbb{E}_q[\log(1 - \prod_{j=1}^k v_j)]$. Using this lower bound, we can get an upper bound of the KL-divergence term. Then, the inference procedure iteratively solves the following steps:

Infer $q(\boldsymbol{\nu})$, $q(\mathbf{Z})$ and $q(\mathbf{W})$: For $q(\mathbf{W})$, we have the update rules

$$\begin{aligned} \Phi_{\cdot k} &= \sum_n \frac{\psi_{nk}}{\sigma_{n0}^2}\left(\mathbf{x}_n - \sum_{j\neq k}\psi_{nj}\Phi_{\cdot j}\right)\left(1 + \sum_n \frac{\psi_{nk}}{\sigma_{n0}^2}\right)^{-1} \\ \sigma_k^2 &= \left(1 + \sum_n \frac{\psi_{nk}}{\sigma_{n0}^2}\right)^{-1}. \end{aligned} \tag{31}$$

For $q(\boldsymbol{\nu})$, we have the update rules similar as in (Doshi-Velez, 2009), that is,

$$\begin{aligned} \gamma_{k1} &= \alpha + \sum_{m=k}^K \sum_{n=1}^N \psi_{nm} + \sum_{m=k+1}^K \left(N - \sum_{n=1}^N \psi_{nm}\right)\left(\sum_{i=k+1}^m q_{mi}\right) \\ \gamma_{k2} &= 1 + \sum_{m=k}^K \left(N - \sum_{n=1}^N \psi_{nm}\right) q_{mk}, \end{aligned} \tag{32}$$

where $q_{\cdot k}$ is computed in the same way as in (24). For $q(\mathbf{Z})$, the mean-field update equation for ψ is

$$\psi_{nk} = \frac{1}{1 + e^{-\vartheta_{nk}}}, \tag{33}$$

where

$$\begin{aligned} \vartheta_{nk} = &\sum_{j=1}^k \mathbb{E}_q[\log v_j] - \mathcal{L}_k^\nu(q) - \frac{1}{2\sigma_{n0}^2}\left(D\sigma_k^2 + \Phi_{\cdot k}^\top \Phi_{\cdot k}\right) \\ &+ \frac{1}{\sigma_{n0}^2}\Phi_{\cdot k}^\top \left(\mathbf{x}_n - \sum_{j\neq k}\psi_{nj}\Phi_{\cdot j}\right) + \sum_y \omega_n^y \mathbb{E}_q[\eta_{y_n}^k - \eta_y^k]. \end{aligned}$$

Algorithm 3 Inference Algorithm of iLSVM

Input: data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n\in\mathcal{I}_{\text{tr}}} \cup \{\mathbf{x}_n\}_{n\in\mathcal{I}_{\text{tst}}}$, constants α and C
Output: distributions $q(\boldsymbol{\nu})$, $q(\mathbf{Z})$, $q(\mathbf{W})$, $q(\boldsymbol{\eta})$ and hyper-parameters σ_0^2 and σ_{n0}^2
Initialize $\gamma_{k1} = \alpha$, $\gamma_{k2} = 1$, $\psi_{nk} = 0.5 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.001)$, $\Phi_{\cdot k} = 0$, $\sigma_k^2 = \sigma_0^2 = 1$, $\boldsymbol{\mu} = 0$, σ_{n0}^2 is computed from $\mathcal{D}$.
repeat
  repeat
    update $(\gamma_{k1}, \gamma_{k2})$ using (32), $\forall 1 \le k \le K$;
    update $\Phi_{\cdot k}$ and σ_k^2 using (31), $\forall 1 \le k \le K$;
    update ψ_{nk} using (33), $\forall n \in \mathcal{I}_{\text{tr}}, \forall 1 \le k \le K$;
    update ψ_{nk} using (33), but ϑ_{nk} doesn't have the last term, $\forall n \in \mathcal{I}_{\text{tst}}, \forall 1 \le k \le K$;
  until relative change of L is less than τ (e.g., $1e^{-3}$) or iteration number is T (e.g., 10)
  solve the dual problem (34) (or its primal form) using a multi-class SVM learner.
  update the hyper-parameters σ_0^2 using (35) and σ_{n0}^2 using (36). (*Optional*)
until relative change of L is less than τ' (e.g., $1e^{-4}$) or iteration number is T' (e.g., 20)

For testing data, ϑ_{nk} does not have the last term because of the absence of large-margin constraints.

Infer $q(\boldsymbol{\eta})$ and solve for $\boldsymbol{\omega}$: By the convex duality theory, we have

$$q(\boldsymbol{\eta}) \propto \pi(\boldsymbol{\eta}) \exp\left\{\boldsymbol{\eta}^\top \left(\sum_{n\in\mathcal{I}_{\text{tr}}} \sum_y \omega_n^y \mathbb{E}_q[\mathbf{g}(y_n, \mathbf{x}_n, \mathbf{z}_n) - \mathbf{g}(y, \mathbf{x}_n, \mathbf{z}_n)]\right)\right\}.$$

For the standard normal prior $\pi(\boldsymbol{\eta})$, we have that $q(\boldsymbol{\eta})$ is also normal, with mean

$$\boldsymbol{\mu} = \sum_{n\in\mathcal{I}_{\text{tr}}} \sum_y \omega_d^y \mathbb{E}_q[\mathbf{g}(y_n, \mathbf{x}_n, \mathbf{z}_n) - \mathbf{g}(y, \mathbf{x}_n, \mathbf{z}_n)]$$

and identity covariance matrix. The dual problem is

$$\sup_{\boldsymbol{\omega}} \; -\frac{1}{2}\boldsymbol{\mu}^\top\boldsymbol{\mu} + \sum_{n\in\mathcal{I}_{\text{tr}}} \sum_y \omega_n^y \qquad \text{s.t.. :} \;\; \omega_n^y \ge 0, \; \sum_y \omega_n^y = C, \forall n \in \mathcal{I}_{\text{tr}}, \tag{34}$$

which (and its primal form) can be efficiently solved with a multi-class SVM solver.

Similar as in MT-iLSVM, the hyperparameters σ_0^2 and σ_{n0}^2 can be set a priori or estimated from the data. The empirical estimation can be easily done with closed form solutions. For iLSVM, we have

$$\sigma_0^2 = \frac{\sum_{k=1}^K (D\sigma_k^2 + \Phi_{\cdot k}^\top \Phi_k)}{KD} \tag{35}$$

$$\sigma_{n0}^2 = \frac{\mathbf{x}_n^\top \mathbf{x}_n - 2\mathbf{x}_n^\top \Phi \mathbb{E}_p[\mathbf{z}_n]^\top + \mathbb{E}_q[\mathbf{z}_n \mathbf{A} \mathbf{z}_n^\top]}{D}. \tag{36}$$

References

Yasemin Altun and Alex Smola. Unifying divergence minimization and statistical inference via convex duality. In *International Conference on Learning Theory*, pages 139–153, 2006.

Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, (6):1817–1853, 2005.

Charles E. Antoniak. Mixture of Dirichlet process with applications to Bayesian nonparametric problems. *Annals of Statistics*, (273):1152–1174, 1974.

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems*, pages 41–48, 2007.

Bart Bakker and Tom Heskes. Task clustering and gating for Bayesian multitask learning. *Journal of Machine Learning Research*, (4):83–99, 2003.

Andrew Barron, Mark Schervish, and Larry Wasserman. The consistency of posterior distributions in nonparametric problems. *The Annals of Statistics*, 27(2):536–561, 1999.

Matthew J. Beal, Zoubin Ghahramani, and Carl E. Rasmussen. The infinite hidden Markov model. In *Advances in Neural Information Processing Systems*, pages 577–584, 2002.

Kedar Bellare, Gregory Druck, and Andrew McCallum. Alternating projections for learning with expectation constraints. In *Uncertainty in Artificial Intelligence*, 2009.

David Blei and Peter Frazier. Distance dependent Chinese restaurant process. In *International Conference on Machine Learning*, pages 87–94, 2010.

David Blei, Andrew Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, (3):993–1022, 2003.

Edwin Bonilla, Kian Ming Chai, and Christopher Williams. Multi-task Gaussian process prediction. In *Advances in Neural Information Processing Systems*, pages 153–160, 2008.

Jonathan Borwein and Qiji Zhu. *Techniques of Variational Analysis: An Introduction.* Springer, New York, NY, 2005.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. Guiding semi-supervision with constraint-driven learning. In *the 45th Annual Meeting of the Association of Computational Linguistics*, pages 280–287, 2007.

Ning Chen, Jun Zhu, Fuchun Sun, and Eric P. Xing. Large-margin predictive latent subspace learning for multiview data analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(12):2365–2378, 2012.

Ning Chen, Jun Zhu, Fuchun Sun, and Bo Zhang. Learning harmonium models with infinite latent features. *IEEE Transactions on Neural Networks and Learning Systems*, 25(3): 520–532, 2014.

Taeryon Choi and R. V. Ramamoorthi. Remarks on consistency of posterior distributions. *IMS Collections 3. Pushing the limits of contemporary statistics: Contributions in honor of Jayanta K. Ghosh. eds. S. Ghosal and B. Clarke*, pages 170–186, 2008.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B*, 39(1):1–38, 1977.

Finale Doshi-Velez. The Indian buffet process: Scalable inference and extensions. Master's thesis, The University of Cambridge, Aug 2009.

Mirosla Dudík, Steven J. Phillips, and Robert E. Schapire. Maximum entropy density estimation with generalized regularization and an application to species distribution modeling. *Journal of Machine Learning Research*, (8):1217–1260, 2007.

David Dunson and Shyamal Peddada. Bayesian nonparametric inferences on stochastic ordering. *ISDS Discussion Paper*, 2, 2007.

Thomas Ferguson. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1(2):209–230, 1973.

Morten Frydenberg. The chain graph Markov property. *Scandinavian Journal of Statistics*, 17:333–353, 1990.

Kuzman Ganchev, João. Graca, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, (11):2001–2094, 2010.

Paul Garthwaite, Joseph Kadane, and Anthony O'Hagan. Statistical methods for eliciting probability distributions. *Journal of the American Statistical Association*, 100(470):680–700, 2005.

Jayanta K. Ghosh and R.V. Ramamoorthi. *Bayesian Nonparametrics*. Springer, New York, NY, 2003.

João Graca, Kuzman Ganchev, and Ben Taskar. Expectation maximization and posterior constraints. In *Advances in Neural Information Processing Systems*, pages 569–576, 2007.

Thomas Griffiths and Zoubin Ghahramani. Infinite latent feature models and the Indian buffet process. Technical report, University College London, GCNU TR2005-001, 2005.

Thomas L. Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235, 2004.

Peter D. Hoff. Bayesian methods for partial stochastic orderings. *Biometrika*, 90:303–317, 2003.

Seungil Huh and Stephen Fienberg. Discriminative topic modeling based on manifold learning. *ACM Transactions on Knowledge Discovery from Data*, 5(4), 2012.

Kazufumi Ito and Karl Kunisch. *Lagrange Multiplier Approach to Variational Problems and Applications*. Advances in Design and Control, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008.

Tommi Jaakkola, Meila Meila, and Tony Jebara. Maximum entropy discrimination. In *Advances in Neural Information Processing Systems*, pages 470–476, 1999.

Tony Jebara. *Discriminative, Generative and Imitative Learning*. PhD thesis, Media Laboratory, MIT, Dec 2001.

Tony Jebara. Multitask sparsity via maximum entropy discrimination. *Journal of Machine Learning Research*, (12):75–110, 2011.

Qixia Jiang, Jun Zhu, Maosong Sun, and Eric P. Xing. Monte Carlo methods for maximum margin supervised topic models. In *Advances in Neural Information Processing Systems*, pages 1592–1600, 2012.

Thorsten Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning*, 1999.

Michael I. Jordan, Zoubin Ghahramani, Tommis Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

Mohammad E. Khan, Guillaume Bouchard, Benjamin Marlin, and Kevin Murphy. Variational bounds for mixed-data factor analysis. In *Advances in Neural Information Processing Systems*, pages 1108–1116, 2010.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, pages 282–289, 2001.

Percy Liang, Michael I. Jordan, and Dan Klein. Learning from measurements in exponential families. In *International Conference on Machine Learning*, pages 641–648, 2009.

Steven N. MacEachern. Dependent nonparametric process. In *the Section on Bayesian Statistical Science of ASA*, 1999.

Thomas L. Magnanti. Fenchel and Lagrange duality are equivalent. *Mathematical Programming*, (7):253–258, 1974.

Gideon Mann and Andrew McCallum. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Machine Learning Research*, (11):955–984, 2010.

Shike Mei, Jun Zhu, and Xiaojin Zhu. Robust RegBayes: Selectively incorporating first-order logic domain knowledge into bayesian models. In *International Conference on Machine Learning*, pages 253–261, 2014.

Kurt Miller, Thomas Griffiths, and Michael I. Jordan. Nonparametric latent feature models for link prediction. In *Advances in Neural Information Processing Systems*, pages 1276–1284, 2009.

Iain Murray and Zoubin Ghahramani. Bayesian learning in undirected graphical models: Approximate MCMC algorithms. In *Uncertainty in Artificial Intelligence*, 2004.

Peter Orbanz. Nonparametric priors on complete separable metric spaces. *Working paper*, 2012.

Yuan (Alan) Qi, Martin Szummer, and Thomas P. Minka. Bayesian conditional random fields. In *International Conference on Artificial Intelligence and Statistics*, pages 269–276, 2005.

Piyush Rai and Hal Daume III. Infinite predictor subspace models for multitask learning. In *International Conference on Artificial Intelligence and Statistics*, pages 613–620, 2010.

Charles E. Rasmussen and Zoubin Ghahramani. Infinite mixtures of Gaussian process experts. In *Advances in Neural Information Processing Systems*, pages 881–888, 2002.

Christian P. Robert. Simulation of truncated normal variables. *Statistics and Computing*, 5(2):121–125, 1995.

Edward Schofield. *Fitting maximum-entropy models on large sample spaces*. PhD thesis, PhD thesis, Department of Computing, Imperial College London, 2006.

Wolfgang Stummer and Igor Vajda. On bregman distances and divergences of probability measures. *IEEE Trans. on Information Theory*, 58(3):1277–1288, 2012.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*, 2003.

Yee Whye Teh, Dilan Görür, and Zoubin Ghahramani. Stick-breaking construction of the Indian buffet process. In *International Conference on Artificial Intelligence and Statistics*, pages 556–563, 2007.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David Blei. Hierarchical Dirichlet process. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

Romain Thibaux and Michael I. Jordan. Hierarchical Beta processes and the Indian buffet process. In *International Conference on Artificial Intelligence and Statistics*, pages 564–571, 2007.

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

Martin Wainright and Michael I. Jordan. Graphical models, exponential family, and variational methods. *Foundations and Trends in Machine Learning*, 1(1):1–305, 2008.

Max Welling and Sridevi Parise. Bayesian random fields: The Bethe-Laplace approximation. In *Uncertainty in Artificial Intelligence*, 2006.

Max Welling, Ian Porteous, and Kenichi Kurihara. Exchangeable inconsistent priors for Bayesian posterior inference. In *Workshop on Information Theory and Applications*, 2012.

Max Welling, M. Rosen-Zvi, and G. Hinton. Exponential family harmoniums with an application to information retrieval. In *Advances in Neural Information Processing Systems*, pages 1481–1488, 2004.

Peter M. Williams. Bayesian conditionalisation and the principle of minimum information. *The British Journal for the Philosophy of Science*, 31(2), 1980.

Sinead Williamson, Peter Orbanz, and Zoubin Ghahramani. Dependent Indian buffet processes. In *International Conference on Artifficial Intelligence and Statistics*, pages 924–931, 2010.

Minjie Xu, Jun Zhu, and Bo Zhang. Bayesian nonparametric max-margin matrix factorization for collaborative prediction. In *Advances in Neural Information Processing Systems*, pages 64–72, 2012.

Ya Xue, David Dunson, and Lawrence Carin. The matrix stick-breaking process for flexible multi-task learning. In *International Conference on Machine Learning*, pages 1063–1070, 2007.

Yu Zhang and Dit-Yan Yeung. A convex formulation for learning task relationships in multi-task learning. In *Uncertainty in Artificial Intelligence*, 2010.

Jun Zhu. Max-margin nonparametric latent feature relational models for link prediction. In *International Conference on Machine Learning*, pages 719–726, 2012.

Jun Zhu, Amir Ahmed, and Eric P. Xing. MedLDA: Maximum margin supervised topic models for regression and classification. In *International Conference on Machine Learning*, pages 1257–1264, 2009.

Jun Zhu, Ning Chen, Hugh Perkins, and Bo Zhang. Gibbs max-margin topic models with fast inference algorithms. In *International Conference on Machine Learning*, pages 124–132, 2013.

Jun Zhu, Ning Chen, and Eric P. Xing. Infinite latent SVM for classification and multi-task learning. In *Advances in Neural Information Processing Systems*, pages 1620–1628, 2011a.

Jun Zhu, Ning Chen, and Eric P. Xing. Infinite SVM: a Dirichlet process mixture of large-margin kernel machines. In *International Conference on Machine Learning*, pages 617–624, 2011b.

Jun Zhu and Eric P. Xing. Maximum entropy discrimination Markov networks. *Journal of Machine Learning Research*, (10):2531–2569, 2009.

Journal of Machine Learning Research 15 (2014) 1849-1901 Submitted 3/13; Revised 3/14; Published 5/14

Expectation Propagation for Neural Networks with Sparsity-Promoting Priors

Pasi Jylänki[*] PASI.JYLANKI@AALTO.FI
Department of Biomedical Engineering and Computational Science
Aalto University School of Science
P.O. Box 12200, FI-00076 Aalto, Finland

Aapo Nummenmaa NUMMENMA@NMR.MGH.HARVARD.EDU
Athinoula A. Martinos Center for Biomedical Imaging
Massachusetts General Hospital, Harvard Medical School
Boston, MA 02129, USA

Aki Vehtari AKI.VEHTARI@AALTO.FI
Department of Biomedical Engineering and Computational Science
Aalto University School of Science
P.O. Box 12200, FI-00076 Aalto, Finland

Editor: Manfred Opper

Abstract

We propose a novel approach for nonlinear regression using a two-layer neural network (NN) model structure with sparsity-favoring hierarchical priors on the network weights. We present an expectation propagation (EP) approach for approximate integration over the posterior distribution of the weights, the hierarchical scale parameters of the priors, and the residual scale. Using a factorized posterior approximation we derive a computationally efficient algorithm, whose complexity scales similarly to an ensemble of independent sparse linear models. The approach enables flexible definition of weight priors with different sparseness properties such as independent Laplace priors with a common scale parameter or Gaussian automatic relevance determination (ARD) priors with different relevance parameters for all inputs. The approach can be extended beyond standard activation functions and NN model structures to form flexible nonlinear predictors from multiple sparse linear models. The effects of the hierarchical priors and the predictive performance of the algorithm are assessed using both simulated and real-world data. Comparisons are made to two alternative models with ARD priors: a Gaussian process with a NN covariance function and marginal maximum a posteriori estimates of the relevance parameters, and a NN with Markov chain Monte Carlo integration over all the unknown model parameters.

Keywords: expectation propagation, neural network, multilayer perceptron, linear model, sparse prior, automatic relevance determination

1. Introduction

Gaussian priors may not be the best possible choice for the input layer weights of a feed-forward neural network (NN) because allowing, *a priori*, a large weight w_j for a potentially

∗. Also at Donders Institute for Brain, Cognition, and Behavior, Radboud University Nijmegen, 6525 HR Nijmegen, The Netherlands.

irrelevant feature x_j may deteriorate the predictive performance. This behavior is analogous to a linear model because the input layer weights associated with each hidden unit of a multilayer perceptron (MLP) network can be interpreted as separate linear models whose outputs are combined nonlinearly in the next layer. Integrating over the posterior uncertainty of the unknown input weights mitigates the potentially harmful effects of irrelevant features but it may not be sufficient if the number of input features, or the total number of unknown variables, grows large compared with the number of observations. In such cases, an alternative strategy is required to suppress the effect of the irrelevant features. In this article we focus on a two-layer MLP model structure but aim to form a more general framework that can be used to combine linear models with sparsity-promoting priors using general activation functions and interaction terms between the hidden units.

A popular approach has been to apply hierarchical automatic relevance determination (ARD) priors (Mackay, 1995; Neal, 1996), where individual Gaussian priors are assigned for each weight, $w_j \sim \mathcal{N}(0, \alpha_{l_j})$, with separate variance hyperparameters α_{l_j} controlling the relevance of the group of weights related to each feature. Mackay (1995) described an ARD approach for NNs, where point estimates for the relevance parameters α_{l_j} along with other model hyperparameters, such as the noise level, are determined using a marginal likelihood estimate obtained by approximate integration over the weights with Laplace's method. Neal (1996) proposed an alternative Markov chain Monte Carlo (MCMC) approach, where approximate integration is performed over the posterior uncertainty of all the model parameters including w_j and α_{l_j}. In connection with linear models, various computationally efficient algorithms have been proposed for determining marginal likelihood based point estimates for the relevance parameters (Tipping, 2001; Qi et al., 2004; Wipf and Nagarajan, 2008). The point-estimate based methods, however, may suffer from overfitting because the maximum a posteriori (MAP) estimate of α_{l_j} may be close to zero also for relevant features as demonstrated by Qi et al. (2004). The same applies also for infinite neural networks implemented using Gaussian process (GP) priors when separate hyperparameters controlling the nonlinearity of each input are optimized (Williams, 1998; Rasmussen and Williams, 2006).

Recently, appealing surrogates for ARD priors have been presented for linear models. These approaches are based on sparsity favoring priors, such as the Laplace prior (Seeger, 2008) and the spike and slab prior (Hernández-Lobato et al., 2008, 2010). The methods utilize the expectation propagation (EP) (Minka, 2001b) algorithm to efficiently integrate over the analytically intractable posterior distributions. Importantly, these sparse priors do not suffer from similar overfitting as many ARD approaches because point estimates of feature specific parameters such as α_{l_j} are not used, but instead, approximate integration is done over the posterior uncertainty resulting from a sparse prior on the weights. Expectation propagation provides a useful alternative to MCMC for carrying out the approximate integration because it has been found computationally efficient and very accurate in many practical applications (Nickisch and Rasmussen, 2008; Hernández-Lobato et al., 2010).

In nonlinear regression, sparsity favoring Laplace priors have been considered for NNs, for instance, by Williams (1995), where the inference is performed using the Laplace approximation. However, a problem with the Laplace approximation is that the curvature of the log-posterior density at the posterior mode may not be well defined for all types of prior distributions, such as, the Laplace distribution whose derivatives are not continuous

at the origin (Williams, 1995; Seeger, 2008). Implementing a successful algorithm requires some additional approximations as described by Williams (1995), whereas with EP the implementation is straightforward since it relies only on expectations of the prior terms with respect to a Gaussian measure.

Another possibly undesired characteristic of the Laplace approximation is that it approximates the posterior mean of the unknowns with their MAP estimate and their posterior covariance with the negative Hessian of the posterior distribution at the mode. This local estimate may not represent well the overall uncertainty on the unknown variables and it may lead to worse predictive performance for example when the posterior distribution is skewed (Nickisch and Rasmussen, 2008) or multimodal (Jylänki et al., 2011). Furthermore, when there are many unknowns compared to the effective number of observations, which is typical in practical NN applications, the MAP solution may differ significantly from the posterior mean. For example, with linear models the Laplace prior leads to strictly sparse estimates with many zero weight values only when the MAP estimator of the weights is used. The posterior mean estimate, on the other hand, can result in many clearly nonzero values for the same weights whose MAP estimates are zero (Seeger, 2008). In such case the Laplace approximation underestimates the uncertainty of the feature relevances, that is, the joint mode is sharply peaked at zero but the bulk of the probability mass is distributed widely at nonzero weight values. Recently, it has also been shown that in connection with linear models the ARD solution is exactly equivalent to a MAP estimate of the coefficients obtained using a particular class of non-factorized coefficient prior distributions which includes models that have desirable advantages over MAP weight estimates (Wipf and Nagarajan, 2008; Wipf et al., 2011). This connection suggests that the Laplace approximation of the input weights with a sparse prior may possess more similar characteristics with the point-estimate based ARD solution compared to the posterior mean solution.

Our aim is to introduce the benefits of the sparse linear models (Seeger, 2008; Hernández-Lobato et al., 2008) into nonlinear regression by combining the sparse priors with a two-layer NN in a computationally efficient EP framework. We propose a logical extension of the linear regression models by adopting the algorithms presented for sparse linear models to MLPs with a linear input layer. For this purpose, the main challenge is constructing a reliable Gaussian EP approximation for the analytically intractable likelihood resulting from the NN observation model. In addition to the already discussed Laplace's method (see, e.g., Mackay, 1995; Williams, 1995), Gaussian approximations for NN models have been formed using variational Bayesian (VB) methods (Hinton and van Camp, 1993; Barber and Bishop, 1998; Honkela and Valpola, 2005), the extended Kalman filter (EKF) (Freitas, 1999), and the unscented Kalman filter (UKF) (Wan and van der Merwe, 2000). Alternative mean field approaches possessing similar characteristic with EP have been proposed by Opper and Winther (1996) and Winther (2001). Comparisons between Laplace approximation, VB, and MCMC have been made by Heskes et al. (2002) and Bakker et al. (2004). Recently, Graves (2011) proposed stochastic VB method applicable for multi-layered network architectures with various regularizing priors.

We extend the ideas from the UKF approach by utilizing similar decoupling approximations for the weights as proposed by Puskorius and Feldkamp (1991) for EKF-based inference and derive a computationally efficient algorithm that does not require numerical sigma point approximations for multi-dimensional integrals. We propose a posterior ap-

proximation that assumes the weights associated with the output-layer and each hidden unit independent. The complexity of the EP updates in the resulting algorithm scale linearly with respect to the number of hidden units and they require only one-dimensional numerical quadratures. The complexity of the posterior computations scale similarly to an ensemble of independent sparse linear models (one for each hidden unit) with one additional linear predictor associated with the output layer. It follows that all existing methodology on sparse linear models (e.g., methods that facilitate computations with large number of inputs) can be applied separately on the sparse linear model corresponding to each hidden unit. Furthermore, the complexity of the algorithm scales linearly with respect to the number of observations, which is beneficial for large data sets. The proposed approach can also be extended beyond standard activation functions and NN model structures, for example, by including a linear hidden unit or predefined interactions between the linear input-layer models.

In addition to generalizing the standard EP framework for sparse linear models we introduce an efficient EP approach for inference on the unknown hyperparameters, such as the noise level and the scale parameters of the weight priors. This framework enables flexible definition of different hierarchical priors, such as one common scale parameter for all input weights, or a separate scale parameter for all weights associated with one input variable (i.e., an integrated ARD prior). For example, assigning independent Laplace priors on the input weights with a common unknown scale parameter does not produce very sparse approximate posterior mean solutions, but, if required, more sparse solutions can be obtained by adjusting the common hyperprior of the scale parameters with the ARD specification. We show that by making independent approximations for the hyperparameters, the inference on them can be done simultaneously within the EP algorithm for the network weights, without the need for separate optimization steps which is common for many EP approaches for sparse linear models and GPs (Rasmussen and Williams, 2006; Seeger, 2008), as well as combined EKF and expectation maximization (EM) algorithms for NNs (Freitas, 1999). Additional benefits are achieved by introducing left-truncated priors on the output weights which prevent possible convergence problems in the EP algorithm resulting from inherent unidentifiability in the MLP network specification.

The main contributions of the paper can be summarized as:

- An efficiently scaling EP approximation for the non-Gaussian likelihood resulting from the MLP-model that requires numerical approximations only for one-dimensional integrals. We derive closed-form solutions for the parameters of the site term approximations, which can be interpreted as pseudo-observations of a linear model associated with each hidden unit (Sections 3.1–3.4 and Appendices A–E).

- An EP approach for integrating over the posterior uncertainty of the input weights and their hierarchical scale parameters assigned on predefined weight groups (Section 3.2.2). The approach could be applied also for sparse linear models to construct factorized approximations for predefined weight groups with shared hyperparameters.

- Approximate integration over the posterior uncertainty of the observation noise simultaneously within the EP algorithm for inference on the weights of a MLP-network (see Appendix D). Using factorizing approximations, the approach could be extended also

for approximate inference on other hyperparameters associated with the likelihood terms and could be applied, for example, in recursive filtering.

Key properties of the proposed model are first demonstrated with three artificial case studies in which comparisons are made with a neural network with infinitely many hidden units implemented as a GP regression model with a NN covariance function and an ARD prior (Williams, 1998; Rasmussen and Williams, 2006). Finally, the predictive performance of the proposed approach is assessed using four real-world data sets and comparisons are made with two alternative models with ARD priors: a GP with a NN covariance function where point estimates of the relevance hyperparameters are determined by optimizing their marginal posterior distribution, and a NN where approximate inference on all unknowns is done using MCMC (Neal, 1996).

2. The Model

We focus on two layer NNs where the unknown function value $f_i = f(\mathbf{x}_i)$ related to a d-dimensional input vector $\mathbf{x}_i$ is modeled as

$$f(\mathbf{x}_i) = \sum_{k=1}^{K} v_k g(\mathbf{w}_k^{\mathrm{T}} \mathbf{x}_i) + v_0, \tag{1}$$

where $g(x)$ is a nonlinear activation function, K the number of hidden units, and v_0 the output bias. Vector $\mathbf{w}_k = [w_{k,1}, w_{k,2}, ..., w_{k,d}]^{\mathrm{T}}$ contains the input layer weights related to hidden unit k and v_k is the corresponding output layer weight. Input biases can be introduced by adding a constant term to the input vectors $\mathbf{x}_k$. In the sequel, all weights are denoted by vector $\mathbf{z} = [\mathbf{w}^{\mathrm{T}}, \mathbf{v}^{\mathrm{T}}]^{\mathrm{T}}$, where $\mathbf{w} = [\mathbf{w}_1^{\mathrm{T}}, ..., \mathbf{w}_K^{\mathrm{T}}]^{\mathrm{T}}$, and $\mathbf{v} = [v_1, ..., v_K, v_0]^{\mathrm{T}}$.

In this work, we use the following activation function:

$$g(x) = \frac{1}{\sqrt{K}} \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right) = \frac{2}{\sqrt{K}} \left(\Phi(x) - 0.5\right), \tag{2}$$

where $\Phi(x) = \int_{-\infty}^{x} \mathcal{N}(t|0,1)dt$, and the scaling by $1/\sqrt{K}$ ensures that the prior variance of $f(\mathbf{x}_i)$ does not increase with K assuming fixed Gaussian priors on v_k and w_{kj}. We focus on regression problems with Gaussian observation model $p(y_i|f_i, \sigma^2) = \mathcal{N}(y_i|f_i, \sigma^2)$, where σ^2 is the noise variance. However, the proposed approach could be extended for other activation functions and observations models, for example, the probit model for binary classification.

2.1 Prior Definitions

To reduce the effects of irrelevant features, sparsity-promoting priors $p(w_j|\phi_{l_j})$ with hierarchical scale parameters $\phi_{l_j} \in \boldsymbol{\phi} = \{\phi_l\}_{l=1}^{L}$ are placed on the input layer weights. Each input weight w_j corresponding to the j:th element of $\mathbf{w}$ is assigned to one of L predefined groups indicated by the index variable $l_j \in \{1, ..., L\}$, and ϕ_l is a joint hyperparameter controlling the prior variance of all the input weights belonging to group l, that is, $\mathrm{Var}(w_j|\boldsymbol{\phi}) = \exp(\phi_l)$ for all $j \in \{1, ..., Kd|l_j = l\}$. We consider two types of input weight priors conditioned on

$\boldsymbol{\phi}$; Laplace priors and Gaussian priors defined as

$$\begin{aligned} \text{Laplace: } p(w_j|\phi_{l_j}) &= \frac{1}{\sqrt{2}\lambda_{l_j}} \exp\left(-\frac{\sqrt{2}}{\lambda_{l_j}}|w_j|\right) \\ \text{Gaussian: } p(w_j|\phi_{l_j}) &= \mathcal{N}(w_j|0, \lambda_{l_j}^2), \end{aligned} \tag{3}$$

where $\lambda_{l_j}^2 = \exp(\phi_{l_j}) = \text{Var}(w_j|\phi_{l_j})$. The grouping of the weights can be chosen freely and also other weight prior distributions can be used in place of the Laplace or Gaussian distributions in (3). The approximate inference on the variance parameters $\lambda_l^2 > 0$ is carried out using the log-transformed scale parameters $\phi_l = \log(\lambda_l^2) \in \mathbb{R}$ to facilitate an EP algorithm based on Gaussian approximating families as described in Section 3. By defining a suitable prior on the unknown group scales ϕ_l, useful hierarchical priors can be implemented on the input layer. Possible definitions include one common scale parameter for all inputs ($L = 1$), and a separate scale parameter for all the weights related to each feature, which implements an ARD prior ($L = d$). The traditional ARD setting is obtained by using a Gaussian distributions for $p(w_j|\phi_{l_j})$ as defined in (3) and choosing simply $l_j = 1, ..., d$ for all $j = K(k-1)+l_j$. We assign Gaussian hyperpriors to the scale parameters $\boldsymbol{\phi} = \{\phi_l\}_{l=1}^L$:

$$p(\phi_l) = \mathcal{N}(\mu_{\phi,0}, \sigma_{\phi,0}^2), \tag{4}$$

where a common mean $\mu_{\phi,0}$ and a variance $\sigma_{\phi,0}^2$ have been defined for all groups $l = 1, ..., L$. Definition (4) corresponds to a log-normal prior for the associated prior variance $\lambda_l^2 = \exp(\phi_l)$ of the weights in group l.

Because of the symmetry $g(x) = -g(-x)$ of the activation function, changing the signs of output weight v_k and the corresponding input weights $\mathbf{w}_k$ results in the same prediction $f(\mathbf{x})$. This unidentifiability may cause converge problems in the EP algorithm: if the approximate posterior probability mass of output weight v_k concentrates too close to zero, expected values of v_k and $\mathbf{w}_k$ may start fluctuating between small positive and negative values. Therefore the output weights are constrained to positive values by assigning left-truncated heavy-tailed priors to them:

$$p(v_k) = 2t_\nu(v_k|0, \sigma_{v,0}^2), \tag{5}$$

where $v_k \geq 0$, $k = 1, ..., K$, and $t_\nu(v_k|0, \sigma_{v,0}^2)$ denotes a Student-t distribution with degrees of freedom ν, mean zero, and scale parameter $\sigma_{v,0}^2$. The prior scale is fixed to $\sigma_{v,0}^2 = 1$ and the degrees of freedom to $\nu = 4$, which by experiments was found to produce sufficiently large posterior variations of $f(\mathbf{x})$ when the activation function is scaled according to (2) and the observations are normalized to zero mean and unit variance. The heavy-tailed prior (5) enables very large output weights if required, for example, when some hidden unit is forming almost a linear predictor (see, e.g, Section 4.2). A zero-mean Gaussian prior is assigned to the output bias: $p(v_0) = \mathcal{N}(0, \sigma_{v_0,0}^2)$, where the scale parameter is fixed to $\sigma_{v_0,0}^2 = 1$ because it was also found to be a sufficient simplification with the same data normalization conditions. The noise level σ^2 is assumed unknown and therefore a log-normal prior is assigned to it corresponding to a normal prior on $\theta = \log(\sigma^2)$:

$$p(\theta) = \mathcal{N}(\mu_{\theta,0}, \sigma_{\theta,0}^2) \tag{6}$$

with prior mean $\mu_{\theta,0}$ and variance $\sigma_{\theta,0}^2$.

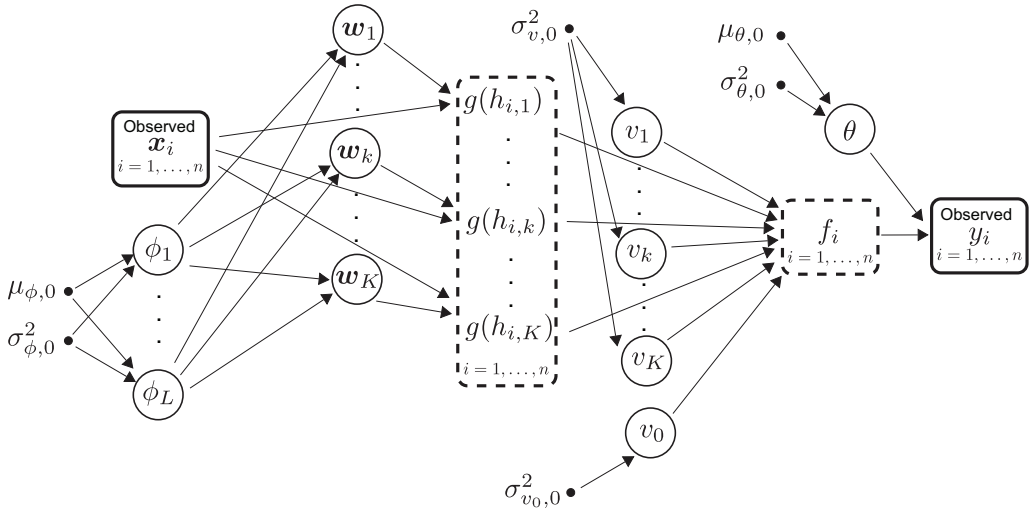


Figure 1: A directed graph representing the joint distribution of all the model parameters written in equation (7) resulting from the observation model and prior definitions summarized in Section 2. The observed variables indexed with $i = 1, ..., n$ are denoted with boxes, the unobserved random variables are denoted with circles, and the fixed prior parameters are denoted with dots. For each input $\mathbf{x}_i$, $i = 1, ..., n$, two intermediate random variables are visualized: The linear hidden unit activations defined as $h_{i,k} = \mathbf{w}_k^{\mathrm{T}}\mathbf{x}_i$ and the latent function value given by $f_i = \sum_{k=1}^{K} v_k g(h_{i,k}) + v_0$.

2.2 The Posterior Distribution

Given the previous prior definitions and a set of n observations $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, where $\mathbf{y} = [y_1, ..., y_n]^{\mathrm{T}}$ and $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]^{\mathrm{T}}$, the joint posterior distribution of $\mathbf{w}$, $\mathbf{v}$, $\boldsymbol{\phi}$, and θ is given by

$$p(\mathbf{w}, \mathbf{v}, \theta, \boldsymbol{\phi}|\mathcal{D}, \boldsymbol{\gamma}) = Z^{-1} \prod_{i=1}^{n} p(y_i|f_i, \theta) \prod_{j=1}^{Kd} p(w_j|\phi_{l_j}) \prod_{k=0}^{K} p(v_k|\boldsymbol{\gamma}) \prod_{l=1}^{L} p(\phi_l|\boldsymbol{\gamma}) p(\theta|\boldsymbol{\gamma}), \tag{7}$$

where $f_i = \sum_{k=1}^{K} v_k g(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i) + v_0$, $\boldsymbol{\gamma} = \{\sigma^2_{v,0}, \sigma^2_{v_0,0}, \mu_{\phi,0}, \sigma^2_{\phi,0}, \mu_{\theta,0}, \sigma^2_{\theta,0}\}$ contains all the fixed hyperparameters and Z is the marginal likelihood of the observations conditioned on $\boldsymbol{\gamma}$:

$$Z = p(\mathbf{y}|\mathbf{X}, \boldsymbol{\gamma}) = \int p(\mathbf{y}|\mathbf{w}, \mathbf{v}, \mathbf{X}, \theta) p(\mathbf{w}|\boldsymbol{\phi}) p(\mathbf{v}|\boldsymbol{\gamma}) p(\boldsymbol{\phi}|\boldsymbol{\gamma}) p(\theta|\boldsymbol{\gamma}) d\mathbf{w} d\mathbf{v} d\boldsymbol{\phi} d\theta. \tag{8}$$

Figure 1 shows a directed graph representing the joint distribution (7) where we have also included intermediate random variables $h_{i,k} = \mathbf{w}_k^{\mathrm{T}}\mathbf{x}_i$ and $f_i = \sum_{k=1}^{K} v_k g(h_{i,k}) + v_0$ related to each data point to clarify the upcoming description of the approximate inference scheme.

2.3 General Properties of the Model

The values of the hyperparameters $\lambda_l = \exp(\phi_l/2)$ and $\sigma^2_{v,0}$ affect the smoothness properties of the model in different ways. In the following discussion we first assume that there is only one input scale parameter λ_1 ($L = 1$) for clarity. Choosing a smaller value for λ_1 penalizes more strongly for larger input weights and produces smoother functions with respect to changes in the input features. More precisely, in the two-layer NN model (1) the magnitudes of the input weights (or equivalently the ARD scale parameters) are related to the nonlinearity of the latent function $f(\mathbf{x})$ with respect to the corresponding inputs $\mathbf{x}$. Strong nonlinearities require large input weights whereas almost a linear function is obtained with a very large output weight and very small input weights for a certain hidden unit (see Section 4.2 for illustration).

Because the values of the activation function $g(x)$ are constrained to the interval $[-1, 1]$, hyperparameter $\sigma^2_{v,0}$ controls the overall magnitude of the latent function $f(\mathbf{x})$. Larger values of $\sigma^2_{v,0}$ increase the magnitude of the steps the hidden unit activation $v_k g(\mathbf{w}_k^{\mathrm{T}}\mathbf{x})$ can take in the direction of weight vector $\mathbf{w}_k$ in the feature space. Choosing a smaller value for $\sigma^2_{v,0}$ can improve the predictive performance by constraining the overall flexibility of the model but too small value can prevent the model from explaining all the variance in the target variable y. In this work, we keep $\sigma^2_{v,0}$ fixed to a sufficiently large value and infer λ_l from data promoting simultaneously smoother solutions with the prior on $\phi_l = \log(\lambda_l^2)$. If only one common scale parameter ϕ_1 is used, the sparsity-inducing properties of the prior depend on the shape of the joint distribution $p(\mathbf{w}|\lambda_1) = \prod_j p(w_j|\lambda_1)$ resulting from the choice of the prior terms (3). By decreasing $\mu_{\phi,0}$, we can favor smaller input weight values overall, and with $\sigma^2_{\phi,0}$, we can adjust the thickness of the tails of $p(\mathbf{w}|\lambda_1)$. On the other hand, if individual scale parameters are assigned for all inputs according to the ARD setting, a family of sparsity-promoting priors is obtained by adjusting $\mu_{\phi,0}$ and $\sigma^2_{\phi,0}$. If $\mu_{\phi,0}$ is set to a small value, say 0.01, and $\sigma^2_{\phi,0}$ is increased, sparser solutions are favored by allocating increasing amounts of prior probability on the axes of the input weight space. A sparse prior could be introduced also on the output weights v_k to suppress redundant hidden units but this was not found necessary in the experiments because the proposed EP updates have a fixed point at $\mathrm{E}(v_k) = 0$ and $\mathrm{E}(\mathbf{w}_k) = \mathbf{0}$ for each $k = 1, ..., K$ and during the iterations unused hidden units are gradually driven towards zero (see Section 3.5.3 and Appendix E).

3. Approximate Inference

In this section, we describe how approximate Bayesian inference on the unknown model parameters $\mathbf{w}$, $\mathbf{v}$, θ, and $\boldsymbol{\phi} = [\phi_1, ..., \phi_L]^{\mathrm{T}}$ can be done efficiently using EP. First, in Section 3.1, we describe how the posterior approximation can be formed using local factorized site approximations and then, in Section 3.2, we summarize a general EP algorithm suitable for determining their parameters. In Section 3.3, we discuss suitable parametric forms for the local site approximations and properties of the resulting approximate posterior distributions. We discuss the various computational blocks required in the EP algorithm first in Section 3.4 and give detailed descriptions of the methods in Appendices A– I. Finally, we give an algorithm description with references to the different building blocks in Section 3.5.

3.1 The Posterior Approximation

To form an analytically tractable approximation for the posterior distribution (7), all the non-tractable likelihood and prior terms are approximated with unnormalized Gaussian site functions, which provide a suitable approximating family for random vectors defined in the real vector space. The Gaussian distribution is a commonly used approximate family for the weights of linear models and neural networks (see, e.g, Seeger, 2008; Freitas, 1999) but in our case it is also a suitable family for the hyperparameters $\phi_l = \log(\lambda_l^2)$ and $\theta = \log(\sigma^2)$, because of the logarithmic transformations and Gaussian prior definitions (4) and (6). Alternatively one could consider other exponential family distributions such as the inverse-gamma for the variance parameters λ_l^2 and σ^2 directly. We approximate the posterior distribution (7) as

$$\begin{aligned} p(\mathbf{z}, \theta, \boldsymbol{\phi}|\mathcal{D}) &= Z^{-1} \prod_{i=1}^{n} p(y_i|f_i, \theta) \prod_{j=1}^{Kd} p(w_j|\phi_{l_j}) \prod_{k=0}^{K} p(v_k) \prod_{l=1}^{L} p(\phi_l) p(\theta) \qquad (9) \\ &\approx Z_{\mathrm{EP}}^{-1} \prod_{i=1}^{n} \tilde{Z}_{y,i} \tilde{t}_{\mathbf{z},i}(\mathbf{z}) \tilde{t}_{\theta,i}(\theta) \prod_{j=1}^{Kd} \tilde{Z}_{w,j} \tilde{t}_{w,j}(w_j) \tilde{t}_{\phi,j}(\phi_{l_j}) \prod_{k=1}^{K} \tilde{Z}_{v,k} \tilde{t}_{v,k}(v_k) p(v_0) \prod_{l=1}^{L} p(\phi_l) p(\theta), \end{aligned}$$

where $\mathbf{z} = [\mathbf{w}^{\mathrm{T}}, \mathbf{v}^{\mathrm{T}}]^{\mathrm{T}}$ and Z_{EP} is the EP approximation of the marginal likelihood $Z = p(\mathbf{y}|\mathbf{X}, \boldsymbol{\gamma})$ defined in (8) (for details, see Appendix I). We have excluded the fixed hyperparameters $\boldsymbol{\gamma} = \{\sigma_{v,0}^2, \sigma_{v_0,0}^2, \mu_{\phi,0}, \sigma_{\phi,0}^2, \mu_{\theta,0}, \sigma_{\theta,0}^2\}$ from the notation in equation (9) and will do that also in the following sections, because they are assumed to be fixed during the EP iterations.

3.1.1 The Likelihood Term Approximations

The likelihood terms that depend on the weights $\mathbf{z} = [\mathbf{w}^{\mathrm{T}}, \mathbf{v}^{\mathrm{T}}]^{\mathrm{T}}$ through f_i according to (1) are approximated with a product of two unnormalized Gaussian site functions:

$$p(y_i|f_i, \theta) \approx \tilde{Z}_{y,i} \tilde{t}_{\mathbf{z},i}(\mathbf{z}) \tilde{t}_{\theta,i}(\theta), \qquad (10)$$

where $\tilde{Z}_{y,i}$ is a scalar scaling parameter. Because the approximate posterior correlations between the components of $\mathbf{z}$ are defined by the likelihood site approximations $\tilde{t}_{\mathbf{z},i}(\mathbf{z})$, their parametric structure is crucial for computationally efficient EP updates especially when K and d are large. Section 3.3 discusses alternative structures for $\tilde{t}_{\mathbf{z},i}(\mathbf{z})$ and proposes factorized Gaussian site approximations of the form

$$\tilde{t}_{\mathbf{z},i}(\mathbf{z}) = \tilde{t}_{\mathbf{v},i}(\mathbf{v}) \prod_{k=1}^{K} \tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k) \qquad (11)$$

that result in independent approximations for $\mathbf{v}$, $\mathbf{w}_1$,..., $\mathbf{w}_K$ and computationally more efficient EP updates compared to fully-coupled site approximations. The second likelihood site approximation dependent on the scalar $\theta = \log \sigma^2$ is parameterized as

$$\tilde{t}_{\theta,i}(\theta) = \exp\left(-\frac{1}{2}\tilde{\sigma}_{\theta,i}^{-2}\theta^2 + \tilde{\mu}_{\theta,i}\tilde{\sigma}_{\theta,i}^{-2}\theta\right) \propto \mathcal{N}(\theta|\tilde{\mu}_{\theta,i}, \tilde{\sigma}_{\theta,i}^2), \qquad (12)$$

where the site parameters $\tilde{\mu}_{\theta,i}$ and $\tilde{\sigma}^2_{\theta,i}$ control the location and the scale of the site approximation, respectively. Combined with the known Gaussian prior term $p(\theta)$ this results in a Gaussian posterior approximation for θ that corresponds to a log-normal approximation for σ^2.

3.1.2 The Prior Term Approximations

The prior terms of the output weights v_k, for $k = 1, ..., K$, are approximated with

$$p(v_k) \approx \tilde{Z}_{v,k}\tilde{t}_{v,k}(v_k) \propto \mathcal{N}(v_k|\tilde{\mu}_{v,k}, \tilde{\sigma}^2_{v,k}), \tag{13}$$

where $\tilde{Z}_{v,k}$ is a scalar scaling parameter, $\tilde{t}_{v,k}(v_k)$ has a similar exponential form as (12), and the site parameters $\tilde{\mu}_{v,k}$ and $\tilde{\sigma}^2_{v,k}$ control the location and scale of the site approximation, respectively. If the prior scales ϕ_l are assumed unknown, the prior terms of the input weights $\{w_j|j = 1, ..., Kd\}$, are approximated with

$$p(w_j|\phi_{l_j}) \approx \tilde{Z}_{w,j}\tilde{t}_{w,j}(w_j)\tilde{t}_{\phi,j}(\phi_{l_j}) \propto \mathcal{N}(w_j|\tilde{\mu}_{w,j}, \tilde{\sigma}^2_{w,j})\mathcal{N}(\phi_{l_j}|\tilde{\mu}_{\phi,j}, \tilde{\sigma}^2_{\phi,j}), \tag{14}$$

where a factorized site approximation with location parameters $\tilde{\mu}_{w,j}$ and $\tilde{\mu}_{\phi,j}$, and scale parameters $\tilde{\sigma}^2_{w,j}$ and $\tilde{\sigma}^2_{\phi,j}$, is assigned to weight w_j and the associated scale parameter ϕ_{l_j}, respectively. A similar exponential form to equation (12) is assumed for both $\tilde{t}_{w,j}(w_j)$ and $\tilde{t}_{\phi,j}(\phi_{l_j})$. This factorizing site approximation results in independent posterior approximations for $\mathbf{w}$ and each component of $\boldsymbol{\phi}$.

3.1.3 Forming the Joint Posterior Approximation

Multiplying the site approximations together according to (9) and normalizing the resulting expression gives the following factorized posterior approximation:

$$p(\mathbf{z}, \theta, \boldsymbol{\phi}|\mathcal{D}, \gamma) \approx q(\mathbf{z})q(\theta)\prod_{l=1}^{L} q(\phi_l), \tag{15}$$

where

$$\begin{aligned}
q(\mathbf{z}) &= \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto \prod_{i=1}^{n} \tilde{t}_{\mathbf{z},i}(\mathbf{z}) \prod_{j=1}^{Kd} \tilde{t}_{w,j}(w_j) \prod_{k=1}^{K} \tilde{t}_{v,k}(v_k)p(v_0) \\
q(\phi_l) &= \mathcal{N}(\phi_l|\mu^2_{\phi_l}, \sigma^2_{\phi_l}) \propto \prod_{j=1, l_j=l}^{Kd} \tilde{t}_{\phi,j}(\phi_l)p(\phi_l) \quad l = 1, ..., L \\
q(\theta) &= \mathcal{N}(\theta|\mu^2_\theta, \sigma^2_\theta) \propto \prod_{i=1}^{n} \tilde{t}_{\theta,i}(\theta)p(\theta).
\end{aligned} \tag{16}$$

Multiplying the likelihood site approximations $\tilde{t}_{\theta,i}(\theta)$ defined in (12) together according to (16) results in a Gaussian approximation $q(\theta) = \mathcal{N}(\mu_\theta, \sigma^2_\theta)$, where the mean and variance are given by

$$\sigma^2_\theta = \left(\sum_{i=1}^{n} \tilde{\sigma}^{-2}_{\theta,i} + \sigma^{-2}_{\theta,0}\right)^{-1} \quad \text{and} \quad \mu_\theta = \sigma^2_{\theta,0}\left(\sum_{i=1}^{n} \tilde{\sigma}^{-2}_{\theta,i}\tilde{\mu}_{\theta,i} + \sigma^{-2}_{\theta,0}\mu_{\theta,0}\right). \tag{17}$$

Similarly, combining the prior site approximations $\tilde{t}_{\phi,j}(\phi_{l_j})$ from (14) results in a Gaussian approximation $q(\phi_l) = \mathcal{N}(\mu_{\phi_l}, \sigma^2_{\phi_l})$ with the mean and variance given by

$$\sigma^2_{\phi_l} = \left(\sum_{j=1, l_j=l}^{Kd} \tilde{\sigma}^{-2}_{\phi,j} + \sigma^{-2}_{\phi_l,0} \right)^{-1} \quad \text{and} \quad \mu_{\phi_l} = \sigma^2_{\phi,0} \left(\sum_{j=1, l_j=l}^{Kd} \tilde{\sigma}^{-2}_{\phi,j} \tilde{\mu}_{\phi,j} + \sigma^{-2}_{\phi,0} \mu_{\phi,0} \right). \tag{18}$$

Note that in (18) only the approximations of the prior terms $p(w_j|\phi_{l_j})$ linked to scale parameter ϕ_l via $l_j = l$ affect the summations.

Adopting the factorized site approximation (11) results in a posterior approximation where the weights related to the different hidden units and the output layer decouple:

$$q(\mathbf{z}) = q(\mathbf{v}) \prod_{k=1}^{K} q(\mathbf{w}_k), \tag{19}$$

where

$$q(\mathbf{w}_k) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{w}_k}, \boldsymbol{\Sigma}_{\mathbf{w}_k}) \propto \prod_{i=1}^{n} \tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k) \prod_{j=m+1}^{m+d} \tilde{t}_{w,j}(w_j) \quad k = 1, ..., K, \text{ and } m = K(k-1)$$

$$q(\mathbf{v}) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{v}}, \boldsymbol{\Sigma}_{\mathbf{v}}) \propto \prod_{i=1}^{n} \tilde{t}_{\mathbf{v},i}(\mathbf{v}) \prod_{k=1}^{K} \tilde{t}_{v,k}(v_k) p(v_0). \tag{20}$$

The exact parametric forms of the Gaussian posterior approximations $q(\mathbf{z})$, $q(\mathbf{v})$, and $q(\mathbf{w}_k)$ are presented in Section 3.3.

3.2 Expectation Propagation

The parameters of the local site approximations that define the approximate posterior distribution (15) are determined using the EP algorithm (Minka, 2001b). In the following, we give general descriptions of the EP updates separately for the likelihood terms and the weight prior terms.

3.2.1 EP Updates for the Likelihood Terms

Here we consider the procedure for updating the likelihood sites $\tilde{t}_{\mathbf{z},i}$ and $\tilde{t}_{\theta,i}$ defined in equations (10)–(12) and assume that the prior site approximations (13) and (14) are kept fixed. Because the likelihood terms $p(y_i|f_i, \theta)$ do not depend on $\boldsymbol{\phi}$ and the posterior approximation can be factorized as $q(\mathbf{z}, \theta, \boldsymbol{\phi}) = q(\mathbf{z})q(\theta)q(\boldsymbol{\phi})$, we need to consider only the approximations for $\mathbf{z}$ and θ. Furthermore, independent approximations for $\mathbf{w}_k$ and $\mathbf{v}$ are obtained by using (11) and (19) in place of $\tilde{t}_{\mathbf{z},i}$ and $q(\mathbf{z})$, respectively.

At each iteration, first a proportion η of the i:th site term is removed from the posterior approximation to obtain a cavity distribution:

$$q_{-i}(\mathbf{z}, \theta) = q_{-i}(\mathbf{z}) q_{-i}(\theta) \propto q(\mathbf{z}) q(\theta) \big(\tilde{Z}_{y,i} \tilde{t}_{\mathbf{z},i}(\mathbf{z}) \tilde{t}_{\theta,i}(\theta) \big)^{-\eta}, \tag{21}$$

where $\eta \in (0, 1]$ is a fraction parameter that can be adjusted to implement fractional (or power) EP updates (Minka, 2004, 2005) (regular EP updates are obtained by setting $\eta = 1$).

Then, the removed fraction of the i:th site approximation is replaced with a corresponding fraction of the exact likelihood term to form a tilted distribution

$$\hat{p}_i(\mathbf{z}, \theta) = \hat{Z}_i^{-1} q_{-i}(\mathbf{z}, \theta) p(y_i|\mathbf{z}, \theta, \mathbf{x}_i)^\eta, \tag{22}$$

where $\hat{Z}_i = \int q_{-i}(\mathbf{z}, \theta) p(y_i|\mathbf{z}, \theta, \mathbf{x}_i)^\eta d\mathbf{z} d\theta$ is a normalization factor. The tilted distribution (22) can be regarded as a more refined approximation to the true posterior distribution assuming that all the other local approximations that form the cavity distribution are sufficiently accurate. Next, the algorithm attempts to match the approximate posterior distribution $q(\mathbf{z}, \theta)$ with $\hat{p}_i(\mathbf{z}, \theta)$ by finding first a member of the chosen approximate family, $\hat{q}_i(\mathbf{z}, \theta) = \hat{q}_i(\mathbf{z})\hat{q}_i(\theta)$, that satisfies

$$\hat{q}_i(\mathbf{z}, \theta) = \arg\min_{q_i} \mathrm{KL}\left(\hat{p}_i(\mathbf{z}, \theta) || q_i(\mathbf{z}, \theta)\right),$$

where KL denotes the Kullback-Leibler divergence. When $q(\mathbf{z}, \theta)$ is chosen to be a Gaussian distribution this is equivalent to setting the approximate mean vectors and covariance matrices that determine $\hat{q}_i(\mathbf{z})$ and $\hat{q}_i(\theta)$ equal to the marginal mean vectors and covariance matrices of $\mathbf{z}$ and θ with respect to $\hat{p}_i$. Then, the parameters of the i:th site terms are updated so that the new posterior approximation $q(\mathbf{z}, \theta)^{\text{new}}$ that would result from the site update is consistent with $\hat{q}_i(\mathbf{z}, \theta)$:

$$q(\mathbf{z}, \theta)^{\text{new}} = \hat{Z}_i^{-1} q_{-i}(\mathbf{z}, \theta)\left(\tilde{Z}_{y,i}^{\text{new}} \tilde{t}_{\mathbf{z},i}(\mathbf{z})^{\text{new}} \tilde{t}_{\theta,i}(\theta)^{\text{new}}\right)^\eta = \hat{q}_i(\mathbf{z}, \theta). \tag{23}$$

Finally, the posterior approximation $q(\mathbf{z}, \theta)$ is updated according to the changes in the site parameters. These steps are repeated for all sites in some suitable order until convergence, that is, when all the n tilted distributions (22) are consistent with the approximation $q(\mathbf{z}, \theta)$. From now on, we refer to the previously described EP update scheme as sequential EP. If the update of the posterior approximation $q(\mathbf{z}, \theta)$ in the last step is done only after new parameter values have been determined for all sites (in this case the n likelihood term approximations), we refer to parallel EP (see, e.g., Gerven et al., 2009).

The actual numerical values of the normalization parameters $\tilde{Z}_{y,i}$ (or $\tilde{Z}_{v,k}$ and $\tilde{Z}_{w,j}$ with the prior term updates in Section 3.2.2) are not required during the iterations of the EP algorithm because with exponential approximating families it suffices to update only the natural parameters of $q(\mathbf{z}, \theta)$ so that the expected sufficient statistics of $q(\mathbf{z}, \theta)$ are matched with $\hat{p}_i(\mathbf{z}, \theta)$. However, the effect of the normalization parameters must be taken into account if one wishes to form an EP approximation for the marginal likelihood $Z = p(\mathbf{y}|\mathbf{X}, \boldsymbol{\gamma})$ (see Appendix I). This estimate could be utilized to compare models or to alternatively determine type-II MAP estimates for the hyperparameters $\boldsymbol{\gamma}$ or parameters θ and $\{\phi_l\}_{l=1}^L$ in case they are not inferred within the EP framework.

Setting the fraction parameter to $\eta = 1$ corresponds to regular EP updates whereas choosing a smaller value produces a slightly different approximation that puts less emphasis on preserving all the nonzero probability mass of the tilted distributions (Minka, 2005). Consequently, choosing a smaller value of $\eta < 1$ tries to represent possible multimodalities in (22) but ignores modes far away from the main probability mass, which results in a tendency to underestimate variances. Taking the limit $\eta \to 0$ corresponds to minimizing the reverse KL divergence $\hat{q}_i(\mathbf{z}, \theta) = \arg\min_{q_i} \mathrm{KL}\left(q_i(\mathbf{z}, \theta) || \hat{p}_i(\mathbf{z}, \theta)\right)$ resulting in a local approximation that tries to represent only one mode of the tilted distribution. In practice, decreasing η can

improve the overall numerical stability of the algorithm and alleviate convergence problems resulting from possible multimodalities in case the unimodal approximation is not a good fit for the true posterior distribution (Minka, 2001b, 2005; Seeger, 2008; Jylänki et al., 2011).

In case the likelihood term approximations are updated with $\eta = 1$, the cavity distribution (21) can be interpreted as an approximation to the leave-one-out (LOO) posterior distribution where the contribution of the i:th likelihood term $p(y_i|f_i, \theta)$ is removed from $q(\mathbf{z}, \theta)$. Furthermore, the normalization factor of the tilted distribution (22) can be thought of as an approximation to the LOO predictive density of the excluded data point y_i: $p(y_i|\mathcal{D}_{-i}, \mathbf{x}_i) \approx \hat{Z}_i = \int q_{-i}(\mathbf{z}, \theta) p(y_i|\mathbf{z}, \theta, \mathbf{x}_i) d\mathbf{z} d\theta$. In Section 3.5 we use these normalization factors as a one measure of the model fit.

There is no theoretical convergence guarantee for the standard EP algorithm but damping the site parameter updates can help to achieve convergence in harder problems (Minka and Lafferty, 2002; Heskes and Zoeter, 2002).[1] With exponential approximate families damping can be understood as leaving part of the old site approximation in the posterior approximation according to $q(\mathbf{z})^{\text{new}} = q(\mathbf{z}) \tilde{t}_{\mathbf{z},i}(\mathbf{z})^{-\delta} (\tilde{t}_{\mathbf{z},i}(\mathbf{z})^{\text{new}})^{\delta}$, where $\delta \in (0, 1]$ is a damping factor. This corresponds to updating the site parameters (in their natural exponential forms) to a convex combination of their old values and the new values resulting from (23) (see, e.g, equation (60) in Appendix E). The convergence problems are usually seen as oscillations over iterations in the site parameter values and they may occur, for example, if there are inaccuracies in the tilted moment evaluations, or if the approximate distribution is not a suitable proxy for the true posterior because of multimodalities. With damping, smaller steps are taken in the site parameter updates, which can reduce the oscillations and alleviate numerical problems caused by ill-conditioned approximate posterior (or cavity) covariance matrices.

3.2.2 EP Updates for the Weight Prior Terms

Assuming that the likelihood term approximations (10) are fixed, the EP algorithm for determining the parameters of the prior term approximations (13) and (14) can be implemented in the same way as with sparse linear models (see, e.g., Seeger, 2008; Hernández-Lobato et al., 2008; Gerven et al., 2010; Hernández-Lobato et al., 2013).

To derive EP updates for the prior term approximations of the output weights $\mathbf{v}$ assuming the factorized approximation (19), we need to consider only the prior site approximations $p(v_k) \approx \tilde{Z}_{v,k} \tilde{t}_{v,k}(v_k)$ from (13) and the approximate posterior $q(\mathbf{v}) = \mathcal{N}(\mathbf{v}|\boldsymbol{\mu}_{\mathbf{v}}, \boldsymbol{\Sigma}_{\mathbf{v}})$ defined in equation (20). All approximate posterior information from the observations $\mathcal{D} = \{\mathbf{y}, \mathbf{X}\}$ and the priors on the input weights $\mathbf{w}$ are conveyed by the likelihood term approximations $\{\tilde{t}_{\mathbf{v},i}(\mathbf{v})\}_{i=1}^{n}$ that are determined during the EP iterations for the likelihood sites which is why a standard EP implementation (see, e.g., Seeger, 2008) can be readily applied to determine $t_{\mathbf{v},k}(v_k)$ by using $\prod_{i=1}^{n} \tilde{t}_{\mathbf{v},i}(\mathbf{v})$ as an approximate Gaussian likelihood. The EP updates can be derived by following the same general scheme that was described in 3.2.1. Because the prior terms $p(v_k)$ depend only on one random variable v_k, deriving the parameters of the cavity distributions $q_{-k}(v_k) \propto q(v_k) \tilde{t}_{v,k}(v_k|\tilde{\mu}_{v,k}, \tilde{\sigma}^2_{v,k})^{-\eta}$ and updates for the site

1. Alternative provably convergent double-loop algorithms exist but usually they require more computational effort in the form of additional inner-loop iterations (Minka, 2001b; Heskes and Zoeter, 2002; Opper and Winther, 2005; Seeger and Nickisch, 2011).

parameters $\tilde{\mu}_{v,k}$ and $\tilde{\sigma}^2_{v,k}$ require only manipulating univariate Gaussians. The moments of the tilted distribution $\hat{p}_k(v_k) \propto q_{-k}(v_k)p(v_k)^\eta$ can be computed either analytically or using one-dimensional numerical quadratures depending on the functional form of the exact prior term $p(v_k)$. Appendix F presents an algorithm description that can used to implement these steps in practice.

To derive EP updates for the site approximations of the hierarchical prior terms $p(w_j|\phi_{l_j})$ assuming the factorized approximation (19), we need to consider the approximate posterior distributions $q(\mathbf{w}_k) = \mathcal{N}(\mathbf{w}_k|\boldsymbol{\mu}_{\mathbf{w}_k}, \boldsymbol{\Sigma}_{\mathbf{w}_k})$ from (20) for $k = 1, ..., K$ together with the corresponding prior site approximations $p(w_j|\phi_{l_j}) \approx \tilde{Z}_{w,j}\tilde{t}_{w,j}(w_j)\tilde{t}_{\phi,j}(\phi_{l_j})$ from (14) for indices $j = (k-1)+1, ..., (k-1)+d$. Separate EP algorithms can be run for each of the hidden units if they are not coupled through shared scale parameters ϕ_l. Since all approximate posterior information from the observations $\mathcal{D}$ is conveyed by the likelihood term approximations $\{\tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k)\}_{i=1}^n$ that are determined during the EP updates for the likelihood sites, the EP updates to determine $\tilde{Z}_{w,j}$, $\tilde{t}_{w,j}(w_j)$, and $\tilde{t}_{\phi,j}(\phi_{l_j})$ can now be derived using $\prod_{i=1}^n \tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k)$ as an approximate Gaussian likelihood for $\mathbf{w}_k$, and $\{p(\phi_l)\}_{l=1}^L$ as fixed priors for ϕ_l in the posterior approximations $q(\mathbf{w}_k)$ and $q(\phi_l)$ defined in (16) and (20). EP algorithms for sparse linear models that operate on site terms depending on a nonlinear combination of multiple random variables have been described earlier, e.g., by Hernández-Lobato et al. (2008) and Gerven et al. (2009).

Because the j:th exact prior term (3) depends on both the weight w_j and the corresponding log-transformed scale parameter ϕ_{l_j}, the j:th cavity distribution is formed by removing a fraction η of both site approximations $\tilde{t}_{w,j}(w_j)$ and $\tilde{t}_{\phi,j}(\phi_{l_j})$ from the approximate posterior:

$$q_{-j}(w_j, \phi_{l_j}) = q_{-j}(w_j)q_{-j}(\phi_{l_j}) \propto q(w_j)q(\phi_{l_j}) \left(\tilde{Z}_{w,j}\tilde{t}_{w,j}(w_j)\tilde{t}_{\phi,j}(\phi_{l_j})\right)^{-\eta}, \tag{24}$$

where $q(w_j)$ is the j:th marginal density extracted from the corresponding approximation $q(\mathbf{w}_k)$ and $q(\phi_{l_j})$ is defined by (16) and (18). The j:th tilted distribution is formed by replacing the removed site terms with a fraction η of the exact prior term $p(w_j|\phi_{l_j})$:

$$\hat{p}_j(w_j, \phi_{l_j}) = \hat{Z}^{-1}_{w,j} q_{-j}(w_j)q_{-j}(\phi_{l_j})p(w_j|\phi_{l_j})^\eta \equiv \hat{q}(w_j, \phi_{l_j}), \tag{25}$$

where $\hat{q}(w_j, \phi_{l_j})$ is a Gaussian approximation formed by determining the mean and covariance of $\hat{p}_j(w_j, \phi_{l_j})$. The site parameters are updated so that the resulting posterior approximation is consistent with the marginal means and variances of $\hat{q}(w_j, \phi_{l_j})$:

$$\hat{q}_j(w_j)\hat{q}_j(\phi_{l_j}) = \hat{Z}^{-1}_{w,j} q_{-j}(w_j)q_{-j}(\phi_{l_j}) \left(\tilde{Z}^{\text{new}}_{w,j}\tilde{t}_{w,j}(w_j)^{\text{new}}\tilde{t}_{\phi,j}(\phi_{l_j})^{\text{new}}\right)^{\eta}. \tag{26}$$

Because of the factorized approximation, the cavity computations (24) and the site updates (26) require only scalar operations. An algorithm description implementing the update steps (24)–(26) is presented in Appendix F.

Determining the moments of (25) can be done efficiently using one-dimensional quadratures if the means and variances of w_j with respect to the conditional distribution $\hat{p}_j(w_j|\phi_{l_j})$ can be determined analytically. This can be readily done when $p(w_j|\phi_{l_j})$ is a Laplace distribution or a finite mixture of Gaussians. The marginal tilted distribution for ϕ_{l_j} is given

by

$$\begin{aligned}\hat{p}(\phi_{l_j}) &= \hat{Z}_{w,j}^{-1} \int q_{-j}(w_j) q_{-j}(\phi_{l_j}) p(w_j|\phi_{l_j})^\eta dw_j = \hat{Z}_{w,j}^{-1} Z(\phi_{l_j}, \eta) q_{-j}(\phi_{l_j}) \\ &\approx \mathcal{N}(\phi_{l_j}|\hat{\mu}_{\phi,j}, \hat{\sigma}^2_{\phi,j}),\end{aligned} \tag{27}$$

where it is assumed that $Z(\phi_{l_j}, \eta) = \int q_{-j}(w_j) p(w_j|\phi_{l_j})^\eta dw_j$ can be calculated analytically. The normalization term $\hat{Z}_{w,j} = \int Z(\phi_{l_j}, \eta) q_{-j}(\phi_{l_j}) d\phi_{l_j}$, the marginal mean $\hat{\mu}_{\phi,j}$, and the variance $\hat{\sigma}^2_{\phi,j}$ can be determined using numerical quadratures.

The marginal tilted mean and variance of w_j can be determined by integrating numerically the conditional expectations of w_j and w_j^2 over $\hat{p}_j(\phi_{l_j})$:

$$\begin{aligned}\mathrm{E}(w_j) &= \hat{Z}_{w,j}^{-1} \int w_j \hat{p}_j(w_j|\phi_{l_j}) Z(\phi_{l_j}, \eta) q_{-j}(\phi_{l_j}) dw_j d\phi_{l_j} = \int \mathrm{E}(w_j|\phi_{l_j}, \eta) \hat{p}_j(\phi_{l_j}) d\phi_{l_j} \\ \mathrm{Var}(w_j) &= \int \mathrm{E}(w_j^2|\phi_{l_j}, \eta) \hat{p}_j(\phi_{l_j}) d\phi_{l_j} - \mathrm{E}(w_j)^2,\end{aligned} \tag{28}$$

where $\hat{p}_j(w_j|\phi_{l_j}) = Z(\phi_{l_j}, \eta)^{-1} q_{-j}(w_j) p(w_j|\phi_{l_j})^\eta$, and it is assumed that the conditional expectations $\mathrm{E}(w_j|\phi_{l_j}, \eta)$ and $\mathrm{E}(w_j^2|\phi_{l_j}, \eta)$ can be calculated analytically. The fraction parameter can also be handled conveniently because the exponentiation with η results in a distribution of the same family multiplied by a tractable function of η and ϕ_{l_j} when the prior distribution $p(w_j|\phi_{l_j})$ belongs to the exponential family. Computing the marginal moments using equations (27) and (28) requires a total of five one-dimensional quadrature integrations but in practice this can be done efficiently by utilizing the same function evaluations of $\hat{p}_j(\phi_{l_j})$ and taking into account the prior specific forms of $\mathrm{E}(w_j|\phi_{l_j}, \eta)$ and $\mathrm{E}(w_j^2|\phi_{l_j}, \eta)$.

3.3 Structure of the Weight Approximation

In this section we consider different possibilities for approximating the likelihood terms $p(y_i|f_i, \theta)$ which depend on the noise parameter $\theta = \log \sigma^2$ and the weight vectors $\mathbf{w}$ and $\mathbf{v}$ through the latent function value f_i as

$$f_i = \mathbf{v}^\mathrm{T} \mathbf{g}(\tilde{\mathbf{x}}_i^\mathrm{T} \mathbf{w}) = \mathbf{v}^\mathrm{T} \mathbf{g}(\mathbf{h}_i), \tag{29}$$

where $\tilde{\mathbf{x}}_i = \mathbf{I}_K \otimes \mathbf{x}_i$ is a $Kd \times K$ auxiliary matrix formed as a Kronecker product. It can be used to write all the linear input layer activations $\mathbf{h}_i$ related to $\mathbf{x}_i$ as $\mathbf{h}_i = \mathbf{h}(\mathbf{x}_i) = \tilde{\mathbf{x}}_i^\mathrm{T} \mathbf{w}$. The vector valued function $\mathbf{g}(\mathbf{h}_i)$ applies the nonlinear transformation (2) on each component of $\mathbf{h}_i$ according to $\mathbf{g}(\mathbf{h}_i) = [g(h_{i,1}), g(h_{i,2}), ..., g(h_{i,K}), 1]^\mathrm{T}$, where the last element corresponds to the output bias v_0.

3.3.1 Fully-Coupled Approximation for the Network Weights

If we approximate the posterior distribution of all the weights $\mathbf{z} = [\mathbf{w}^\mathrm{T}, \mathbf{v}^\mathrm{T}]^\mathrm{T}$ with a multivariate Gaussian approximation $q(\mathbf{z})$ from (16) that is independent of all the other unknowns including ϕ and θ, the resulting EP algorithm requires fast evaluation of the means and covariances of tilted distributions of the form

$$\hat{p}_i(\mathbf{z}) \propto p(y_i|\mathbf{v}^T \mathbf{g}(\mathbf{h}_i), \theta)^\eta q_{-i}(\mathbf{z}), \tag{30}$$

which is equivalent to (22) except that θ is assumed fixed for clarity. Approximating the tilted moments with unknown θ is described in Appendix D. Assuming Gaussian site approximations $\tilde{t}_{\mathbf{z},i}(\mathbf{z})$ and using (21) results in a Gaussian cavity distribution $q_{-i}(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_{-i}, \boldsymbol{\Sigma}_{-i})$, where $\boldsymbol{\mu}_{-i}$ is a $d_z \times 1$ mean vector and $\boldsymbol{\Sigma}_{-i}$ a $d_z \times d_z$ covariance matrix with $d_z = Kd + K + 1$.

Because the non-Gaussian likelihood term depends on $\mathbf{w}$ only through the linear transformation $\mathbf{h}_i = \tilde{\mathbf{x}}_i^{\mathrm{T}}\mathbf{w}$, it can be shown (e.g., by differentiating (30) twice with respect to $\boldsymbol{\mu}_{-i}$) that the normalization term, mean and covariance of $\hat{p}_i(\mathbf{z})$ can be exactly determined by using the corresponding moments of the transformed lower dimensional random vector $\mathbf{u}_i = \mathbf{B}_i^{\mathrm{T}}\mathbf{z} = [\mathbf{w}^{\mathrm{T}}\tilde{\mathbf{x}}_i, \mathbf{v}^{\mathrm{T}}]^{\mathrm{T}} = [\mathbf{h}_i^{\mathrm{T}}, \mathbf{v}^{\mathrm{T}}]^{\mathrm{T}}$, where the transformation matrix $\mathbf{B}_i$ can be written as

$$\mathbf{B}_i = \begin{bmatrix} \tilde{\mathbf{x}}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{K+1} \end{bmatrix}. \tag{31}$$

This results in significant computational savings because the size of $\mathbf{B}_i$ is $d_z \times d_u$, where the dimensions of $\mathbf{u}_i$ and $\mathbf{z}$ are $d_u = 2K + 1$ and $d_z = Kd + K + 1$ respectively. It follows that the EP algorithm can be implemented by propagating the moments of $\mathbf{u}_i$ using, for example, the general algorithm described by Cseke and Heskes (2011, appendix C). The same principle has been utilized to form computationally efficient algorithms also for linear binary classification (Minka, 2001a; Qi et al., 2004) and multi-class classification (Riihimäki et al., 2013).

Because of the previously described property, the first likelihood site approximation $\tilde{t}_{\mathbf{z},i}(\mathbf{z})$ in (10) depends on $\mathbf{z}$ only through the linear transformation $\mathbf{B}_i^{\mathrm{T}}\mathbf{z}$ (Cseke and Heskes, 2011):

$$\tilde{t}_{\mathbf{z},i}(\mathbf{z}) = \exp\left(-\frac{1}{2}\mathbf{z}^{\mathrm{T}}\mathbf{B}_i\tilde{\mathbf{T}}_i\mathbf{B}_i^{\mathrm{T}}\mathbf{z} + \mathbf{z}^{\mathrm{T}}\mathbf{B}_i\tilde{\mathbf{b}}_i\right), \tag{32}$$

where $\tilde{\mathbf{b}}_i$ is a $d_u \times 1$ vector of location parameters, and $\tilde{\mathbf{T}}_i$ a $d_u \times d_u$ site precision matrix. Multiplying the site approximations (32) together according to (16) results in a Gaussian approximation $q(\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where the mean vector and covariance matrix are given by

$$\boldsymbol{\Sigma} = \left(\sum_{i=1}^{n}\mathbf{B}_i\tilde{\mathbf{T}}_i\mathbf{B}_i^{\mathrm{T}} + \boldsymbol{\Sigma}_0^{-1}\right)^{-1} \quad \text{and} \quad \boldsymbol{\mu} = \boldsymbol{\Sigma}\left(\sum_{i=1}^{n}\mathbf{B}_i\tilde{\mathbf{b}}_i + \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0\right). \tag{33}$$

In (33) the parameters of the prior term approximations $\tilde{t}_{w,j}(w_j) \propto \mathcal{N}(\tilde{\mu}_{w,j}, \tilde{\sigma}_{w,j}^2)$ and $\tilde{t}_{v,k}(v_k) \propto \mathcal{N}(\tilde{\mu}_{v,k}, \tilde{\sigma}_{v,k}^2)$ and the prior $p(v_0) = \mathcal{N}(0, \sigma_{v_0,0}^2)$ are collected together in $\boldsymbol{\Sigma}_0 = \mathrm{diag}([\tilde{\sigma}_{w,1}^2, ..., \tilde{\sigma}_{w,Kd}^2, \tilde{\sigma}_{v,1}^2, ..., \tilde{\sigma}_{v,K}^2, \sigma_{v_0,0}^2])$ and $\boldsymbol{\mu}_0 = [\tilde{\mu}_{w,1}, ..., \tilde{\mu}_{w,Kd}, \tilde{\mu}_{v,1}, ..., \tilde{\mu}_{v,K}, 0]^{\mathrm{T}}$. The fully-coupled approximation defined by (32) and (33) can capture correlations between all components of $\mathbf{z}$ because the off-diagonal elements of $\tilde{\mathbf{T}}_i$ will typically become non-zero during the EP updates. Because the base computational scaling of these updates is $\mathcal{O}(d_u^3)$ and determining the tilted moments requires multi-dimensional numerical integrations, the fully-coupled approximation is feasible only with a small number of hidden units or with additional low rank approximations for the site precision parameters $\tilde{\mathbf{T}}_i$. These issues together with computationally more efficient approximations are considered in the next section.

3.3.2 Factorized Approximation for the Network Weights

A drawback with the fully-coupled approximation (33) is that computing the covariance matrix $\boldsymbol{\Sigma}$ scales as $\mathcal{O}\big(\min\big(Kd{+}K{+}1, \sum_i \text{rank}(\tilde{\mathbf{T}}_i)\big)^3\big)$, which may not be feasible with large values of d or K. In addition, an EP update for each likelihood site would require multiple $\text{rank}(\tilde{\mathbf{T}}_i)$ matrix inversion (or decomposition) to compute the mean and covariance of the cavity distribution (21) and the new site parameters using (23). Determining the mean and covariance of $\mathbf{u}_i = \mathbf{B}_i\mathbf{z} = [\mathbf{h}_i^{\mathrm{T}}, \mathbf{v}^{\mathrm{T}}]^{\mathrm{T}}$ is also computationally challenging when $\mathbf{z}$ is distributed according to (30). If the observation model is Gaussian and θ is fixed, this requires at least K-dimensional numerical quadratures (or other alternative approximations) that may quickly become infeasible as K increases. By adopting suitable independence assumptions between $\mathbf{v}$ and the input weights $\mathbf{w}_k$ associated with the different hidden units, the mean and covariance of $\mathbf{u}_i$ can be approximated using only 1-dimensional numerical quadratures as will be described in Section 3.4.

The structure of the correlations in the approximation (33) can be studied by dividing $\tilde{\mathbf{T}}_i$ into four blocks as follows:

$$\tilde{\mathbf{T}}_i = \begin{bmatrix} \tilde{\mathbf{T}}_{\mathbf{h}_i\mathbf{h}_i} & \tilde{\mathbf{T}}_{\mathbf{h}_i\mathbf{v}} \\ \tilde{\mathbf{T}}_{\mathbf{v}\mathbf{h}_i} & \tilde{\mathbf{T}}_{i,\mathbf{v}\mathbf{v}} \end{bmatrix}, \tag{34}$$

where $\tilde{\mathbf{T}}_{\mathbf{h}_i\mathbf{h}_i}$ is a $K \times K$ matrix, $\tilde{\mathbf{T}}_{\mathbf{h}_i\mathbf{v}} = \tilde{\mathbf{T}}_{\mathbf{v}\mathbf{h}_i}^{\mathrm{T}}$ a $K \times K{+}1$ matrix, and $\tilde{\mathbf{T}}_{i,\mathbf{v}\mathbf{v}}$ a $K{+}1 \times K{+}1$ matrix. The element $[\tilde{\mathbf{T}}_{\mathbf{h}_i\mathbf{h}_i}]_{k,k'}$ contributes to the approximate posterior covariance between $\mathbf{w}_k$ and $\mathbf{w}_{k'}$, and the k:th row of sub-matrix $\tilde{\mathbf{T}}_{\mathbf{h}_i\mathbf{v}}$ contributes to the approximate covariance between $\mathbf{w}_k$ and $\mathbf{v}$. To form an alternative computationally more efficient approximation we propose a simpler structure for $\tilde{\mathbf{T}}_i$. First, we approximate $\tilde{\mathbf{T}}_{\mathbf{h}_i\mathbf{h}_i}$ with a diagonal matrix, that is, $\tilde{\mathbf{T}}_{\mathbf{h}_i\mathbf{h}_i} = \text{diag}(\tilde{\boldsymbol{\tau}}_i)$, where only the k:th component of the vector $\tilde{\boldsymbol{\tau}}_i$ contributes to the posterior covariance of $\mathbf{w}_k$. Secondly, we set $\tilde{\mathbf{T}}_{\mathbf{h}_i\mathbf{v}} = \tilde{\mathbf{T}}_{\mathbf{v}\mathbf{h}_i}^{\mathrm{T}} = \mathbf{0}$, and approximate $\tilde{\mathbf{T}}_{i,\mathbf{v}\mathbf{v}}$ with an outer-product of the form $\tilde{\mathbf{T}}_{i,\mathbf{v}\mathbf{v}} = \tilde{\boldsymbol{\alpha}}_i\tilde{\boldsymbol{\alpha}}_i^{\mathrm{T}}$. With this precision structure the site approximation (32) can be factorized into terms depending only on the output weights $\mathbf{v}$ or the input weights $\mathbf{w}_k$ associated with the different hidden units $k = 1, ..., K$:

$$\tilde{t}_{\mathbf{z},i}(\mathbf{z}) = \underbrace{\exp\left(-\frac{1}{2}(\tilde{\boldsymbol{\alpha}}_i^{\mathrm{T}}\mathbf{v})^2 + \mathbf{v}^{\mathrm{T}}\tilde{\boldsymbol{\beta}}_i\right)}_{=\tilde{t}_{\mathbf{v},i}(\mathbf{v}|\tilde{\boldsymbol{\alpha}}_i,\tilde{\boldsymbol{\beta}}_i)} \prod_{k=1}^{K} \underbrace{\exp\left(-\frac{1}{2}\tilde{\tau}_{i,k}(\mathbf{x}_i^{\mathrm{T}}\mathbf{w}_k)^2 + \tilde{\nu}_{i,k}\mathbf{w}_k^{\mathrm{T}}\mathbf{x}_i\right)}_{=\tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k|\tilde{\tau}_{i,k},\tilde{\nu}_{i,k})}, \tag{35}$$

where the scalar site location parameters $\tilde{\nu}_{i,k}$ now correspond to the first K elements of $\tilde{\mathbf{b}}_i$ in equation (32), that is, $\tilde{\boldsymbol{\nu}}_i = [\tilde{\nu}_{i,1}, ...\tilde{\nu}_{i,K}]^{\mathrm{T}} = [\tilde{\mathbf{b}}_{i,1}, ..., \tilde{\mathbf{b}}_{i,K}]^{\mathrm{T}}$, and analogously, the site location vector $\tilde{\boldsymbol{\beta}}_i$ corresponds to the last $K{+}1$ entries of $\tilde{\mathbf{b}}_i$, that is, $\tilde{\boldsymbol{\beta}}_i = [\tilde{\mathbf{b}}_{i,K+1}, ..., \tilde{\mathbf{b}}_{i,2K+1}]^{\mathrm{T}}$. Equation (35) defines the parametric structure of the factorized likelihood site approximation already introduced in (11).

Combining the site approximations (35) according to (20) results in an independent posterior approximation $q(\mathbf{v}) = \mathcal{N}(\boldsymbol{\mu}_v, \boldsymbol{\Sigma}_v)$ for the output weights and independent approximations $q(\mathbf{w}_k) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{w}_k}, \boldsymbol{\Sigma}_{\mathbf{w}_k})$ for the input weights associated with the different hidden units $k = 1, ..., K$. The approximate mean and covariance of $\mathbf{w}_k$ are given by

$$\boldsymbol{\Sigma}_{\mathbf{w}_k} = \left(\mathbf{X}^{\mathrm{T}}\tilde{\mathbf{T}}_{\mathbf{w}_k}\mathbf{X} + \boldsymbol{\Sigma}_{\mathbf{w}_k,0}^{-1}\right)^{-1} \quad \text{and} \quad \boldsymbol{\mu}_{\mathbf{w}_k} = \boldsymbol{\Sigma}_{\mathbf{w}_k}\left(\mathbf{X}^{\mathrm{T}}\tilde{\boldsymbol{\nu}}_{\mathbf{w}_k} + \boldsymbol{\Sigma}_{\mathbf{w}_k,0}^{-1}\boldsymbol{\mu}_{\mathbf{w}_k,0}\right), \tag{36}$$

where the diagonal matrix $\tilde{\mathbf{T}}_{\mathbf{w}_k} = \text{diag}(\tilde{\boldsymbol{\tau}}_{\mathbf{w}_k})$ and the vector $\tilde{\boldsymbol{\nu}}_{\mathbf{w}_k}$ collect all the site parameters related to hidden unit k: $\tilde{\boldsymbol{\tau}}_{\mathbf{w}_k} = [\tilde{\tau}_{1,k}, ..., \tilde{\tau}_{n,k}]^{\mathrm{T}}$ and $\tilde{\boldsymbol{\nu}}_{\mathbf{w}_k} = [\tilde{\nu}_{1,k}, ..., \tilde{\nu}_{n,k}]^{\mathrm{T}}$. The parameters of the prior term approximations $\tilde{t}_{w,j}(w_j) \propto \mathcal{N}(\tilde{\mu}_{w,j}, \tilde{\sigma}^2_{w,j})$ related to hidden unit k are collected in the diagonal matrix $\boldsymbol{\Sigma}_{\mathbf{w}_k,0} = \text{diag}(\tilde{\sigma}^2_{w,m+1}, ..., \tilde{\sigma}^2_{w,m+d})$ and vector $\boldsymbol{\mu}_{\mathbf{w}_k,0} = [\tilde{\mu}_{w,m+1}, ..., \tilde{\mu}_{w,m+d}]^{\mathrm{T}}$, where $m = K(k-1)$. The approximate mean and covariance of the output weights $\mathbf{v}$ are given by

$$\boldsymbol{\Sigma}_{\mathbf{v}} = \left(\sum_{i=1}^{n} \tilde{\boldsymbol{\alpha}}_i \tilde{\boldsymbol{\alpha}}_i^{\mathrm{T}} + \boldsymbol{\Sigma}_{\mathbf{v},0}^{-1} \right)^{-1} \quad \text{and} \quad \boldsymbol{\mu}_{\mathbf{v}} = \boldsymbol{\Sigma}_{\mathbf{v}} \left(\sum_{i=1}^{n} \tilde{\boldsymbol{\beta}}_i + \boldsymbol{\Sigma}_{\mathbf{v}}^{-1} \boldsymbol{\mu}_{\mathbf{v},0} \right), \tag{37}$$

where the parameters of the prior term approximations $\tilde{t}_{v,k}(v_k) \propto \mathcal{N}(\tilde{\mu}_{v,k}, \tilde{\sigma}^2_{v,k})$ are collected in the diagonal matrix $\boldsymbol{\Sigma}_{\mathbf{v},0} = \text{diag}(\tilde{\sigma}^2_{v,1}, ..., \tilde{\sigma}^2_{v,K})$ and vector $\boldsymbol{\mu}_{\mathbf{v},0} = [\tilde{\mu}_{v,1}, ..., \tilde{\mu}_{v,K}]^{\mathrm{T}}$.

For each hidden unit k, approximation (36) can be interpreted as an independent linear model with Gaussian likelihood terms $\mathcal{N}(\tilde{y}_{i,k} | \mathbf{x}_i^{\mathrm{T}} \mathbf{w}_k, \tilde{\tau}_{i,k}^{-1})$, where the pseudo-observations are given by $\tilde{y}_{i,k} = \tilde{\tau}_{i,k}^{-1} \tilde{\nu}_{i,k}$. The approximation for the output weights (37) has no explicit dependence on the input vectors $\mathbf{x}_i$ but later we will show that the independence assumption results in a similar dependence on expected values of the hidden unit activations $\mathbf{g}_i = \mathbf{g}(\mathbf{h}_i)$ taken with respect to the cavity distributions $q_{-i}(\mathbf{w})$ and $q_{-i}(\mathbf{v})$ (see Appendix E).

One sequential EP update for each of the n likelihood sites requires either one rank$(\tilde{\mathbf{T}}_i)$ covariance matrix update for the fully-correlated approximation (33), or $K+1$ rank-one covariance matrix updates for each of the factorized approximations (36) and (37). In parallel EP these updates are replaced with a single re-computation of the posterior representation after each sweep over all the n sites. In practice, one parallel iteration step over all the sites can be much faster compared to a sequential EP iteration, especially if d or K are large, but parallel EP may require larger number of iterations for overall convergence.

3.4 Implementing the EP Algorithm

In this section, we describe the computational implementation of the EP algorithm resulting from the choice of the approximating family described in Section 3.3. Because the non-Gaussian likelihood term in the tilted distribution (22) depends on $\mathbf{z} = [\mathbf{w}^{\mathrm{T}}, \mathbf{v}^{\mathrm{T}}]^{\mathrm{T}}$ only through the linear transformation $\mathbf{u}_i = [\mathbf{h}_i^{\mathrm{T}}, \mathbf{v}^{\mathrm{T}}]^{\mathrm{T}} = \mathbf{B}_i^{\mathrm{T}} \mathbf{z}$, the EP algorithm can be implemented by iteratively determining and matching the moments of the lower-dimensional random vector $\mathbf{u}_i$ instead of $\mathbf{z}$ (Cseke and Heskes, 2011, appendix C). The computations can be further facilitated by using the factorized approximation (19): Because the hidden activations $h_{i,k} = \mathbf{x}_i^{\mathrm{T}} \mathbf{w}_k$ related to the different hidden units $k = 1, ..., K$ are independent of each other and $\mathbf{v}$, it is only required to propagate the marginal means and covariances of $h_{i,k}$ and $\mathbf{v}$ to determine the new site parameters. This enables more efficient formulas for determining the cavity distributions (21), the tilted distributions (22), and site parameter updates from (23). Details of the computations required for updating the likelihood site approximations are presented in Appendices A–E. The main properties of the procedure can be summarized as follows:

- Appendix A presents the formulas for computing the parameters of the cavity distributions (21). The factorized approximation (19) leads to efficient computations, because the cavity distribution can be factored as $q_{-i}(\mathbf{z}) = q_{-i}(\mathbf{v}) \prod_{k=1}^{K} q_{-i}(\mathbf{w}_k)$.

The parameters of $q_{-i}(h_{i,k})$ resulting from the transformation $h_{i,k} = \mathbf{x}_i^{\mathrm{T}}\mathbf{w}_k$ can be computed using only scalar manipulations of the mean and covariance of $q(h_{i,k}) = \mathcal{N}(\mathbf{x}_i^{\mathrm{T}}\boldsymbol{\mu}_{\mathbf{w}_k}, \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathbf{w}_k}\mathbf{x}_i)$. Because of the outer-product structure of $\tilde{t}_{\mathbf{v},i}(\mathbf{v})$ in equation (35), also the parameters of $q_{-i}(\mathbf{v})$ can be computed using rank-one matrix updates.

- Appendix B describes how the marginal mean and covariance of $\mathbf{v}$ with respect to the tilted distribution (22) can be approximated efficiently using a similar Gaussian approximation as is used in the UKF filter (Wan and van der Merwe, 2000). Because of the factorized approximation (19) only one-dimensional quadratures are required to compute the means and variances of $g(h_{i,k})$ with respect to $q_{-i}(h_{i,k})$ and no multivariate quadrature or sigma-point approximations are needed.

- Appendix C presents a new way to approximate the marginal distribution of $\hat{p}_i(h_{i,k})$ resulting from (22). In preliminary simulations we found that an approach based on the unscented transform and the approximate linear filtering paradigm used in Appendix B did not capture well the information from the left-out observation y_i. This behavior was more problematic when there was a large discrepancy between the information provided by the likelihood term through the marginal tilted distribution $\hat{p}_i(y_i|h_{i,k}) = \int p(y_i|f_i,\theta)^{\eta} q_{-i}(\mathbf{v}) q_{-i}(\mathbf{h}_{i,-k}) d\mathbf{v} d\mathbf{h}_{i,-k}$ and the cavity distribution $q_{-i}(h_{i,k})$, where $\mathbf{h}_{i,-k}$ includes all components of $\mathbf{h}_i$ except $h_{i,k}$.[2]

  The improved numerical approximation of $\hat{p}_i(h_{i,k})$ is obtained by approximating the cavity distribution $q_{-i}(f_i|h_{i,k})$, that is, the distribution of the latent function value $f_i = \sum_{k=1}^{K} g(h_{i,k}) + v_0$ resulting from $q_{-i}(\mathbf{h}_{i,-k}, \mathbf{v}|h_{i,k}) = q_{-i}(\mathbf{v}) \prod_{k' \neq k} q_{-i}(h_{i,-k})$, with a Gaussian distribution and carrying out the integration over f_i analytically. According to the central limit theorem we expect this approximation to become more accurate as K increases. A similar approach has been used by Ribeiro and Opper (2011) to form factorized EP approximations for the input weights with a linear single-layer model structure. They used the central limit argument to form Laplace approximations for the marginal tilted distributions resulting from univariate Gaussian approximations for the input weights. We utilize the same idea to approximate the tilted moments of the transformed variables $h_{i,k} = \mathbf{w}_k^{\mathrm{T}}\mathbf{x}_i$ using numerical quadratures and an input weight approximation that can be factorized between the different hidden units.

- Appendix D generalizes the tilted moment estimations of Appendices B and C for approximate integration over the posterior uncertainty of $\theta = \log \sigma^2$. Computationally convenient marginal mean and covariance estimates for $\mathbf{v}$, $\{h_{i,k}\}_{k=1}^{K}$, and θ can be obtained by utilizing the independent posterior approximation for θ and the same Gaussian approximation for $q_{-i}(f_i)$ as in Appendix C. Compared to the tilted moments approximations of $\mathbf{v}$ and $\mathbf{h}_i$ with fixed θ, the approach requires five additional univariate quadratures for each likelihood term, which can be facilitated by utilizing the same function evaluations.

2. The UKF approach was found to perform better with smaller values of η because then a fraction of the site approximation from the previous iteration is left in the cavity, which can reduce the possible multimodality of $\hat{p}_i(h_{i,k})$.

- Appendix E presents expressions for the new site parameters obtained by applying the results of Appendices A–D in the moment matching condition (23). Because of the factorization assumption in (19) and the UKF-style approximation in the tilted moment estimations (Appendix B), the parameters of the likelihood term approximations related to $\mathbf{v}$ (see (35)) can be written as $\tilde{\boldsymbol{\alpha}}_i = \mathbf{m}_{\mathbf{g}_i} \tilde{\sigma}^{-1}_{\mathbf{v},i}$ and $\tilde{\boldsymbol{\beta}}_i = \mathbf{m}_{\mathbf{g}_i} \tilde{\sigma}^{-2}_{\mathbf{v},i} \tilde{y}_{\mathbf{v},i}$, where $[\mathbf{m}_{\mathbf{g}_i}]_k = \int g(h_{i,k}) q_{-i}(h_{i,k}) dh_{i,k}$ and $\tilde{y}_{\mathbf{v},i}$ can be interpreted as Gaussian pseudo-observations with noise variances $\tilde{\sigma}^2_{\mathbf{v},i}$ (compare with equation (55) and (56)). Thus, by comparing the parameter expressions with (37), the output-layer approximation $q(\mathbf{v})$ can be interpreted as a linear model where the cavity expectations of the hidden unit outputs $g(h_{i,k}) = g(\mathbf{w}_k^{\mathrm{T}} \mathbf{x}_i)$ are used as input features. The EP updates for the site parameters $\tilde{\tau}_{i,k}$ and $\tilde{\nu}_{i,k}$ related to the input weight approximations $q(\mathbf{w}_k)$ require only scalar operations similarly to other standard EP implementations (Minka, 2001a; Rasmussen and Williams, 2006).

Appendix F summarizes an EP algorithm that can be used to implement the EP updates for the prior site approximations described in Section 3.2.2. Appendix G presents some practical tips to improve the numerical stability of the EP updates proposed in Appendices A–F. Appendix H describes how the predictive distribution $p(y_*|\mathbf{x}_*)$ related to a new input vector $\mathbf{x}_*$ can be approximated efficiently using $q(\mathbf{v})$, $\{q(\mathbf{w}_k)\}_{k=1}^K$, and $q(\theta)$. Note that the prior scale approximations $\{q(\phi_l)\}_{l=1}^L$ are not needed in the predictions because information from the hierarchical input weight priors is approximately absorbed in $\{q(\mathbf{w}_k)\}_{k=1}^K$ during the EP iterations. Appendix I shows how the EP marginal likelihood approximation, $\log Z_{\mathrm{EP}} \approx \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\gamma})$, conditioned on fixed hyperparameters $\boldsymbol{\gamma}$, can be computed in a numerically efficient and stable manner. The marginal likelihood estimate can be used to monitor convergence of the EP iterations, to determine marginal MAP estimates of the fixed hyperparameters, and to compare different model structures.

3.5 General Algorithm Description and Practical Considerations

Algorithm 1 collects together all the computational components described in Section 3.4 and Appendices A–F into a single EP algorithm. In this section we will discuss the initialization, the order of updates between the different term approximations, and the convergence properties of the algorithm.

3.5.1 Scheduling Between the Likelihood and Prior term updates

In algorithm 1, we have combined the EP updates for the site approximations of the likelihood terms $p(y_i|\mathbf{v}^{\mathrm{T}}\mathbf{g}(\tilde{\mathbf{x}}_i^{\mathrm{T}}\mathbf{w}), \theta)$ (lines 2-7) and the prior terms $p(w_j|\phi_{l_j})$ (line 1) and $p(v_k)$ (line 8) by running them in turn. Because all information from the observations $\mathbf{y}$ is conveyed by the likelihood term approximations, it is sensible to iterate first the parameters $\tilde{\boldsymbol{\tau}}_i$ and $\tilde{\boldsymbol{\nu}}_i$ of $\{\tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k)\}_{k=1}^K$ together with the parameters $\tilde{\boldsymbol{\alpha}}_i$ and $\tilde{\boldsymbol{\beta}}_i$ of $\tilde{t}_{\mathbf{v},i}(\mathbf{v})$ to obtain a good data fit while keeping the prior term approximations of $p(w_j|\phi_{l_j})$ and $p(v_k)$ fixed so that all the output weights remain effectively positive and all the input weights have equal prior distributions. Otherwise, depending on the scales of the priors, many hidden units and input weights could be effectively pruned out of the model by the prior sites $\{\tilde{t}_{v,k}(v_k|\tilde{\mu}_{v,k}, \tilde{\sigma}^2_{v,k})\}_{k=1}^K$ and $\{\tilde{t}_{w,j}(w_j|\tilde{\mu}_{w,j}, \tilde{\sigma}^2_{w,j}), \tilde{t}_{\phi,j}(\phi_{l_j}|\tilde{\mu}_{\phi,j}, \tilde{\sigma}^2_{\phi,j})\}_{j=1}^{Kd}$: For example, the location parameters $\tilde{\mu}_{w,j}$

Algorithm 1: An EP algorithm for a two-layer MLP-network with non-Gaussian hierarchical priors on the weights.

Initialize $q(\mathbf{v}|\boldsymbol{\mu}_\mathbf{v}, \boldsymbol{\Sigma}_\mathbf{v})$, $\{q(\mathbf{w}_k|\boldsymbol{\mu}_{\mathbf{w}_k}, \boldsymbol{\Sigma}_{\mathbf{w}_k})\}_{k=1}^K$, $q(\theta|\mu_\theta, \sigma_\theta^2)$, and $\{q(\phi_l|\mu_{\phi_l}, \sigma_{\phi_l}^2)\}_{l=1}^L$ using site approximations $\left\{\tilde{t}_{\mathbf{w},i}(\mathbf{w}|\tilde{\boldsymbol{\tau}}_i, \tilde{\boldsymbol{\nu}}_i), \tilde{t}_{\mathbf{v},i}(\mathbf{v}|\tilde{\boldsymbol{\alpha}}_i, \tilde{\boldsymbol{\beta}}_i), \tilde{t}_{\theta,i}(\theta|\tilde{\mu}_{\theta,i}, \tilde{\sigma}_{\theta,i}^2)\right\}_{i=1}^n$, $\left\{\tilde{t}_{v,k}(v_k|\tilde{\mu}_{v,k}, \tilde{\sigma}_{v,k}^2\right\}_{k=1}^K$, and $\left\{\tilde{t}_{w,j}(w_j|\tilde{\mu}_{w,j}, \tilde{\sigma}_{w,j}^2), \tilde{t}_{\phi,j}(\phi_{l_j}|\tilde{\mu}_{\phi,j}, \tilde{\sigma}_{\phi,j}^2\right\}_{j=1}^{Kd}$ (equations (17), (18), (36), and (37)).

repeat

  if *sufficient convergence in* $\{\tilde{\boldsymbol{\tau}}_i, \tilde{\boldsymbol{\nu}}_i, \tilde{\boldsymbol{\alpha}}_i, \tilde{\boldsymbol{\beta}}_i, \tilde{\mu}_{\theta,i}, \tilde{\sigma}_{\theta,i}^2\}_{i=1}^n$ *and* $\{\tilde{\mu}_{v,k}, \tilde{\sigma}_{v,k}^2\}_{k=1}^K$ **then**

1 Run the EP algorithm from Appendix F to update the parameters $\{\tilde{\mu}_{w,j}, \tilde{\sigma}_{w,j}^2, \tilde{\mu}_{\phi,j}, \tilde{\sigma}_{\phi,j}^2\}_{j=1}^{Kd}$ of the prior site approximations $\tilde{t}_{w,j}(w_j)$ and $\tilde{t}_{\phi,j}(\phi_{l_j})$ from (14).

  end

  Loop over the likelihood terms to update $\tilde{t}_{\mathbf{v},i}(\mathbf{v})$, $\{\tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k)\}_{k=1}^K$, and $\tilde{t}_{\theta,i}(\theta)$:

  for $i \leftarrow 1$ **to** n **do**

2 Compute the means and covariances of the cavity distributions: $\{q_{-i}(h_{i,k})\}_{k=1}^K$ and $q_{-i}(\mathbf{v})$ using equations (39) and (40).
  If θ unknown, compute the cavity distribution $q_{-i}(\theta)$ using (41).

3 Compute the means and covariances of the tilted distributions $\hat{q}_i(\mathbf{v}) = \mathcal{N}(\hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{\Sigma}}_i)$ and $\hat{q}_i(h_{i,j}) = \mathcal{N}(\hat{m}_i, \hat{V}_i)$ for $k = 1, ..., K$:
  If θ known, use (45) and (49).
  Otherwise, use (51), (52), and (54), and compute $\hat{q}_i(\theta) = \mathcal{N}(\hat{\mu}_{\theta,i}, \hat{\sigma}_{\theta,i}^2)$ from (50).

4 Update the site parameters $\tilde{\boldsymbol{\tau}}_i$, $\tilde{\boldsymbol{\nu}}_i$, $\tilde{\boldsymbol{\alpha}}_i$, $\tilde{\boldsymbol{\beta}}_i$ using (57), (59), and (60).
  If θ unknown, update $\tilde{\mu}_{\theta,i}$, $\tilde{\sigma}_{\theta,i}^2$ using (61).

  if *sequential updates* **then**

5 Rank-1 updates for $\{q(\mathbf{w}_k)\}_{k=1}^K$ according to the changes in $\{\tilde{\tau}_{i,k}, \tilde{\nu}_{i,k}\}_{k=1}^K$.
  If θ unknown, update $q(\theta)$ according to the changes in $\{\tilde{\mu}_{\theta,i}, \tilde{\sigma}_{\theta,i}^2\}$.

  end

  end

  if *parallel updates* **then**

6 Recompute the approximations $\{q(\mathbf{w}_k)\}_{k=1}^K$ using $\{\tilde{\boldsymbol{\tau}}_i, \tilde{\boldsymbol{\nu}}_i\}_{i=1}^n$ and $\{\tilde{\mu}_{w,j}, \tilde{\sigma}_{w,j}^2\}_{j=1}^{Kd}$.
  If θ unknown, recompute $q(\theta)$ using $\left\{\tilde{\mu}_{\theta,i}, \tilde{\sigma}_{\theta,i}^2\right\}_{i=1}^n$ and $\{\mu_{\theta,0}, \sigma_{\theta,0}^2\}$.

  end

7 Recompute (parallel update) $q(\mathbf{v})$ using $\{\tilde{\boldsymbol{\alpha}}_i, \tilde{\boldsymbol{\beta}}_i\}_{i=1}^n$ and $\{\tilde{\mu}_{v,k}, \tilde{\sigma}_{v,k}^2\}_{k=1}^K$.

  if *sufficient data fit* **then**

8 Run the EP algorithm from Appendix F to update the parameters $\{\tilde{\mu}_{v,k}, \tilde{\sigma}_{v,k}^2\}_{k=1}^K$ of the prior site approximations $\tilde{t}_{v,k}(v_k)$ from (13) and recompute $q(\mathbf{v}) = \mathcal{N}(\boldsymbol{\mu}_\mathbf{v}, \boldsymbol{\Sigma}_\mathbf{v})$.

  end

until *convergence or maximum number of iterations exceeded*

would push the approximate marginal distribution $q(w_j)$ towards zero and the scale parameter $\tilde{\sigma}_{w,j}^2$ would adjust the approximate variance of w_j to the level reflecting the fixed scale prior definition $p(\phi_{l_j}) = \mathcal{N}(\mu_{\phi,0}, \sigma_{\phi,0}^2)$ in case enough information was not conveyed from

the observations due to poorly fitted likelihood site approximations. To enable convenient implementation of various learning strategies, different damping factors were assigned to the different types of likelihood term approximations $\tilde{t}_{\mathbf{v},i}(\mathbf{v})$, $\tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k)$, and $\tilde{t}_{\theta,i}(\theta)$. For example, only one of the approximations, say $q(\mathbf{v})$, can by updated simply by setting the damping factors related to $\tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k)$ and $\tilde{t}_{\theta,i}(\theta)$ to zero. Similarly individual damping factors were assigned to the prior term approximations $\tilde{t}_{v,k}(v_k)$ and $\{\tilde{t}_{w,j}(w_j), \tilde{t}_{\phi,j}(\phi_{l_j})\}$. A more detailed discussion about damping and scheduling of the likelihood and prior site updates will be given in Sections 3.5.5 and 3.5.6.

3.5.2 Monitoring Convergence and Model Quality

During the iterations, the data fit can be assessed by monitoring the convergence of the approximate LOO predictive density $\log Z_{\text{LOO}} = \sum_i \log p(y_i|\mathbf{y}_{-i}, \mathbf{X}) \approx \sum_i \log \hat{Z}_i$ that should increase steadily in the beginning of the learning process as the model adapts to the observations $\mathbf{y}$. In contrast, the approximate marginal likelihood $\log Z_{\text{EP}} \approx \log p(\mathbf{y}|\mathbf{X})$ depends more on the model complexity and usually fluctuates more during the learning process because many different model structures can produce similar predictions. Convergence of the EP algorithm and quality of the approximation can be assessed by checking the consistency between the tilted distributions and the posterior approximation. For the likelihood site updates, we monitor the differences of the means and covariances of $\hat{p}_i(\mathbf{v})$, $\hat{p}_i(h_{i,k})$, and $\hat{p}_i(\theta)$ from the corresponding marginal approximations $q(\mathbf{v}) = \mathcal{N}(\boldsymbol{\mu}_\mathbf{v}, \boldsymbol{\Sigma}_\mathbf{v})$, $q(h_{i,k}) = \mathcal{N}(\mathbf{x}_i^{\mathrm{T}}\boldsymbol{\mu}_{\mathbf{w}_k}, \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathbf{w}_k}\mathbf{x}_i)$, and $q(\theta) = \mathcal{N}(\mu_\theta, \sigma_\theta^2)$ for all sites $i = 1, ..., n$ and hidden units $k = 1, ..., K$. Similarly for the prior site updates, we monitor the differences of $\hat{p}_j(w_j)$, $\hat{p}_j(\phi_{l_j})$, and $\hat{p}_k(v_k)$ from the corresponding marginal approximations for all $j = 1, ..., Kd$ and $k = 1, ..., K$, respectively. Note that the site parameter updates in (60), (61), and (65) become zero when these consistency conditions are satisfied.

3.5.3 Initial Parameter Values and Early Iterations

We initialized the algorithm with 10-20 iterations over the likelihood sites with θ fixed to a sufficiently small value, such as $\sigma^2 = \exp(\theta) = 0.3^2$, to obtain a good data fit before learning θ from the data. The parameters of the input weight priors were initialized to $\tilde{\mu}_{w,j} = 0$ and $\tilde{\sigma}_{w,j}^2 = 0.5$, where we have assumed that the target variables $\mathbf{y}$ and the columns of $\mathbf{X}$ containing the input variables are normalized to zero mean and unit variance. Larger initial variances $\tilde{\sigma}_{w,j}^2 = 2^2$ can be used for the input bias terms $j = d, 2d, ..., Kd$ so that the network is able to distribute the hidden units more flexibly to different locations of the input space. The prior means of the output weights $\tilde{\mu}_{v,k}$ were initialized with linear spacing in some appropriate interval, for example $[1, 2]$, and the prior variances were set to sufficiently small values such as $\tilde{\sigma}_{v,k}^2 = 0.2^2$ so that the elements of the approximate mean vector $\boldsymbol{\mu}_\mathbf{v}$ remain positive during the initial iterations.

We initialized the likelihood site parameters $\{\tilde{\boldsymbol{\tau}}_i, \tilde{\boldsymbol{\nu}}_i, \tilde{\boldsymbol{\alpha}}_i, \tilde{\boldsymbol{\beta}}_i\}_{i=1}^n$ to zero, which means that in the beginning all hidden units produce zero expected outputs $\mathbf{m}_{\mathbf{g}_i} = \mathbf{0}$ resulting in zero messages to the output weight approximation $\tilde{t}_{\mathbf{v},i}(\mathbf{v}|\tilde{\boldsymbol{\alpha}}_i, \tilde{\boldsymbol{\beta}}_i)$ in equations (55) and (56). However, because of the initialization of $\tilde{\mu}_{v,k}$ and $\tilde{\sigma}_{v,k}^2$, the initial approximate means of the output weights $[\boldsymbol{\mu}_\mathbf{v}]_k = \tilde{\mu}_{v,k}$ will be positive and nonidentical. It follows that different nonzero messages will be sent to the input weights according to (60) because the tilted

moments $\hat{m}_{i,k}$ and $\hat{V}_{i,k}$ of $h_{i,k}$ as given by (49) will differ from the corresponding marginal approximations $m_{i,k} = \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\mu}_{\mathbf{w}_k}$ and $V_{i,k} = \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathbf{w}_k} \mathbf{x}_i$. If in the beginning all the hidden units were updated simultaneously with the same priors for the output weights, they would get very similar approximate posteriors. In this case all the computational units would do more or less the same thing but sufficiently many iterations would eventually result in different values for all the input weight approximations $q(\mathbf{w}_k)$. This learning process can be accelerated by the previously described linearly spaced prior means $\tilde{\mu}_{v,j}$ or by updating only one hidden unit in the beginning and increasing the number of updated units one by one after every few iterations. The rationale behind the latter incremental scheme is that the first unit will usually explain the dominant linear relationships between $\mathbf{x}$ and y and the remaining units will fit to more local nonlinearities.

An alternative approach that can speed up the learning is to initialize the prior location parameters $\tilde{\mu}_{w,j}$ related to the input bias terms ($j = d, 2d, ..., Kd$) to random values, which can be interpreted as placing the hidden unit activations randomly in different locations of the input space. Also the prior location parameters $\tilde{\mu}_{w,j}$ of the input weights ($j = k + 1, ..., k + d - 1$ for $k = 1, ..., K$) could be initialized to random or some preselected values, which can be interpreted as starting the learning process from an initial feature embedding. The prior scale parameters $\tilde{\sigma}^2_{w,j}$ can be adjusted to control how strongly these prior constraints are enforced. After some iterations for the likelihood sites, the prior parameters can be relaxed and learned from data as described in the next section.

3.5.4 Relaxing the Initial Weight Prior and Noise Approximations

The positive Gaussian output weight priors defined at the initialization of $\tilde{\mu}_{v,k}$ and $\tilde{\sigma}^2_{v,k}$ can be relaxed after the initial iterations by running the EP algorithm on the approximations $\tilde{t}_{v,k}(v_k)$ of the truncated prior terms (5) (line 8 in Algorithm 1). The EP updates for the truncated prior terms ensure that the mass of the approximate density $q(\mathbf{v})$ will remain on the positive values for each component of $\mathbf{v}$. For this same reason we do only parallel updates on $q(\mathbf{v})$ on line 7 of Algorithm 1, because otherwise we would have to run the EP updates for the output weight priors (line 8) after each sequential update of $q(\mathbf{v})$. This parallel update scheme is discussed further in 3.5.5.

The EP updates for the observation noise θ (lines 2-5 in Algorithm 1) can be started after the initial iterations with fixed θ and weight priors. We initialized the site parameters $\{\tilde{\mu}_{\theta,i}, \tilde{\sigma}^2_{\theta,i}\}_{i=1}^n$ to zero, and at the first iteration for θ we also kept parameters $\tilde{\boldsymbol{\tau}}_i$, $\tilde{\boldsymbol{\nu}}_i$, $\tilde{\boldsymbol{\alpha}}_i$, and $\tilde{\boldsymbol{\beta}}_i$ fixed so that the initial fluctuations of $\tilde{\mu}_{\theta,i}$ and $\tilde{\sigma}^2_{\theta,i}$ do not affect the approximations $q(\mathbf{v})$ and $q(\mathbf{w}_k)$.

After sufficient convergence is obtained in the EP iterations for the parameters of the likelihood sites $\{\tilde{\boldsymbol{\tau}}_i, \tilde{\boldsymbol{\nu}}_i, \tilde{\boldsymbol{\alpha}}_i, \tilde{\boldsymbol{\beta}}_i, \tilde{\mu}_{\theta,i}, \tilde{\sigma}^2_{\theta,i}\}_{i=1}^n$ and the parameters of the output weight prior sites $\{\tilde{\mu}_{v,k}, \tilde{\sigma}^2_{v,k}\}_{k=1}^K$, EP updates can be started for the parameters $\{\tilde{\mu}_{w,j}, \tilde{\sigma}^2_{w,j}, \tilde{\mu}_{\phi,j}, \tilde{\sigma}^2_{\phi,j}\}_{j=1}^{Kd}$ of the prior term approximations $\tilde{t}_{w,j}(w_j)$ and $\tilde{t}_{\phi,j}(\phi_{l_j})$ (line 1 in Algorithm 1). This ensures that input weights and hidden units are not pruned out of the model before enough information is propagated from the observations to the likelihood term approximations.

3.5.5 Scheduling and Convergence of the Likelihood Term Iterations

If all the prior term approximations together with $\{\tilde{t}_{\mathbf{w},i}(\mathbf{w}|\tilde{\boldsymbol{\tau}}_i, \tilde{\boldsymbol{\nu}}_i)\}_{i=1}^n$ are kept fixed, that is, $q(\mathbf{w}_k)$ are not updated, the EP algorithm for the site approximations $\tilde{t}_{\mathbf{v},i}(\mathbf{v}|\tilde{\boldsymbol{\alpha}}_i, \tilde{\boldsymbol{\beta}}_i)$ related to $q(\mathbf{v})$ converges typically in 5-10 iterations. In addition, if only the site approximations $\tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k|\tilde{\tau}_{i,k}, \tilde{\nu}_{i,k})$ related to only one hidden unit k are updated, the algorithm will typically converge in less than 10 iterations. The fast convergence is expected in both settings because in both cases the iterations can be interpreted as a standard EP algorithm on a linear model with known input variables. However, updating only one hidden unit at a time will induce moment inconsistencies between the approximations and the corresponding tilted distributions of the other $K-1$ hidden unit activations $h_{i,k}$ and the output weights $\mathbf{v}$. This means that such update scheme would require many separate EP runs for each hidden unit and $\mathbf{v}$ to achieve overall convergence, and in practice it was found more efficient to update all of them together simultaneously with a sufficient level of damping.

The updates on $\tilde{\boldsymbol{\alpha}}_i$ and $\tilde{\boldsymbol{\beta}}_i$ were damped more strongly by $\delta \in 0.2$ so that subsequent changes in $q(\mathbf{v})$ would not inflict unnecessary fluctuations in the parameters of $q(\mathbf{w}_k)$, which are more difficult to determine and converge more slowly compared with $q(\mathbf{v})$. In other words, we wanted to change the output weight approximations more slowly so that there is enough time for the messages to propagate between the different hidden units. For the same reason, on the line 7 of Algorithm 1, parallel updates are done on $q(\mathbf{v})$ whereas the user can choose between sequential and parallel updates for $q(\mathbf{w}_k)$ (lines 5 and 6). With sequential posterior updates for $q(\mathbf{w}_k)$, damping the updates of $\tilde{\boldsymbol{\tau}}_i$ and $\tilde{\boldsymbol{\nu}}_i$ with $\delta \in [0.5, 0.8]$ was found sufficient whereas with parallel updates much more damping ($\delta < 0.5$) was usually required. If the number of input features is very large, it can be more efficient to use parallel updates for $q(\mathbf{w}_k)$ with a larger amount of damping in a similar framework as described by Gerven et al. (2009). In this large scale parallel learning scheme it may also be useful to update only one hidden unit or a subset of them at a time.

3.5.6 Scheduling and Convergence of the Prior Term Iterations

The EP updates for the site approximations of the prior terms $p(v_k)$ and $p(w_j|\phi_{l_j})$ are computationally less expensive and converge faster compared with the likelihood term approximations. With fixed values of $\{\tilde{\boldsymbol{\tau}}_i, \tilde{\boldsymbol{\nu}}_i, \tilde{\boldsymbol{\alpha}}_i, \tilde{\boldsymbol{\beta}}_i\}_{i=1}^n$, typically only 5-10 iterations were required for convergence of the updates on the prior term approximations $\tilde{t}_{v,k}(v_k)$ in line 8 of Algorithm 1, because $q(\mathbf{v})$ was allowed to change relatively slowly by damping the updates on $\tilde{\boldsymbol{\alpha}}_i$ and $\tilde{\boldsymbol{\beta}}_i$ in line 4. Because the relative time required for these computations is negligible compared with the likelihood term updates in lines 2-7, we ran the EP algorithm for $\tilde{t}_{v,k}(v_k)$ to convergence after each parallel update of $q(\mathbf{v})$ on line 7 to make sure that the components of $\mathbf{v}$ are distributed at positive values at all times.

With fixed likelihood term approximations, typically 10-40 iterations were required for convergence of the EP updates on the site approximations of $p(w_j|\phi_{l_j})$ in line 1 of Algorithm 1. More iterations are required compared to the EP algorithm on $p(v_k)$, because information needs to be propagated in multiple passes between the different hidden unit approximations $q(\mathbf{w}_k)$ via the hierarchical scale parameter approximations $q(\phi_l)$. After sufficient convergence is achieved for the likelihood term updates with the initial Gaussian priors defined

using $\tilde{\mu}_{w,j}$ and $\tilde{\sigma}^2_{w,j}$, at least two sensible update schemes can be considered for updating the site approximations of the input weight priors:

1. The EP algorithm in line 1 is run only once until convergence and then the other parameters $\{\tilde{\boldsymbol{\tau}}_i, \tilde{\boldsymbol{\nu}}_i, \tilde{\boldsymbol{\alpha}}_i, \tilde{\boldsymbol{\beta}}_i, \tilde{\mu}_{\theta,i}, \tilde{\sigma}^2_{\theta,i}\}_{i=1}^n$ and $\{\tilde{\mu}_{v,k}, \tilde{\sigma}^2_{v,k}\}_{k=1}^K$ are iterated to convergence with fixed $\{\tilde{\mu}_{w,j}, \tilde{\sigma}^2_{w,j}\}_{j=1}^{Kd}$.

2. The EP algorithm in line 1 is run once until convergence and after that only one inner iteration is done on $\{\tilde{\mu}_{w,j}, \tilde{\sigma}^2_{w,j}, \tilde{\mu}_{\phi,j}, \tilde{\sigma}^2_{\phi,j}\}_{j=1}^{Kd}$ in line 1.

In the first scheme a fixed sparsity-favoring Gaussian prior is constructed using the current likelihood term approximations whereas in the second scheme the prior is iterated further within the EP algorithm for the likelihood terms. The second scheme usually converges more slowly and requires more damping. In our experiments, damping the updates by $\delta \in [0.5, 0.7]$ and choosing a fraction parameter $\eta \in [0.7, 0.9]$ resulted in numerically stable updates and convergence for the EP algorithms on the prior term approximations.

3.5.7 Fractional EP Updates

Adjusting the fraction parameter η when the likelihood term approximations are updated according to equations (21) – (23) can have a significant effect on the behavior of the algorithm and the quality of the resulting approximation. The tilted distribution $\hat{p}_i(h_{i,k})$ approximated using (49) or (54) can become multimodal if the prediction resulting from the cavity distributions $q_{-i}(\mathbf{v})$ and $q_{-i}(\mathbf{h}_i)$ does not fit well the left out observation y_i. More precisely, there can exist one mode corresponding to the cavity $q_{-i}(h_{i,k})$ and one to the values of $h_{i,k}$ that result in good fit for the observation y_i. If η is close to one and the discrepancy between y_i and the cavity prediction is large, the resulting multimodal tilted distribution is approximated with a wide Gaussian distribution $\hat{q}_i(h_{i,k}) = \mathcal{N}(h_{i,k}|\hat{m}_{i,k}, \hat{V}_{i,k})$ to represent the uncertainty of both modes. If there are no other data points supporting the deviating information provided by y_i, the approximation needs to widen the predictive distribution at $\mathbf{x}_i$ considerably requiring large changes to $\tilde{\boldsymbol{\tau}}_i$ and $\tilde{\boldsymbol{\nu}}_i$ based on only one site term.

These kind of large local updates corresponding to sites with large discrepancies are inherently more challenging in terms of finding stable fixed points of the message passing algorithm and require therefore more damping. Furthermore, the approximation may not fit well the training data if there are isolated data points that cannot be considered as outliers. If smaller value of η is chosen, for example $\eta \in [0.4, 0.7]$, a fraction $1 - \eta$ of the site approximations $\{\tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k|\tilde{\tau}_{i,k}, \tilde{\nu}_{i,k})\}_{k=1}^K$ from the previous iteration is left in the cavity distribution and the discrepancy between the cavity prediction and y_i is usually smaller. Consequently, the model fits more strongly to the training data, the EP updates are numerically more robust, and usually less damping is required. However, in the experiments we found that with smaller values of η the model can also overfit, because more and more past information is accumulated in $\hat{p}_i(h_{i,k})$ during subsequent iterations. Therefore we set $\eta = 0.95$ and applied more damping for the likelihood term updates as described already in Section 3.5.5. The tilted distributions related to the EP updates for the prior terms approximations are not likely to become multimodal unless d or K are not very large compared to n, which is the typical setting in highly underdetermined linear models, or extremely sparse

prior settings are chosen. In our experiments the prior term approximations were not found to be sensitive to the choice of the fraction parameter which is why we used smaller values $\eta = [0.7, 0.9]$ to improve numerical stability as described in Section 3.5.6.

4. Experiments

First, three case studies with simulated data were carried out to illustrate the properties of the proposed EP-based neural network approach with sparse priors (NN-EP). Case 1 compares the effects of integration over the uncertainty resulting from a sparsity-favoring prior with a point-estimate based ARD solution. Case 2 illustrates the benefits of sparse ARD priors on regularizing the proposed NN-EP solution in the presence of irrelevant features and various input effects with different degree of nonlinearity. Case 3 compares the parametric NN-EP solution to an infinite Gaussian process network using observations from a discontinuous latent function. In cases 1 and 3, comparisons are made with an infinite network (GP-ARD) implemented using a Gaussian process with a neural network covariance function and ARD-priors with separate variance parameters for all input weights (Williams, 1998; Rasmussen and Williams, 2006). The neural-network covariance function for the GP-prior can be derived by letting the number of hidden units approach infinity in a 2-layer MLP network that has cumulative Gaussian activation functions and fixed zero-mean Gaussian priors with separate variance (ARD) parameters on the input-layer weights related to each input variable (Williams, 1998). Point estimates for the ARD parameters, the variance parameter of the output weights, and the noise variance were determined by optimizing the marginal likelihood with uniform priors on the log-scale. Finally, the predictive accuracy of NN-EP is assessed with four real-world data sets and comparisons are made with a neural network GP with a single variance parameter for all input features (GP), a GP with ARD priors (GP-ARD), and a neural network with hierarchical ARD priors (NN-MC) inferred using MCMC as described by Neal (1996).

4.1 Case 1: Overfitting of the ARD

The first case illustrates the overfitting of ARD with a similar example as presented by Qi et al. (2004). Figure 2 shows a two-dimensional regression problem with two relevant inputs x_1 and x_2. The data points are obtained from three clusters, $\{f(\mathbf{x}) = 1 | x_1 > 0.5, x_2 > 0.5\}$, $\{f(\mathbf{x}) = 0 | 0.5 > x_1 > -0.5, 0.5 > x_2 > -0.5\}$, and $\{f(\mathbf{x}) = 0.8 | x_1 < -0.5, x_2 < -0.5\}$. The noisy observations were generated according to $y = f(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.1^2)$. The observations can be explained by using a combination of two step functions with only either one of the input features but a more robust model can be obtained by using both of them.

Subfigure (a) shows the predictive mean of the latent function $f(\mathbf{x})$ obtained with the optimized GP-ARD solution. Input x_2 is effectively pruned out and almost a step function is obtained with respect to input x_1. Subfigure (b) shows the NN-EP solution with $K = 10$ hidden units and Laplace priors with one common unknown scale parameter ϕ_1 on the input weights $\mathbf{w}$. The prior for ϕ_1 was defined as $\phi_1 \sim \mathcal{N}(\mu_{\phi,0}, \sigma^2_{\phi,0})$, where $\mu_{\phi,0} = 2\log(0.1)$ and $\sigma^2_{\phi,0} = 1.5^2$. The noise variance σ^2 was inferred using the same prior definition for both models: $\theta = \log(\sigma^2) \sim \mathcal{N}(\mu_{\theta,0}, \sigma^2_{\theta,0})$, where $\mu_{\theta,0} = 2\log(0.05)$ and $\sigma^2_{\theta,0} = 1.5^2$. NN-EP produces a much smoother step function that uses both of the input features. Despite of

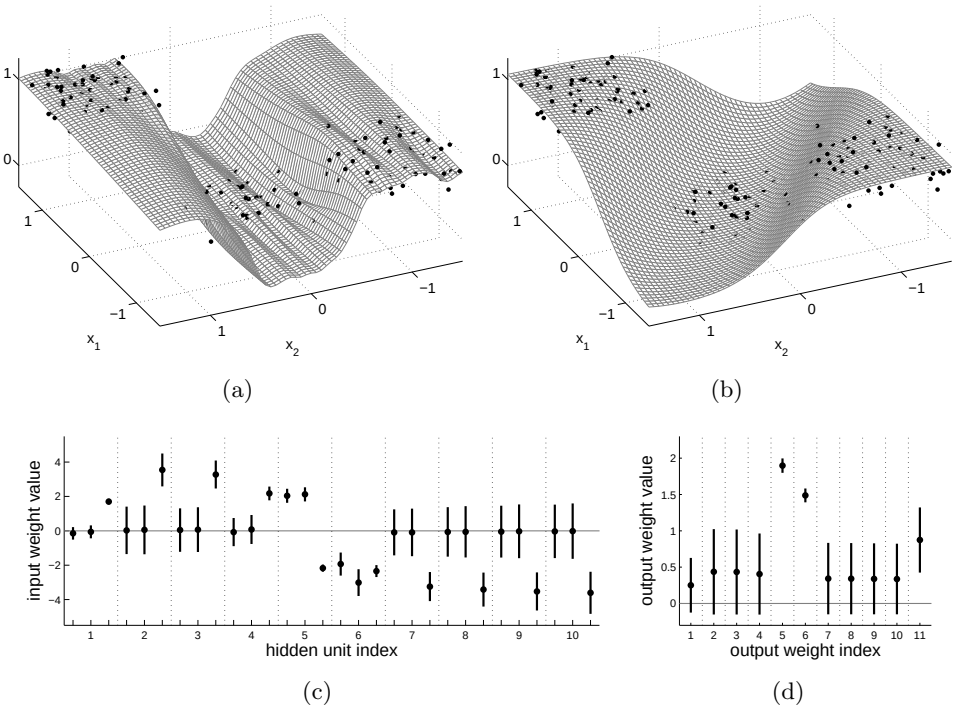


Figure 2: Case 1: An example of the overfitting of the point-estimate based ARD on a simulated data set with two relevant input features. (a) A GP model with a neural network covariance function and point-estimates for the ARD parameters. (b) An EP approximation for a neural network with 10 hidden units and independent Laplace priors with one common unknown scale parameter ϕ on the input weights. (c) and (d) The 95 % approximate marginal posterior probability intervals for the input weights and the output weights of the EP-based neural network.

the sparsity favoring Laplace prior, the NN-EP solution preserves the uncertainty on the input variable relevances. This shows that the approximate integration over the weight prior can help to avoid pruning out potentially relevant inputs.

Subfigure (c) shows the 95% approximate marginal posterior probability intervals derived from the Gaussian approximations $q(\mathbf{w}_k)$ with the same ordering of the weights as in vector $\mathbf{z}^{\mathrm{T}} = [\mathbf{w}_1^{\mathrm{T}}, ..., \mathbf{w}_K^{\mathrm{T}}]$ (every third weight corresponds to the input bias term). The vertical dotted lines separate the input weights associated with the different hidden units. Subfigure (d) shows the same marginal posterior intervals for the output weights computed using $q(\mathbf{v})$. Only hidden units 5 and 6 have clearly nonzero output weights and input weights corresponding to the input variables x_1 and x_2 (see the first two weight distributions in triplets 5 and 6 in panel (c)). For the other hidden units, the input weights related to x_1 and x_2 are distributed around zero and they have negligible effect on the predictions. In panel (c), the third input weight distribution corresponding to the bias term in each triplet are distributed in nonzero values for many unused hidden units but these bias effects

affect only the mean level of the predictions. These nonzero bias weight values may be caused by the observations not being normalized to zero mean. The weights corresponding to hidden unit 1 differ from the other unused units, because a linear action function was assigned to it for illustration purposes. If required, a truly sparse model could be obtained by removing the unused hidden units and running additional EP iterations until convergence.

4.2 Case 2: The Effect of Sparse Priors in a Regression Problem Consisting of Additive Input Effects with Different Degree of Nonlinearity

The second case study illustrates the effects of sparse priors using a similar regression example as considered by Lampinen and Vehtari (2001). In our experiments we found two main effects from applying sparsity-promoting priors with adaptive scale parameters $\boldsymbol{\phi} = [\phi_1, ..., \phi_L]$ on the input-layer. Firstly, the sparse priors can help to suppress the effects of irrelevant features and protect from overfitting effects in input variable relevance determination as illustrated in Case 1 (Section 4.1). Secondly, sparsity-promoting priors with adaptive prior scale parameters $\boldsymbol{\phi}$ can mitigate the effects of unsuitable initial Gaussian prior definitions on the input layer (too large or too small initial prior variances $\tilde{\sigma}^2_{w,j}$, see Section 3.5 for discussion on the initialization). More precisely, the sparse priors with adaptive scale parameters can help to obtain better data fit and more accurate predictions by shrinking the uncertainty on the weights related to irrelevant features towards zero and by allowing the relevant input weights to gain larger values which are needed in modeling strongly nonlinear (or step) functions. Placing very large initial prior variances $\tilde{\sigma}^2_{w,j}$ on all weights enables the model to fit strong nonlinearities but the initial learning phase is more challenging and prone to end up in poor local minima. In this section, we demonstrate that switching to Gaussian ARD priors with adaptive scale parameter $\phi_1, ..., \phi_d$ after a converged EP solution is obtained with fixed Gaussian priors can reduce the effects of irrelevant features, decrease the posterior uncertainties on the predictions on $f(\mathbf{x})$, and enable the model to fit more accurately latent nonlinear effects.

A data set with 200 observations and ten input variables with different additive effects on the target variable was simulated. The black lines in Figure 3 show the additive effects as a function of each input variable $x_{i,j}$. The targets y_i were calculated by summing the additive effects together and adding Gaussian noise with a standard deviation of 0.2. The first input variable is irrelevant and variables 2-5 have an increasing linear effect on the target. The effects of input variables 6-10 are increasingly nonlinear and the last three of them require at least three hidden units for explaining the observations.

Figure 3(a) shows the converged NN-EP solution with fixed zero-mean Gaussian priors on the input weights. The number of hidden units was set to $K = 10$ and the noise variance σ^2 was inferred using the prior definition $\mu_{\theta,0} = 2\log(0.05)$ and $\sigma^2_{\theta,0} = 2^2$. The Gaussian priors were defined by initializing the prior site parameters of the input weights as $\{\tilde{\mu}_{w,j} = 0, \tilde{\sigma}^2_{w,j} = 0.4^2\}_{j=1}^{Kd}$. The dark grey lines illustrate the posterior mean predictions and the shaded light gray area the 95% approximate posterior predictive intervals of the latent function $f(\mathbf{x})$. The graphs are obtained by changing the value of each input in turn from -5 to 5 while keeping the others fixed at zero. The training observations are obtained by sampling all input variables linearly from the interval $x_{i,j} \in [-\pi, \pi]$. Panel (b) shows the resulting NN-EP solution when the Gaussian priors of the network in panel (a) are

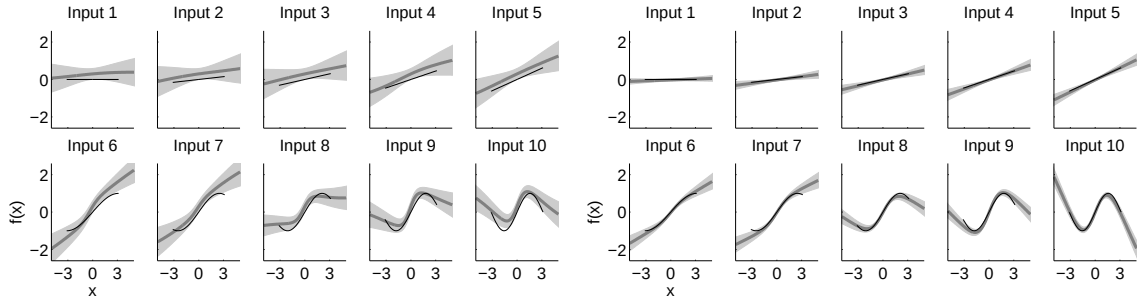
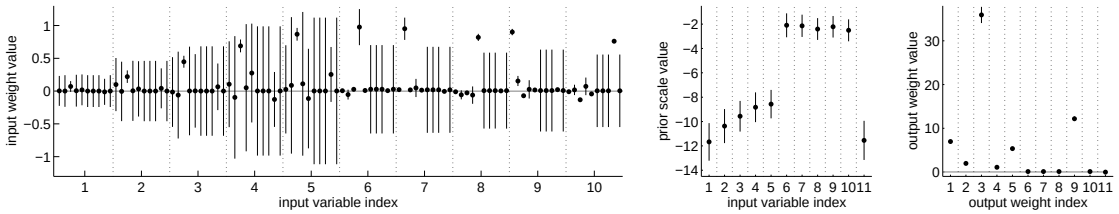


Figure 3: Case 2: An artificial regression problem where the observations are formed as a sum of additive input effects dependent on ten input features. The true additive effects are shown with black lines and the estimated mean predictions with dark grey lines. The 95% posterior predictive intervals are shaded with light grey. (a) A converged EP approximation for a neural network with ten hidden units and fixed zero-mean Gaussian priors on the input weights. (b) The resulting EP approximation when the Gaussian priors of the network in panel (a) are replaced with Gaussian ARD priors with separate scale parameters $\phi_1, ..., \phi_d$ for all input variables, and additional EP iterations are done until a new converged solution is obtained. Figure 4 visualizes the approximate posterior distributions of the parameters of the ARD network from panel (b).

replaced with Gaussian ARD priors with adaptive scale parameter $\phi_1, ..., \phi_d$ and additional EP iterations are done until convergence. Prior distributions for the scale parameters were defined as $\phi_l \sim \mathcal{N}(\mu_{\phi,0}, \sigma^2_{\phi,0})$, where $\mu_{\phi,0} = 2\log(0.01)$ and $\sigma^2_{\phi,0} = 2.5^2$. This prior definitions favors small input variances close to 0.01 but enables also larger values around one. It should be noted that the actual variance parameters $\tilde{\sigma}^2_{w,j}$ of the prior site approximations can attain much larger values from the EP updates.

With the Gaussian priors (Figure 3(a)), the predictions do not capture the nonlinear effects very accurately and the model produces a small nonzero effect on the irrelevant input 1. Applying the ARD priors (Figure 3(b)) with additional iterations produces clearly more accurate predictions on the latent input effects and effectively removes the predictive effect of input 1. The overall approximate posterior uncertainties on the latent function $f(\mathbf{x})$ are also smaller compared with the initial Gaussian priors. We should note that the result of panel (a) depends on the initial Gaussian prior definitions and choosing a smaller $\tilde{\sigma}^2_{w,j} = 0.2^2$ or optimizing it could give more accurate predictions compared with the solution visualized in panel (a).

Figure 4 shows the 95% posterior credible intervals for the input weights $\mathbf{w}$ (a), the prior scale parameters $\phi_1, ..., \phi_d$ (b), and the output weights $\mathbf{v}$ (c) of the NN-EP approximation with ARD priors visualized in Figure 3(b). In panel (a) the input weights from the different hidden units are grouped together according to the different additive input effects 1–10, and the weights related to the linear effects 1–5 are scaled by 40 for illustration purposes,



(a) (b) (c)

Figure 4: Case 2: Visualization of the model parameters related to the artificial regression problem shown in Figure 3. Panels (a), (b), and (c) show the 95% marginal posterior credible intervals for the input weights $\mathbf{w}$, the scale parameters $\phi_1, ..., \phi_d$, and the output weights $\mathbf{v}$ of the neural network with Gaussian ARD priors from Figure 3(b). In panel (a) the input weights associated with each additive input effect (1-10) are grouped together (the bias terms are not shown). The weight distributions related to the linear input effects 1–5 are much smaller compared with the nonlinear effects 6–10, which is why they are scaled by 40 for better illustration in panel (a).

because they are much smaller compared with the weights associated with the nonlinear input effects 6–10. From panels (a) and (c) we see that only hidden units are 1–5 and 9 have clearly non-zero effect on the predictions. The linear effects of inputs 1–5 are modeled by unit 3 that has very small but clearly nonzero input weights in panel (a) and a very large output weight in panel (a). The input weights related to the irrelevant input 1 are all zero in the 95% posterior credible level. By comparing panels (a) and (c) we can also see that hidden units 1, 2, 4, 5, and 9 are most probably responsible for modeling the nonlinear input effects 6-7 because of large input weights values. Panel (b) gives further evidence on this interpretation because the scale parameters associated with the nonlinear input effects 6–10 are clearly larger compared to effects 1–5. The scale parameters associated with the linear input effects 1–5 increase steadily as the magnitudes of the true effects increase. These results are congruent with the findings of Lampinen and Vehtari (2001) who showed by MCMC experiments that with MLP models the magnitudes of the ARD parameters and the associated input weights also reflect the degree of nonlinearity associated with the latent input effects, not only the relevance of the input features.

4.3 Case 3: Comparison of a Finite vs. Infinite Network with Observations from a Latent Function with a Discontinuity

The third case study compares the performance of the finite NN-EP network with an infinite GP network in a one-dimensional regression problem with a strong discontinuity. Figure 5 shows the true underlying function (black lines) that has a discontinuity at zero together with the noisy observations (black dots). Panel (a) shows the predictive distributions obtained using NN-EP with ten hidden units ($K = 10$) and Laplace priors with one common scale parameter ϕ. The prior distribution for the scale parameter was defined

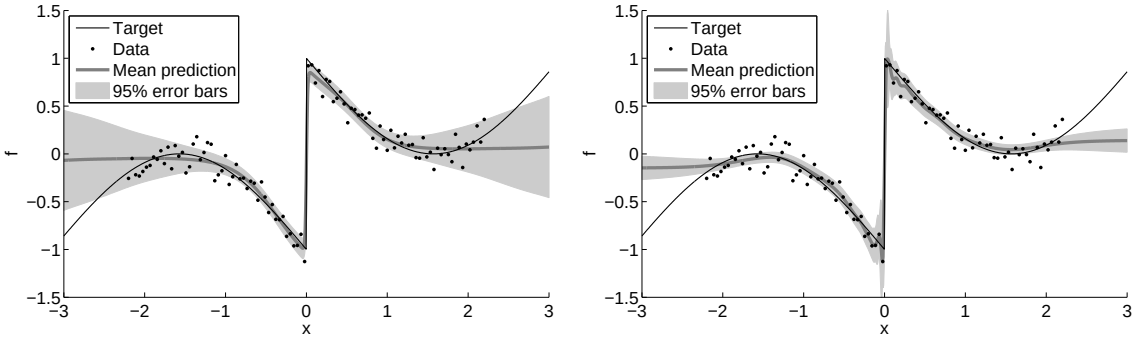


Figure 5: Case 3: An artificial regression problem consisting of noisy observations (black dots) generated from a latent function (black lines) that has a discontinuity at zero. Panel (a) shows the mean predictions (dark grey line) and the 95% credible intervals (light gray shaded area) obtained using the proposed EP approach for a NN with ten hidden units and Laplace priors with one common scale parameter ϕ on the input weights. Panel (b) visualizes the corresponding predictive distribution obtained using a GP with a neural network covariance function.

with $\mu_{\phi,0} = 2\log(0.01)$ and $\sigma^2_{\phi,0} = 2.5^2$, and the noise variance σ^2 was inferred from the data using the prior definition $\mu_{\theta,0} = 2\log(0.05)$ and $\sigma^2_{\theta,0} = 2^2$. Panel (b) shows the corresponding predictions obtained using a GP with a neural network covariance function. With the GP network the noise variance was optimized together with the other hyperparameters using the marginal likelihood. The finite NN-EP network explains the discontinuity with a slightly smoother step compared to infinite GP network, but the GP mean estimate shows fluctuations near the discontinuity. It seems that the infinite GP network fits more strongly to individual observations near the discontinuity. This shows that a flexible parametric model with a limited complexity may generalize better with finite amount of observations even though the GP model includes the correct solution a priori. This is in accordance with the results described by Winther (2001).

4.4 Predictive Comparisons with Real World Data

In this section the predictive performance of NN-EP is compared to three other nonlinear regression methods using the following real-world data sets: the concrete quality data (Concrete) analyzed by Lampinen and Vehtari (2001), the Boston housing data (Housing) and the unnormalized Communities and Crime data (Crime) that can be obtained from the UCI data repository (Bache and Lichman, 2013), and the robot arm data (Kin40k) utilized by Schwaighofer and Tresp (2003).[3] The number of observations n and the number of input features d are shown in Table 1 for each data set. The Kin40k includes originally only 8

3. Kin40k data is based on the same simulation of the forward kinematics of an 8 link all-revolute robot arm as the Kin family of data sets available at `http://www.cs.toronto.edu/~delve/` except for lower noise level and larger amount of observations.

input features but we added 92 irrelevant uniformly sampled random inputs to create a challenging feature selection problem. The columns of the input matrices $\mathbf{X}$ and the output vectors $\mathbf{y}$ were normalized to zero mean and unit variance for all methods. The predictive performance of the models was measured using the log predictive densities and the squared errors evaluated with separate test data. We used 10-fold cross-validation with the Housing, Concrete, and Crime data, whereas with Kin40k we chose randomly 5000 data points for training and used the remaining observations for validation.

The proposed NN-EP solution was computed using two alternative prior definitions: Laplace priors with one common scale parameter ϕ (NN-EP-LA), and Gaussian ARD priors with separate scale parameters $\phi_1, ..., \phi_d$ for all inputs including the input bias terms (NN-EP-ARD). With both prior frameworks, the hyperpriors for the scale parameters were defined as $\phi_l \sim \mathcal{N}(\mu_{\phi,0}, \sigma^2_{\phi,0})$, where $\mu_{\phi,0} = 2\log(0.01)$ and $\sigma^2_{\phi,0} = 2.5^2$. This definition encourages small input weight variances (around 0.01^2) but enables also large input weight values if required for strong nonlinearities assuming the input variables are scaled to unit variance. The noise level $\theta = \log(\sigma^2)$ was inferred from data with a prior distribution defined by $\mu_{\theta,0} = 2\log(0.01)$ and $\sigma^2_{\theta,0} = 2^2$, which is a sufficiently flexible prior when the output variables $\mathbf{y}$ are scaled to unit variance. The methods used for comparison include an MCMC-based MLP network with ARD priors (NN-MC) and two GPs with a neural network covariance function: one with common variance parameter for all inputs (GP), and another with separate variance hyperparameters for all inputs (GP-ARD). With both GP models the hyperparameters were estimated by gradient-based optimization of the analytically tractable marginal likelihood (Rasmussen and Williams, 2006). For NN-MC and NN-EP, we set the number of hidden units to $K = 10$ with the Housing, Concrete, and Crime data sets. With the Kin40k data, we set $K = 30$ because n is large and fewer units were found to produce clearly worse data fits.

Table 1 summarizes the means (mean) and standard deviations (std) of the log predictive densities (LPDs) and the squared errors (SEs). Because the distributions of the LPD values are heavily skewed towards negative values, we summarize also the lower 1% percentiles (prct 1%). Similarly, because the SE values are skewed towards positive values we summarize also the 99% percentiles (prct 99%). These additional measures describe the quality of the worst case predictions of the methods. Table 1 summarizes also the average relative CPU times (cputime) required for parameter estimation and predictions using MATLAB implementations. The GP models were implemented using the GPstuff toolbox (Vanhatalo et al., 2013) and NN-MC was implemented using the MCMCstuff toolbox.[4] The CPU times were averaged over the CV-folds and scaled so that the relative cost for NN-EP is one. These running time measures are highly dependent on the implementation, the tolerance levels in optimization and iterative algorithms, and the number of posterior draws, and therefore they are reported only to summarize the main properties regarding the scalability of the different methods. When assessing the results with respect to the Housing and Concrete data sets, it is worth noting that there is evidence that an outlier-robust observation model is beneficial over the Gaussian model used in this comparison with both data sets (Jylänki et al., 2011).

4. The MCMCstuff toolbox can be obtained from `http://becs.aalto.fi/en/research/bayes/mcmcstuff/` and the GPstuff toolbox from `http://becs.aalto.fi/en/research/bayes/gpstuff/`.

Housing	log predictive density (LPD)			squared error (SE)			
(n=506, d=13)	mean	std	prct 1%	mean	std	prct 99%	cputime
NN-EP-LA	-0.44	1.64	-7.55	0.15	0.45	2.42	1.0
NN-EP-ARD	-0.50	1.66	-6.31	0.17	0.49	1.60	1.0
NN-MC	-0.08	1.17	-4.54	0.11	0.50	1.18	110.5
GP	-0.29	2.35	-7.57	0.13	0.53	1.98	0.3
GP-ARD	-0.20	2.00	-10.71	0.10	0.37	1.53	1.0
Concrete (n=215, d=27)							
NN-EP-LA	0.18	0.85	-3.05	0.05	0.08	0.30	1.0
NN-EP-ARD	0.05	1.03	-4.61	0.05	0.11	0.57	0.8
NN-MC	0.22	1.52	-3.62	0.04	0.08	0.28	103.0
GP	-0.07	1.70	-5.12	0.06	0.11	0.66	0.03
GP-ARD	0.15	1.98	-4.23	0.04	0.08	0.28	0.6
Crime (n=1993, d=102)							
NN-EP-LA	-0.83	0.89	-4.64	0.31	0.55	2.60	1.0
NN-EP-ARD	-0.84	0.89	-4.81	0.31	0.55	2.75	0.2
NN-MC	-0.80	0.93	-4.81	0.29	0.53	2.60	19.8
GP	-0.81	0.91	-4.80	0.30	0.54	2.69	0.2
GP-ARD	-0.81	1.01	-5.49	0.30	0.55	2.75	4.4
Kin40k (n=5000, d=100)							
NN-EP-LA	-0.59	0.89	-4.27	0.19	0.29	1.38	1.0
NN-EP-ARD	0.27	1.19	-4.63	0.03	0.08	0.37	0.9
NN-MC	0.49	1.51	-5.37	0.02	0.07	0.26	48.7
GP	-1.15	0.72	-4.18	0.58	0.83	4.06	0.5
GP-ARD	0.64	1.11	-3.90	0.02	0.05	0.24	32.3

Table 1: A predictive assessment of the proposed EP approach for neural networks with two different prior definitions: Laplace priors with one common scale parameter ϕ (NN-EP-LA) and Gaussian ARD priors with separate scale parameters $\phi_1, ..., \phi_d$ for all inputs (NN-EP-ARD). Comparisons are made with a neural network with ARD priors inferred using MCMC (NN-MC), and two GPs with a neural network covariance: one with a common variance hyperparameter for all inputs (GP), and another with separate variance hyperparameters for all inputs (GP-ARD). The log predictive densities are summarized with their means, standard deviations (std), and lower 1% percentiles (prct 1%). The squared errors are summarized with their means, standard deviations (std), and upper 99% percentiles (prct 99%).

Table1 shows that NN-EP-LA performs slightly better compared to NN-EP-ARD in all data sets except in Kin40k, where NN-EP-ARD gives clearly better results. The main reason for this is probably the stronger sparsity of the NN-EP-ARD solutions: In Kin40k data there are a large number truly irrelevant features that should be completely pruned out of the model, whereas with the other data sets most features have probably some relevance

for predictions or at least they are not generated in a completely random manner. Further evidence for this is given by the clearly better performance of GP-ARD over GP with the Kin40k data.

If the mean log predictive densities (MLPDs) are considered, the NN-MC approach based on a finite network performs best in all data sets except with Kin40k, where the infinite GP-ARD network is slightly better. The main reason for this is probably the strong nonlinearity of the true latent mapping, which requires a large number of hidden units, and consequently the infinite GP network with ARD priors gives very accurate predictions. In pair-wise comparisons the differences in MLPDs are significant in 95% posterior credible level only with Housing and Kin40k data sets. In terms of mean squared errors (MSEs), GP-ARD is best in all data sets except Crime, but with 95% credible level the pair-wise differences are significant only with the Kin40k data. With the Kin40k data, the performance of NN-MC could probably be improved by increasing K or drawing more posterior samples, because learning the nonlinear mapping with a large number of unknown parameters and potentially multimodal posterior distribution may require a very large number of posterior draws.

When compared with NN-MC and GP-ARD, NN-EP gives slightly worse MLPD scores with all data sets except with Concrete. The pair-wise differences in MLPDs are significant with 95% credible level in all cases except with the Concrete data. In terms of MSE scores, NN-EP is also slightly but significantly worse with 95% credible level in all data sets. By inspecting the std:s and 1% percentiles of the LPDs, it can be seen that NN-EP achieves better or comparable worst case performance when compared to GP-ARD. In other words, NN-EP seems to make more cautions predictions by producing less very high or very low LPD values. One possible explanation for this behavior is that it might be an inherent property of the chosen approximation. Approximating the possibly multimodal tilted distribution $\hat{p}(h_{i,k})$, where one mode is near the cavity distribution $q_{-i}(h_{i,k})$ and another at the values of $h_{i,k}$ giving the best fit for y_i, with an unimodal Gaussian approximation as described in Appendix C, may lead to reduced fit to individual observations. Another possibility is that the EP-iterations have converged into a suboptimal stationary solution or the maximum number of iterations has been exceeded. Doing more iterations or using an alternative non-zero initialization for the input-layer weights might result in better data fit. The second possibility is supported by the generally acknowledged benefits from different initializations, for example, the unsupervised schemes discussed by Erhan et al. (2010), and our experiments using the Kin40k data without the extra random inputs. We found that initializing the location parameters $\tilde{\mu}_{v,k}$ and $\tilde{\mu}_{w,j}$ of the prior site approximations (13) and (14) using a gradient-based MAP estimate of the weights $\mathbf{w}$ and $\mathbf{v}$, and relaxing the prior site approximations after initial iterations using the proposed EP framework, can result in better MSE and MLPD scores. However, such alternative initialization schemes were left out of these experiments, because our aim was to test how good performance could be obtained using only the EP algorithm with the zero initialization described in Section 3.5.

The CPU times of Table 1 indicate that with small n the computational cost of NN-EP is larger compared to GP-ARD, which requires only one $\mathcal{O}(n^3)$ Cholesky decomposition per analytically tractable marginal likelihood evaluation. However, as n increases GP-ARD becomes slower, which is why several different sparse approximation schemes have been proposed (see, e.g, Rasmussen and Williams, 2006). Furthermore, assuming a non-Gaussian observation model, such as the binary probit classification model, GP or GP-ARD would

require several $\mathcal{O}(n^3)$ iterations to form Laplace or EP approximations for the marginal likelihood at each hyperparameter configuration. With NN-EP, probit or Gaussian mixture models could be used without additional computations. The computational cost of NN-EP increases linearly with n and K, but as d increases the posterior updates of $q(\mathbf{w}_k)$, which scale as $\mathcal{O}(Kd^3)$, become more demanding. The results of Table 1 were generated using a sequential scheme for updating $q(\mathbf{w}_k)$ (see Algorithm 1), which can be seen as larger computational costs with respect to NN-MC with the Crime and Kin40k data sets. One option with larger d is to use parallel EP updates, but this may require more damping or better initialization for the input weight approximations. Another possibility would be to use fully factorized posterior approximations in place of $q(\mathbf{w}_k)$, or to assign different overlapping subgroups of the input features into the different hidden units and to place hierarchical prior scale parameters between the groups.

5. Discussion

In this article, we have described how approximate inference using EP can be carried out with a two-layer NN model structure with sparse hierarchical priors on the network weights, resulting in a novel method for nonlinear regression problems.

We have described a computationally efficient EP algorithm that utilizes independent approximations for the weights associated with the different hidden units and layers to achieve computational complexity scaling similar to an ensemble of K sparse linear models. More generally, our approach can be regarded as a non-linear adaptation of the various EP methods proposed for sparse linear regression models. This is achieved by constructing a factorized Gaussian approximation for the posterior distribution resulting from the nonlinear MLP model structure with a linear input layer, and adapting the algorithms proposed for sparse linear models on the Gaussian approximations of the hidden units. Because of the structure of the approximation, all existing methodology presented for facilitating the computations in sparse linear models can be applied on the hidden unit approximations separately. We have also introduced an EP framework that enables definition of flexible hierarchical priors using higher level scale parameters that are shared by a group of independent linear models (in our case the hidden units). The proposed EP approach enables efficient approximate integration over these scale parameters simultaneously with the coefficients of the linear models. We used this framework for inferring the common scale parameter of Laplace priors assigned to the input weights, and to implement Gaussian ARD priors for the input-layer. In this article, we have focused on the Gaussian observation model, but the method can be readily extended to others as well (e.g., binary probit classification and robust regression with Gaussian mixture models).

Using simple artificial examples we demonstrated several desirable characteristics of our approach. The sparsity promoting priors can be used to suppress the confounding predictive influences of possibly irrelevant features without the potential risk of overfitting associated with point-estimate-based ARD priors. More precisely, the approximate integration over the posterior uncertainty helps to avoid pruning out potentially relevant features in cases with large uncertainty on the input relevances. Albeit more challenging to estimate, the finite parametric model enables a posteriori inspection of the model structure and feature relevances using the hyperparameter and weight approximations. Furthermore, the para-

metric model structure can also be used to construct more constrained models by assigning different input variables into different hidden units, defining overlapping groups for the inputs using the hierarchical scale priors, using different nonlinear activation functions for the different hidden units, or using fixed interaction terms dependent on certain hidden units as inputs for the output-layer.

In the derivations of the EP algorithm, we have also described different computational techniques that could be useful in other models and approximation methods. These include the EP approximation for the hierarchical priors on the scale parameters of the weights that could be useful in combining sparse linear models associated with different subjects or measurement instances, the noise estimation framework that could be used for estimating the likelihood parameters in sparse linear models or approximate Gaussian filtering methods, and the proposed approach for approximating the tilted distributions of the hidden unit activations that could be useful in forming EP approximations for observation models involving sums of nonlinear functions taken from random variables with factorized Gaussian posterior approximations.

A Matlab demonstration code implementing the proposed EP approach for neural networks will be made available at `http://becs.aalto.fi/en/research/bayes/epnn/`.

Acknowledgments

We thank the anonymous referees for their helpful feedback and suggestions. The research reported in this publication was partially funded by the Academy of Finland (grant 218248).

Appendix A. Cavity Distributions with the Factorized Approximation

Because the likelihood terms $p(y_i|\mathbf{v}^{\mathrm{T}}\mathbf{g}(\mathbf{h}_i),\theta)$ depend on the input weights $\mathbf{w}$ only through the linear transformation $\mathbf{h}_i = [h_{i,1},...,h_{i,K}]^{\mathrm{T}}$, where $h_{i,k} = \mathbf{w}_k^{\mathrm{T}}\mathbf{x}_i$, the EP updates can be implemented by propagating the moments of $\mathbf{h}_i$ and $\mathbf{v}$. Assuming the factorized approximation (19) for $\mathbf{w}_1,...,\mathbf{w}_K$ and $\mathbf{v}$, the parameters of the cavity distribution (21) can be determined from

$$q_{-i}(\mathbf{w},\mathbf{v},\theta) = q_{-i}(\mathbf{v})\prod_{k=1}^{K} q_{-i}(\mathbf{w}_k)q_{-i}(\theta) \propto q(\mathbf{v})\prod_{k=1}^{K} q(\mathbf{w}_k)q(\theta)\Big(\tilde{t}_{\mathbf{v},i}(\mathbf{v})\prod_{k=1}^{K}\tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k)\tilde{t}_{\theta,i}(\theta)\Big)^{-\eta},$$

which can be transformed into

$$q_{-i}(\mathbf{h}_i,\mathbf{v},\theta) = q_{-i}(\mathbf{v})\prod_{k=1}^{K} q_{-i}(h_{i,k})q_{-i}(\theta) \tag{38}$$

by applying the transformation $h_{i,k} = \mathbf{w}_k^{\mathrm{T}}\mathbf{x}_i$. Plugging in $q(\mathbf{w}_k) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{w}_k}, \boldsymbol{\Sigma}_{\mathbf{w}_k})$ from (36) and the site approximations $\tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k|\tilde{\tau}_{i,k},\tilde{\nu}_{i,k})$ from (35), and doing the transformation $h_{i,k} = \mathbf{w}_k^{\mathrm{T}}\mathbf{x}_i$, results in the following scalar mean and variance for $q_{-i}(h_{i,k}) = \mathcal{N}(h_{i,k}|m_{-i,k},V_{-i,k})$:

$$\begin{aligned} V_{-i,k} &= (V_{i,k}^{-1} - \eta\tilde{\tau}_{i,k})^{-1} \\ m_{-i,k} &= V_{-i,k}(V_{i,k}^{-1}m_{i,k} - \eta\tilde{\nu}_{i,k}), \end{aligned} \tag{39}$$

where the mean and variance of $h_{i,k}$ under the current approximation $q(\mathbf{w}_k)$ are denoted with $m_{i,k} = \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\mu}_{\mathbf{w}_k}$ and $V_{i,k} = \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathbf{w}_k} \mathbf{x}_i$, respectively. Similarly, plugging in $q(\mathbf{v}) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{v}}, \boldsymbol{\Sigma}_{\mathbf{v}})$ from (37) and the site approximation $\tilde{t}_{\mathbf{v},i}(\mathbf{v}|\tilde{\boldsymbol{\alpha}}_i, \tilde{\boldsymbol{\beta}}_i)$ from (35) gives the cavity distribution $q_{-i}(\mathbf{v}) = \mathcal{N}(\mathbf{v}|\boldsymbol{\mu}_{-i}, \boldsymbol{\Sigma}_{-i})$ with the mean and covariance given by

$$\begin{aligned} \boldsymbol{\Sigma}_{-i} &= \boldsymbol{\Sigma}_{\mathbf{v}} + \boldsymbol{\Sigma}_{\mathbf{v}} \tilde{\boldsymbol{\alpha}}_i s^{-1} \tilde{\boldsymbol{\alpha}}_i^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathbf{v}} \\ \boldsymbol{\mu}_{-i} &= \mathbf{a} + \boldsymbol{\Sigma}_{\mathbf{v}} \tilde{\boldsymbol{\alpha}}_i s^{-1} \tilde{\boldsymbol{\alpha}}_i^{\mathrm{T}} \mathbf{a}, \end{aligned} \tag{40}$$

where $s = \eta^{-1} - \tilde{\boldsymbol{\alpha}}_i^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathbf{v}} \tilde{\boldsymbol{\alpha}}_i$ and $\mathbf{a} = \boldsymbol{\mu}_{\mathbf{v}} - \eta \boldsymbol{\Sigma}_{\mathbf{v}} \tilde{\boldsymbol{\beta}}_i$. Using $q(\theta) = \mathcal{N}(\mu_\theta, \sigma_\theta^2)$ from (17) and the site approximation $\tilde{t}_{\theta,i}(\theta|\tilde{\mu}_{\theta,i}, \tilde{\sigma}^2_{\theta,i})$ from (12) gives the cavity distribution $q_{-i}(\theta) = \mathcal{N}(\mu_{\theta,-i}, \sigma^2_{\theta,-i})$ with the mean and variance given by

$$\begin{aligned} \sigma^2_{\theta,-i} &= \left(\sigma_\theta^{-2} - \eta \tilde{\tau}_{\theta,i}\right)^{-1} \\ \mu_{\theta,-i} &= \sigma^2_{\theta,-i}(\sigma_\theta^{-2} \mu_\theta - \eta \tilde{\nu}_{\theta,i}), \end{aligned} \tag{41}$$

where the site parameters are written in their natural exponential forms $\tilde{\tau}_{\theta,i} = \tilde{\sigma}^{-2}_{\theta,i}$ and $\tilde{\nu}_{\theta,i} = \tilde{\sigma}^{-2}_{\theta,i} \tilde{\mu}_{\theta,i}$. Using (39), (40), and (41) the cavity evaluations can be implemented efficiently: for the input weights $\mathbf{w}_k$ and the noise parameter θ only scalar moments of $h_{i,1}, ..., h_{i,K}$ and θ need to be determined, and for the output weights $\mathbf{v}$ rank-one matrix updates are required.

Appendix B. Tilted Moments of the Output Weights

To obtain closed-form expressions for the parameters of the likelihood site approximations $\tilde{t}_{\mathbf{v},i}(\mathbf{v}|\tilde{\boldsymbol{\alpha}}_i, \tilde{\boldsymbol{\beta}}_i)$, $\{\tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k|\tilde{\tau}_{i,k}, \tilde{\nu}_{i,k})\}_{k=1}^K$, and $\tilde{t}_{\theta,i}(\theta|\tilde{\mu}_{\theta,i}, \tilde{\sigma}^2_{\theta,i})$ that satisfy the moment matching condition (23), we need to form suitable approximations for the marginal means and covariances of $\{h_{i,k} = \mathbf{w}_k^{\mathrm{T}} \mathbf{x}_i\}_{k=1}^K$, $\mathbf{v}$, and θ resulting from the tilted distribution (22). First, we combine the cavity distribution (38) with the i:th likelihood term to obtain a transformed tilted distribution

$$\hat{p}_i(\mathbf{h}_i, \mathbf{v}, \theta) \propto p(y_i|\mathbf{v}^{\mathrm{T}} \mathbf{g}(\mathbf{h}_i), \theta)^\eta q_{-i}(\mathbf{v}|\boldsymbol{\mu}_{-i}, \boldsymbol{\Sigma}_{-i}) \prod_{k=1}^K q_{-i}(h_{i,k}|m_{-i,k}, V_{-i,k}) q_{-i}(\theta|\mu_{\theta,-i}, \sigma^2_{\theta,-i}), \tag{42}$$

where the cavity parameters are given by (39)–(41). We start by assuming the noise level θ known and present a simple and efficient way to approximate the moments of $\mathbf{v}$ in this appendix. In Appendix C we describe a more accurate approximation scheme for the marginal moments of $h_{i,k}$, and finally extend the presented approach for approximate integration over $q_{-i}(\theta)$ in Appendix D.

In the following we consider an approximate scheme which has already been utilized in the unscented Kalman filtering framework for inferring the weights of a neural network (Wan and van der Merwe, 2000). The approach is based on the assumption that the probability distribution of the random vector $[\mathbf{u}_i^{\mathrm{T}}, \tilde{y}_i]^{\mathrm{T}} = [\mathbf{h}_i^{\mathrm{T}}, \mathbf{v}^{\mathrm{T}}, \tilde{y}_i]^{\mathrm{T}}$ that is given by $\hat{p}_i(\mathbf{h}_i, \mathbf{v}, \tilde{y}_i|\theta) \propto p(\tilde{y}_i|\mathbf{v}^{\mathrm{T}} \mathbf{g}(\mathbf{h}_i), \theta)^\eta q_{-i}(\mathbf{h}_i) q_{-i}(\mathbf{v})$, can be reasonable well approximated with a joint Gaussian approximation $\hat{q}_i(\mathbf{h}_i, \mathbf{v}, \tilde{y}_i)$. Here random variable $\tilde{y}_i$ corresponds to a target

y_i, which we assume first unknown and condition upon later. The Gaussian approximation is constructed as

$$\hat{q}_i(\mathbf{h}_i, \mathbf{v}, \tilde{y}_i) = \mathcal{N}\left(\begin{bmatrix} \mathbf{m}_{-i} \\ \boldsymbol{\mu}_{-i} \\ m_{\tilde{y}_i} \end{bmatrix}, \begin{bmatrix} \mathbf{V}_{-i} & \mathbf{0} & \boldsymbol{\Sigma}_{\mathbf{h}_i,\tilde{y}_i} \\ \mathbf{0} & \boldsymbol{\Sigma}_{-i} & \boldsymbol{\Sigma}_{\mathbf{v},\tilde{y}_i} \\ \boldsymbol{\Sigma}_{\mathbf{h}_i,\tilde{y}_i}^{\mathrm{T}} & \boldsymbol{\Sigma}_{\mathbf{v},\tilde{y}_i}^{\mathrm{T}} & V_{\tilde{y}_i} \end{bmatrix} \right), \tag{43}$$

where the marginal means and covariances of $\mathbf{h}_i$ and $\mathbf{v}$ are set equal to the cavity moments $\mathbf{m}_{-i} = [m_{-i,1}, ..., m_{-i,K}]^{\mathrm{T}}$, $\mathbf{V}_{-i} = \mathrm{diag}(V_{-i,1}, ..., V_{-i,K})$, $\boldsymbol{\mu}_{-i}$ and $\boldsymbol{\Sigma}_{-i}$ defined in (39) and (40), respectively. The dependencies of $\tilde{y}_i$ from $\mathbf{h}_i$ and $\mathbf{v}$ are approximated linearly by determining the central moments $m_{\tilde{y}_i} = \mathrm{E}(\tilde{y}_i|\theta)$, $V_{\tilde{y}_i} = \mathrm{Var}(\tilde{y}_i|\theta)$, $\boldsymbol{\Sigma}_{\mathbf{h}_i,\tilde{y}_i} = \mathrm{Cov}(\mathbf{h}_i, \tilde{y}_i|\theta)$, and $\boldsymbol{\Sigma}_{\mathbf{v},\tilde{y}_i} = \mathrm{Cov}(\mathbf{v}, \tilde{y}_i|\theta)$ with respect to $\hat{p}_i(\mathbf{h}_i, \mathbf{v}, \tilde{y}_i|\theta)$ using, e.g., the unscented transform. Approximations to the mean and covariance of the tilted distribution (42) can now be determined by conditioning on $\tilde{y}_i$ in the joint Gaussian approximation (43) to obtain $\mathrm{E}(\mathbf{u}_i|\tilde{y}_i, \theta)$ and $\mathrm{Cov}(\mathbf{u}_i|\tilde{y}_i, \theta)$, and plugging in the observation $\tilde{y}_i = y_i$.

In our experiments, this approach was found sufficiently accurate for approximating the moments of $\mathbf{v}$, which is most likely explained by the conditional linear dependence of f_i on $\mathbf{v}$ via transformation $f_i = \mathbf{v}^{\mathrm{T}}\mathbf{g}(\mathbf{h}_i)$ in the observation model. To facilitate the upcoming approximate integration over $q_{-i}(\theta)$ in Appendix D, we rewrite the moments $m_{\tilde{y}_i}$, $V_{\tilde{y}_i}$, and $\boldsymbol{\Sigma}_{\mathbf{v},\tilde{y}_i}$ in equation (43) using the latent function value $f_i = \mathbf{v}^{\mathrm{T}}\mathbf{g}(\mathbf{h}_i)$ instead of the noisy observation $\tilde{y}_i$. Because $\hat{p}_i(\tilde{y}_i, \mathbf{h}_i, \mathbf{v}|\theta) = \hat{p}_i(\tilde{y}_i|f_i, \theta)\hat{p}_i(\mathbf{h}_i, \mathbf{v}|\theta)$, where $\hat{p}_i(\tilde{y}_i|f_i, \theta) \propto N(\tilde{y}_i|f_i, \exp(\theta))^\eta \propto N(\tilde{y}_i|f_i, \exp(\theta)/\eta)$ and $\hat{p}_i(\mathbf{h}_i, \mathbf{v}|\theta) \propto q_{-i}(\mathbf{h}_i)q_{-i}(\mathbf{v})$, we can write the required moments as

$$\begin{aligned}
m_{\tilde{y}_i} &= \mathrm{E}\left(\mathrm{E}(\tilde{y}_i|f_i, \theta)|\theta\right) = \mathrm{E}(f_i|\theta) = m_{f_i} \\
V_{\tilde{y}_i} &= \mathrm{Var}\left(\mathrm{E}(\tilde{y}_i|f_i, \theta)|\theta\right) + \mathrm{E}\left(\mathrm{Var}(\tilde{y}_i|f_i, \theta)|\theta\right) = \mathrm{Var}(f_i|\theta) + \mathrm{E}\left(\eta^{-1}\exp(\theta)|\theta\right) \\
&= V_{f_i} + \eta^{-1}\exp(\theta) \\
\boldsymbol{\Sigma}_{\mathbf{v},\tilde{y}_i} &= \mathrm{E}\left(\mathrm{E}(\mathbf{v}, \tilde{y}_i|f_i)\right) - \mathrm{E}(\mathbf{v})\,\mathrm{E}(\mathrm{E}(\tilde{y}_i|f_i)) = \mathrm{E}\left(\mathrm{E}(\mathbf{v}|f_i)\,\mathrm{E}(\tilde{y}_i|f_i)\right) - \mathrm{E}(\mathbf{v})\,\mathrm{E}(f_i) \\
&= \mathrm{E}\left(\mathrm{E}(\mathbf{v}f_i|f_i)\right) - \mathrm{E}(\mathbf{v})\,\mathrm{E}(f_i) = \mathrm{Cov}(\mathbf{v}, f_i) = \boldsymbol{\Sigma}_{\mathbf{v},f_i},
\end{aligned} \tag{44}$$

where integrals over f_i are taken with respect to $\hat{p}_i(\mathbf{h}_i, \mathbf{v}|\theta)$ using substitution $f_i = \mathbf{v}^{\mathrm{T}}\mathbf{g}(\mathbf{h}_i)$, and on the last two lines we have omitted the conditioning on θ for clarity. Using (43) and (44), we form the approximation to the marginal tilted distribution of $\mathbf{v}$ as $\hat{p}_i(\mathbf{v}|\theta) \approx \mathcal{N}(\hat{\boldsymbol{\mu}}_i(\theta), \hat{\boldsymbol{\Sigma}}_i(\theta))$ with the mean and covariance given by

$$\begin{aligned}
\hat{\boldsymbol{\mu}}_i(\theta) &= \boldsymbol{\mu}_{-i} + \boldsymbol{\Sigma}_{\mathbf{v},f_i}V_{\tilde{y}_i}^{-1}(y_i - m_{f_i}) \\
\hat{\boldsymbol{\Sigma}}_i(\theta) &= \boldsymbol{\Sigma}_{-i} - \boldsymbol{\Sigma}_{\mathbf{v},f_i}V_{\tilde{y}_i}^{-1}\boldsymbol{\Sigma}_{\mathbf{v},f_i}^{\mathrm{T}},
\end{aligned} \tag{45}$$

where $V_{\tilde{y}_i} = V_{f_i} + \eta^{-1}\exp(\theta)$. Because $\hat{p}_i(\mathbf{h}_i, \mathbf{v}|\theta) \propto q_{-i}(\mathbf{h}_i)q_{-i}(\mathbf{v})$ factorizes between $h_{i,1}, ..., h_{i,K}$ and $\mathbf{v}$ according to (38), the central moments of $f_i = \mathbf{v}^{\mathrm{T}}\mathbf{g}(\mathbf{h}_i)$ required in (45) can be computed efficiently as

$$\begin{aligned}
m_{f_i} &= \mathrm{E}(f_i) = \boldsymbol{\mu}_{-i}^{\mathrm{T}}\mathbf{m}_{\mathbf{g}_i} \\
V_{f_i} &= \mathrm{Var}(f_i) = \mathbf{m}_{\mathbf{g}_i}^{\mathrm{T}}\boldsymbol{\Sigma}_{-i}\mathbf{m}_{\mathbf{g}_i} + \mathbf{V}_{g_i}^{\mathrm{T}}(\mathrm{diag}(\boldsymbol{\Sigma}_{-i}) + \boldsymbol{\mu}_{-i} \circ \boldsymbol{\mu}_{-i}) \\
\boldsymbol{\Sigma}_{\mathbf{v},f_i} &= \mathrm{Cov}(\mathbf{v}, f_i) = \boldsymbol{\Sigma}_{-i}\mathbf{m}_{\mathbf{g}_i},
\end{aligned} \tag{46}$$

where $\circ$ denotes the element-wise matrix product, and the $(K+1) \times 1$ vectors $\mathbf{m}_{\mathbf{g}_i} = \mathrm{E}(\mathbf{g}(\mathbf{h}_i))$ and $\mathbf{V}_{\mathbf{g}_i} = \mathrm{Var}(\mathbf{g}(\mathbf{h}_i))$ are formed by computing the means and variances from each component of $\mathbf{g}_i = \mathbf{g}(\mathbf{h}_i) = [g(h_{i,1}), ..., g(h_{i,K}), 1]^{\mathrm{T}}$ with respect to $q_{-i}(\mathbf{h}_i)$ defined in (39). Note that the last elements of $\mathbf{m}_{\mathbf{g}_i}$ and $\mathbf{V}_{\mathbf{g}_i}$ are one and zero corresponding to the output bias term v_0.

With the probit activation function (2) the elements of $\mathbf{m}_{\mathbf{g}_i}$ can be computed analytically as

$$\mathrm{E}(g(h_{i,k})) = 2K^{-1/2} \left(\Phi \left(m_{-i,k} (1 + V_{-i,k})^{-1/2} \right) - 0.5 \right),$$

and for computing the variance vector $\mathbf{V}_{\mathbf{g}_i}$, the following integral has to be evaluated numerically for all $k = 1, ..., K$:

$$\mathrm{Var}(g(h_{i,k})) = 2(K\pi)^{-1} \int_0^{\sin^{-1}(\rho)} \exp \left(- \frac{m_{-i,k}^2}{(1 + V_{-i,k})(1 + \sin(x))} \right) dx,$$

where $\rho = V_{-i,k}(1 + V_{-i,k})^{-1}$. Other activation functions could be incorporated by using one-dimensional numerical quadratures. Note that with the full posterior couplings (33), K-dimensional numerical integrations would be required to approximate m_{f_i}, V_{f_i}, and $\mathbf{\Sigma}_{\mathbf{v},f_i}$.

Appendix C. Tilted Moments for the Hidden Unit Activations

To determine the parameters of the site approximations $\{\tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k|\tilde{\tau}_{i,k}, \tilde{\nu}_{i,k})\}_{k=1}^K$, we need to form suitable approximations for the marginal means and covariances of $\{h_{i,k} = \mathbf{w}_k^{\mathrm{T}} \mathbf{x}_i\}_{k=1}^K$ resulting from the transformed tilted distribution (42). In this appendix we approximate these tilted moments with known θ and extend the approach for unknown noise level in Appendix D. The marginal conditional tilted distribution of $h_{i,k}$ is given by

$$\hat{p}_i(h_{i,k}|\theta) \propto \iint p(y_i|\mathbf{v}^{\mathrm{T}} \mathbf{g}(\mathbf{h}_i), \theta)^{\eta} q_{-i}(\mathbf{v}|\boldsymbol{\mu}_{-i}, \mathbf{\Sigma}_{-i}) d\mathbf{v} \prod_{k'=1}^{K} q_{-i}(h_{i,k'}|m_{-i,k'}, V_{-i,k'}) d\mathbf{h}_{i,-k}, \tag{47}$$

where $\mathbf{h}_{i,-k}$ contains all other hidden unit activations except $h_{i,k}$. The challenge in approximating the mean and variance of $\hat{p}_i(h_{i,k}|\theta)$ is that this marginal density can have multiple distinct modes, one related to the high-density areas of the cavity distribution $q_{-i}(\mathbf{h}_i)$ and another one related to the likelihood $p(y_i|\mathbf{v}^{\mathrm{T}} \mathbf{g}(\mathbf{h}_i), \theta)$, that is, to the values of $h_{i,k}$ that give better fit for the left-out observation y_i. In our numerical experiments, the simple approach from Appendix B that is based on a joint Gaussian approximation to $[\mathbf{h}_i^{\mathrm{T}}, \mathbf{v}^{\mathrm{T}}, f_i]$ was found to underestimate the marginal probability mass of the latter mode related to y_i especially in cases where the modes were clearly separated from each other. This problem was found to be mitigated by decreasing η, which probably stems from leaving a fraction of the old site approximation $\tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k|\tilde{\tau}_{i,k}, \tilde{\nu}_{i,k})$ from the previous iteration in the approximation that in turn shifts the cavity towards the observation y_i. With some difficult data sets, η-values as small as 0.5 were found necessary for obtaining a good data fit but usually this also required more iterations for achieving convergence compared to larger values of η.

To form robust approximations to the marginal tilted distributions $\hat{p}_i(h_{i,k}|\theta)$ also in the presence of multiple modes, we propose an alternative approximate method that enables

numerical integration over the values of $h_{i,k}$ using one-dimensional quadratures. More precisely, we aim to form a computationally cheap approximation to the integration over $\mathbf{v}$ and $\mathbf{h}_{i,-k}$ in (47) and use it to explore numerically the effect of $h_{i,k}$ on the marginal density $\hat{p}_i(h_{i,k}|\theta)$. The key difference from the Gaussian approximation of Appendix B is that more complex dependencies between $h_{i,k}$ and $f_i = \mathbf{v}^{\mathrm{T}}\mathbf{g}(\mathbf{h}_i)$ can be taken into account by numerically inspecting an approximation to $\hat{p}_i(h_{i,k}|\theta)$ at different values of $h_{i,k}$ in contrast to relaying only on linear dependencies encoded by $\mathrm{Cov}(h_{i,k}, f_i)$ (or equivalently $\mathrm{Cov}(h_{i,k}, \tilde{y}_i)$) in approximation (43).

To approximate the marginalization over $\mathbf{h}_{i,-k}$ and $\mathbf{v}$ in equation (47), we utilize the fact that the likelihood term $p(y_i|\mathbf{v}^{\mathrm{T}}\mathbf{g}(\mathbf{h}_i), \theta)$ depends on $\mathbf{v}$ and $\mathbf{h}_i$ only through the transformed scalar function value $f_i = \mathbf{v}^{\mathrm{T}}\mathbf{g}(\mathbf{h}_i)$. We first approximately transform the integration variables from $\{\mathbf{h}_{-i}, \mathbf{v}\}$ to the univariate latent function value $f_i = \mathbf{v}^{\mathrm{T}}\mathbf{g}(\mathbf{h}_i) = \mathbf{v}_{-k}^{\mathrm{T}}\mathbf{g}(\mathbf{h}_{i,-k}) + v_k g(h_{i,k})$ that depends on $h_{i,k}$, and subsequently integrate analytically over f_i. For the likelihood term in (47), we plug in the transformed variable $f_i = \mathbf{v}^{\mathrm{T}}\mathbf{g}(\mathbf{h}_i)$, but for the cavity distributions $q_{-i}(\mathbf{v})\prod_{k\neq k'} q_{-i}(h_{i,k'})$ we need to make a transformation to obtain $q_{-i}(f_i|h_{i,k})$, that is, the cavity distribution of f_i conditioned on $h_{i,k}$. Because, $q_{-i}(f_i|h_{i,k})$ cannot be computed analytically, we utilize the analytical moments from (46) to approximate it with a univariate Gaussian as

$$q_{-i}(f_i|h_{i,k}) \approx \mathcal{N}\left(f_i|m_{f_i}(h_{i,k}), V_{f_i}(h_{i,k})\right), \tag{48}$$

where $m_{f_i}(h_{i,k})$ and $V_{f_i}(h_{i,k})$ are the mean and variance of f_i computed with respect to $q_{-i}(\mathbf{v}, \mathbf{h}_{i,-k}) = q_{-i}(\mathbf{v}|\boldsymbol{\mu}_{-i}, \boldsymbol{\Sigma}_{-i})\prod_{k\neq k'} q_{-i}(h_{i,k'}|m_{-i,k}, V_{-i,k})$ with fixed $h_{i,k}$. The required conditional moments $m_{f_i}(h_{i,k})$ and $V_{f_i}(h_{i,k})$ can be computed efficiently using equation (46) by modifying the k:th element of $\mathbf{m}_{\mathbf{g}_i} = \mathrm{E}(\mathbf{g}(\mathbf{h}_i))$ and $\mathbf{V}_{\mathbf{g}_i} = \mathrm{Var}(\mathbf{g}(\mathbf{h}_i))$ corresponding to the known values of $h_{i,k}$, that is, setting $[\mathbf{m}_{\mathbf{g}_i}]_k = g(h_{i,k})$ and $[\mathbf{V}_{\mathbf{g}_i}]_k = 0$. Using equation (48), we can write the following approximation for the marginal tilted distribution of $h_{i,k}$:

$$\begin{aligned}
\hat{p}_i(h_{i,k}|\theta) &\propto \int \mathcal{N}(y_i|\mathbf{v}_{-k}^{\mathrm{T}}\mathbf{g}(\mathbf{h}_{i,-k}) + v_k g(h_{i,k}), \exp(\theta))^{\eta} q_{-i}(\mathbf{v}) \prod_{k'=1}^{K} q_{-i}(h_{i,k'}) d\mathbf{v} d\mathbf{h}_{i,-k} \\
&= \int \mathcal{N}\left(y_i|f_i, \exp(\theta)\right)^{\eta} q_{-i}\left(f_i|h_{i,k}\right) q_{-i}(h_{i,k}) df_i \\
&\approx Z(\theta)\mathcal{N}\left(y_i|m_{f_i}(h_{i,k}), V_{f_i}(h_{i,k}) + \eta^{-1}\exp(\theta)\right) q_{-i}(h_{i,k}) \\
&\approx \hat{Z}_{i,k}(\theta)\hat{q}_i\big(h_{i,k}|\hat{m}_{i,k}(\theta), \hat{V}_{i,k}(\theta)\big),
\end{aligned} \tag{49}$$

where all output weights excluding v_k are denoted by $\mathbf{v}_{-k}$, $\hat{Z}_{i,k}(\theta)$ is a normalizing constant, and $\hat{q}_i(h_{i,k}|\hat{m}_{i,k}(\theta), \hat{V}_{i,k}(\theta)) = \mathcal{N}(h_{i,k}|\hat{m}_{i,k}(\theta), \hat{V}_{i,k}(\theta))$ is the final Gaussian approximation to $\hat{p}_i(h_{i,k}|\theta)$. In the last step we have substituted approximation (48) and carried out the integration over f_i analytically to give $Z(\theta) = (2\pi\exp(\theta))^{(1-\eta)/2}\eta^{-1/2}$. Approximation (49) enables numerical inspection for possible multimodality of $\hat{p}_i(h_{i,k}|\theta)$, and it can be used for approximating the conditional tilted means $\hat{m}_{i,k}(\theta)$ and variances $\hat{V}_{i,k}(\theta)$ efficiently with one-dimensional numerical quadratures.

In our implementation, for each hidden unit $k = 1, ..., K$, we first computed $m_{f_i}(h_{i,k})$ and $V_{f_i}(h_{i,k})$ using (46) in all quadrature points that were selected to cover all the relevant cavity density $q_{-i}(h_{i,k})$. In this step we reused the means $\mathbf{m}_{\mathbf{g}_i} = \mathrm{E}(\mathbf{g}(\mathbf{h}_i))$ and variances

$\mathbf{V}_{\mathbf{g}_i} = \mathrm{Var}(\mathbf{g}(\mathbf{h}_i))$ that were computed previously to determine the moments of $\hat{p}_i(\mathbf{v}|\theta)$ with (45). Note that only terms dependent on $[\mathbf{m}_{\mathbf{g}_i}]_k$ have to be re-evaluated for each value of $h_{i,k}$, because $[\mathbf{m}_{\mathbf{g}_i}]_k = g(h_{i,k})$ and $[\mathbf{V}_{\mathbf{g}_i}]_k = 0$. Then we computed the tilted mean $\hat{m}_{i,k}(\theta)$ and variance $\hat{V}_{i,k}(\theta)$ using the same values of the integrand in the third line of (49) for each $k = 1, ..., K$.

The approximation (48) can be justified using the central limit theorem according to which the distribution of the sum in $f_i = \sum_{k'=1}^{K} v_{k'} g(h_{i,k'}) + v_0$ given $h_{i,k}$ approaches a normal distribution as K increases. Therefore, the approximate transformation used in (48) and (49) becomes more accurate as the number of hidden units increase. However, the proposed approximation can be very useful also with smaller values of K, because the predictions are made using exactly the same scheme (see Appendix H). During training, the input weight approximations are adjusted so that the Gaussian approximations of $q(f_i|h_{i,k})$ in (48) encompass the high-density regions of the likelihood terms $p(y_i|f_i, \theta)$ in equation (49). Therefore, the approximation should be able to produce high predictive densities also for test observations.

A similar approach has been used by Ribeiro and Opper (2011) to form factorized EP approximations for the input weights with linear single-layer models. They used the central limit argument to form second-order Taylor approximations for the marginal tilted distributions resulting from univariate Gaussian approximations for the input weights. We utilize the same idea to approximate the tilted moments of the transformed variables $h_{i,k} = \mathbf{w}_k^{\mathrm{T}} \mathbf{x}_i$ using numerical quadratures and an input weight approximation that can be factorized between the different hidden units.

Appendix D. Tilted Moments with Unknown Noise Level

In this appendix we propose ways to approximate the moments of $\mathbf{v}$, $\{h_{i,k} = \mathbf{w}_k^{\mathrm{T}} \mathbf{x}_i\}_{k=1}^{K}$, and θ resulting from the transformed tilted distribution (42) by extending the derivations of Appendices B and C for approximate integration over θ for the setting where the noise level is assumed unknown and estimated using the proposed EP framework. The mean $\hat{\mu}_{\theta,i}$ and variance $\hat{\sigma}^2_{\theta,i}$ of the marginal tilted distribution

$$\hat{p}_i(\theta) \propto \int p(y_i|\mathbf{v}^{\mathrm{T}} \mathbf{g}(\mathbf{h}_i), \theta)^{\eta} q_{-i}(\mathbf{v}|\boldsymbol{\mu}_{-i}, \boldsymbol{\Sigma}_{-i}) d\mathbf{v} \prod_{k=1}^{K} q_{-i}(h_{i,k}|m_{-i,k}, V_{-i,k}) d\mathbf{h}_i q_{-i}(\theta|\mu_{\theta,-i}, \sigma^2_{\theta,-i}),$$

can be approximated with a similar approach to the one that was used to determine the moments of $\hat{p}_i(h_{i,k}|\theta)$ in Appendix C. We first transform the integration over $\mathbf{v}$ and $\mathbf{h}_i$ to integration over $f_i = \mathbf{v}^{\mathrm{T}} \mathbf{g}(\mathbf{h}_i)$ by forming a Gaussian approximation to the cavity distribution of f_i as

$$q_{-i}(f_i|\theta) = q_{-i}(f_i) \approx \mathcal{N}(f_i|m_{f_i}, V_{f_i}),$$

where the mean m_{f_i} and variance V_{f_i} are computed using (46). Note that $q_{-i}(f_i|\theta)$ is independent of θ, because of the factorized approximation. Then, assuming a Gaussian observation model, we can integrate analytically over f_i to obtain a numerical approximation

for the tilted distribution of θ:

$$\begin{aligned}\hat{p}_i(\theta) &\propto \int \mathcal{N}(y_i|\mathbf{v}^{\mathrm{T}}\mathbf{g}(\mathbf{h}_i), \exp(\theta))^{\eta} q_{-i}(\mathbf{v}) q_{-i}(\mathbf{h}_i) d\mathbf{v} d\mathbf{h}_i q_{-i}(\theta) \\ &= \int \mathcal{N}\left(y_i|f_i, \exp(\theta)\right)^{\eta} q_{-i}(f_i) q_{-i}(\theta) df_i \\ &\approx Z(\theta)\mathcal{N}\left(y_i|m_{f_i}, V_{f_i} + \eta^{-1}\exp(\theta)\right) q_{-i}(\theta) \approx \hat{Z}_i \hat{q}_i(\theta|\hat{\mu}_{\theta,i}, \hat{\sigma}^2_{\theta,i}),\end{aligned} \tag{50}$$

where $Z(\theta) = (2\pi\exp(\theta))^{(1-\eta)/2}\eta^{-1/2}$, $\hat{Z}_i$ is an approximation to the normalization term of the tilted distribution (22), and $\hat{q}_i(\theta|\hat{\mu}_{\theta,i}, \hat{\sigma}^2_{\theta,i}) = \mathcal{N}(\theta|\hat{\mu}_{\theta,i}, \hat{\sigma}^2_{\theta,i})$ is our final Gaussian approximation to the marginal tilted distribution $\hat{p}_i(\theta)$. The approximate tilted mean $\hat{\mu}_{\theta,i}$, variance $\hat{\sigma}^2_{\theta,i}$, and normalization term $\hat{Z}_i$ can be computed by integrating numerically over the integrand on the third line of (50) using a quadrature. From (50) we also see that the normalization term $\hat{Z}_i$ can by approximated with $\hat{Z}_i(\theta) = Z(\theta)\mathcal{N}\left(y_i|m_{f_i}, V_{f_i} + \eta^{-1}\exp(\theta)\right)$, if θ is known or fixed.

To approximate the marginal mean and covariance of $\mathbf{v}$ with unknown θ, we can utilize the conditional tilted moments from equation (45) by taking expectations with respect to

$$\tilde{q}_i(\theta) = \hat{Z}_i^{-1} Z(\theta)\mathcal{N}\left(y_i|m_{f_i}, V_{f_i} + \eta^{-1}\exp(\theta)\right) q_{-i}(\theta),$$

because the conditional moments are determined using an approximation to $\hat{p}_i(\mathbf{h}_i, \mathbf{v}|\theta)$ and from (50) we see that $\hat{p}_i(\mathbf{h}_i, \mathbf{v}, \theta) \approx \hat{Z}_i^{-1}\hat{p}_i(\mathbf{h}_i, \mathbf{v}|\theta)\tilde{q}_i(\theta)$. In case of the simple joint Gaussian approximation for $\mathbf{v}$ we can write

$$\begin{aligned}\hat{\boldsymbol{\mu}}_i = \mathrm{E}_{\hat{p}_i(\mathbf{v})}(\mathbf{v}) &= \mathrm{E}_{\hat{p}_i(\theta)}\left(\mathrm{E}_{\hat{p}_i(\mathbf{v}|\theta)}(\mathbf{v}|\theta)\right) \approx \mathrm{E}_{\tilde{q}_i(\theta)}(\hat{\boldsymbol{\mu}}_i(\theta)) \\ &= \boldsymbol{\mu}_{-i} + \boldsymbol{\Sigma}_{\mathbf{v},f_i}\mathrm{E}_{\tilde{q}_i(\theta)}\left(V_{y_i}^{-1}\right)(y_i - m_{f_i}),\end{aligned} \tag{51}$$

where the conditional mean of $\mathbf{v}$ with respect to $\hat{p}_i(\mathbf{v}|\theta)$ is approximated using (45), and the integration over $V_{y_i}^{-1} = (V_{f_i} + \eta^{-1}\exp(\theta))^{-1}$ can be done using a one-dimensional quadrature. Similarly, for the marginal covariance of $\mathbf{v}$ we can write

$$\begin{aligned}\hat{\boldsymbol{\Sigma}}_i = \mathrm{Cov}_{\hat{p}_i(\mathbf{v})}(\mathbf{v}) &= \mathrm{E}_{\hat{p}_i(\theta)}\left(\mathrm{Cov}_{\hat{p}_i(\mathbf{v}|\theta)}(\mathbf{v}|\theta)\right) + \mathrm{Cov}_{\hat{p}_i(\theta)}\left(\mathrm{E}_{\hat{p}_i(\mathbf{v}|\theta)}(\mathbf{v}|\theta)\right) \\ &\approx \mathrm{E}_{\tilde{q}_i(\theta)}\left(\hat{\boldsymbol{\Sigma}}_i(\theta)\right) + \mathrm{E}_{\tilde{q}_i(\theta)}\left(\left(\hat{\boldsymbol{\mu}}_i(\theta) - \hat{\boldsymbol{\mu}}_i\right)\left(\hat{\boldsymbol{\mu}}_i(\theta) - \hat{\boldsymbol{\mu}}_i\right)^{\mathrm{T}}\right) \\ &= \boldsymbol{\Sigma}_{-i} - \boldsymbol{\Sigma}_{\mathbf{v},f_i}\left(\mathrm{E}_{\tilde{q}_i(\theta)}\left(V_{y_i}^{-1}\right) - (y_i - m_{f_i})^2\,\mathrm{Var}_{\tilde{q}_i(\theta)}\left(V_{y_i}^{-1}\right)\right)\boldsymbol{\Sigma}_{\mathbf{v},f_i}^{\mathrm{T}},\end{aligned} \tag{52}$$

where the conditional covariance of $\mathbf{v}$ with respect to $\hat{p}_i(\mathbf{v}|\theta)$ is approximated using (45) and $\mathrm{Var}_{\tilde{q}_i(\theta)}\left(V_{y_i}^{-1}\right) = \mathrm{E}_{\tilde{q}_i(\theta)}\left((V_{y_i}^{-1} - \mathrm{E}_{\tilde{q}_i(\theta)}(V_{y_i}^{-1}))^2\right)$ can be computed with a numerical quadrature. For the output weights $\mathbf{v}$ the integration over the uncertainty of θ can be done without significant additional computational cost. The mean $\mathrm{E}_{\tilde{q}_i(\theta)}(V_{y_i}^{-1})$ and variance $\mathrm{Var}_{\tilde{q}_i(\theta)}(V_{y_i}^{-1})$ can be determined by reusing the same function evaluations that are needed in the quadrature integrations of $\hat{\mu}_{\theta,i}$, $\hat{\sigma}^2_{\theta,i}$, and $\hat{Z}_i$ according to equation (50).

Approximating the marginal tilted moments of the hidden unit activations $h_{i,k}$ with unknown θ is more demanding because determining the means and variances of $\hat{p}_i(h_{i,k})$ using the approximation (49) requires two-dimensional numerical quadratures over both $h_{i,k}$ and θ in

$$\hat{p}_i(h_{i,k}, \theta) \approx \hat{Z}_i^{-1} Z(\theta)\mathcal{N}\left(y_i|m_{f_i}(h_{i,k}), V_{f_i}(h_{i,k}) + \eta^{-1}\exp(\theta)\right) q_{-i}(\theta) q_{-i}(h_{i,k}), \tag{53}$$

for each hidden unit $k = 1, ..., K$. To reduce the computational burden, we approximate the probability density of $\hat{p}_i(h_{i,k}, \theta)$ to be relatively sharply peaked near the marginal expected value $\hat{\mu}_{\theta,i}$ determined using (50) leading to approximation

$$\begin{aligned}\hat{p}_i(h_{i,k}) &\approx \hat{Z}_i^{-1} Z(\theta) \mathcal{N}\left(y_i | m(h_{i,k}), V(h_{i,k}) + \eta^{-1} \exp(\hat{\mu}_{\theta,i})\right) q_{-i}(h_{i,k}) \\ &\approx \hat{q}_i\big(h_{i,k} | \hat{m}_{i,k}(\hat{\mu}_{\theta,i}), \hat{V}_{i,k}(\hat{\mu}_{\theta,i})\big),\end{aligned} \tag{54}$$

where $\hat{q}_i(h_{i,k}|\hat{m}_{i,k}(\hat{\mu}_{\theta,i}), \hat{V}_{i,k}(\hat{\mu}_{\theta,i})) = \mathcal{N}(h_{i,k}|\hat{m}_{i,k}(\hat{\mu}_{\theta,i}), \hat{V}_{i,k}(\hat{\mu}_{\theta,i}))$ is our final Gaussian approximation for $\hat{p}_i(h_{i,k})$. This approximation does not require any additional computational effort compared to the conditional estimate (49) and the difference in accuracy compared to the two-dimensional quadrature estimate based on (53) was found small after a few iterations provided that there are enough observations.

Appendix E. Site Parameters and Damped Updates

In this appendix we present closed form expressions for the parameters of the likelihood site approximations $\tilde{t}_{\mathbf{v},i}(\mathbf{v}|\tilde{\boldsymbol{\alpha}}_i, \tilde{\boldsymbol{\beta}}_i)$, $\{\tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k|\tilde{\tau}_{i,k}, \tilde{\nu}_{i,k})\}_{k=1}^K$, and $\tilde{t}_{\theta,i}(\theta|\tilde{\mu}_{\theta,i}, \tilde{\sigma}^2_{\theta,i})$ that are obtained by applying the moment matching condition (23) with the approximate tilted moments derived in Appendices B–D.

Using the moment matching condition $\hat{\boldsymbol{\Sigma}}_i^{-1} = \boldsymbol{\Sigma}_{-i}^{-1} + \eta \tilde{\boldsymbol{\alpha}}_i \tilde{\boldsymbol{\alpha}}_i^{\mathrm{T}}$ resulting from (23) and approximate tilted covariance $\hat{\boldsymbol{\Sigma}}_i$ from (45) or (52), we can write the following expression for the scale parameter vector $\tilde{\boldsymbol{\alpha}}_i$ of the i:th approximate site term $\tilde{t}_{\mathbf{v},i}(\mathbf{v}|\tilde{\boldsymbol{\alpha}}_i, \tilde{\boldsymbol{\beta}}_i)$ defined in (35):

$$\tilde{\boldsymbol{\alpha}}_i = \mathbf{m}_{\mathbf{g}_i} \mathrm{sign}(\hat{a}_i) |\hat{a}_i|^{1/2} \left(1 - \hat{a}_i \mathbf{m}_{\mathbf{g}_i}^{\mathrm{T}} \boldsymbol{\Sigma}_{-i} \mathbf{m}_{\mathbf{g}_i}\right)^{-1/2} \eta^{-1/2}, \tag{55}$$

where $\hat{a}_i = V_{y_i}^{-1} = (V_{f_i} + \eta^{-1} \exp(\theta))^{-1} > 0$ with known θ (see equation (45)), and $\hat{a}_i = \mathrm{E}_{\tilde{q}_i(\theta)}\left(V_{y_i}^{-1}\right) - (y_i - m_{f_i})^2 \mathrm{Var}_{\tilde{q}_i(\theta)}\left(V_{y_i}^{-1}\right)$ with unknown θ (see equation (52)). Similarly for the location parameter vector $\tilde{\boldsymbol{\beta}}_i$, equation (23) results in the moment matching condition $\hat{\boldsymbol{\Sigma}}_i^{-1} \hat{\boldsymbol{\mu}}_i = \boldsymbol{\Sigma}_{-i}^{-1} \boldsymbol{\mu}_{-i} + \eta \tilde{\boldsymbol{\beta}}_i$ that together with the approximate tilted mean $\hat{\boldsymbol{\mu}}_i$ from equation (45) or (51) gives

$$\tilde{\boldsymbol{\beta}}_i = \mathbf{m}_{\mathbf{g}_i} \left(1 - \hat{a}_i \mathbf{m}_{\mathbf{g}_i}^{\mathrm{T}} \boldsymbol{\Sigma}_{-i} \mathbf{m}_{\mathbf{g}_i}\right)^{-1} \left(\hat{a}_i \mathbf{m}_{\mathbf{g}_i}^{\mathrm{T}} \boldsymbol{\mu}_{-i} + \hat{b}_i (y_i - m_{f_i})\right) \eta^{-1} \tag{56}$$

where $\hat{a}_i$ is defined similarly with the previous equation, $\hat{b}_i = V_{y_i}^{-1}$ with known θ (see equation (45)), and $\hat{b}_i = \mathrm{E}_{\hat{q}_i(\theta)}\left(V_{y_i}^{-1}\right)$ when θ is unknown (see equation (51)).

By looking at equations (55) and (56) we can now extend our previous discussion about the structure of the site parameters after equation (37) in Section 3.3. The mean and covariance of the posterior approximation $q(\mathbf{v})$ defined in equation (37) can be interpreted as the posterior distribution of a linear model where the input features are replaced with the expected values of the nonlinearly transformed input layer activations $\mathbf{m}_{\mathbf{g}_i} = \mathrm{E}_{q_{-i}}(\mathbf{g}(\tilde{\mathbf{x}}_i^{\mathrm{T}} \mathbf{w}))$ and pseudo observations $\tilde{y}_i = \mathbf{m}_{\mathbf{g}_i}^{\mathrm{T}} \boldsymbol{\mu}_{-i} + \hat{a}_i^{-1} \hat{b}_i (y_i - m_{f_i})$ are made according to an observation model $\mathcal{N}(\tilde{y}_i | \mathbf{m}_{\mathbf{g}_i}^{\mathrm{T}} \mathbf{v}, \hat{a}_i^{-1} - \mathbf{m}_{\mathbf{g}_i}^{\mathrm{T}} \boldsymbol{\Sigma}_{-i} \mathbf{m}_{\mathbf{g}_i})$.

Damping the site updates can improve the numerical robustness and convergence of the EP algorithm, but applying damping on the site precision structure $\tilde{\mathbf{T}}_{i,\mathbf{vv}} = \tilde{\boldsymbol{\alpha}}_i \tilde{\boldsymbol{\alpha}}_i^{\mathrm{T}}$

resulting from equations (37) and (55), that is, $\tilde{\mathbf{T}}^{\text{new}}_{i,\mathbf{vv}} = (1-\delta)\tilde{\boldsymbol{\alpha}}^{\text{old}}_i(\tilde{\boldsymbol{\alpha}}^{\text{old}}_i)^{\text{T}} + \delta\tilde{\boldsymbol{\alpha}}_i\tilde{\boldsymbol{\alpha}}_i^{\text{T}}$, would break the outer product form of the likelihood site approximations (35) and produce a computationally more demanding rank-K site precision after K iterations. In case the input weight approximations $q(\mathbf{w}_k)$ were kept fixed while updating the output weights $\mathbf{v}$, the expected activations $\mathbf{m}(\mathbf{g}_i)$ would remain constant and one could consider damping only the scalar terms on the right hand side of equations (55) and (56).

In the more general case where also the site parameters $\tilde{\tau}_{i,k}$ and $\tilde{\nu}_{i,k}$ related to the input weights are updated simultaneously, we can approximate the new site precision structure $\tilde{\mathbf{T}}^{\text{new}}_{i,\mathbf{vv}} = \mathbf{A}_i\mathbf{A}_i^{\text{T}}$, where $\mathbf{A}_i = [(1-\delta)^{1/2}\tilde{\boldsymbol{\alpha}}^{\text{old}}_i, \delta^{1/2}\tilde{\boldsymbol{\alpha}}_i]$ and $\tilde{\boldsymbol{\alpha}}_i$ is obtained from (55), with its largest eigenvector at each site update step. This requires solving the eigenvector $\mathbf{v}_i$ corresponding to the largest eigenvalue λ_i of the 2×2 matrix $\mathbf{A}_i^{\text{T}}\mathbf{A}_i \approx \mathbf{v}_i\lambda_i\mathbf{v}_i^{\text{T}}$ after which the new damped site parameter vector can be approximated as

$$\tilde{\boldsymbol{\alpha}}^{\text{new}}_i = \mathbf{A}_i\mathbf{v}_i. \tag{57}$$

Damping the site location vector $\tilde{\boldsymbol{\beta}}_i$ is straightforward because update $\tilde{\boldsymbol{\beta}}^{\text{new}}_i = (1-\delta)\tilde{\boldsymbol{\beta}}^{\text{old}}_i + \delta\tilde{\boldsymbol{\beta}}_i = \mathbf{b}_i$, where $\tilde{\boldsymbol{\beta}}_i$ is obtained from (56), will preserve the structure of the site approximation (35). However, approximation $\tilde{\boldsymbol{\alpha}}^{\text{new}}_i = \mathbf{A}_i\mathbf{v}_i$ changes the moment consistency conditions used in deriving (56) which is why $\tilde{\boldsymbol{\beta}}^{\text{new}}_i$ has to be modified so that combining it with $\tilde{\boldsymbol{\alpha}}^{\text{new}}_i$ according to the moment matching rule (23) results in the same mean vector $\boldsymbol{\mu}_{\mathbf{v}}$ as the rank-2 site $\mathbf{A}_i\mathbf{A}_i^{\text{T}}$ combined with $\mathbf{b}_i$:

$$\begin{aligned}\boldsymbol{\mu}_{\mathbf{v}} &= \left(\boldsymbol{\Sigma}_{-i}^{-1} + \eta\tilde{\boldsymbol{\alpha}}^{\text{new}}_i(\tilde{\boldsymbol{\alpha}}^{\text{new}}_i)^{\text{T}}\right)^{-1}\left(\boldsymbol{\Sigma}_{-i}^{-1}\boldsymbol{\mu}_{-i} + \eta\tilde{\boldsymbol{\beta}}^{\text{new}}_i\right)\\ &= \left(\boldsymbol{\Sigma}_{-i}^{-1} + \eta\mathbf{A}_i\mathbf{A}_i^{\text{T}}\right)^{-1}\left(\boldsymbol{\Sigma}_{-i}^{-1}\boldsymbol{\mu}_{-i} + \eta\mathbf{b}_i\right).\end{aligned} \tag{58}$$

In other words, we approximate the posterior covariance $\boldsymbol{\Sigma}_{\mathbf{v}} = (\boldsymbol{\Sigma}_{-i}^{-1} + \eta\mathbf{A}_i\mathbf{A}_i^{\text{T}})^{-1}$ resulting from the rank-two damped update with the rank-one update $\boldsymbol{\Sigma}_{\mathbf{v}} \approx (\boldsymbol{\Sigma}_{-i}^{-1} + \eta\tilde{\boldsymbol{\alpha}}^{\text{new}}_i(\tilde{\boldsymbol{\alpha}}^{\text{new}}_i)^{\text{T}})^{-1}$ but choose $\tilde{\boldsymbol{\beta}}^{\text{new}}_i$ so that the mean $\boldsymbol{\mu}_{\mathbf{v}}$ will be exact. Plugging in $\tilde{\boldsymbol{\alpha}}^{\text{new}}_i = \mathbf{A}_i\mathbf{v}_i$ and solving for $\tilde{\boldsymbol{\beta}}^{\text{new}}_i$ gives the following update rule

$$\tilde{\boldsymbol{\beta}}^{\text{new}}_i = \mathbf{b}_i + \eta^{-1}\mathbf{A}_i(\mathbf{v}_i\mathbf{v}_i^{\text{T}} - \mathbf{I})(\mathbf{A}_i^{\text{T}}\boldsymbol{\Sigma}_{-i}\mathbf{A}_i + \eta^{-1}\mathbf{I})^{-1}\mathbf{A}_i^{\text{T}}(\boldsymbol{\mu}_{-i} + \eta\boldsymbol{\Sigma}_{-i}\mathbf{b}_i), \tag{59}$$

where $\mathbf{b}_i = (1-\delta)\tilde{\boldsymbol{\beta}}^{\text{old}}_i + \delta\tilde{\boldsymbol{\beta}}_i$ with $\tilde{\boldsymbol{\beta}}_i$ given by (56).

Because of the factorized posterior approximation (19), the likelihood site approximation terms associated with the input weights decouple over the different hidden units as $\prod_{k=1}^{K}\tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}_k|\tilde{\tau}_{i,k},\tilde{\nu}_{i,k})$ and consequently the moment matching condition (23) results in simple scalar site parameter updates. Using the moment matching condition with the cavity definitions (39) and the tilted moments approximated with either (49) or (54) gives the following site updates

$$\begin{aligned}\tilde{\tau}^{\text{new}}_{i,k} &= (1-\delta)\tilde{\tau}_{i,k} + \delta\eta^{-1}(\hat{V}_{i,k}^{-1} - V_{-i,k}^{-1}) = \tilde{\tau}_{i,k} + \delta\eta^{-1}(\hat{V}_{i,k}^{-1} - V_{i,k}^{-1}) \\ \tilde{\nu}^{\text{new}}_{i,k} &= (1-\delta)\tilde{\nu}_{i,k} + \delta\eta^{-1}(\hat{V}_{i,k}^{-1}\hat{m}_{i,k} - V_{-i,k}^{-1}m_{-i,k}) = \tilde{\nu}_{i,k} + \delta\eta^{-1}(\hat{V}_{i,k}^{-1}\hat{m}_{i,k} - V_{i,k}^{-1}m_{i,k}),\end{aligned} \tag{60}$$

where $\delta \in (0,1]$ is a damping factor and the marginal tilted mean $\hat{m}_{i,k}$ and variance $\hat{V}_{i,k}$ are computed using (49) or (54) depending on whether θ is known or unknown. Equation (60)

shows that the EP iterations on the input weights $\mathbf{w}_k$ have converged when the approximate marginal means $m_{i,k}$ and variances $V_{i,k}$ of the activations $h_{i,k}$ from all hidden units are consistent with all tilted distributions.

In case θ is inferred using EP, parameter updates for the site approximations $\tilde{t}_{\theta,i}(\theta|\tilde{\mu}_{\theta,i}, \tilde{\sigma}^2_{\theta,i})$ can be derived by combining the cavity definitions (41) with the tilted moment approximations (50) according to the moment consistency conditions (23), which results in

$$\begin{aligned} \tilde{\tau}^{\text{new}}_{\theta,i} &= \tilde{\tau}_{\theta,i} + \delta\eta^{-1}\left(\hat{\sigma}^{-2}_{\theta,i} - \sigma^{-2}_{\theta}\right) \\ \tilde{\nu}^{\text{new}}_{\theta,i} &= \tilde{\nu}_{\theta,i} + \delta\eta^{-1}(\hat{\sigma}^{-2}_{\theta,i}\hat{\mu}_{\theta,i} - \sigma^{-2}_{\theta}\mu_{\theta}), \end{aligned} \tag{61}$$

where we have written the site parameters in their natural exponential forms as $\tilde{\tau}_{\theta,i} = \tilde{\sigma}^{-2}_{\theta,i}$ and $\tilde{\nu}_{\theta,i} = \tilde{\sigma}^{-2}_{\theta,i}\tilde{\mu}_{\theta,i}$.

Appendix F. EP Algorithm for the Weight Prior Terms

This appendix summarizes an EP algorithm that can be used to determine the site approximations of the weight prior terms (13) and (14) as discussed in Section 3.2.2. The following algorithm is written for the input weight terms

$$p(w_j|\phi_{l_j}) \approx \tilde{Z}_{w,j}\tilde{t}_{w,j}(w_j)\tilde{t}_{\phi,j}(\phi_{l_j}) \propto \mathcal{N}(w_j|\tilde{\mu}_{w,j}, \tilde{\sigma}^2_{w,j})\mathcal{N}(\phi_{l_j}|\tilde{\mu}_{\phi,j}, \tilde{\sigma}^2_{\phi,j})$$

that depend also on the scale parameters $\{\phi_l\}_{l=1}^L$. We denote the parameters of the site approximations in their natural exponential forms as $\tilde{\tau}_{w,j} = \tilde{\sigma}^{-2}_{w,j}$, $\tilde{\nu}_{w,j} = \tilde{\sigma}^{-2}_{w,j}\tilde{\mu}_{w,j}$, $\tilde{\tau}_{\phi,j} = \tilde{\sigma}^{-2}_{\phi,j}$ and $\tilde{\nu}_{\phi,j} = \tilde{\sigma}^{-2}_{\phi,j}\tilde{\mu}_{\phi,j}$. The algorithm can be applied for updating the output weight terms for $k = 1, ..., K$,

$$p(v_k) \approx \tilde{Z}_{v,k}\tilde{t}_{v,k}(v_k) \propto \mathcal{N}(v_k|\tilde{\mu}_{v,k}, \tilde{\sigma}^2_{v,k}),$$

by leaving out the computations related to parameters ϕ_l, and replacing the natural parameters $\tilde{\nu}_{w,j}$ and $\tilde{\tau}_{w,j}$ with $\tilde{\tau}_{v,k} = \tilde{\sigma}^{-2}_{v,k}\tilde{\mu}_{v,k}$ and $\tilde{\nu}_{v,k} = \tilde{\sigma}^{-2}_{w,k}$, and the posterior approximation $q(\mathbf{w}_k) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{w}_k}, \boldsymbol{\Sigma}_{\mathbf{w}_k})$ with $q(\mathbf{v}) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{v}}, \boldsymbol{\Sigma}_{\mathbf{v}})$. One iteration of the algorithm consist of the following update steps for all site approximations $j = K(k-1)+1, ..., K(k-1)+d$ related to all hidden units $k = 1, ..., K$:

1. Compute the mean and covariance of the cavity distribution $q_{-j}(w_j) = \mathcal{N}(m_{w,-j}, V_{w,-j})$:

$$\begin{aligned} V_{w,-j} &= (V^{-1}_{w,j} - \eta\tilde{\tau}_{w,j})^{-1} \\ m_{w,-j} &= V_{w,-j}(V^{-1}_{w,j}m_{w,j} - \eta\tilde{\nu}_{w,j}), \end{aligned} \tag{62}$$

   where the approximate mean and variance of w_j are given by $m_{w,j} = [\boldsymbol{\mu}_{\mathbf{w}_k}]_i = \boldsymbol{\mu}_{\mathbf{w}_k}\mathbf{e}_i$ and $V_{w,j} = [\boldsymbol{\Sigma}_{\mathbf{w}_k}]_{i,i} = \mathbf{e}_i^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathbf{w}_k}\mathbf{e}_i$ with $\boldsymbol{\mu}_{\mathbf{w}_k}$ and $\boldsymbol{\Sigma}_{\mathbf{w}_k}$ defined by (36), $i = j - K(k-1)$, and $\mathbf{e}_i$ the i:th unit vector. Compute also the mean and covariance of the cavity distribution $q_{-j}(\phi_{l_j}) = \mathcal{N}(m_{\phi,-j}, V_{\phi,-j})$:

$$\begin{aligned} V_{\phi,-j} &= (\sigma^{-2}_{\phi_{l_j},} - \eta\tilde{\tau}_{\phi,j})^{-1} \\ m_{\phi,-j} &= V_{\phi,-j}(\sigma^{-2}_{\phi_{l_j}}\mu_{\phi_{l_j}} - \eta\tilde{\nu}_{\phi,j}), \end{aligned} \tag{63}$$

   where $\mu_{\phi_{l_j}}$ and $\sigma^2_{\phi_{l_j}}$ are the mean and covariance of $q(\phi_{l_j})$ given by (18).

2. Compute the marginal moments $\hat{Z}_{w,j}$, $\hat{m}_{w,j} = \mathrm{E}(w_j)$, $\hat{V}_{w,j} = \mathrm{Var}(w_j)$, $\hat{m}_{\phi,j} = \mathrm{E}(\phi_{l_j})$, and $\hat{V}_{\phi,j} = \mathrm{Var}(\phi_{l_j})$ of the tilted distribution $\hat{p}_j(w_j, \phi_{l_j})$ either analytically or using a numerical quadrature depending on the functional form of the prior term $p(w_j|\phi_{l_j})$:

$$\begin{aligned}\hat{p}_j(w_j, \phi_{l_j}) &= \hat{Z}_{w,j}^{-1} q_{-j}(w_j) q_{-j}(\phi_{l_j}) p(w_j|\phi_{l_j})^\eta \\ &\approx \mathcal{N}(w_j|\hat{m}_{w,j}, \hat{V}_{w,j}) \mathcal{N}(\phi_{l_j}|\hat{m}_{\phi,j}, \hat{V}_{\phi,j}),\end{aligned} \tag{64}$$

where $\hat{Z}_{w,j} = \int q_{-j}(w_j) q_{-j}(\phi_{l_j}) p(w_j|\phi_{l_j})^\eta dw_j d\phi_{l_j}$.

3. Update the site parameters related to $\tilde{t}_{w,j}(w_j)$ as $\tilde{\tau}_{w,j}^{\text{new}} = \tilde{\tau}_{w,j} + \Delta\tilde{\tau}_{w,j}$ and $\tilde{\nu}_{w,j}^{\text{new}} = \tilde{\nu}_{w,j} + \Delta\tilde{\nu}_{w,j}$ together with the parameters related to $\tilde{t}_{\phi,j}(\phi_{l_j})$ as $\tilde{\tau}_{\phi,j}^{\text{new}} = \tilde{\tau}_{\phi,j} + \Delta\tilde{\tau}_{\phi,j}$ and $\tilde{\nu}_{\phi,j}^{\text{new}} = \tilde{\nu}_{\phi,j} + \Delta\tilde{\nu}_{\phi,j}$, where the parameter adjustments damped by $\delta \in (0, 1]$ are given by

$$\begin{aligned}\Delta\tilde{\tau}_{w,j} &= \delta\eta^{-1}(\hat{V}_{w,j}^{-1} - V_{w,j}^{-1}) \\ \Delta\tilde{\nu}_{w,j} &= \delta\eta^{-1}(\hat{V}_{w,j}^{-1}\hat{m}_{w,j} - V_{w,j}^{-1} m_{w,j}) \\ \Delta\tilde{\tau}_{\phi,j} &= \delta\eta^{-1}(\hat{V}_{\phi,j}^{-1} - \sigma_{\phi_{l_j}}^{-2}) \\ \Delta\tilde{\nu}_{\phi,j} &= \delta\eta^{-1}(\hat{V}_{\phi,j}^{-1}\hat{m}_{\phi,j} - \sigma_{\phi_{l_j}}^{-2}\mu_{\phi_{l_j}}).\end{aligned} \tag{65}$$

4. If sequential EP is used, update the posterior approximation $q(\mathbf{w}_k) = \mathcal{N}(\boldsymbol{\mu}_\mathbf{w}, \boldsymbol{\Sigma}_\mathbf{w})$ using a rank-one update:

$$\begin{aligned}\boldsymbol{\Sigma}_{\mathbf{w}_k}^{\text{new}} &= \boldsymbol{\Sigma}_{\mathbf{w}_k} - \mathbf{a}_j \Delta\tilde{\tau}_{w,j} s_j^{-1} \mathbf{a}_j^{\mathrm{T}} \\ \boldsymbol{\mu}_{\mathbf{w}_k}^{\text{new}} &= \boldsymbol{\mu}_{\mathbf{w}_k} + \mathbf{a}_j s_j^{-1}(\Delta\tilde{\nu}_{w,j} - \Delta\tilde{\tau}_{w,j} m_{w,j}),\end{aligned} \tag{66}$$

where $s_j = 1 + \Delta\tilde{\tau}_{w,j} V_{w,j}$ and $\mathbf{a}_j = \boldsymbol{\Sigma}_\mathbf{w} \mathbf{e}_i$ with $i = j - K(k-1)$. Also the determinant of $\boldsymbol{\Sigma}_{\mathbf{w}_k}$ can be updated sequentially as $\log|\boldsymbol{\Sigma}_{\mathbf{w}_k}^{\text{new}}| = \log|\boldsymbol{\Sigma}_{\mathbf{w}_k}| - \log(s_j)$, which can be used in evaluating the approximate marginal likelihood as described in Appendix I. For the scalar ϕ_{l_j}, the posterior $q(\phi_{l_j}) = \mathcal{N}(\mu_{\phi_{l_j}}, \sigma_{\phi_{l_j}}^2)$ can be updated as

$$\begin{aligned}\sigma_{\phi_{l_j}}^{2\ \text{new}} &= \left(\sigma_{\phi_{l_j}}^{-2} + \Delta\tilde{\tau}_{\phi,j}\right)^{-1} \\ \mu_{\phi_{l_j}}^{\text{new}} &= \sigma_{\phi_{l_j}}^{2\ \text{new}} \left(\sigma_{\phi_{l_j}}^{-2}\mu_{\phi_{l_j}} + \Delta\tilde{\nu}_{\phi,j}\right).\end{aligned} \tag{67}$$

Steps 1–4 are repeated until all the tilted distributions are consistent with the approximate posterior, that is, $\hat{m}_{w,j} = m_{w,j}$, $\hat{V}_{w,j} = V_{w,j}$, $\hat{m}_{\phi,j} = \mu_{\phi_{l_j}}$ and $\hat{V}_{\phi,j} = \sigma_{\phi_{l_j}}^2$. In parallel EP, step 4 is replaced with a single re-computation of $\{\boldsymbol{\mu}_{\mathbf{w}_k}\}_{k=1}^K$ and $\{\boldsymbol{\Sigma}_{\mathbf{w}_k}\}_{k=1}^K$ using, e.g., K Cholesky decompositions after each sweep over all the site approximations $j = K(k-1)+1, ..., K(k-1)+d$ for all the hidden units $k = 1, ..., K$.

Appendix G. Improving the Numerical Stability of the EP algorithm

This appendix outlines some practical procedures for conducting the updates (57), (59), (60), and (65) so that the EP algorithm 1 remains numerically stable. From (66) we see

that the approximate posterior $\mathbf{\Sigma}_{\mathbf{w}_k}$ becomes ill-conditioned (or negative definite) in a sequential update if $s_j \leq 0$, that is, when $\Delta\tilde{\tau}_{w,j} \leq -V_{w,j}^{-1}$, because $|\mathbf{\Sigma}_{\mathbf{w}_k}^{\text{new}}| = |\mathbf{\Sigma}_{\mathbf{w}_k}|/s_j$. According to (65), this can result from site updates where $\Delta\tilde{\tau}_{w,j} \leq -V_{w,j}^{-1}$, corresponding to cases with $\hat{V}_{w,j}^{-1} \leq (1-\eta/\delta)V_{w,j}^{-1}$. If no damping is used , i.e., $\delta = 1$, this requires that the tilted precision $\hat{V}_{w,j}^{-1}$ becomes very small (or even negative if $\eta = 1$) corresponding to a very (or infinitely) large approximate posterior uncertainty in $\hat{p}_j(w_j)$. Therefore, it is sensible to make sure that each EP update is done only if the corresponding tilted distribution is proper. From $\hat{V}_{w,j}^{-1} \leq (1-\eta/\delta)V_{w,j}^{-1}$ we can also see that using damping $\delta < 1$ helps to avoid problems arising from inaccurate tilted moment derivations. This discussion applies also to the rank-one updates in the EP iterations for the likelihood terms in line 5 of Algorithm 1 with $s_i = 1 + \Delta\tilde{\tau}_{i,k}V_{i,k}$, $V_{i,k} = \mathbf{x}_i^{\mathrm{T}}\mathbf{\Sigma}_{\mathbf{w}_k}\mathbf{x}_i$, and $\mathbf{a}_i = \mathbf{\Sigma}_{\mathbf{w}_k}\mathbf{x}_i$.

Another type of problem can arise from the rank-one update (66) if $-V_{w,j}^{-1} < \Delta\tilde{\tau}_{w,j} < 0$, although the approximate covariance remains positive definite. If the site precisions $\tilde{\tau}_{w,j}$ that are used to construct $\mathbf{\Sigma}_{\mathbf{w}_k}$ according to (36) are allowed to become negative, a large negative site precision adjustment $\Delta\tilde{\tau}_{w,j}$ can cause some of the cavity precisions $\{V_{w,-l}^{-1}\}_{l\neq j}$ related to the other terms to become very small or even negative at subsequent cavity computation steps (62) (Jylänki et al., 2011). Negative cavity precisions should not occur if all site precision parameters are non-negative but still, with certain models, the cavity variances can become very large causing unstable tilted moment integrations and site updates (Minka, 2001a; Seeger, 2008). Because negative cavity precisions are associated with too large negative adjustments $\Delta\tilde{\tau}_{w,j}$, a useful heuristic way to mitigate these problems is to apply more damping in the updates with $\Delta\tilde{\tau}_{w,j} < 0$ in (65). As a result, more cautious steps are taken whenever the posterior variances are increased locally but otherwise greedier updates are done to decrease the posterior uncertainty. In our experiments, this was found helpful especially with the updates of the likelihood term approximations $\tilde{t}_{\mathbf{w}_k,i}(\mathbf{w}|\tilde{\tau}_{i,k}, \tilde{\nu}_{i,k})$ in (60). In case of parallel updates in lines 6 and 7 of Algorithm 1, the posterior covariances can be recomputed by gradually increasing damping (especially for the negative site precision adjustments) as many times as required so that all the resulting cavity distributions are well defined.

Constraining the site precision parameters to positive values can improve the stability and convergence of the EP algorithm but it can also change the properties of the posterior approximation because the moment consistency conditions (23) may not be satisfied anymore. In addition, constraining the site precisions may not be sensible with certain models. For example, a robust non-log-concave observation model can result in negative precision parameters for the likelihood terms related to the outlying observations meaning that such observations increase the posterior uncertainty locally. Thus, we do not wish to tamper with the likelihood site parameters, because our observation model is also non-log-concave and prone to multimodal tilted distributions. On the other hand, constraining the precision parameters of the prior site approximations can be viewed as setting a limit to the maximum prior uncertainty on the unknown model parameters. Therefore, we chose to leave the parameters $\tilde{\tau}_{i,k}$, $\tilde{\alpha}_i$, and $\tilde{\tau}_{\theta,i} = \tilde{\sigma}_{\theta,i}^{-2}$ related to the likelihood term approximations (12) and (35) unconstrained but assign constraints $\tilde{\tau}_{v,k} = \tilde{\sigma}_{v,k}^{-2} \geq \tilde{\tau}_{\min}$ and $\tilde{\tau}_{w,j} = \tilde{\sigma}_{w,j}^{-2} \geq \tilde{\tau}_{\min}$ to the precision parameters of the prior term approximations (13) and (14) with some small positive value such as 0.1^2 for $\tilde{\tau}_{\min}$. In practice, this is implemented

by modifying the outcome of the tilted moment derivations (64) in the algorithm of Appendix F with $\hat{V}_{w,j}^{-1} = V_{w,j}^{-1} + \delta^{-1}\eta(\tilde{\tau}_{\min} - \tilde{\tau}_{w,j})$ whenever the unconstrained update (65) results in $\tilde{\tau}_{w,j}^{\text{new}} < \tilde{\tau}_{\min}$. Recomputing the update (65) with this modified tilted variance results in slightly underestimated variances only in case of very wide tilted distributions but the tilted means $\hat{m}_{w,j}$ are matched exactly. In our experiments, this improved the stability of the challenging likelihood term updates in lines 2–7 of Algorithm 1 by preventing the effective weight prior variances $\tilde{\sigma}_{v,k}^2$ and $\tilde{\sigma}_{w,j}^2$ from becoming very large.

Appendix H. Computing the Predictions

The prediction for a new test input $\mathbf{x}_*$ can be computed using approximations (17), (36) and (37), as follows

$$\begin{aligned} p(y_*|\mathbf{x}_*) &\approx \int p(y_*|f(\mathbf{x}_*),\theta) q(\mathbf{v}|\boldsymbol{\mu}_\mathbf{v}, \boldsymbol{\Sigma}_\mathbf{v}) \prod_{k=1}^{K} q(\mathbf{w}_k|\boldsymbol{\mu}_{\mathbf{w}_k}, \boldsymbol{\Sigma}_{\mathbf{w}_k}) q(\theta|\mu_\theta, \sigma_\theta^2) d\mathbf{v} d\mathbf{w} d\theta \\ &\approx \int \mathcal{N}(y_*|f_*, \exp(\theta)) \mathcal{N}(f_*|m_{f_*}, V_{f_*}) q(\theta) df_* d\theta \\ &= \int \mathcal{N}(y_*|m_{f_*}, V_{f_*} + \exp(\theta)) q(\theta) d\theta, \end{aligned} \tag{68}$$

where the approximate mean m_{f_*} and V_{f_*} of the latent function value $f(\mathbf{x}_*) = f_* = \sum_{k=1}^{K} v_k g(\mathbf{w}_k^{\mathrm{T}} \mathbf{x}_*) + v_0$ is approximated in the same way as in equation (46). The cavity mean $\boldsymbol{\mu}_{-i}$ and covariance $\boldsymbol{\Sigma}_{-i}$ are replaced with $\boldsymbol{\mu}_\mathbf{v}$ and $\boldsymbol{\Sigma}_\mathbf{v}$, and the means $\mathbf{m}_{\mathbf{g}_*} = \mathrm{E}(\mathbf{g}(\mathbf{h}_*))$ and variances $\mathbf{V}_{\mathbf{g}_*} = \mathrm{Var}(\mathbf{g}(\mathbf{h}_*))$ of the hidden unit activations are computed with respect to the approximations $q(\mathbf{w}_k) = \mathcal{N}(\mathbf{w}_k|\boldsymbol{\mu}_{\mathbf{w}_k}, \boldsymbol{\Sigma}_{\mathbf{w}_k})$. The predictive mean is given by $\mathrm{E}(y_*|\mathbf{x}_*) = \mathrm{E}(\mathrm{E}(y_*|\mathbf{x}_*, \theta)) = \mathrm{E}(m_{f_*}) = m_{f_*}$. The predictive variances $\mathrm{Var}(y_*|\mathbf{x}_*) = \mathrm{E}(\mathrm{Var}(y_*|\mathbf{x}_*, \theta)) + \mathrm{Var}(\mathrm{E}(y_*|\mathbf{x}_*, \theta)) = V_{f_*} + \mathrm{E}(\exp(\theta))$ and the predictive densities $p(y_*|\mathbf{x}_*)$, can be approximated either with a plug-in value for $\theta = \mu_\theta$ or by integrating over θ using a numerical quadrature (in the experiments we used numerical quadratures).

Appendix I. Marginal Likelihood Approximation

An EP approximation to the log marginal likelihood $\log Z = \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\gamma})$ conditioned on the fixed hyperparameters $\boldsymbol{\gamma}$ as defined in (8) can be computed in a numerically stable and efficient manner following the general EP formulation for Gaussian approximating families summarized by Cseke and Heskes (2011, appendix C). Adopting the formulation for our approximate family gives

$$\begin{aligned} \log Z_{\text{EP}} = {} & \Psi(\boldsymbol{\mu}_\mathbf{v}, \boldsymbol{\Sigma}_\mathbf{v}) + \sum_{k=1}^{K} \Psi(\boldsymbol{\mu}_{\mathbf{w}_k}, \boldsymbol{\Sigma}_{\mathbf{w}_k}) + \Psi(\mu_\theta, \sigma_\theta^2) + \sum_{l=1}^{L} \Psi(\mu_{\phi,l}, \sigma_{\phi,l}^2) \\ & + \frac{1}{\eta} \sum_{i=1}^{n} \Big(\ln \hat{Z}_i + \Psi(\mu_{\theta,-i}, \sigma_{\theta,-i}^2) - \Psi(\mu_\theta, \sigma_\theta^2) + \sum_{k=1}^{K} \big(\Psi(m_{-i,k}, V_{-i,k}) - \Psi(m_{i,k}, V_{i,k}) \big) \Big) \\ & + \frac{1}{\eta} \sum_{i=1}^{n} \Big(\frac{1}{2} \big(s_i^{-1} (\mathbf{a}_i^{\mathrm{T}} \tilde{\boldsymbol{\alpha}}_i)^2 - \eta \tilde{\boldsymbol{\beta}}_i^{\mathrm{T}} (\boldsymbol{\mu}_\mathbf{v} + \mathbf{a}_i) - \ln(s_i \eta) \big) \Big) \end{aligned}$$

$$+\frac{1}{\eta_w}\sum_{j=1}^{Kd}\Big(\log\hat{Z}_{w,j}+\Psi(m_{w,-j},V_{w,-j})-\Psi(m_{w,j},V_{w,j})+\Psi(\mu_{\phi,-j},\sigma^2_{\phi,-j})-\Psi(\mu_{\phi_{l_j}},\sigma^2_{\phi_{l_j}})\Big)$$
$$+\frac{1}{\eta_v}\sum_{k=1}^{K}\Big(\ln\hat{Z}_{v,k}+\Psi(m_{v,-k},V_{v,-k})-\Psi(m_{v,k},V_{v,k})\Big)$$
$$-\Psi(\mu_{v_0},\sigma^2_{v_0})-\Psi(\mu_{\theta,0},\sigma^2_{\theta,0})-\sum_{l=1}^{L}\Psi(\mu_{\phi,0},\sigma^2_{\phi,0}), \tag{69}$$

where η, η_w, and η_v are the fraction parameters related to the model terms $p(y_i|f_i,\theta)$, $p(w_j|\phi_{l_j})$, and $p(v_k)$, respectively, and $s_i=\eta^{-1}-\tilde{\boldsymbol{\alpha}}_i^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathbf{v}}\tilde{\boldsymbol{\alpha}}_i$ together with $\mathbf{a}_i=\boldsymbol{\mu}_{\mathbf{v}}-\eta\boldsymbol{\Sigma}_{\mathbf{v}}\tilde{\boldsymbol{\beta}}_i$ can be computed during the cavity computations (40). The normalization terms $\Psi(\cdot,\cdot)$ related to unnormalized Gaussian densities (also known as log partition functions) computed for various Gaussian cavity and marginal distributions in (69) are defined as

$$\Psi(\boldsymbol{\mu},\boldsymbol{\Sigma})=\log\int\exp\left(-\frac{1}{2}\mathbf{w}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{w}+\boldsymbol{\nu}^{\mathrm{T}}\mathbf{w}\right)d\mathbf{w}=\frac{1}{2}\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\nu}+\frac{1}{2}\log|\boldsymbol{\Sigma}|+\frac{d}{2}\log(2\pi),$$

where $\mathbf{w}$, $\boldsymbol{\mu}$, and $\boldsymbol{\nu}=\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ are $d\times 1$ vectors and $\boldsymbol{\Sigma}$ is a $d\times d$ matrix. The approximate means and covariances in line one of (69) are given by equations (37), (36), (18), and (17) in respective order. The cavity and marginal moments in line two related to the likelihood sites are defined in (41) and (39). Line three corresponds to $\Psi(\boldsymbol{\mu}_{-i},\boldsymbol{\Sigma}_{-i})-\Psi(\boldsymbol{\mu}_{\mathbf{v}},\boldsymbol{\Sigma}_{\mathbf{v}})$, which can be computed efficiently using $s_i=\eta^{-1}-\tilde{\boldsymbol{\alpha}}_i^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathbf{v}}\tilde{\boldsymbol{\alpha}}_i$ and $\mathbf{a}_i=\boldsymbol{\mu}_{\mathbf{v}}-\eta\boldsymbol{\Sigma}_{\mathbf{v}}\tilde{\boldsymbol{\beta}}_i$ as defined in the cavity computations (40). The cavity and marginal moments in line four associated with the prior terms $p(w_j|\phi_{l_j})$ are computed using (62), and analogous definitions can also be used in line five that is related to prior terms $p(v_k)$. The last line of (69) contains the constant normalization terms related to the fixed Gaussian priors including $p(v_0)=\mathcal{N}(\mu_{v_0},\sigma^2_{v_0})$ for the output bias, $p(\theta)=\mathcal{N}(\mu_{\theta,0},\sigma^2_{\theta,0})$ for the noise level, and $p(\phi_l)=\mathcal{N}(\mu_{\phi,0},\sigma^2_{\phi,0})$ for the input weight scales for $l=1,...,L$.

When θ is inferred using EP, the normalization terms of the tilted distributions in line two of (69), which are defined by

$$\hat{Z}_i\approx\int p(y_i|\mathbf{v}^{\mathrm{T}}\mathbf{g}(\mathbf{h}_i),\theta)^{\eta}q_{-i}(\mathbf{v},\mathbf{h}_i,\theta)d\mathbf{v}d\mathbf{h}_i d\theta,$$

can be computed using approximation (50). Otherwise, they can be computed using the expression $\hat{Z}_i=Z(\theta)\mathcal{N}\left(y_i|m_{f_i},V_{f_i}+\eta^{-1}\exp(\theta)\right)$ from (50) with the known value of θ. The normalization terms of the other tilted distributions related to the prior terms on lines four and five are defined as

$$\hat{Z}_{v,k}=\int p(v_k|\sigma^2_{v,0})^{\eta_v}q_{-k}(v_k)dv_k\quad\text{and}\quad\hat{Z}_{w,j}=\int p(w_j|\phi_{l_j})^{\eta_w}q_{-j}(w_j)q_{-j}(\phi_{l_j})dw_j d\phi_{l_j},$$

and they can be computed during the step 2 of the EP algorithm summarized in Appendix F.

All terms of equation (69) excluding $\Psi(\boldsymbol{\mu}_{\mathbf{v}},\boldsymbol{\Sigma}_{\mathbf{v}})$ and $\Psi(\boldsymbol{\mu}_{\mathbf{w}_k},\boldsymbol{\Sigma}_{\mathbf{w}_k})$ can be computed without significant additional cost simultaneously during the EP update of the corresponding site approximation. Term $\Psi(\boldsymbol{\mu}_{\mathbf{v}},\boldsymbol{\Sigma}_{\mathbf{v}})$ can be computed using one Cholesky decomposition

at each parallel update step of $q(\mathbf{w}_k)$ in line 7 of Algorithm 1. Similarly, if parallel updates are used for the input weight approximations, $\Psi(\boldsymbol{\mu}_{\mathbf{w}_k}, \boldsymbol{\Sigma}_{\mathbf{w}_k})$ can be computed using the same Cholesky decompositions that are used to recompute $q(\mathbf{w}_k)$ in line 6 of Algorithm 1. In case sequential EP is used for $q(\mathbf{w}_k)$ in line 5 of Algorithm 1, vectors $\boldsymbol{\nu}_{\mathbf{w}_k} = \boldsymbol{\Sigma}_{\mathbf{w}_k}^{-1}\boldsymbol{\mu}_{\mathbf{w}_k}$ and determinant term $\log|\boldsymbol{\Sigma}_{\mathbf{v}}|$ can be updated simultaneously with the rank-1 updates of $\boldsymbol{\mu}_{\mathbf{w}_k}$ and $\boldsymbol{\Sigma}_{\mathbf{w}_k}$ that are given by (66).

The EP approximation $\log Z_{EP}$ has the appealing property that its partial derivatives with respect to the site parameters in their canonical forms[5] are zero when the algorithm has been iterated until convergence (Opper and Winther, 2005). This follows form the fact that the fixed points of the EP algorithm correspond to the stationary points of (69) with respect to the site parameters (or equivalently the cavity parameters) using constraints of the form $V_{-i,k}^{-1} = V_{i,k}^{-1} - \eta\tilde{\tau}_{i,k}$ and $V_{-i,k}^{-1}m_{-i,k} = V_{i,k}^{-1}m_{i,k} - \eta\tilde{\nu}_{i,k}$, which are equivalent to the cavity definitions. Thereby, the marginal likelihood approximation can be used for gradient-based estimation of the hyperparameters $\sigma_{v,0}^2$, $\sigma_{v_0,0}^2$, $\mu_{\phi,0}^2$ and $\sigma_{\phi,0}^2$, and also parameters θ and $\{\phi_l\}_{l=1}^L$ in case they are not inferred within the EP framework for determining $\{q(\mathbf{w}_k)\}_{k=1}^K$ and $q(\mathbf{v})$. Because the convergence of the likelihood approximation can take many iterations it is advisable to initialize the hyperparameters to sensible values and run the EP algorithm once until sufficient convergence starting from a zero initialization for the site parameters. After that, gradient-based local update steps can be taken for the hyperparameter values by continuing the EP iterations from the previous site parameter values at each new hyperparameter configuration.

References

Kevin Bache and Moshe Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Bart Bakker, Tom Heskes, Jan Neijt, and Bert Kappen. Improving Cox survival analysis with a neural-Bayesian approach. *Statistics in Medicine*, 23:2989–3012, 2004.

David Barber and Christopher M. Bishop. Ensemble learning for multi-layer networks. In M.I. Jordan, M.J. Kearns, and S.A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 395–401. MIT Press, 1998.

Botond Cseke and Tom Heskes. Approximate marginals in latent Gaussian models. *Journal of Machine Learning Research*, 12:417–454, 2011.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660, 2010.

Joao F. G. de Freitas. *Bayesian Methods for Neural Networks*. PhD thesis, University of Cambridge, 1999.

5. In our model the canonical parameters consist of $\{\tilde{\tau}_{i,k}, \tilde{\nu}_{i,k}, \tilde{\mathbf{T}}_{i,\mathbf{vv}} = \tilde{\boldsymbol{\alpha}}_i\tilde{\boldsymbol{\alpha}}_i^{\mathrm{T}}, \tilde{\boldsymbol{\beta}}_i, \tilde{\tau}_{\theta,i} = \tilde{\sigma}_{\theta,i}^{-2}, \tilde{\nu}_{\theta,i} = \tilde{\sigma}_{\theta,i}^{-2}\tilde{\mu}_{\theta,i}\}$ for the likelihood sites (10), $\{\tilde{\tau}_{v,k} = \tilde{\sigma}_{v,k}^{-2}, \tilde{\nu}_{v,k} = \tilde{\sigma}_{v,k}^{-2}\tilde{\mu}_{v,k}\}$ for the prior sites (13), and $\{\tilde{\tau}_{w,j} = \tilde{\sigma}_{w,j}^{-2}, \tilde{\nu}_{w,j} = \tilde{\sigma}_{w,j}^{-2}\tilde{\mu}_{w,j}, \tilde{\tau}_{\phi,j} = \tilde{\sigma}_{\phi,j}^{-2}, \tilde{\nu}_{\phi,j} = \tilde{\sigma}_{\phi,j}^{-2}\tilde{\mu}_{\phi,j}\}$ for the prior sites (14).

Marcel van Gerven, Botond Cseke, Robert Oostenveld, and Tom Heskes. Bayesian source localization with the multivariate Laplace prior. In Y. Bengio, D. Schuurmans, J. Lafferty, C.K.I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1901–1909. Curran Associates, Inc., 2009.

Marcel van Gerven, Botond Cseke, Floris de Lange, and Tom Heskes. Efficient Bayesian multivariate fMRI analysis using a sparsifying spatio-temporal prior. *NeuroImage*, 50: 150–161, 2010.

Alex Graves. Practical variational inference for neural networks. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2348–2356. Curran Associates, Inc., 2011.

Daniel Hernández-Lobato, José M. Hernández-Lobato, and A. Suárez. Expectation propagation for microarray data classification. *Pattern Recognition Letters*, 31(12):1618–1626, 2010.

Daniel Hernández-Lobato, José M. Hernández-Lobato, and Pierre Dupont. Generalized spike-and-slab priors for Bayesian group feature selection using expectation propagation. *Journal of Machine Learning Research*, 14:1891–1945, 2013.

José M. Hernández-Lobato, Tjeerd Dijkstra, and Tom Heskes. Regulator discovery from gene expression time series of malaria parasites: a hierarchical approach. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 649–656. Curran Associates, Inc., 2008.

Tom Heskes and Onno Zoeter. Expectation propagation for approximate inference in dynamic Bayesian networks. In *Proceedings of the Eighteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 216–223. Morgan Kaufmann, 2002.

Tom Heskes, Bart Bakker, and Bert Kappen. Approximate algorithms for neural-Bayesian approaches. *Theoretical Computer Science*, 287:219–238, 2002.

Geoffrey E. Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, COLT '93, pages 5–13. ACM, 1993.

Antti Honkela and Harri Valpola. Unsupervised variational Bayesian learning of nonlinear models. In L.K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 593–600. MIT Press, 2005.

Pasi Jylänki, Jarno Vanhatalo, and Aki Vehtari. Gaussian process regression with a Student-t likelihood. *Journal of Machine Learning Research*, 12:3227–3257, 2011.

Jouko Lampinen and Aki Vehtari. Bayesian approach for neural networks – review and case studies. *Neural Networks*, 14(3):7–24, 2001.

David J. C. Mackay. Probable networks and plausible predictions – a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3):469–505, 1995.

Thomas Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, 2001a.

Thomas Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 362–369. Morgan Kaufmann, 2001b.

Thomas Minka. Power EP. Technical report, Microsoft Research, Cambridge, 2004.

Thomas Minka. Divergence measures and message passing. Technical report, Microsoft Research, Cambridge, 2005.

Thomas Minka and John Lafferty. Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 352–359. Morgan Kaufmann, 2002.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer, 1996.

Hannes Nickisch and Carl E. Rasmussen. Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 9:2035–2078, Oct. 2008.

Manfred Opper and Ole Winther. Mean field approach to Bayes learning in feed-forward neural networks. *Physical Review Letters*, 76:1964–1967, Mar. 1996.

Manfred Opper and Ole Winther. Expectation consistent approximate inference. *Journal of Machine Learning Research*, 6:2177–2204, December 2005.

Gintaras V. Puskorius and Lee A. Feldkamp. Decoupled extended Kalman filter training of feedforward layered networks. In *Proceedings of the International Joint Conference on Neural Networks*, pages 307–312. Seattle, 1991.

Yuan (Alan) Qi, Thomas P. Minka, Rosalind W. Picard, and Zoubin Ghahramani. Predictive automatic relevance determination by expectation propagation. In *Proceedings of Twenty-first International Conference on Machine Learning*, pages 671–678, 2004.

Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

Fabiano Ribeiro and Manfred Opper. Expectation propagation with factorizing distributions: A Gaussian approximation and performance results for simple models. *Neural Computation*, 23(4):1047–1069, 2011.

Jaakko Riihimäki, Pasi Jylänki, and Aki Vehtari. Nested expectation propagation for Gaussian process classification with a multinomial probit likelihood. *Journal of Machine Learning Research*, 14:75–109, 2013.

Anton Schwaighofer and Volker Tresp. Transductive and inductive methods for approximate Gaussian process regression. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 953–960. MIT Press, 2003.

Matthias Seeger. Bayesian inference and optimal design for the sparse linear model. *Journal of Machine Learning Research*, 9:759–813, 2008.

Matthias Seeger and Hannes Nickisch. Fast convergent algorithms for expectation propagation approximate Bayesian inference. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 652–660. JMLR W&CP, vol. 15, 2011.

Michael E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, Dec. 2001.

Jarno Vanhatalo, Jaakko Riihimäki, Jouni Hartikainen, Pasi Jylänki, Ville Tolvanen, and Aki Vehtari. GPstuff: Bayesian modeling with Gaussian processes. *Journal of Machine Learning Research*, 14:1175–1179, 2013.

Eric A. Wan and Rudolph van der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proceedings of IEEE Symposium on Adaptive Systems for Signal Processing, Communications, and Control (AS-SPCC)*, pages 153–158, 2000.

Christopher K. I. Williams. Computation with infinite neural networks. *Neural Computation*, 10(5):1203–1216, 1998.

Peter M. Williams. Bayesian regularisation and pruning using a Laplace prior. *Neural Computation*, 7(1):117–143, 1995.

Ole Winther. Computing with finite and infinite networks. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 336–342. MIT Press, 2001.

David Wipf and Srikantan Nagarajan. A new view of automatic relevance determination. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1625–1632. Curran Associates, Inc., 2008.

David Wipf, Bhaskar D. Rao, and Srikantan Nagarajan. Latent variable Bayesian models for promoting sparsity. *IEEE Transactions on Information Theory*, 57(9):6236–6255, Sept. 2011.

Journal of Machine Learning Research 15 (2014) 1903-1928 Submitted 5/10; Revised 1/14; Published 6/14

Pattern Alternating Maximization Algorithm for Missing Data in High-Dimensional Problems

Nicolas Städler N.STADLER@NKI.NL
The Netherlands Cancer Institute
Plesmanlaan 121
1066 CX Amsterdam, The Netherlands

Daniel J. Stekhoven STEKHOVEN@QUANTIK.CH
Quantik AG
Bahnhofstrasse 57
8965 Berikon, Switzerland

Peter Bühlmann BUHLMANN@STAT.MATH.ETHZ.CH
Seminar for Statistics, ETH Zurich
Rämistrasse 101
8092 Zurich, Switzerland

Editor: Tommi Jaakkola

Abstract

We propose a novel and efficient algorithm for maximizing the observed log-likelihood of a multivariate normal data matrix with missing values. We show that our procedure, based on iteratively regressing the missing on the observed variables, generalizes the standard EM algorithm by alternating between different complete data spaces and performing the E-Step incrementally. In this non-standard setup we prove numerical convergence to a stationary point of the observed log-likelihood. For high-dimensional data, where the number of variables may greatly exceed sample size, we perform regularization using a Lasso-type penalty. This introduces sparsity in the regression coefficients used for imputation, permits fast computation and warrants competitive performance in terms of estimating the missing entries. We show on simulated and real data that the new method often improves upon other modern imputation techniques such as k-nearest neighbors imputation, nuclear norm minimization or a penalized likelihood approach with an ℓ_1-penalty on the concentration matrix.

Keywords: missing data, observed likelihood, (partial) E- and M-Step, Lasso, penalized variational free energy

1. Introduction and Motivation

Missing data imputation for large data sets is a significant challenge in many complex data applications. One well-known example are microarray data sets which contain expression profiles of p genes from a series of n experiments, where p is typically much larger than

n (Troyanskaya et al., 2001; Aittokallio, 2010). In this paper, we propose a novel and computationally efficient imputation algorithm based on missingness pattern alternating maximization in the high-dimensional multivariate normal model. The Gaussian assumption in our model is used for computation of the likelihood but empirical findings suggest that the method is applicable to a wide range of problems where continuous data is arranged in the form of a $n \times p$ matrix with $p \gg n$.

There is a growing literature on missing values in the high-dimensional context (Allen and Tibshirani, 2010; Josse et al., 2011; Loh and Wainwright, 2012; Rosenbaum and Tsybakov, 2010). In recent years, a special focus has been given to the so-called matrix completion problem, where the goal is to recover a low-rank matrix from an incomplete set of entries. It has been shown in a series of fascinating papers that one can recover the missing data entries by solving a convex optimization problem, namely, nuclear-norm minimization subject to data constraints (Candès and Recht, 2009; Candès and Tao, 2010; Keshavan et al., 2010). Efficient algorithms for the matrix completion problem were proposed by Cai et al. (2010) and Mazumder et al. (2010). However, many incomplete data problems do not arise from a near low rank matrix scenario. In these cases there is substantial room to improve upon the convex matrix completion algorithms. We will empirically demonstrate this point for some high-throughput biological data.

In this manuscript we assume a multivariate normal model (MVN) with p-dimensional covariance matrix Σ and address the missing data problem through a likelihood approach (Little and Rubin, 1987; Schafer, 1997). Recently, in the high-dimensional setup, Städler and Bühlmann (2012) proposed to maximize the penalized observed log-likelihood with an ℓ_1-penalty on the concentration matrix Σ^{-1}. They called their method *MissGLasso*, as an extension of the graphical Lasso (Friedman et al., 2008) for missing data. MissGLasso induces sparsity in the concentration matrix and uses an EM algorithm for optimization. Roughly, the algorithm can be summarized as follows: in the E-Step, for each sample, the regression coefficients of the missing against the observed variables are computed from the current estimate $\hat{\Sigma}^{-1}$; in the following M-Step, the missing values are imputed by linear regressions and $\hat{\Sigma}^{-1}$ is re-estimated by applying the graphical Lasso on completed data. There are two main drawbacks of this algorithm in a high-dimensional context. First, the E-Step is rather complex as it involves (for each sample) inversion and multiplication of large matrices in order to compute the regression coefficients. Secondly, a sparse concentration matrix does not imply sparse regression coefficients while we believe that in high-dimensions, sparse regression coefficients would enhance imputations. Our new algorithm, *MissPALasso* in this paper, generalizes the E-Step in order to resolve the disadvantages of MissGLasso. In particular, inversion of a matrix (in order to compute the regression coefficients) will be replaced by a simple soft-thresholding operator. In addition, the regression coefficients will be sparse, which leads to a new sparsity concept for missing data estimation.

MissPALasso emerges from the missingness pattern alternating maximization algorithm (*MissPA*) which we propose for optimizing the (unpenalized) observed log-likelihood. We show that this method generalizes the E- *and* M-Step of the EM algorithm by alternating between different complete data spaces *and* performing the E-Step incrementally (Dempster et al., 1977; Fessler and Hero, 1994; Neal and Hinton, 1998). Such a generalization does not

fit into any of the existing methodologies which extend the standard EM. We analyse our procedure using the variational free energy (Jordan et al., 1999) and prove convergence to a stationary point of the observed log-likelihood.

The further organization of the paper is as follows: Section 2 introduces the setup and the useful notion of missingness patterns. In Section 3 we present our new methodology based on (missingness) pattern alternating maximization and develop MissPALasso for imputation in the high-dimensional scenario. Section 4 compares performance of MissPALasso with other competitive methods and reports on computational efficiency. Finally, in Section 5, we present some theory to gain insights into the numerical properties of the procedure.

2. Setup

We assume $X = (X_1, \ldots, X_p) \sim \mathcal{N}(\mu, \Sigma)$ has a p-variate normal distribution with mean μ and covariance matrix Σ. In order to simplify the notation we set without loss of generality $\mu = 0$: for $\mu \neq 0$, some of the formulae involve the parameter μ and an intercept column of $(1, \ldots, 1)$ in the design matrices but conceptually, we can proceed as for the case with $\mu = 0$. We then write $\mathbf{X} = (\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{mis}})$, where $\mathbf{X}$ represents an i.i.d. random sample of size n, $\mathbf{X}_{\text{obs}}$ denotes the set of observed values, and $\mathbf{X}_{\text{mis}}$ the missing data.

2.1 Missingness Patterns and Different Parametrizations

For our purpose it will be convenient to group rows of the matrix $\mathbf{X}$ according to their missingness patterns (Schafer, 1997). We index the unique missingness patterns that actually appear in our data by $k = 1, \ldots, s$. Furthermore, with $o_k \subset \{1, \ldots, p\}$ and $m_k = \{1, \ldots, p\} \setminus o_k$ we denote the set of observed variables and the set of missing variables, respectively. $\mathcal{I}_k$ is the index set of the samples (row numbers) which belong to pattern k, whereas $\mathcal{I}_k^c = \{1, \ldots, n\} \setminus \mathcal{I}_k$ stands for the row numbers which do not belong to that pattern. By convention, samples with all variables observed do not belong to a missingness pattern.

Consider a partition $X = (X_{o_k}, X_{m_k})$ of a single Gaussian random vector. It is well known that $X_{m_k}|X_{o_k}$ follows a linear regression on X_{o_k} with regression coefficients $B_{m_k|o_k}$ and covariance $\Sigma_{m_k|o_k}$ given by

$$\begin{aligned} B_{m_k|o_k} &= \Sigma_{m_k,o_k}\Sigma_{o_k}^{-1}, \\ \Sigma_{m_k|o_k} &= \Sigma_{m_k} - \Sigma_{m_k,o_k}\Sigma_{o_k}^{-1}\Sigma_{o_k,m_k}. \end{aligned} \tag{1}$$

Consequently, we can write the density function $p(x; \Sigma)$ of X as

$$p(x; \Sigma) = p(x_{m_k}|x_{o_k}; B_{m_k|o_k}, \Sigma_{m_k|o_k})p(x_{o_k}; \Sigma_{o_k}),$$

i.e., the density can be characterized by either the parameter Σ or $(\Sigma_{o_k}, B_{m_k|o_k}, \Sigma_{m_k|o_k})$. The transformation (1) allows us to switch between both parametrizations.

2.2 Observed Log-Likelihood and Maximum Likelihood Estimation (MLE)

A systematic approach to estimate the parameter of interest Σ from $\mathbf{X}_{\text{obs}}$ maximizes the observed log-likelihood $\ell(\Sigma; \mathbf{X}_{\text{obs}})$ given by

$$\ell(\Sigma; \mathbf{X}_{\text{obs}}) = \sum_{i \notin \bigcup_k \mathcal{I}_k} \log p(x_i; \Sigma) + \sum_{k=1}^{s} \sum_{i \in \mathcal{I}_k} \log p(x_{i,o_k}; \Sigma_{o_k}). \tag{2}$$

Inference for Σ can be based on (2) if the underlying missing data mechanism is *ignorable*. The missing data mechanism is said to be *ignorable* if the probability that an observation is missing may depend on $\mathbf{X}_{\text{obs}}$ but not on $\mathbf{X}_{\text{mis}}$ (*Missing at Random*) and if the parameters of the data model and the parameters of the missingness mechanism are *distinct*. For a precise definition see Little and Rubin (1987).

Explicit maximization of $\ell(\Sigma; \mathbf{X}_{\text{obs}})$ is only possible for special missing data patterns. Most prominent are examples with a so-called monotone missing data pattern (Little and Rubin, 1987; Schafer, 1997), where X_1 is more observed than X_2, which is more observed than X_3, and so on. In this case, the observed log-likelihood factorizes and explicit maximization is achieved by performing several regressions. For a general pattern of missing data, the standard EM algorithm is often used for optimization of (2). See Schafer (1997) for a detailed description of the algorithm. In the next section we present an alternative method for maximizing the observed log-likelihood. We will argue that this new algorithm is computationally more efficient than the standard EM.

3. Missingness Pattern Alternating Maximization

For each missingness pattern, indexed by $k = 1, \ldots, s$, we introduce some further notation:

$$\begin{aligned} \mathbf{X}^k &= (x_{i,j}) \quad \text{with} \quad i \in \mathcal{I}_k, \quad j = 1, \ldots, p \\ \mathbf{X}^{-k} &= (x_{i,j}) \quad \text{with} \quad i \in \mathcal{I}_k^c, \quad j = 1, \ldots, p. \end{aligned}$$

Thus, $\mathbf{X}^k$ is the $|\mathcal{I}_k| \times p$ submatrix of $\mathbf{X}$ with rows belonging to the kth pattern. Similarly, $\mathbf{X}^{-k}$ is the $|\mathcal{I}_k^c| \times p$ matrix with rows not belonging to the kth pattern. In the same way we define $\mathbf{X}^k_{o_k}, \mathbf{X}^k_{m_k}, \mathbf{X}^{-k}_{o_k}$ and $\mathbf{X}^{-k}_{m_k}$. For example, $\mathbf{X}^k_{o_k}$ is defined as the $|\mathcal{I}_k| \times |o_k|$ matrix with

$$\mathbf{X}^k_{o_k} = (x_{i,j}) \quad \text{with} \quad i \in \mathcal{I}_k, \quad j \in o_k.$$

3.1 MLE for Data with a Single Missingness Pattern

Assume that the data matrix $\mathbf{X}$ has only one single missingness pattern, denoted by s. This is the most simple example of a monotone pattern. The observed log-likelihood factorizes according to:

$$\ell(\Sigma; \mathbf{X}_{\text{obs}}) = \sum_{i \in \mathcal{I}_s} \log p(x_{i,o_s}; \Sigma_{o_s}) + \sum_{i \in \mathcal{I}_s^c} \log p(x_i; \Sigma)$$
$$= \sum_{i=1}^{n} \log p(x_{i,o_s}; \Sigma_{o_s}) + \sum_{i \in \mathcal{I}_s^c} \log p(x_{i,m_s} | x_{i,o_s}; B_{m_s|o_s}, \Sigma_{m_s|o_s}). \quad (3)$$

The left and right part in Equation (3) can be maximized separately. The first part is maximized by the sample covariance of the observed variables based on *all samples*, whereas the second part is maximized by a regression of the missing against observed variables based on only the *fully observed samples*. In formulae:

$$\hat{\Sigma}_{o_s} = {}^t\mathbf{X}_{o_s}\mathbf{X}_{o_s}/n, \quad (4)$$

and

$$\hat{B}_{m_s|o_s} = {}^t\mathbf{X}_{m_s}^{-s}\mathbf{X}_{o_s}^{-s}({}^t\mathbf{X}_{o_s}^{-s}\mathbf{X}_{o_s}^{-s})^{-1},$$
$$\hat{\Sigma}_{m_s|o_s} = {}^t(\mathbf{X}_{m_s}^{-s} - \mathbf{X}_{o_s}^{-s}\,{}^t\hat{B}_{m_s|o_s})(\mathbf{X}_{m_s}^{-s} - \mathbf{X}_{o_s}^{-s}\,{}^t\hat{B}_{m_s|o_s})/|\mathcal{I}_s^c|. \quad (5)$$

Having these estimates at hand, it is easy to impute the missing data:

$$\hat{x}_{i,m_s} = \hat{B}_{m_s|o_s}\,{}^t x_{i,o_s} \text{ for all } i \in \mathcal{I}_s, \text{ or, in matrix notation, } \hat{\mathbf{X}}_{m_s}^s = \mathbf{X}_{o_s}^s\,{}^t\hat{B}_{m_s|o_s}.$$

It is important to note, that, if interested in imputation, only the regression part of the MLE is needed and the estimate $\hat{\Sigma}_{o_s}$ in (4) is superfluous.

3.2 MLE for General Missing Data Pattern

We turn now to the general case, where we have more than one missingness pattern, indexed by $k = 1, \ldots, s$. The general idea of the new algorithm is as follows. Assume we have some initial imputations for all missing values. Our goal is to improve on these imputations. For this purpose, we iterate as follows:

- Keep all imputations except those of the 1st missingness pattern fixed and compute the single pattern MLE (for the first pattern) as explained in Section 3.1. In particular, compute the regression coefficients of the missing 1st pattern against all other variables (treated as "observed") based on all samples which do not belong to the 1st pattern.
- Use the resulting estimates (regression coefficients) to impute the missing values from only the 1st pattern.

Next, turn to the 2nd pattern and repeat the above steps. In this way we continue cycling through the different patterns until convergence.

We now describe the missingness pattern alternating maximization algorithm (MissPA) which makes the above idea precise. Let $T = {}^t\mathbf{X}\mathbf{X}$ be the sufficient statistic in the multivariate normal model. Furthermore, we let $T^k = {}^t(\mathbf{X}^k)\mathbf{X}^k$ and $T^{-k} = {}^t(\mathbf{X}^{-k})\mathbf{X}^{-k} = \sum_{l \neq k} T^l$.

Let $\mathcal{T}$ and $\mathcal{T}^k\,(k=1,\ldots,s)$ be some initial guess of T and $T^k\,(k=1,\ldots,s)$, for example, using zero imputation. Our algorithm proceeds as follows:

Algorithm 1: MissPA

(1) $\mathcal{T}$, $\mathcal{T}^k$: initial guess of T and T^k $(k=1,\ldots,s)$.

(2) For $k=1,\ldots,s$ do:

M-Step: Compute the MLE $\hat{B}_{m_k|o_k}$, and $\hat{\Sigma}_{m_k|o_k}$, based on $\mathcal{T}^{-k}=\mathcal{T}-\mathcal{T}^k$:

$$\hat{B}_{m_k|o_k}=\mathcal{T}^{-k}_{m_k,o_k}(\mathcal{T}^{-k}_{o_k,o_k})^{-1},$$
$$\hat{\Sigma}_{m_k|o_k}=\left(\mathcal{T}^{-k}_{m_k,m_k}-\mathcal{T}^{-k}_{m_k,o_k}(\mathcal{T}^{-k}_{o_k,o_k})^{-1}\mathcal{T}^{-k}_{o_k,m_k}\right)/|\mathcal{I}^c_k|.$$

Partial E-Step:

Set $\mathcal{T}^l=\mathcal{T}^l$ for all $l\neq k$ (this takes no time),
Set $\mathcal{T}^k=\mathbb{E}[T^k|\mathbf{X}^k_{o_k},\hat{B}_{m_k|o_k},\hat{\Sigma}_{m_k|o_k}]$,
Update $\mathcal{T}=\mathcal{T}^{-k}+\mathcal{T}^k$.

(3) Repeat step (2) until some convergence criterion is met.

(4) Compute the final maximum likelihood estimator $\hat{\Sigma}$ via:

$\hat{\Sigma}_{o_s}=\mathcal{T}_{o_s,o_s}/n$, $\hat{\Sigma}_{m_s,o_s}=\hat{B}_{m_s|o_s}\hat{\Sigma}_{o_s}$ and $\hat{\Sigma}_{m_s}=\hat{\Sigma}_{m_s|o_s}+\hat{B}_{m_s|o_s}\hat{\Sigma}_{o_s,m_s}$.

Note, that we refer to the maximization step as M-Step and to the imputation step as *partial* E-Step. The word partial refers to the fact that the expectation is only performed with respect to samples belonging to the current pattern. The partial E-Step takes the following simple form:

$$\mathcal{T}^k_{o_k,m_k}={}^t(\mathbf{X}^k_{o_k})\hat{\mathbf{X}}^k_{m_k},$$
$$\mathcal{T}^k_{m_k,m_k}={}^t(\hat{\mathbf{X}}^k_{m_k})\hat{\mathbf{X}}^k_{m_k}+|\mathcal{I}_k|\;\hat{\Sigma}_{m_k|o_k},$$

where $\hat{\mathbf{X}}^k_{m_k}=\mathbb{E}[\mathbf{X}^k_{m_k}|\mathbf{X}^k_{o_k},\hat{B}_{m_k|o_k},\hat{\Sigma}_{m_k|o_k}]=\mathbf{X}^k_{o_k}\,{}^t\hat{B}_{m_k|o_k}$.

Algorithm 1 does not require an evaluation of $\hat{\Sigma}_{o_k}$ in the M-Step, as it is not used in the following partial E-Step. But, if we are interested in the observed log-likelihood or the maximum likelihood estimator $\hat{\Sigma}$ at convergence, we compute $\hat{\Sigma}_{o_s}$ (at convergence), use it together with $\hat{B}_{m_s|o_s}$ and $\hat{\Sigma}_{m_s|o_s}$ to get $\hat{\Sigma}$ via the transformations (1) as explained in step (4).

MissPA is computationally more efficient than the standard EM for missing data: one cycle through all patterns $(k=1,\ldots,s)$ takes about the same time as one iteration of the standard EM. But our algorithm makes more progress since the information from the partial E-Step is employed immediately to perform the next M-Step. We will demonstrate empirically the gain of computational efficiency in Section 4.2. The new MissPA generalizes the standard EM in two ways. Firstly, MissPA alternates between different complete data spaces in the sense of Fessler and Hero (1994). Secondly, the E-Step is performed incrementally (Neal and Hinton, 1998). In Section 5 we will expand on these generalizations and we will provide an appropriate framework which allows analyzing the convergence properties of MissPA.

Finally, a small modification of MissPA, namely replacing in Algorithm 1 the M-Step by

M-Step2: Compute the MLE $\hat{B}_{m_k|o_k}$, and $\hat{\Sigma}_{m_k|o_k}$, based on $\mathcal{T}$:

$$\hat{B}_{m_k|o_k} = \mathcal{T}_{m_k,o_k}(\mathcal{T}_{o_k,o_k})^{-1}$$
$$\hat{\Sigma}_{m_k|o_k} = \left(\mathcal{T}_{m_k,m_k} - \mathcal{T}_{m_k,o_k}(\mathcal{T}_{o_k,o_k})^{-1}\mathcal{T}_{o_k,m_k}\right)/n,$$

results in an alternative algorithm. We show in Section 5 that Algorithm 1 with M-Step2 is equivalent to an incremental EM in the sense of Neal and Hinton (1998).

3.3 High-Dimensionality and Lasso Penalty

The M-Step of Algorithm 1 is basically a multivariate regression of the missing (X_{m_k}) against the observed variables (X_{o_k}). In a high-dimensional framework with $p \gg n$ the number of observed variables $|o_k|$ will be large and therefore some regularization is necessary. The main idea is, in order to regularize, to replace regressions with Lasso analogues (Tibshirani, 1996). We give now the details.

Estimation of $B_{m_k|o_k}$: The estimation of the multivariate regression coefficients in the M-Step2 can be expressed as $|m_k|$ separate minimization problems of the form

$$\hat{B}_{j|o_k} \quad = \quad \arg\min_{\beta} -\mathcal{T}_{j,o_k}\beta + {}^t\beta\mathcal{T}_{o_k,o_k}\beta/2,$$

where $j \in m_k$. Here, $\hat{B}_{j|o_k}$ denotes the jth row vector of the $(|m_k| \times |o_k|)$-matrix $\hat{B}_{m_k|o_k}$ and represents the regression of variable j against the variables from o_k.

Consider now the objective function

$$-\mathcal{T}_{j,o_k}\beta + {}^t\beta\mathcal{T}_{o_k,o_k}\beta/2 + \lambda\|\beta\|_1, \tag{6}$$

with an additional Lasso penalty. Instead of minimizing (6) with respect to β (for all $j \in m_k$), it is computationally much more efficient to perform coordinate-wise improvements from the old parameters (computed in the last cycle through all patterns). For that purpose, let $B^{(r)}_{m_k|o_k}$ be the regression coefficients for pattern k in cycle r and $B^{(r)}_{j|o_k}$ its jth row vector. In cycle $r+1$ we compute $B^{(r+1)}_{j|o_k}$ by minimizing (6) with respect to each of the components of β, holding the other components fixed at their current value. Closed-form updates have the form:

$$B^{(r+1)}_{j|l} \quad = \quad \frac{\mathrm{Soft}\left(\mathcal{T}_{l,l}B^{(r)}_{j|l} - S^{(r)}_l, \lambda\right)}{\mathcal{T}_{l,l}}, \qquad \text{for all } l \in o_k, \tag{7}$$

where

- $B^{(r+1)}_{j|l}$ is the lth component of $B^{(r+1)}_{j|o_k}$ equal to the element (j,l) of matrix $B^{(r+1)}_{m_k|o_k}$
- $S^{(r)}_l$, the gradient of $-\mathcal{T}_{j,o_k}\beta + {}^t\beta\mathcal{T}_{o_k,o_k}\beta/2$ with respect to β_l, which equals

$$S^{(r)}_l = -\mathcal{T}_{j,l} + \sum_{\substack{v<l \\ v\in o_k}} \mathcal{T}_{l,v}B^{(r+1)}_{j|v} + \mathcal{T}_{l,l}B^{(r)}_{j|l} + \sum_{\substack{v>l \\ v\in o_k}} \mathcal{T}_{l,v}B^{(r)}_{j|v} \tag{8}$$

- $\text{Soft}(z, \lambda) = \begin{cases} z - \lambda & \text{if } z > \lambda \\ z + \lambda & \text{if } z < -\lambda \\ 0 & \text{if } |z| \leq \lambda \end{cases}$, is the standard soft-thresholding operator.

In a sparse setup the soft-thresholding update (7) can be evaluated very quickly as l varies. Often coefficients which are zero remain zero after thresholding and therefore nothing has to be changed in (8). See also the *naive-* or *covariance update* of Friedman et al. (2010) for efficient computation of (7) and (8).

Estimation of $\Sigma_{m_k|o_k}$: We update the residual covariance matrix as:

$$\Sigma^{(r+1)}_{m_k|o_k} = \Big(\mathcal{T}_{m_k,m_k} - \mathcal{T}_{m_k,o_k}{}^t B^{(r+1)}_{m_k|o_k} - B^{(r+1)}_{m_k|o_k}\mathcal{T}_{o_k,m_k} + B^{(r+1)}_{m_k|o_k}\mathcal{T}_{o_k,o_k}{}^t B^{(r+1)}_{m_k|o_k}\Big)/n. \tag{9}$$

Formula (9) can be viewed as a generalized version of Equation (5), when multiplying out the matrix product in (5) and taking conditional expectations.

Our regularized algorithm, MissPALasso, is summarized in Algorithm 2. Note, that we update the sufficient statistic in the partial E-Step according to $\mathcal{T} = \gamma\mathcal{T} + \mathcal{T}^k$ where $\gamma = 1 - |\mathcal{I}_k|/n$. This update, motivated by Nowlan (1991), calculates $\mathcal{T}$ as an exponentially decaying average of recently-visited data points. It prevents MissPALasso from storing $\mathcal{T}^k$ for all $k = 1, \ldots, s$ which gets problematic for large p. As we are mainly interested in estimating the missing values, we will output the data matrix with missing values imputed by the regression coefficients $\hat{B}_{m_k|o_k}$ $(k = 1, \ldots, s)$ as indicated in step (4) of Algorithm 2. MissPALasso provides not only the imputed data matrix $\hat{\mathbf{X}}$ but also $\hat{\mathcal{T}}$, the completed version of the sufficient statistic ${}^t\mathbf{X}\mathbf{X}$. The latter can be very useful if MissPALasso is used as a pre-processing step followed by a learning method which is expressible in terms of the sufficient statistic. Examples include regularized regression (e.g., Lasso), discriminant analysis, or estimation of directed acyclic graphs with the PC-algorithm (Spirtes et al., 2000).

By construction, the regression estimates $\hat{B}_{m_k|o_k}$ are sparse due to the employed ℓ_1-penalty, and therefore imputation of missing values $\hat{\mathbf{X}}^k_{m_k} = \mathbf{X}^k_{o_k}{}^t\hat{B}_{m_k|o_k}$ is based on sparse regressions. This is in sharp contrast to the MissGLasso approach (see Section 4.1) which places sparsity on Σ^{-1}. But this does not imply that regressions of variables in m_k on variables in o_k are sparse since the inverse of sub-matrices of a sparse Σ^{-1} are not sparse in general. MissPALasso employs another type of sparsity and this seems to be the main reason for its better statistical performance than MissGLasso.

In practice, we propose to run MissPALasso for a decreasing sequence of values for λ, using each solution as a warm start for the next problem with smaller λ. This pathwise strategy is computationally very attractive and our algorithm converges (for each λ) after a few cycles.

Algorithm 2: MissPALasso

(1) Set $r = 0$ and start with initial guess for $\mathcal{T}$ and $B^{(0)}_{m_k|o_k}$ $(k = 1, \ldots, s)$.

(2) In cycle $r + 1$; for $k = 1, \ldots, s$ do:

Penalized M-Step2:

For all $j \in m_k$, compute $B^{(r+1)}_{j|o_k}$ by improving $-\mathcal{T}_{j,o_k}\beta + {}^t\beta\mathcal{T}_{o_k,o_k}\beta/2 + \lambda\|\beta\|_1$ in a coordinate-wise manner from $B^{(r)}_{j|o_k}$.

Set $\Sigma^{(r+1)}_{m_k|o_k} = \Big(\mathcal{T}_{m_k,m_k} - \mathcal{T}_{m_k,o_k}\,{}^tB^{(r+1)}_{m_k|o_k} - B^{(r+1)}_{m_k|o_k}\mathcal{T}_{o_k,m_k} + B^{(r+1)}_{m_k|o_k}\mathcal{T}_{o_k,o_k}\,{}^tB^{(r+1)}_{m_k|o_k}\Big)/n.$

Partial E-Step:

Set $\mathcal{T}^k = \mathbb{E}[T^k|\mathbf{X}^k_{o_k}, B^{(r+1)}_{m_k|o_k}, \Sigma^{(r+1)}_{m_k|o_k}]$,

Update $\mathcal{T} = \gamma\mathcal{T} + \mathcal{T}^k$ where $\gamma = 1 - |\mathcal{I}_k|/n$.

Increase: $r \leftarrow r + 1$.

(3) Repeat step (2) until some convergence criterion is met.

(4) Output the imputed data matrix $\hat{\mathbf{X}}$, with missing values estimated by: $\hat{\mathbf{X}}^k_{m_k} = \mathbf{X}^k_{o_k}\,{}^t\hat{B}_{m_k|o_k}$, $k = 1, \ldots, s$.

4. Numerical Experiments

In this section we explore the performance of MissPALasso in recovering missing entries and we report on computational efficiency of the algorithm.

4.1 Performance of MissPALasso

Our new approach is compared with the following imputation methods which are well-suited for the high-dimensional context:

- *KnnImpute.* Impute the missing values by the K-nearest neighbors imputation method introduced by Troyanskaya et al. (2001).
- *SoftImpute.* The soft imputation algorithm is proposed by Mazumder et al. (2010) in order to solve the matrix completion problem. They propose to approximate the incomplete data matrix $\mathbf{X}$ by a complete (low-rank) matrix $\mathbf{Z}$ minimizing

$$\frac{1}{2}\sum_{(i,j)\in\Omega}(z_{ij} - x_{ij})^2 + \lambda\|\mathbf{Z}\|_*.$$

Here, Ω denotes the indices of observed entries and $\|\cdot\|_*$ is the nuclear norm, or the sum of the singular values. The missing values of $\mathbf{X}$ are imputed by the corresponding values of $\mathbf{Z}$.

- *MissGLasso.* Compute $\hat{\Sigma}$ by minimizing $-\ell(\Sigma; \mathbf{X}_{\text{obs}}) + \lambda\|\Sigma^{-1}\|_1$, where $\|\cdot\|_1$ is the entrywise ℓ_1-norm. Then, use this estimate to impute the missing values by conditional mean imputation. MissGLasso is described in Städler and Bühlmann (2012).
- *MissPALasso.* This is the method introduced in Section 3.3.

To assess the performances of the methods we use the normalized root mean squared error (Oba et al., 2003) which is defined by

$$\text{NRMSE} \quad = \quad \sqrt{\frac{\textbf{mean}\left((\mathbf{X}^{\text{true}} - \hat{\mathbf{X}})^2\right)}{\textbf{var}\left(\mathbf{X}^{\text{true}}\right)}}.$$

Here, $\mathbf{X}^{\text{true}}$ is the original data matrix (before deleting values) and $\hat{\mathbf{X}}$ is the imputed matrix. With **mean** and **var** we abbreviate the empirical mean and variance, calculated over only the missing entries.

All methods involve one tuning parameter. In KnnImpute we have to choose the number K of nearest neighbors, while SoftImpute, MissGLasso and MissPALasso involve a regularization parameter which is always denoted by λ. In all of our experiments we select the tuning parameters to obtain optimal prediction of the missing entries in terms of NRMSE.

4.1.1 Simulation Study

We consider both high- and a low-dimensional MVN models with $\sim \mathcal{N}_p(0, \Sigma)$ where

- **Model 1:** $p = 50$ and 500;
  Σ: block diagonal with $p/2$ blocks of the form $\left(\begin{smallmatrix} 1 & 0.9 \\ 0.9 & 1 \end{smallmatrix}\right)$.
- **Model 2:** $p = 100$ and 1000;
  Σ: two blocks B_1, B_2 each of size $\frac{p}{2} \times \frac{p}{2}$ with $\mathrm{B}_1 = \mathrm{I}_{\frac{p}{2}}$ and $(\mathrm{B}_2)_{j,j'} = 0.9^{|j-j'|}$.
- **Model 3:** $p = 55$ and 496;
  Σ: block diagonal with $b = 1, \ldots, 10$ for $p = 55$ and $b = 1, \ldots, 31$ for $p = 496$ (increasing) blocks B_b of the size $b \times b$, with $(\mathrm{B}_b)_{j,j'} = 0.9$ $(j \neq j')$ and $(\mathrm{B}_b)_{j,j} = 1$.
- **Model 4:** $p = 100$ and 500;
  $\Sigma_{j,j'} = 0.9^{|j-j'|}$ for $j, j' = 1, \ldots, p$.

For all four settings we perform 50 independent simulation runs. In each run we generate $n = 50$ i.i.d. samples from the model. We then delete randomly 5%, 10% and 15% of the values in the data matrix, apply an imputation method and compute the NRMSE. The results of the different imputation methods are reported in Table 1 for the low-dimensional models and Table 2 for the high-dimensional models. MissPALasso is very competitive in all setups. SoftImpute works rather poorly, perhaps because the resulting data matrices are not well approximable by low-rank matrices. KnnImpute works very well in model 1 and model 4. Model 1, where each variable is highly correlated with its neighboring variable, represents an example which fits well into the KnnImpute framework. However, in model 2 and model 3, KnnImpute performs rather poorly. The reason is that with an inhomogeneous covariance

matrix, as in model 2 and 3, the optimal number of nearest neighbors is varying among the different blocks, and a single parameter K is too restrictive. For example in model 2, a variable from the first block is not correlated to any other variable, whereas a variable from the second block is correlated to other variables. Except for the low-dimensional model 3 MissGLasso is inferior to MissPALasso. Furthermore, MissPALasso strongly outperforms MissGLasso with respect to computation time (see Figure 4 in Section 4.2). Interestingly, all methods exhibit a quite large NRMSE in the high-dimensional model 3. They seem to have problems coping with the complex covariance structure in higher dimensions. If we look at the same model but with $p = 105$ the NRMSE for 5% missing values is: 0.85 for KnnImpute, 0.86 for SoftImpute, 0.77 for MissGLasso and 0.77 for MissPALasso. This indicates an increase in NRMSE according to the size of p. Arguably, we consider here only multivariate normal models which are ideal, from a distributional point of view, for MissGLasso and our MissPALasso. The more interesting case will be with real data (all from genomics) where model assumptions never hold exactly.

		KnnImpute	SoftImpute	MissGLasso	MissPALasso
Model 1	5%	0.4874 (0.0068)	0.7139 (0.0051)	0.5391 (0.0079)	0.5014 (0.0070)
p=50	10%	0.5227 (0.0051)	0.7447 (0.0038)	0.5866 (0.0057)	0.5392 (0.0055)
	15%	0.5577 (0.0052)	0.7813 (0.0037)	0.6316 (0.0048)	0.5761 (0.0047)
Model 2	5%	0.8395 (0.0101)	0.8301 (0.0076)	0.7960 (0.0082)	0.7786 (0.0075)
p=100	10%	0.8572 (0.0070)	0.8424 (0.0063)	0.8022 (0.0071)	0.7828 (0.0066)
	15%	0.8708 (0.0062)	0.8514 (0.0053)	0.8082 (0.0058)	0.7900 (0.0054)
Model 3	5%	0.4391 (0.0061)	0.4724 (0.0050)	0.3976 (0.0056)	0.4112 (0.0058)
p=55	10%	0.4543 (0.0057)	0.4856 (0.0042)	0.4069 (0.0047)	0.4155 (0.0047)
	15%	0.4624 (0.0054)	0.4986 (0.0036)	0.4131 (0.0043)	0.4182 (0.0044)
Model 4	5%	0.3505 (0.0037)	0.5515 (0.0039)	0.3829 (0.0035)	0.3666 (0.0031)
p=100	10%	0.3717 (0.0033)	0.5623 (0.0033)	0.3936 (0.0027)	0.3724 (0.0026)
	15%	0.3935 (0.0032)	0.5800 (0.0031)	0.4075 (0.0026)	0.3827 (0.0026)

Table 1: Average (SE) NRMSE of KnnImpute, SoftImpute, MissGLasso and MissPALasso with different degrees of missingness in the low-dimensional models.

4.1.2 Real Data Examples

We consider the following four publicly available data sets:

- **Isoprenoid gene network in Arabidopsis thaliana:** The number of genes in the network is $p = 39$. The number of observations (gene expression profiles), corresponding to different experimental conditions, is $n = 118$. More details about the data can be found in Wille et al. (2004).

- **Colon cancer:** In this data set, expression levels of 40 tumor and 22 normal colon tissues ($n = 62$) for $p = 2000$ human genes are measured. For more information see Alon et al. (1999).

		KnnImpute	SoftImpute	MissGLasso	MissPALasso
Model 1	5%	0.4913 (0.0027)	0.9838 (0.0006)	0.6705 (0.0036)	0.5301 (0.0024)
p=500	10%	0.5335 (0.0020)	0.9851 (0.0005)	0.7613 (0.0031)	0.5779 (0.0019)
	15%	0.5681 (0.0016)	0.9870 (0.0004)	0.7781 (0.0013)	0.6200 (0.0015)
Model 2	5%	0.8356 (0.0020)	0.9518 (0.0009)	0.8018 (0.0012)	0.7958 (0.0017)
p=1000	10%	0.8376 (0.0016)	0.9537 (0.0007)	0.8061 (0.0002)	0.7990 (0.0013)
	15%	0.8405 (0.0014)	0.9562 (0.0006)	0.8494 (0.0080)	0.8035 (0.0011)
Model 3	5%	1.0018 (0.0009)	0.9943 (0.0005)	0.9722 (0.0013)	0.9663 (0.0010)
p=496	10%	1.0028 (0.0007)	0.9948 (0.0004)	0.9776 (0.0010)	0.9680 (0.0007)
	15%	1.0036 (0.0006)	0.9948 (0.0003)	0.9834 (0.0010)	0.9691 (0.0007)
Model 4	5%	0.3487 (0.0016)	0.7839 (0.0020)	0.4075 (0.0016)	0.4011 (0.0016)
p=500	10%	0.3721 (0.0014)	0.7929 (0.0015)	0.4211 (0.0012)	0.4139 (0.0013)
	15%	0.3960 (0.0011)	0.8045 (0.0014)	0.4369 (0.0012)	0.4292 (0.0014)

Table 2: Average (SE) NRMSE of KnnImpute, SoftImpute, MissGLasso and MissPALasso with different degrees of missingness in the high-dimensional models.

- **Lymphoma:** This data set, presented in Alizadeh et al. (2000), contains gene expression levels of 42 samples of diffuse large B-cell lymphoma, 9 observations of follicular lymphoma, and 11 cases of chronic lymphocytic leukemia. The total sample size is $n = 62$ and $p = 1332$ complete measured expression profiles are documented.

- **Yeast cell-cycle:** The data set, described in Spellman et al. (1998), monitors expressions of 6178 genes. The data consists of four parts, which are relevant to alpha factor (18 samples), elutriation (14 samples), *cdc15* (24 samples), and *cdc28* (17 samples). The total sample size is $n = 73$. We use the $p = 573$ complete profiles in our study.

For all data sets we standardize the columns (genes) to zero mean and variance one. In order to compare the performance of the different imputation methods we randomly delete values to obtain an overall missing rate of 5%, 10% and 15%. Table 3 shows the results for 50 simulation runs, where in each run another random set of values is deleted.

MissPALasso exhibits in all setups the lowest averaged NRMSE. MissGLasso performs nearly as well as MissPALasso on the Arabidopsis data. However, its R implementation cannot cope with large values of p. If we were to restrict our analysis to the 100 variables exhibiting the most variance we would see that MissGLasso performs slightly less well than MissPALasso (results not included). Compared to KnnImpute, SoftImpute works well for all data sets. Interestingly, for all data sets, KnnImpute performance was very inferior compared to MissPALasso. In light of the simulation results of Section 4.1.1, a reason for the poor performance could be that KnnImpute has difficulties with the inhomogeneous correlation structure between different genes which plausibly could be present in real data sets.

To investigate the effect of already missing values on the imputation performance of the compared methods we use the original lymphoma and yeast cell-cycle data sets which already have "real" missing values. We only consider the 100 most variable genes in these

		KnnImpute	SoftImpute	MissGLasso	MissPALasso
Arabidopsis	5%	0.7732 (0.0086)	0.7076 (0.0065)	0.7107 (0.0076)	0.7029 (0.0077)
n=118	10%	0.7723 (0.0073)	0.7222 (0.0052)	0.7237 (0.0064)	0.7158 (0.0060)
p=39	15%	0.7918 (0.0050)	0.7369 (0.0041)	0.7415 (0.0053)	0.7337 (0.0050)
Colon cancer	5%	0.4884 (0.0011)	0.4921 (0.0011)	-	0.4490 (0.0011)
n=62	10%	0.4948 (0.0008)	0.4973 (0.0006)	-	0.4510 (0.0006)
p=2000	15%	0.5015 (0.0007)	0.5067 (0.0006)	-	0.4562 (0.0007)
Lymphoma	5%	0.7357 (0.0014)	0.6969 (0.0008)	-	0.6247 (0.0012)
n=62	10%	0.7418 (0.0009)	0.7100 (0.0006)	-	0.6384 (0.0009)
p=1332	15%	0.7480 (0.0007)	0.7192 (0.0005)	-	0.6525 (0.0008)
Yeast cell-cycle	5%	0.8083 (0.0018)	0.6969 (0.0012)	-	0.6582 (0.0016)
n=73	10%	0.8156 (0.0011)	0.7265 (0.0010)	-	0.7057 (0.0013)
p=573	15%	0.8240 (0.0009)	0.7488 (0.0007)	-	0.7499 (0.0011)

Table 3: Average (SE) NRMSE of KnnImpute, SoftImpute, MissGLasso and MissPALasso for different real data sets from genomics. The R implementation of MissGLasso is not able to handle real data sets of such high dimensionality.

data sets to be able to compare all four methods with each other. From the left panel of Figures 1 and 2 we can read off how many values are missing for each of the 100 variables. In the right panel of Figures 1 and 2 we show how well the different methods are able to estimate $2\%, 4\%, 6\% \ldots, 16\%$ of additionally deleted entries.

4.2 Computational Efficiency

We first compare the computational efficiency of MissPA (Algorithm 1) with the standard EM for missing values described for example in Schafer (1997). A key attribute of MissPA is that the computational cost of one cycle through all patterns is the same as the cost of a single standard EM-iteration. The reason why our algorithm takes less time to converge is that the latent distribution is updated much more frequently. We emphasize the big contrast of MissPA to the incremental EM, mostly applied to finite mixtures (Thiesson et al., 2001; Ng and McLachlan, 2003), where there is a trade-off between the additional computation time per cycle, or "scan" in the language of Ng and McLachlan (2003), and the fewer number of "scans" required because of the more frequent updating after each partial E-Step. The speed of convergence of the standard EM and MissPA for three data sets are shown in Figure 3, in which the log-likelihood is plotted as a function of the number of iterations (cycles). The left panel corresponds to the subset of the lymphoma data set when only the ten genes with highest missing rate are used. This results in a 62×10 data matrix with 22.85% missing values. For the middle panel we draw a random sample of size 62×10 from $\mathcal{N}_{10}(0, \Sigma)$, $\Sigma_{j,j'} = 0.9^{|j-j'|}$, and delete the same entries which are missing in the reduced lymphoma data. For the right panel we draw from the multivariate t-model with one degree of freedom and again with the same values deleted. As can be seen, MissPA converges after fewer cycles. A very extreme example is obtained with the multivariate t-model where the standard EM reaches the log-likelihood level of MissPA about 400 iterations later. We note

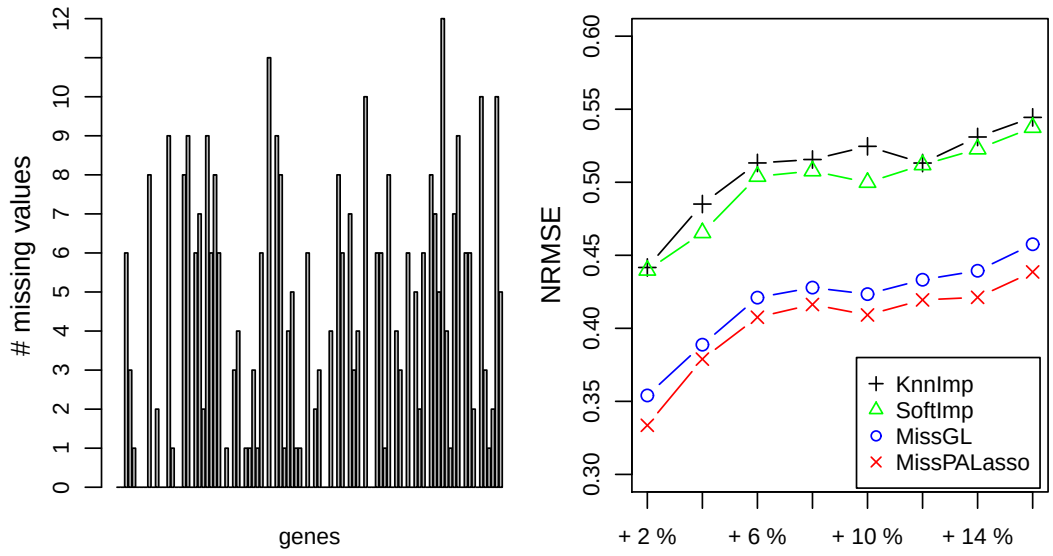


Figure 1: Lymphoma data set. Left panel: Barplots which count the number of missing values for each of the 100 genes. Right panel: NRMSE for KnnImpute, SoftImpute, MissGLasso and MissPALasso if we introduce additional $2\%, 4\%, 6\%, \ldots, 16\%$ missing values.

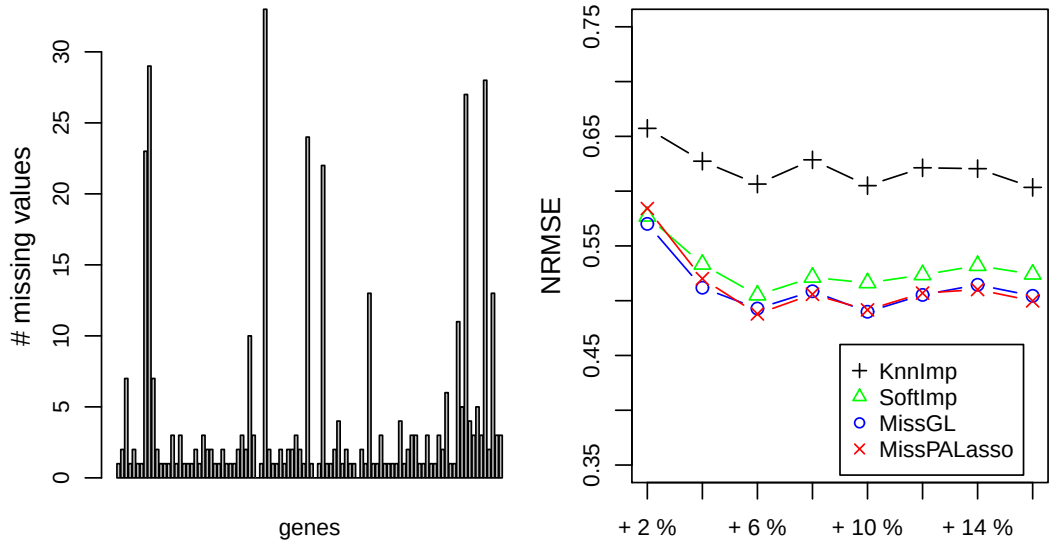


Figure 2: Yeast cell-cycle data set. Left panel: Barplots which count the number of missing values for each of the 100 genes. Right panel: NRMSE for KnnImpute, SoftImpute, MissGLasso and MissPALasso if we introduce additional $2\%, 4\%, 6\%, \ldots, 16\%$ missing values.

here, that the results shown in the middle and right panels highly depend on the realized random sample. With other realizations, we get less and more extreme results than the one shown in Figure 3.

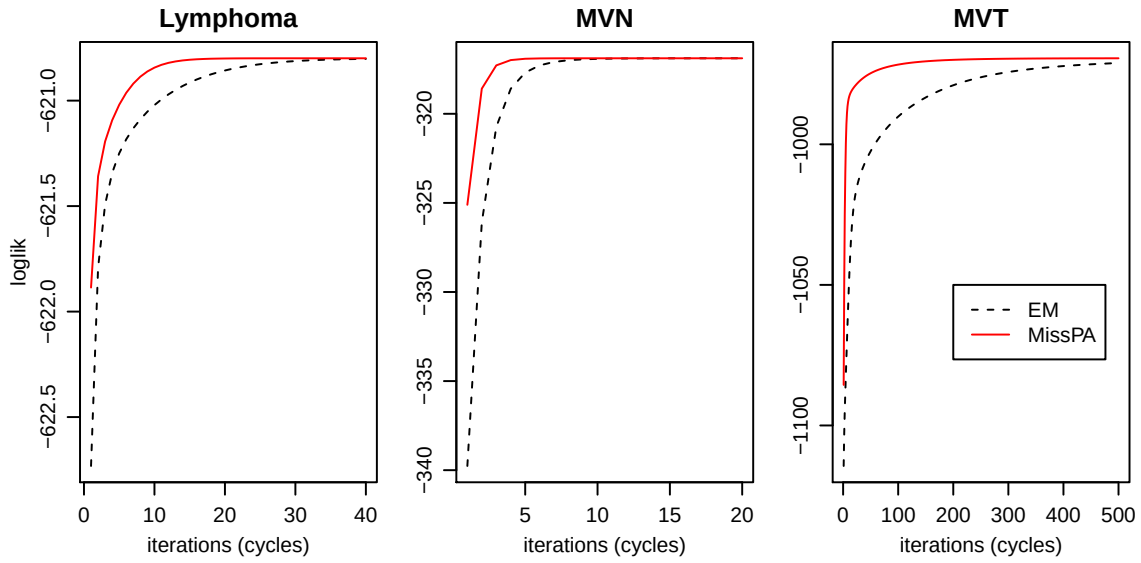


Figure 3: Log-likelihood as a function of the number of iterations (cycles) for standard EM and MissPA. Left panel: subset of the lymphoma data ($n = 62$, $p = 10$ and 22.85% missing values). Middle panel: random sample of size 62×10 from the multivariate normal model with the same missing entries as in the reduced lymphoma data. Right panel: random sample of the size 62×10 from the multivariate t-model again with the same missing values.

We end this section by illustrating the computational timings of MissPALasso and MissGLasso implemented with the statistical computing language R. We consider two settings. Firstly, model 4 of Section 4.1.1 with $n = 50$ and a growing number of variables p ranging from 10 to 500. Secondly, the colon cancer data set from Section 4.1.2 with $n = 62$ and also a growing number of variables where we sorted the variables according to the empirical variance. For each p we delete 10% of the data, run MissPALasso and MissGLasso ten times on a decreasing grid (on the log-scale) of λ values with thirty grid points. For a fixed λ we stop the algorithm if the relative change in imputation satisfies,

$$\frac{\|\hat{\mathbf{X}}^{(r+1)} - \hat{\mathbf{X}}^{(r)}\|^2}{\|\hat{\mathbf{X}}^{(r+1)}\|^2} \leq 10^{-5}.$$

In Figure 4 the CPU times in seconds are plotted for various values of p in the two settings. As shown, with MissPALasso we are typically able to solve a problem of size $p = 100$ in about 9 seconds and a problem of size $p = 500$ in about 400 seconds. For MissGLasso these times are highly increased to 27 and 4300 seconds respectively. Furthermore, we can see that MissPALasso has much smaller variability in runtimes. The computational complexity of MissGLasso is $O\big(p^3 + \sum_{k=1}^{s}(\max\{|m_k|, |o_k|\}|m_k|^2) + np^2\big)$: the

graphical Lasso algorithm costs $O(p^3)$, calculating the coefficients needed in the E-Step involves $O(\sum_{k=1}^{s} \max\{|m_k|, |o_k|\}|m_k|^2)$ operations and updating the sufficient statistic costs $O(np^2)$. In contrast, in a sparse setting, the complexity of MissPALasso is considerably smaller: MissPALasso costs $O\big(\sum_{k=1}^{s} \big(\max\{|m_k|, |o_k|\} \sum_{j \in m_k} q_j\big) + np^2\big)$ operations where q_j denotes the average number of nonzero elements in $B^{(r)}_{j|o_k}$.

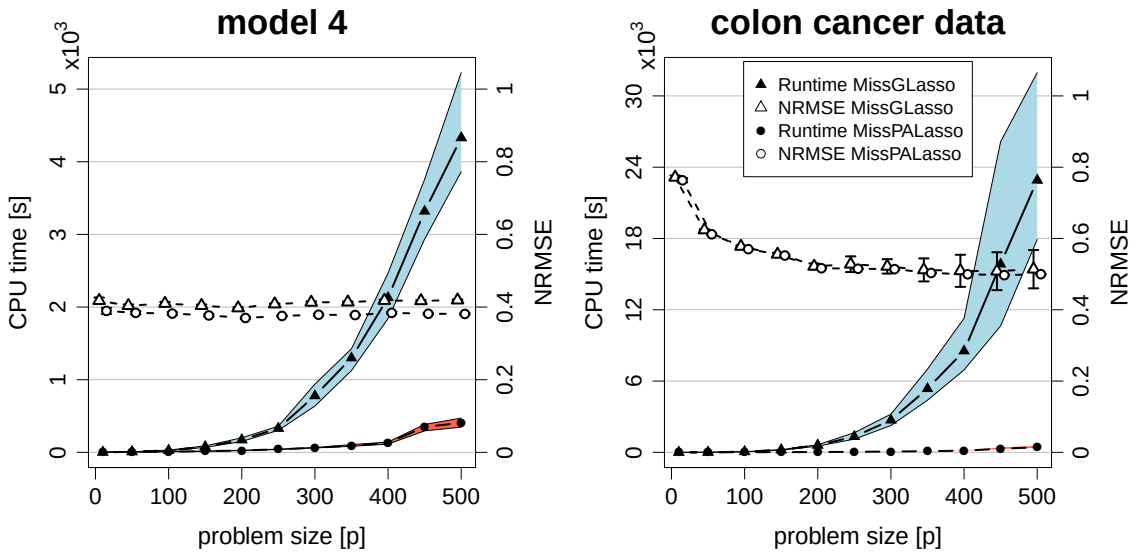


Figure 4: CPU times (filled points, left axis) and NRMSE (hollow points, right axis) vs. problem size p of MissPALasso (circles) and MissGLasso (triangles) in simulation model 4 (left panel) and the colon cancer data (right panel). MissPALasso and MissGLasso are applied on a grid of thirty λ values. The shaded area shows the full range of CPU times over 10 simulation runs. Measurements of NRMSE include standard error bars which are due to their small size ($\sim 10^{-3}$) mostly not visible except for MissGLasso in the real data example.

5. Theory

A key characteristic of pattern alternating maximization (MissPA, Algorithm 1 in Section 3.2) is that the E-Step is only performed on those samples belonging to a single pattern. We already mentioned the close connection to the incremental EM introduced by Neal and Hinton (1998). In fact, if the density of $\mathbf{X}^k$, $k \in \{1, \ldots, s\}$, is denoted by $\mathrm{P}_\Sigma(\mathbf{X}^k) = \prod_{i \in \mathcal{I}_k} p(x_i; \Sigma)$, then the negative variational free energy (Neal and Hinton, 1998; Jordan et al., 1999) equals

$$\mathcal{F}[\Sigma \| \Psi_1, \ldots, \Psi_s] \quad = \quad \sum_{k=1}^{s} \big(\mathbb{E}_{\Psi_k}[\log \mathrm{P}_\Sigma(\mathbf{X}^k) | \mathbf{X}^k_{o_k}] + \mathcal{H}_k[\Psi_k]\big). \tag{10}$$

Here, $\Psi_k = \left(B_{k,m_k|o_k}, \Sigma_{k,m_k|o_k}\right)$ denotes the regression parameter of the latent distribution

$$\mathrm{P}_{\Psi_k}(\mathbf{X}^k_{m_k}|\mathbf{X}^k_{o_k}) \;=\; \prod_{i\in\mathcal{I}_k} p(x_{i,m_k}|x_{i,o_k}; B_{k,m_k|o_k}, \Sigma_{k,m_k|o_k})$$

and $\mathcal{H}_k[\Psi_k] = -\mathbb{E}_{\Psi_k}[\log \mathrm{P}_{\Psi_k}(\mathbf{X}^k_{m_k}|\mathbf{X}^k_{o_k})|\mathbf{X}^k_{o_k}]$ is the entropy. An iterative procedure alternating between maximization of $\mathcal{F}$ with respect to Σ,

$$\begin{aligned}\hat{\Sigma} &= \arg\max_{\Sigma} \mathcal{F}[\Sigma\|\Psi_1,\ldots,\Psi_s] \\ &= \frac{1}{n}\sum_{k=1}^{s} \mathbb{E}_{\Psi_k}[{}^t\mathbf{X}^k\mathbf{X}^k|\mathbf{X}^k_{o_k}] =: \frac{1}{n}\mathcal{T},\end{aligned}$$

and maximizing $\mathcal{F}$ with respect to Ψ_k,

$$\begin{aligned}(\hat{B}_{k,m_k|o_k}, \hat{\Sigma}_{k,m_k|o_k}) &= \arg\max_{\Psi_k} \mathcal{F}[\hat{\Sigma}\|\Psi_1,\ldots,\Psi_s] \\ &= \arg\max_{\Psi_k} \mathbb{E}_{\Psi_k}[\log \mathrm{P}_{\hat{\Sigma}}(\mathbf{X}^k)|\mathbf{X}^k_{o_k}] + \mathcal{H}_k[\Psi_k] \\ &= \left(\mathcal{T}_{m_k,o_k}\mathcal{T}^{-1}_{o_k,o_k}, \frac{1}{n}(\mathcal{T}_{m_k,m_k} - \mathcal{T}_{m_k,o_k}\mathcal{T}^{-1}_{o_k,o_k}\mathcal{T}_{o_k,m_k})\right),\end{aligned}$$

is equivalent to Algorithm 1 with $\mathcal{T}^{-k}$ replaced by $\mathcal{T}$ (see M-Step2 in Section 3.2). Alternating maximization of (10) is a GAM procedure in the sense of Gunawardana and Byrne (2005) for which convergence to a stationary point of the observed log-likelihood can be established easily.

Unfortunately, MissPA does not quite fit into the GAM formulation as it extends the standard EM in an additional manner, namely by using for each pattern a different complete data space (for each pattern k only those samples are augmented which do not belong to pattern k). From this point of view MissPA is related to the SAGE procedure (Fessler and Hero, 1994). To see this, consider Σ in the parameterization $\theta = \left(\Sigma_{o_k}, B_{m_k|o_k}, \Sigma_{m_k|o_k}\right)$ introduced in Section 2. From

$$\mathrm{P}_\theta(\mathbf{X}_{\mathrm{obs}}, \mathbf{X}^{-k}) = \mathrm{P}_\theta(\mathbf{X}_{\mathrm{obs}}|\mathbf{X}^{-k})\mathrm{P}_\theta(\mathbf{X}^{-k})$$

and observing that $\mathrm{P}_\theta(\mathbf{X}_{\mathrm{obs}}|\mathbf{X}^{-k}) = \mathrm{P}_{\Sigma_{o_k}}(\mathbf{X}_{o_k})$ we conclude that $\mathbf{X}^{-k}$ is an admissible hidden-data space with respect to $(B_{m_k|o_k}, \Sigma_{m_k|o_k})$ in the sense of Fessler and Hero (1994). The M-Step of MissPA then maximizes a conditional expectation of the log-likelihood $\log \mathrm{P}_\theta(\mathbf{X}^{-k})$ with respect to the parameters $(B_{m_k|o_k}, \Sigma_{m_k|o_k})$. Different from SAGE is the conditional distribution involved in the expectation: after each M-Step, our algorithm updates only the conditional distribution for a single pattern, consequently we do not need to compute estimates for Σ_{o_k}.

In summary, MissPA has similarities with GAM and SAGE. However, neither of these frameworks fit our purpose. In the next section we provide theory which justifies alternating between complete data spaces *and* incrementally performing the E-Step. In particular, we prove convergence to a stationary point of the observed log-likelihood.

5.1 Convergence Analysis of Missingness Pattern Alternating Maximization

In this section we study the numerical properties of MissPA.

5.1.1 Pattern-Depending Lower Bounds

Denote the density of $\mathbf{X}^k$, $k \in \{1,\ldots,s\}$, by $\mathrm{P}_\Sigma(\mathbf{X}^k) = \prod_{i\in\mathcal{I}_k} p(x_i;\Sigma)$ and define for $k,l \in \{1,\ldots,s\}$

$$\begin{aligned}\mathrm{P}_\Sigma(\mathbf{X}^l_{o_k}) &= \prod_{i\in\mathcal{I}_l} p(x_{i,o_k};\Sigma_{o_k}) \quad \text{and}\\ \mathrm{P}_\Sigma(\mathbf{X}^l_{m_k}|\mathbf{X}^l_{o_k}) &= \prod_{i\in\mathcal{I}_l} p(x_{i,m_k}|x_{i,o_k};B_{m_k|o_k},\Sigma_{m_k|o_k}).\end{aligned}$$

Set $\{\Sigma_l\}_{l\neq k} = (\Sigma_1,\ldots,\Sigma_{k-1},\Sigma_{k+1},\ldots,\Sigma_s)$ and consider for $k = 1,\ldots,s$

$$\mathcal{F}_k[\Sigma_k||\{\Sigma_l\}_{l\neq k}] = \log \mathrm{P}_{\Sigma_k}(\mathbf{X}^k_{o_k}) + \sum_{l\neq k}\big(\mathbb{E}_{\Sigma_l}[\log \mathrm{P}_{\Sigma_k}(\mathbf{X}^l)|\mathbf{X}^l_{o_l}] + \mathcal{H}_l[\Sigma_l]\big).$$

Here $\mathcal{H}_l[\tilde\Sigma] = -\mathbb{E}_{\tilde\Sigma}[\log \mathrm{P}_{\tilde\Sigma}(\mathbf{X}^l_{m_l}|\mathbf{X}^l_{o_l})|\mathbf{X}^l_{o_l}]$ denotes the entropy. Note that $\mathcal{F}_k$ is defined for fixed observed data $\mathbf{X}_{\mathrm{obs}}$. The subscript k highlights the dependence on the pattern k. Furthermore, for fixed $\mathbf{X}_{\mathrm{obs}}$ and fixed k, $\mathcal{F}_k$ is a function in the parameters $(\Sigma_1,\ldots,\Sigma_s)$. As a further tool we write the Kullback-Leibler divergence in the following form:

$$\mathcal{D}_l[\tilde\Sigma||\Sigma] = \mathbb{E}_{\tilde\Sigma}[-\log\big(\mathrm{P}_\Sigma(\mathbf{X}^l_{m_l}|\mathbf{X}^l_{o_l})/\mathrm{P}_{\tilde\Sigma}(\mathbf{X}^l_{m_l}|\mathbf{X}^l_{o_l})\big)|\mathbf{X}^l_{o_l}]. \tag{11}$$

An important property of the Kullback-Leibler divergence is its non-negativity:

$$\begin{aligned}&\mathcal{D}_l[\tilde\Sigma||\Sigma] \geq 0, \quad \text{with equality if and only if}\\ &\mathrm{P}_{\tilde\Sigma}(\mathbf{X}^l_{m_l}|\mathbf{X}^l_{o_l}) = \mathrm{P}_\Sigma(\mathbf{X}^l_{m_l}|\mathbf{X}^l_{o_l}).\end{aligned}$$

A simple calculation shows that

$$\mathbb{E}_{\tilde\Sigma}[\log \mathrm{P}_\Sigma(\mathbf{X}^l)|\mathbf{X}^l_{o_l}] + \mathcal{H}_l[\tilde\Sigma] = -\mathcal{D}_l[\tilde\Sigma||\Sigma] + \log \mathrm{P}_\Sigma(\mathbf{X}^l_{o_l}) \tag{12}$$

and that $\mathcal{F}_k[\Sigma_k||\{\Sigma_l\}_{l\neq k}]$ can be written as

$$\mathcal{F}_k[\Sigma_k||\{\Sigma_l\}_{l\neq k}] = \ell(\Sigma_k;\mathbf{X}_{\mathrm{obs}}) - \sum_{l\neq k}\mathcal{D}_l[\Sigma_l||\Sigma_k]. \tag{13}$$

In particular, for fixed values of $\{\Sigma_l\}_{l\neq k}$, $\mathcal{F}_k[\,\cdot\,||\{\Sigma_l\}_{l\neq k}]$ lower bounds the observed log-likelihood $\ell(\,\cdot\,;\mathbf{X}_{\mathrm{obs}})$ due to the non-negativity of the Kullback-Leibler divergence.

5.1.2 Optimization Transfer to Pattern-Depending Lower Bounds

We give now an alternative description of the MissPA algorithm. In cycle $r+1$ through all patterns, generate $(\Sigma_1^{(r+1)},\ldots,\Sigma_s^{(r+1)})$ given $(\Sigma_1^{(r)},\ldots,\Sigma_s^{(r)})$ according to

$$\Sigma_k^{(r+1)} = \arg\max_\Sigma \mathcal{F}_k[\Sigma||\mathrm{Z}_k^{(r+1)}], \quad k = 1,\ldots,s, \tag{14}$$

with $\mathrm{Z}_k^{(r+1)} = (\Sigma_1^{(r+1)}, \ldots, \Sigma_{k-1}^{(r+1)}, \Sigma_{k+1}^{(r)}, \ldots, \Sigma_s^{(r)})$.

We have

$$\begin{aligned}\mathcal{F}_k[\Sigma||\mathrm{Z}_k^{(r+1)}] &= \log \mathrm{P}_\Sigma(\mathbf{X}_{o_k}^k) + \sum_{l<k}\left(\mathbb{E}_{\Sigma_l^{(r+1)}}[\log \mathrm{P}_\Sigma(\mathbf{X}^l)|\mathbf{X}_{o_l}^l] + \mathcal{H}_l[\Sigma_l^{(r+1)}]\right) \\ &\quad + \sum_{l>k}\left(\mathbb{E}_{\Sigma_l^{(r)}}[\log \mathrm{P}_\Sigma(\mathbf{X}^l)|\mathbf{X}_{o_l}^l] + \mathcal{H}_l[\Sigma_l^{(r)}]\right).\end{aligned}$$

The entropy terms do not depend on the optimization parameter Σ, therefore,

$$\begin{aligned}\mathcal{F}_k[\Sigma||\mathrm{Z}_k^{(r+1)}] &= \text{const} + \log \mathrm{P}_\Sigma(\mathbf{X}_{o_k}^k) + \sum_{l<k}\mathbb{E}_{\Sigma_l^{(r+1)}}[\log \mathrm{P}_\Sigma(\mathbf{X}^l)|\mathbf{X}_{o_l}^l] \\ &\quad + \sum_{l>k}\mathbb{E}_{\Sigma_l^{(r)}}[\log \mathrm{P}_\Sigma(\mathbf{X}^l)|\mathbf{X}_{o_l}^l].\end{aligned}$$

Using the factorization $\log \mathrm{P}_\Sigma(\mathbf{X}^l) = \log \mathrm{P}(\mathbf{X}_{o_k}^l; \Sigma_{o_k}) + \log \mathrm{P}(\mathbf{X}_{m_k}^l|\mathbf{X}_{o_k}^l; B_{m_k|o_k}, \Sigma_{m_k|o_k})$ (for all $l \neq k$), and separate maximization with respect to Σ_{o_k} and $(B_{m_k|o_k}, \Sigma_{m_k|o_k})$ we end up with the expressions from the M-Step of MissPA. Summarizing the above, we have recovered the M-Step as a maximization of $\mathcal{F}_k[\Sigma||\mathrm{Z}_k^{(r+1)}]$ which is a lower bound of the observed log-likelihood. Or in the language of Lange et al. (2000), optimization of $\ell(\,\cdot\,; \mathbf{X}_{\text{obs}})$ is transferred to the surrogate objective $\mathcal{F}_k[\,\cdot\,||\mathrm{Z}_k^{(r+1)}]$.

There is still an important piece missing: In M-Step k of cycle $r+1$ we are maximizing $\mathcal{F}_k[\,\cdot\,||\mathrm{Z}_k^{(r+1)}]$ whereas in the following M-Step $(k+1)$ we optimize $\mathcal{F}_{k+1}[\,\cdot\,||\mathrm{Z}_{k+1}^{(r+1)}]$. In order for the algorithm to make progress, it is essential that $\mathcal{F}_{k+1}[\,\cdot\,||\mathrm{Z}_{k+1}^{(r+1)}]$ attains higher values than its predecessor $\mathcal{F}_k[\,\cdot\,||\mathrm{Z}_k^{(r+1)}]$. In this sense the following proposition is crucial.

Proposition 1 *For $r = 0, 1, 2, \ldots$ we have that*

$$\begin{aligned}&\mathcal{F}_s[\Sigma_s^{(r)}||\mathrm{Z}_s^{(r)}] \leq \mathcal{F}_1[\Sigma_s^{(r)}||\mathrm{Z}_1^{(r+1)}], \quad \textit{and} \\ &\mathcal{F}_k[\Sigma_k^{(r+1)}||\mathrm{Z}_k^{(r+1)}] \leq \mathcal{F}_{k+1}[\Sigma_k^{(r+1)}||\mathrm{Z}_{k+1}^{(r+1)}] \quad \textit{for } k = 1, \ldots, s-1.\end{aligned}$$

Proof. We have,

$$\mathcal{F}_k[\Sigma_k^{(r+1)}||\mathrm{Z}_k^{(r+1)}] = \log \mathrm{P}_{\Sigma_k^{(r+1)}}(\mathbf{X}_{o_k}^k) + \mathbb{E}_{\Sigma_{k+1}^{(r)}}[\log \mathrm{P}_{\Sigma_k^{(r+1)}}(\mathbf{X}^{k+1})|\mathbf{X}_{o_{k+1}}^{k+1}] + \mathcal{H}_{k+1}[\Sigma_{k+1}^{(r)}] + \mathrm{A}$$

and

$$\mathcal{F}_{k+1}[\Sigma_k^{(r+1)}||\mathrm{Z}_{k+1}^{(r+1)}] = \log \mathrm{P}_{\Sigma_k^{(r+1)}}(\mathbf{X}_{o_{k+1}}^{k+1}) + \mathbb{E}_{\Sigma_k^{(r+1)}}[\log \mathrm{P}_{\Sigma_k^{(r+1)}}(\mathbf{X}^k)|\mathbf{X}_{o_k}^k] + \mathcal{H}_k[\Sigma_k^{(r+1)}] + \mathrm{A}$$

where

$$\mathrm{A} = \sum_{l<k}\mathbb{E}_{\Sigma_l^{(r+1)}}[\log \mathrm{P}_{\Sigma_k^{(r+1)}}(\mathbf{X}^l)|\mathbf{X}_{o_l}^l] + \mathcal{H}_l[\Sigma_l^{(r+1)}] + \sum_{l>k+1}\mathbb{E}_{\Sigma_l^{(r)}}[\log \mathrm{P}_{\Sigma_k^{(r+1)}}(\mathbf{X}^l)|\mathbf{X}_{o_l}^l] + \mathcal{H}_l[\Sigma_l^{(r)}].$$

Furthermore, using (12) and noting that $\mathcal{D}_k[\Sigma_k^{(r+1)}||\Sigma_k^{(r+1)}] = 0$, we obtain

$$\mathcal{F}_k[\Sigma_k^{(r+1)}||\mathrm{Z}_k^{(r+1)}] - \mathcal{F}_{k+1}[\Sigma_k^{(r+1)}||\mathrm{Z}_{k+1}^{(r+1)}] = \mathcal{D}_k[\Sigma_k^{(r+1)}||\Sigma_k^{(r+1)}] - \mathcal{D}_{k+1}[\Sigma_{k+1}^{(r)}||\Sigma_k^{(r+1)}]$$
$$= -\mathcal{D}_{k+1}[\Sigma_{k+1}^{(r)}||\Sigma_k^{(r+1)}] \leq 0.$$

Note that equality holds if and only if $\mathrm{P}_{\Sigma_k^{(r+1)}}(\mathbf{X}_{m_{k+1}}^{k+1}|\mathbf{X}_{o_{k+1}}^{k+1}) = \mathrm{P}_{\Sigma_{k+1}^{(r)}}(\mathbf{X}_{m_{k+1}}^{k+1}|\mathbf{X}_{o_{k+1}}^{k+1})$. ■

In light of Proposition 1 it is clear that (14) generates a monotonically increasing sequence of the form:

$$\mathcal{F}_s[\Sigma_s^{(0)}||\mathrm{Z}_s^{(0)}] \leq \mathcal{F}_1[\Sigma_s^{(0)}||\mathrm{Z}_1^{(1)}] \leq \mathcal{F}_1[\Sigma_1^{(1)}||\mathrm{Z}_1^{(1)}] \leq \mathcal{F}_2[\Sigma_1^{(1)}||\mathrm{Z}_2^{(1)}] \leq \mathcal{F}_2[\Sigma_2^{(1)}||\mathrm{Z}_2^{(1)}] \leq \cdots$$
$$\cdots \leq \mathcal{F}_k[\Sigma_k^{(r+1)}||\mathrm{Z}_k^{(r+1)}] \leq \mathcal{F}_{k+1}[\Sigma_k^{(r+1)}||\mathrm{Z}_{k+1}^{(r+1)}] \leq \mathcal{F}_{k+1}[\Sigma_{k+1}^{(r+1)}||\mathrm{Z}_{k+1}^{(r+1)}] \leq \cdots$$

For example, we can deduce that $\{\mathcal{F}_s[\Sigma_s^{(r)}||\mathrm{Z}_s^{(r)}]\}_{r=0,1,2,\ldots}$ is a monotone increasing sequence in r.

5.1.3 Convergence to Stationary Points

Ideally we would like to show that a limit point of the sequence generated by MissPA is a global maximum of $\ell(\cdot;\mathbf{X}_{\mathrm{obs}})$. Unfortunately, this is too ambitious because for general missing data patterns the observed log-likelihood is a non-concave function with several local maxima. Thus, the most we can expect is that our algorithm converges to a stationary point. This is ensured by the following theorem which we prove in the Appendix.

Theorem 2 *Assume that* $\mathcal{K} = \{(\Sigma_1, \ldots, \Sigma_s) : \mathcal{F}_s[\Sigma_s||\Sigma_1, \ldots, \Sigma_{s-1}] \geq \mathcal{F}_s[\Sigma_s^{(0)}||\mathrm{Z}_s^{(0)}]\}$ *is compact. Then every limit point* $\bar{\Sigma}_s$ *of* $\{\Sigma_s^{(r)}\}_{r=0,1,2,\ldots}$ *is a stationary point of* $\ell(\,\cdot\,;\mathbf{X}_{\mathrm{obs}})$.

6. Discussion and Extensions

We presented a novel methodology for maximizing the observed log-likelihood for a multivariate normal data matrix with missing values. Simplified, our algorithm iteratively cycles through the different missingness patterns, performs multivariate regressions of the missing on the observed variables and uses the regression coefficients for partial imputation of the missing values. We argued theoretically and gave numerical examples showing that our procedure is computationally more efficient than the standard EM algorithm. Furthermore, we analyzed the numerical properties using non-standard arguments and proved that solutions of our algorithm converge to stationary points of the observed log-likelihood.

In a high-dimensional setup regularization is achieved by replacing least squares regressions with Lasso analogues. Our proposed algorithm, MissPALasso, is built upon coordinate descent approximation of the corresponding Lasso problem in order to gain speed. On simulated and four real data sets (all from genomics) we demonstrated that MissPALasso outperforms other imputation techniques such as k-nearest neighbors imputation, nuclear norm minimization or a penalized likelihood approach with an ℓ_1-penalty on the inverse covariance matrix.

MissPALasso is a "heuristic" motivated by the aim of having sparse regression coefficients for imputation. It is unclear which objective function is optimized by MissPALasso. The comments of two referees on this point made us think of another way of imposing sparsity in the regression coefficients: Consider the penalized variational free energy

$$-\mathcal{F}[\Sigma\|\Psi_1,\ldots,\Psi_s] + \mathrm{Pen}(\Sigma,\Psi_1,\ldots,\Psi_s), \tag{15}$$

with $\mathcal{F}[\Sigma\|\Psi_1,\ldots,\Psi_s]$ defined in equation (10) and $\mathrm{Pen}(\Sigma,\Psi_1,\ldots,\Psi_s)$ some penalty function. If we take

$$\mathrm{Pen}(\Sigma,\Psi_1,\ldots,\Psi_s) = \lambda\sum_{k=1}^{s}\|B_{k,m_k|o_k}\|_1,$$

then, alternating minimization of (15) with respect to Σ and Ψ_k leads to an algorithm with sparse regression coefficients. This algorithm is different from MissPALasso, in fact, minimizing (15) with respect to $\Sigma_{k,m_k|o_k}$ and $B_{k,m_k|o_k}$ gives $\Sigma_{k,m_k|o_k} = \hat{\Sigma}_{m_k|o_k}$ and $\hat{B}_{k,m_k|o_k}$ satisfies the subgradient equation

$$0 = \left(\Omega_{m_k,m_k}\hat{B}_{k,m_k|o_k} - \Omega_{m_k,o_k}\right){}^t\mathbf{X}^k_{o_k}\mathbf{X}^k_{o_k} + \lambda\Gamma(\hat{B}_{k,m_k|o_k}),$$

where $\Omega = \Sigma^{-1}$ and $\Gamma(x)$ is the subgradient of $|x|$, applied componentwise to the elements of a matrix. We do not currently have knowledge of the theoretical or empirical properties of such an algorithm.

In this manuscript we only considered applications to microarray data sets. Our approach is not specifically designed for microarrays and is potentially very useful for many other high-dimensional applications: examples include mass spectrometry-based proteomics, climate field reconstructions and image analysis in cosmology (Karpievitch et al., 2009; Schneider, 2001; Starck and Bobin, 2010). We note that different imputation methods can be beneficial depending on the application context. For example estimating missing entries in gene expression data is a separate problem from dealing with missing values in recommender systems: the Netflix data set (Bennett and Lanning, 2007) involves "large n *and* large p" (480'000 customers, 17'000 movies) with about 98% of the movie ratings missing, in contrast, microarrays have the typical "large p, small n" form and have a much smaller fraction of the values missing. We think that the formulation (15) of our pattern alternating maximization framework is very compelling and can motivate new and efficient algorithms for missing data imputation with application-specific regularization strategies.

Acknowledgments

N.S. acknowledges financial support from Novartis International AG, Basel, Switzerland.

Appendix A.

In this appendix we prove Theorem 2. First, note that the sequence $\{(\Sigma_1^{(r)},\ldots,\Sigma_s^{(r)})\}_{r=0,1,2,\ldots}$ lies in the compact set $\mathcal{K}$. Now, let $\Sigma_s^{(r_j)}$ be a subsequence converging to $\bar{\Sigma}_s$ as $j\to\infty$. By

invoking compactness, we can assume w.l.o.g (by restricting to a subsequence) that

$$(\Sigma_1^{(r_j)}, \ldots, \Sigma_s^{(r_j)}) \to (\bar{\Sigma}_1, \ldots, \bar{\Sigma}_s).$$

As a direct consequence of the monotonicity of the sequence $\{\mathcal{F}_s[\Sigma_s^{(r)}||\mathrm{Z}_s^{(r)}]\}_{r=0,1,2,\ldots}$ we obtain

$$\lim_r \mathcal{F}_s[\Sigma_s^{(r)}||\mathrm{Z}_s^{(r)}] = \mathcal{F}_s[\bar{\Sigma}_s||\bar{\Sigma}_1, \ldots, \bar{\Sigma}_{s-1}] =: \bar{\mathcal{F}}.$$

From (14) and Proposition 1, for $k = 1, \ldots, s-1$ and $r = 0, 1, 2, \ldots$, the following "sandwich"-formulae hold:

$$\mathcal{F}_s[\Sigma_s^{(r)}||\mathrm{Z}_s^{(r)}] \leq \mathcal{F}_1[\Sigma_s^{(r)}||\mathrm{Z}_1^{(r+1)}] \leq \mathcal{F}_1[\Sigma_1^{(r+1)}||\mathrm{Z}_1^{(r+1)}] \leq \mathcal{F}_s[\Sigma_s^{(r+1)}||\mathrm{Z}_s^{(r+1)}],$$
$$\mathcal{F}_s[\Sigma_s^{(r)}||\mathrm{Z}_s^{(r)}] \leq \mathcal{F}_{k+1}[\Sigma_k^{(r+1)}||\mathrm{Z}_{k+1}^{(r+1)}] \leq \mathcal{F}_{k+1}[\Sigma_{k+1}^{(r+1)}||\mathrm{Z}_{k+1}^{(r+1)}] \leq \mathcal{F}_s[\Sigma_s^{(r+1)}||\mathrm{Z}_s^{(r+1)}].$$

As a consequence we have for $k = 1, \ldots, s-1$

$$\lim_r \mathcal{F}_1[\Sigma_s^{(r)}||\mathrm{Z}_1^{(r+1)}] = \lim_r \mathcal{F}_1[\Sigma_1^{(r+1)}||\mathrm{Z}_1^{(r+1)}] = \bar{\mathcal{F}} \tag{16}$$

and

$$\lim_r \mathcal{F}_{k+1}[\Sigma_k^{(r+1)}||\mathrm{Z}_{k+1}^{(r+1)}] = \lim_r \mathcal{F}_{k+1}[\Sigma_{k+1}^{(r+1)}||\mathrm{Z}_{k+1}^{(r+1)}] = \bar{\mathcal{F}}. \tag{17}$$

Now consider the sequence $(\Sigma_1^{(r_j+1)}, \ldots, \Sigma_s^{(r_j+1)})$. By compactness of $\mathcal{K}$ this sequence converges w.l.o.g to some $(\Sigma_1^*, \ldots, \Sigma_s^*)$. We now show by induction that

$$\bar{\Sigma}_s = \Sigma_1^* = \ldots = \Sigma_s^*.$$

From the 1st M-Step of cycle $r_j + 1$ we have

$$\mathcal{F}_1[\Sigma_1^{(r_j+1)}||\mathrm{Z}_1^{(r_j+1)}] \geq \mathcal{F}_1[\Sigma||\mathrm{Z}_1^{(r_j+1)}] \quad \text{for all} \quad \Sigma.$$

Taking the limit $j \to \infty$ we get:

$$\mathcal{F}_1[\Sigma_1^*||\{\bar{\Sigma}_l\}_{l>1}] \geq \mathcal{F}_1[\Sigma||\{\bar{\Sigma}_l\}_{l>1}] \quad \text{for all} \quad \Sigma.$$

In particular, Σ_1^* is the (unique) maximizer of $\mathcal{F}_1[\,\cdot\,||\{\bar{\Sigma}_l\}_{l>1}]$. Assuming $\Sigma_1^* \neq \bar{\Sigma}_s$ would imply

$$\mathcal{F}_1[\Sigma_1^*||\{\bar{\Sigma}_l\}_{l>1}] > \mathcal{F}_1[\bar{\Sigma}_s||\{\bar{\Sigma}_l\}_{l>1}].$$

But this contradicts $\mathcal{F}_1[\Sigma_1^*||\{\bar{\Sigma}_l\}_{l>1}] = \mathcal{F}_1[\bar{\Sigma}_s||\{\bar{\Sigma}_l\}_{l>1}] = \bar{\mathcal{F}}$, which holds by (16). Therefore we obtain $\Sigma_1^* = \bar{\Sigma}_s$.

Assume that we have proven $\Sigma_1^* = \ldots = \Sigma_k^* = \bar{\Sigma}_s$. We will show that $\Sigma_{k+1}^* = \bar{\Sigma}_s$. From the k+1st M-Step in cycle $r_j + 1$:

$$\mathcal{F}_{k+1}[\Sigma_{k+1}^{(r_j+1)}||\mathrm{Z}_{k+1}^{(r_j+1)}] \geq \mathcal{F}_{k+1}[\Sigma||\mathrm{Z}_{k+1}^{(r_j+1)}] \quad \text{for all} \quad \Sigma.$$

Taking the limit for $j \to \infty$, we conclude that Σ_{k+1}^* is the (unique) maximizer of

$$\mathcal{F}_{k+1}[\,\cdot\,||\{\Sigma_l^*\}_{l<k+1}, \{\bar{\Sigma}_l\}_{l>k+1}].$$

From (17),

$$\mathcal{F}_{k+1}[\Sigma_{k+1}^*||\{\Sigma_l^*\}_{l<k+1}, \{\bar{\Sigma}_l\}_{l>k+1}] = \mathcal{F}_{k+1}[\Sigma_k^*||\{\Sigma_l^*\}_{l<k+1}, \{\bar{\Sigma}_l\}_{l>k+1}] = \bar{\mathcal{F}},$$

and therefore Σ_{k+1}^* must be equal to Σ_k^*. By induction we have $\Sigma_k^* = \bar{\Sigma}_s$ and we have proven that $\Sigma_{k+1}^* = \bar{\Sigma}_s$ holds.

Finally, we show stationarity of $\bar{\Sigma}_s$. Invoking (13) we can write

$$\mathcal{F}_s[\Sigma||\bar{\Sigma}_s, \ldots, \bar{\Sigma}_s] = \ell(\Sigma; \mathbf{X}_{\text{obs}}) - \sum_{l=1}^{s-1} \mathcal{D}_l[\bar{\Sigma}_s||\Sigma].$$

Note that

$$\frac{\partial}{\partial \Sigma} \mathcal{D}_l[\bar{\Sigma}_s||\Sigma]\Big|_{\bar{\Sigma}_s} = 0.$$

Furthermore, as $\Sigma_s^{(r_j+1)}$ maximizes $\mathcal{F}_s[\Sigma||\Sigma_1^{(r_j+1)}, \ldots, \Sigma_{s-1}^{(r_j+1)}]$, we get in the limit as $j \to \infty$

$$\frac{\partial}{\partial \Sigma} \mathcal{F}_s[\Sigma||\bar{\Sigma}_s, \ldots, \bar{\Sigma}_s]\Big|_{\bar{\Sigma}_s} = \frac{\partial}{\partial \Sigma} \mathcal{F}_s[\Sigma||\Sigma_1^*, \ldots, \Sigma_{s-1}^*]\Big|_{\Sigma_s^*} = 0.$$

Therefore, we conclude that $\frac{\partial}{\partial \Sigma} \ell(\Sigma; \mathbf{X}_{\text{obs}})\Big|_{\bar{\Sigma}_s} = 0.$ ■

References

T. Aittokallio. Dealing with missing values in large-scale studies: microarray data imputation and beyond. *Briefings in Bioinformatics*, 11(2):253–264, 2010.

A. Alizadeh, M. Eisen, R. Davis, C. Ma, I. Lossos, A. Rosenwald, J. Boldrick, H. Sabet, T. Tran, X. Yu, J. Powell, L. Yang, G. Marti, T. Moore, J. Hudson, L. Lu, D. Lewis, R. Tibshirani, G. Sherlock, W. Chan, T. Greiner, D. Weisenburger, J. Armitage, R. Warnke, R. Levy, W. Wilson, M. R. Grever, J. Byrd, D. Botstein, P. Brown, and L. Staudt. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, 2000.

G. Allen and R. Tibshirani. Transposable regularized covariance models with an application to missing data imputation. *Annals of Applied Statistics*, 4(2):764–790, 2010.

U. Alon, N. Barkai, D. Notterman, K. Gishdagger, S. Ybarradagger, D. Mackdagger, and A. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences of the United States of America*, 96(12):6745–6750, 1999.

J. Bennett and S. Lanning. The Netflix prize. In *Proceedings of KDD Cup and Workshop*, San Jose, 2007.

J.-F. Cai, E. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

E. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

E. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5), 2010.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

J. Fessler and A. Hero. Space-alternating generalized Expectation-Maximization algorithm. *IEEE Transactions on Signal Processing*, 42(11):2664–2677, 1994.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical Lasso. *Biostatistics*, 9(3):432–441, 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.

A. Gunawardana and W. Byrne. Convergence theorems for generalized alternating minimization procedures. *Journal of Machine Learning Research*, 6:2049–2073, 2005.

M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

J. Josse, J. Pagès, and F. Husson. Multiple imputation in principal component analysis. *Advances in Data Analysis and Classification*, 5(3):231–246, 2011.

Y. V. Karpievitch, J. Stanley, T. Taverner, J. Huang, J. N. Adkins, C. Ansong, F. Heffron, T. O. Metz, W.-J. Qian, H. Yoon, R. D. Smith, and A. R. Dabney. A statistical framework for protein quantitation in bottom-up MS-based proteomics. *Bioinformatics*, 25(16):2028–2034, 2009.

R. Keshavan, S. Oh, and A. Montanari. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6), 2010.

K. Lange, D. Hunter, and I. Yang. Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, 9(1):1–20, 2000.

R. Little and D. Rubin. *Statistical Analysis with Missing Data*. Series in Probability and Mathematical Statistics, Wiley, 1987.

P.-L. Loh and M. J. Wainwright. High-dimensional regression with noisy and missing data: Provable guarantees with nonconvexity. *Annals of Statistics*, 40(3):1637–1664, 2012.

R. Mazumder, T. Hastie, and R. Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 99:2287–2322, 2010.

R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

S. Ng and G. McLachlan. On the choice of the number of blocks with the incremental EM algorithm for the fitting of normal mixtures. *Statistics and Computing*, 13(1):45–55, 2003.

S. Nowlan. *Soft Competitive Adaptation: Neural Network Learning Algorithms based on Fitting Statistical Mixtures*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1991.

S. Oba, M.-A. Sato, I. Takemasa, M. Monden, K.-I. Matsubara, and S. Ishii. A Bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, 19(16): 2088–2096, 2003.

M. Rosenbaum and A. Tsybakov. Sparse recovery under matrix uncertainty. *Annals of Statistics*, 38(5):2620–2651, 2010.

J. Schafer. *Analysis of Incomplete Multivariate Data*. Monographs on Statistics and Applied Probability 72, Chapman and Hall, 1997.

T. Schneider. Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values. *Journal of Climate*, 14(5):853–871, 2001.

P. Spellman, G. Sherlock, M. Zhang, V. Iyer, K. Anders, M. Eisen, P. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9 (12):3273–97, 1998.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. The MIT Press, 2nd edition, 2000.

N. Städler and P. Bühlmann. Missing values: sparse inverse covariance estimation and an extension to sparse regression. *Statistics and Computing*, 22(1):219–235, 2012.

J.-L. Starck and J. Bobin. Astronomical data analysis and sparsity: From wavelets to compressed sensing. *Proceedings of the IEEE*, 98(6):1021–1030, 2010.

B. Thiesson, C. Meek, and D. Heckerman. Accelerating EM for large databases. *Machine Learning*, 45(3):279–299, 2001.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, 2001.

A. Wille, P. Zimmermann, E. Vranova, A. Fürholz, O. Laule, S. Bleuler, L. Hennig, A. Prelic, P. Rohrvon , L. Thiele, E. Zitzler, W. Gruissem, and P. Bühlmann. Sparse graphical Gaussian modeling of the isoprenoid gene network in arabidopsis thaliana. *Genome Biology*, 5(11), 2004.

Journal of Machine Learning Research 15 (2014) 1929-1958 Submitted 11/13; Published 6/14

Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Nitish Srivastava NITISH@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU
Alex Krizhevsky KRIZ@CS.TORONTO.EDU
Ilya Sutskever ILYA@CS.TORONTO.EDU
Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
10 Kings College Road, Rm 3302
Toronto, Ontario, M5S 3G4, Canada.

Editor: Yoshua Bengio

Abstract

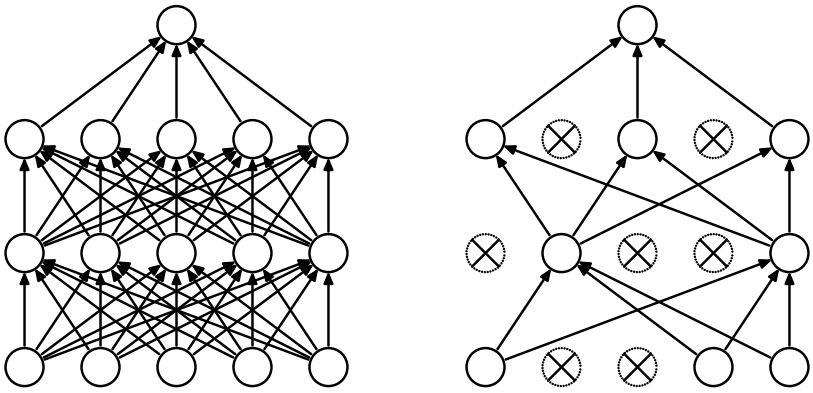
Deep neural nets with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different "thinned" networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods. We show that dropout improves the performance of neural networks on supervised learning tasks in vision, speech recognition, document classification and computational biology, obtaining state-of-the-art results on many benchmark data sets.

Keywords: neural networks, regularization, model combination, deep learning

1. Introduction

Deep neural networks contain multiple non-linear hidden layers and this makes them very expressive models that can learn very complicated relationships between their inputs and outputs. With limited training data, however, many of these complicated relationships will be the result of sampling noise, so they will exist in the training set but not in real test data even if it is drawn from the same distribution. This leads to overfitting and many methods have been developed for reducing it. These include stopping the training as soon as performance on a validation set starts to get worse, introducing weight penalties of various kinds such as L1 and L2 regularization and soft weight sharing (Nowlan and Hinton, 1992).

With unlimited computation, the best way to "regularize" a fixed-sized model is to average the predictions of all possible settings of the parameters, weighting each setting by



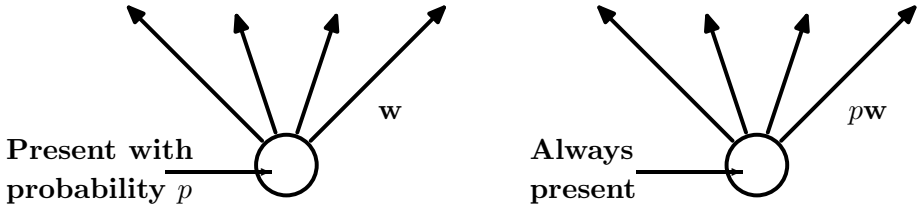
(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

its posterior probability given the training data. This can sometimes be approximated quite well for simple or small models (Xiong et al., 2011; Salakhutdinov and Mnih, 2008), but we would like to approach the performance of the Bayesian gold standard using considerably less computation. We propose to do this by approximating an equally weighted geometric mean of the predictions of an exponential number of learned models that share parameters.

Model combination nearly always improves the performance of machine learning methods. With large neural networks, however, the obvious idea of averaging the outputs of many separately trained nets is prohibitively expensive. Combining several models is most helpful when the individual models are different from each other and in order to make neural net models different, they should either have different architectures or be trained on different data. Training many different architectures is hard because finding optimal hyperparameters for each architecture is a daunting task and training each large network requires a lot of computation. Moreover, large networks normally require large amounts of training data and there may not be enough data available to train different networks on different subsets of the data. Even if one was able to train many different large networks, using them all at test time is infeasible in applications where it is important to respond quickly.

Dropout is a technique that addresses both these issues. It prevents overfitting and provides a way of approximately combining exponentially many different neural network architectures efficiently. The term "dropout" refers to dropping out units (hidden and visible) in a neural network. By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections, as shown in Figure 1. The choice of which units to drop is random. In the simplest case, each unit is retained with a fixed probability p independent of other units, where p can be chosen using a validation set or can simply be set at 0.5, which seems to be close to optimal for a wide range of networks and tasks. For the input units, however, the optimal probability of retention is usually closer to 1 than to 0.5.



(a) At training time (b) At test time

Figure 2: **Left**: A unit at training time that is present with probability p and is connected to units in the next layer with weights $\mathbf{w}$. **Right**: At test time, the unit is always present and the weights are multiplied by p. The output at test time is same as the expected output at training time.

Applying dropout to a neural network amounts to sampling a "thinned" network from it. The thinned network consists of all the units that survived dropout (Figure 1b). A neural net with n units, can be seen as a collection of 2^n possible thinned neural networks. These networks all share weights so that the total number of parameters is still $O(n^2)$, or less. For each presentation of each training case, a new thinned network is sampled and trained. So training a neural network with dropout can be seen as training a collection of 2^n thinned networks with extensive weight sharing, where each thinned network gets trained very rarely, if at all.

At test time, it is not feasible to explicitly average the predictions from exponentially many thinned models. However, a very simple approximate averaging method works well in practice. The idea is to use a single neural net at test time without dropout. The weights of this network are scaled-down versions of the trained weights. If a unit is retained with probability p during training, the outgoing weights of that unit are multiplied by p at test time as shown in Figure 2. This ensures that for any hidden unit the *expected* output (under the distribution used to drop units at training time) is the same as the actual output at test time. By doing this scaling, 2^n networks with shared weights can be combined into a single neural network to be used at test time. We found that training a network with dropout and using this approximate averaging method at test time leads to significantly lower generalization error on a wide variety of classification problems compared to training with other regularization methods.

The idea of dropout is not limited to feed-forward neural nets. It can be more generally applied to graphical models such as Boltzmann Machines. In this paper, we introduce the dropout Restricted Boltzmann Machine model and compare it to standard Restricted Boltzmann Machines (RBM). Our experiments show that dropout RBMs are better than standard RBMs in certain respects.

This paper is structured as follows. Section 2 describes the motivation for this idea. Section 3 describes relevant previous work. Section 4 formally describes the dropout model. Section 5 gives an algorithm for training dropout networks. In Section 6, we present our experimental results where we apply dropout to problems in different domains and compare it with other forms of regularization and model combination. Section 7 analyzes the effect of dropout on different properties of a neural network and describes how dropout interacts with the network's hyperparameters. Section 8 describes the Dropout RBM model. In Section 9 we explore the idea of marginalizing dropout. In Appendix A we present a practical guide

for training dropout nets. This includes a detailed analysis of the practical considerations involved in choosing hyperparameters when training dropout networks.

2. Motivation

A motivation for dropout comes from a theory of the role of sex in evolution (Livnat et al., 2010). Sexual reproduction involves taking half the genes of one parent and half of the other, adding a very small amount of random mutation, and combining them to produce an offspring. The asexual alternative is to create an offspring with a slightly mutated copy of the parent's genes. It seems plausible that asexual reproduction should be a better way to optimize individual fitness because a good set of genes that have come to work well together can be passed on directly to the offspring. On the other hand, sexual reproduction is likely to break up these co-adapted sets of genes, especially if these sets are large and, intuitively, this should decrease the fitness of organisms that have already evolved complicated co-adaptations. However, sexual reproduction is the way most advanced organisms evolved.

One possible explanation for the superiority of sexual reproduction is that, over the long term, the criterion for natural selection may not be individual fitness but rather mix-ability of genes. The ability of a set of genes to be able to work well with another random set of genes makes them more robust. Since a gene cannot rely on a large set of partners to be present at all times, it must learn to do something useful on its own or in collaboration with a *small* number of other genes. According to this theory, the role of sexual reproduction is not just to allow useful new genes to spread throughout the population, but also to facilitate this process by reducing complex co-adaptations that would reduce the chance of a new gene improving the fitness of an individual. Similarly, each hidden unit in a neural network trained with dropout must learn to work with a randomly chosen sample of other units. This should make each hidden unit more robust and drive it towards creating useful features on its own without relying on other hidden units to correct its mistakes. However, the hidden units within a layer will still learn to do different things from each other. One might imagine that the net would become robust against dropout by making many copies of each hidden unit, but this is a poor solution for exactly the same reason as replica codes are a poor way to deal with a noisy channel.

A closely related, but slightly different motivation for dropout comes from thinking about successful conspiracies. Ten conspiracies each involving five people is probably a better way to create havoc than one big conspiracy that requires fifty people to all play their parts correctly. If conditions do not change and there is plenty of time for rehearsal, a big conspiracy can work well, but with non-stationary conditions, the smaller the conspiracy the greater its chance of still working. Complex co-adaptations can be trained to work well on a training set, but on novel test data they are far more likely to fail than multiple simpler co-adaptations that achieve the same thing.

3. Related Work

Dropout can be interpreted as a way of regularizing a neural network by adding noise to its hidden units. The idea of adding noise to the states of units has previously been used in the context of Denoising Autoencoders (DAEs) by Vincent et al. (2008, 2010) where noise

is added to the input units of an autoencoder and the network is trained to reconstruct the noise-free input. Our work extends this idea by showing that dropout can be effectively applied in the hidden layers as well and that it can be interpreted as a form of model averaging. We also show that adding noise is not only useful for unsupervised feature learning but can also be extended to supervised learning problems. In fact, our method can be applied to other neuron-based architectures, for example, Boltzmann Machines. While 5% noise typically works best for DAEs, we found that our weight scaling procedure applied at test time enables us to use much higher noise levels. Dropping out 20% of the input units and 50% of the hidden units was often found to be optimal.

Since dropout can be seen as a *stochastic* regularization technique, it is natural to consider its *deterministic* counterpart which is obtained by marginalizing out the noise. In this paper, we show that, in simple cases, dropout can be analytically marginalized out to obtain deterministic regularization methods. Recently, van der Maaten et al. (2013) also explored deterministic regularizers corresponding to different exponential-family noise distributions, including dropout (which they refer to as "blankout noise"). However, they apply noise to the inputs and only explore models with no hidden layers. Wang and Manning (2013) proposed a method for speeding up dropout by marginalizing dropout noise. Chen et al. (2012) explored marginalization in the context of denoising autoencoders.

In dropout, we minimize the loss function stochastically under a noise distribution. This can be seen as minimizing an expected loss function. Previous work of Globerson and Roweis (2006); Dekel et al. (2010) explored an alternate setting where the loss is minimized when an adversary gets to pick which units to drop. Here, instead of a noise distribution, the maximum number of units that can be dropped is fixed. However, this work also does not explore models with hidden units.

4. Model Description

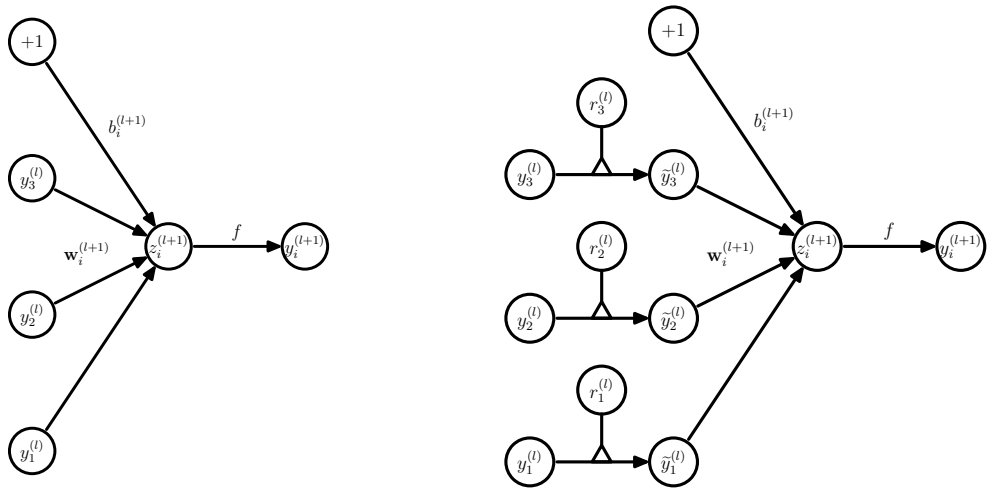
This section describes the dropout neural network model. Consider a neural network with L hidden layers. Let $l \in \{1, \ldots, L\}$ index the hidden layers of the network. Let $\mathbf{z}^{(l)}$ denote the vector of inputs into layer l, $\mathbf{y}^{(l)}$ denote the vector of outputs from layer l ($\mathbf{y}^{(0)} = \mathbf{x}$ is the input). $W^{(l)}$ and $\mathbf{b}^{(l)}$ are the weights and biases at layer l. The feed-forward operation of a standard neural network (Figure 3a) can be described as (for $l \in \{0, \ldots, L-1\}$ and any hidden unit i)

$$\begin{aligned} z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \mathbf{y}^l + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}), \end{aligned}$$

where f is any activation function, for example, $f(x) = 1/\left(1 + \exp(-x)\right)$.

With dropout, the feed-forward operation becomes (Figure 3b)

$$\begin{aligned} r_j^{(l)} &\sim \text{Bernoulli}(p), \\ \widetilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)}, \\ z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \widetilde{\mathbf{y}}^l + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}). \end{aligned}$$



(a) Standard network

(b) Dropout network

Figure 3: Comparison of the basic operations of a standard and dropout network.

Here $*$ denotes an element-wise product. For any layer l, $\mathbf{r}^{(l)}$ is a vector of independent Bernoulli random variables each of which has probability p of being 1. This vector is sampled and multiplied element-wise with the outputs of that layer, $\mathbf{y}^{(l)}$, to create the thinned outputs $\widetilde{\mathbf{y}}^{(l)}$. The thinned outputs are then used as input to the next layer. This process is applied at each layer. This amounts to sampling a sub-network from a larger network. For learning, the derivatives of the loss function are backpropagated through the sub-network. At test time, the weights are scaled as $W_{test}^{(l)} = pW^{(l)}$ as shown in Figure 2. The resulting neural network is used without dropout.

5. Learning Dropout Nets

This section describes a procedure for training dropout neural nets.

5.1 Backpropagation

Dropout neural networks can be trained using stochastic gradient descent in a manner similar to standard neural nets. The only difference is that for each training case in a mini-batch, we sample a thinned network by dropping out units. Forward and backpropagation for that training case are done only on this thinned network. The gradients for each parameter are averaged over the training cases in each mini-batch. Any training case which does not use a parameter contributes a gradient of zero for that parameter. Many methods have been used to improve stochastic gradient descent such as momentum, annealed learning rates and L2 weight decay. Those were found to be useful for dropout neural networks as well.

One particular form of regularization was found to be especially useful for dropout—constraining the norm of the incoming weight vector at each hidden unit to be upper bounded by a fixed constant c. In other words, if $\mathbf{w}$ represents the vector of weights incident on any hidden unit, the neural network was optimized under the constraint $||\mathbf{w}||_2 \leq c$. This constraint was imposed during optimization by projecting $\mathbf{w}$ onto the surface of a ball of radius c, whenever $\mathbf{w}$ went out of it. This is also called max-norm regularization since it implies that the maximum value that the norm of any weight can take is c. The constant

c is a tunable hyperparameter, which is determined using a validation set. Max-norm regularization has been previously used in the context of collaborative filtering (Srebro and Shraibman, 2005). It typically improves the performance of stochastic gradient descent training of deep neural nets, even when no dropout is used.

Although dropout alone gives significant improvements, using dropout along with max-norm regularization, large decaying learning rates and high momentum provides a significant boost over just using dropout. A possible justification is that constraining weight vectors to lie inside a ball of fixed radius makes it possible to use a huge learning rate without the possibility of weights blowing up. The noise provided by dropout then allows the optimization process to explore different regions of the weight space that would have otherwise been difficult to reach. As the learning rate decays, the optimization takes shorter steps, thereby doing less exploration and eventually settles into a minimum.

5.2 Unsupervised Pretraining

Neural networks can be pretrained using stacks of RBMs (Hinton and Salakhutdinov, 2006), autoencoders (Vincent et al., 2010) or Deep Boltzmann Machines (Salakhutdinov and Hinton, 2009). Pretraining is an effective way of making use of unlabeled data. Pretraining followed by finetuning with backpropagation has been shown to give significant performance boosts over finetuning from random initializations in certain cases.

Dropout can be applied to finetune nets that have been pretrained using these techniques. The pretraining procedure stays the same. The weights obtained from pretraining should be scaled up by a factor of $1/p$. This makes sure that for each unit, the expected output from it under random dropout will be the same as the output during pretraining. We were initially concerned that the stochastic nature of dropout might wipe out the information in the pretrained weights. This did happen when the learning rates used during finetuning were comparable to the best learning rates for randomly initialized nets. However, when the learning rates were chosen to be smaller, the information in the pretrained weights seemed to be retained and we were able to get improvements in terms of the final generalization error compared to not using dropout when finetuning.

6. Experimental Results

We trained dropout neural networks for classification problems on data sets in different domains. We found that dropout improved generalization performance on *all* data sets compared to neural networks that did not use dropout. Table 1 gives a brief description of the data sets. The data sets are

- MNIST : A standard toy data set of handwritten digits.
- TIMIT : A standard speech benchmark for clean speech recognition.
- CIFAR-10 and CIFAR-100 : Tiny natural images (Krizhevsky, 2009).
- Street View House Numbers data set (SVHN) : Images of house numbers collected by Google Street View (Netzer et al., 2011).
- ImageNet : A large collection of natural images.
- Reuters-RCV1 : A collection of Reuters newswire articles.

- Alternative Splicing data set: RNA features for predicting alternative gene splicing (Xiong et al., 2011).

We chose a diverse set of data sets to demonstrate that dropout is a general technique for improving neural nets and is not specific to any particular application domain. In this section, we present some key results that show the effectiveness of dropout. A more detailed description of all the experiments and data sets is provided in Appendix B.

Data Set	**Domain**	**Dimensionality**	**Training Set**	**Test Set**
MNIST	Vision	784 (28×28 grayscale)	60K	10K
SVHN	Vision	3072 (32×32 color)	600K	26K
CIFAR-10/100	Vision	3072 (32×32 color)	60K	10K
ImageNet (ILSVRC-2012)	Vision	65536 (256×256 color)	1.2M	150K
TIMIT	Speech	2520 (120-dim, 21 frames)	1.1M frames	58K frames
Reuters-RCV1	Text	2000	200K	200K
Alternative Splicing	Genetics	1014	2932	733

Table 1: Overview of the data sets used in this paper.

6.1 Results on Image Data Sets

We used five image data sets to evaluate dropout—MNIST, SVHN, CIFAR-10, CIFAR-100 and ImageNet. These data sets include different image types and training set sizes. Models which achieve state-of-the-art results on *all* of these data sets use dropout.

6.1.1 MNIST

Method	**Unit Type**	**Architecture**	**Error %**
Standard Neural Net (Simard et al., 2003)	Logistic	2 layers, 800 units	1.60
SVM Gaussian kernel	NA	NA	1.40
Dropout NN	Logistic	3 layers, 1024 units	1.35
Dropout NN	ReLU	3 layers, 1024 units	1.25
Dropout NN + max-norm constraint	ReLU	3 layers, 1024 units	1.06
Dropout NN + max-norm constraint	ReLU	3 layers, 2048 units	1.04
Dropout NN + max-norm constraint	ReLU	2 layers, 4096 units	1.01
Dropout NN + max-norm constraint	ReLU	2 layers, 8192 units	0.95
Dropout NN + max-norm constraint (Goodfellow et al., 2013)	Maxout	2 layers, (5×240) units	0.94
DBN + finetuning (Hinton and Salakhutdinov, 2006)	Logistic	500-500-2000	1.18
DBM + finetuning (Salakhutdinov and Hinton, 2009)	Logistic	500-500-2000	0.96
DBN + dropout finetuning	Logistic	500-500-2000	0.92
DBM + dropout finetuning	Logistic	500-500-2000	**0.79**

Table 2: Comparison of different models on MNIST.

The MNIST data set consists of 28×28 pixel handwritten digit images. The task is to classify the images into 10 digit classes. Table 2 compares the performance of dropout with other techniques. The best performing neural networks for the permutation invariant

setting that do not use dropout or unsupervised pretraining achieve an error of about 1.60% (Simard et al., 2003). With dropout the error reduces to 1.35%. Replacing logistic units with rectified linear units (ReLUs) (Jarrett et al., 2009) further reduces the error to 1.25%. Adding max-norm regularization again reduces it to 1.06%. Increasing the size of the network leads to better results. A neural net with 2 layers and 8192 units per layer gets down to 0.95% error. Note that this network has more than 65 million parameters and is being trained on a data set of size 60,000. Training a network of this size to give good generalization error is very hard with standard regularization methods and early stopping. Dropout, on the other hand, prevents overfitting, even in this case. It does not even need early stopping. Goodfellow et al. (2013) showed that results can be further improved to 0.94% by replacing ReLU units with maxout units. All dropout nets use $p = 0.5$ for hidden units and $p = 0.8$ for input units. More experimental details can be found in Appendix B.1.

Dropout nets pretrained with stacks of RBMs and Deep Boltzmann Machines also give improvements as shown in Table 2. DBM—pretrained dropout nets achieve a test error of 0.79% which is the best performance ever reported for the permutation invariant setting. We note that it possible to obtain better results by using 2-D spatial information and augmenting the training set with distorted versions of images from the standard training set. We demonstrate the effectiveness of dropout in that setting on more interesting data sets.

In order to test the robustness of dropout, classification experiments were done with networks of many different architectures keeping all hyperparameters, including p, fixed. Figure 4 shows the test error rates obtained for these different architectures as training progresses. The same architectures trained with and without dropout have drastically different test errors as seen as by the two separate clusters of trajectories. Dropout gives a huge improvement across all architectures, without using hyperparameters that were tuned specifically for each architecture.

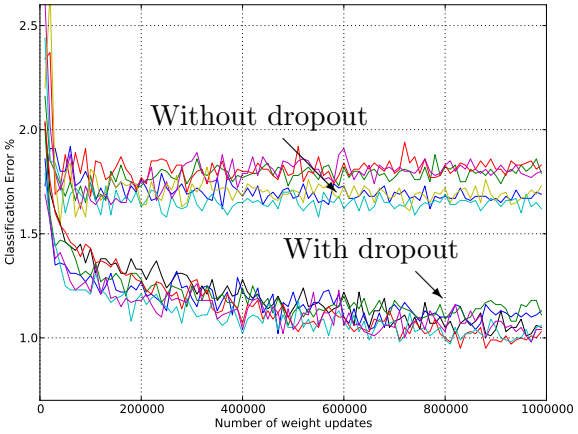


Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

6.1.2 Street View House Numbers

The Street View House Numbers (SVHN) Data Set (Netzer et al., 2011) consists of color images of house numbers collected by Google Street View. Figure 5a shows some examples of images from this data set. The part of the data set that we use in our experiments consists of 32×32 color images roughly centered on a digit in a house number. The task is to identify that digit.

For this data set, we applied dropout to convolutional neural networks (LeCun et al., 1989). The best architecture that we found has three convolutional layers followed by 2 fully connected hidden layers. All hidden units were ReLUs. Each convolutional layer was

Method	**Error %**
Binary Features (WDCH) (Netzer et al., 2011)	36.7
HOG (Netzer et al., 2011)	15.0
Stacked Sparse Autoencoders (Netzer et al., 2011)	10.3
KMeans (Netzer et al., 2011)	9.4
Multi-stage Conv Net with average pooling (Sermanet et al., 2012)	9.06
Multi-stage Conv Net + L2 pooling (Sermanet et al., 2012)	5.36
Multi-stage Conv Net + L4 pooling + padding (Sermanet et al., 2012)	4.90
Conv Net + max-pooling	3.95
Conv Net + max pooling + dropout in fully connected layers	3.02
Conv Net + stochastic pooling (Zeiler and Fergus, 2013)	2.80
Conv Net + max pooling + dropout in all layers	2.55
Conv Net + maxout (Goodfellow et al., 2013)	**2.47**
Human Performance	2.0

Table 3: Results on the Street View House Numbers data set.

followed by a max-pooling layer. Appendix B.2 describes the architecture in more detail. Dropout was applied to all the layers of the network with the probability of retaining a hidden unit being $p = (0.9, 0.75, 0.75, 0.5, 0.5, 0.5)$ for the different layers of the network (going from input to convolutional layers to fully connected layers). Max-norm regularization was used for weights in both convolutional and fully connected layers. Table 3 compares the results obtained by different methods. We find that convolutional nets outperform other methods. The best performing convolutional nets that do not use dropout achieve an error rate of 3.95%. Adding dropout only to the fully connected layers reduces the error to 3.02%. Adding dropout to the convolutional layers as well further reduces the error to 2.55%. Even more gains can be obtained by using maxout units.

The additional gain in performance obtained by adding dropout in the convolutional layers (3.02% to 2.55%) is worth noting. One may have presumed that since the convolutional layers don't have a lot of parameters, overfitting is not a problem and therefore dropout would not have much effect. However, dropout in the lower layers still helps because it provides noisy inputs for the higher fully connected layers which prevents them from overfitting.

6.1.3 CIFAR-10 and CIFAR-100

The CIFAR-10 and CIFAR-100 data sets consist of 32×32 color images drawn from 10 and 100 categories respectively. Figure 5b shows some examples of images from this data set. A detailed description of the data sets, input preprocessing, network architectures and other experimental details is given in Appendix B.3. Table 4 shows the error rate obtained by different methods on these data sets. Without any data augmentation, Snoek et al. (2012) used Bayesian hyperparameter optimization to obtained an error rate of 14.98% on CIFAR-10. Using dropout in the fully connected layers reduces that to 14.32% and adding dropout in every layer further reduces the error to 12.61%. Goodfellow et al. (2013) showed that the error is further reduced to 11.68% by replacing ReLU units with maxout units. On CIFAR-100, dropout reduces the error from 43.48% to 37.20% which is a huge improvement. No data augmentation was used for either data set (apart from the input dropout).

(a) Street View House Numbers (SVHN) (b) CIFAR-10

Figure 5: Samples from image data sets. Each row corresponds to a different category.

Method	**CIFAR-10**	**CIFAR-100**
Conv Net + max pooling (hand tuned)	15.60	43.48
Conv Net + stochastic pooling (Zeiler and Fergus, 2013)	15.13	42.51
Conv Net + max pooling (Snoek et al., 2012)	14.98	-
Conv Net + max pooling + dropout fully connected layers	14.32	41.26
Conv Net + max pooling + dropout in all layers	12.61	**37.20**
Conv Net + maxout (Goodfellow et al., 2013)	**11.68**	38.57

Table 4: Error rates on CIFAR-10 and CIFAR-100.

6.1.4 ImageNet

ImageNet is a data set of over 15 million labeled high-resolution images belonging to roughly 22,000 categories. Starting in 2010, as part of the Pascal Visual Object Challenge, an annual competition called the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) has been held. A subset of ImageNet with roughly 1000 images in each of 1000 categories is used in this challenge. Since the number of categories is rather large, it is conventional to report two error rates: top-1 and top-5, where the top-5 error rate is the fraction of test images for which the correct label is not among the five labels considered most probable by the model. Figure 6 shows some predictions made by our model on a few test images.

ILSVRC-2010 is the only version of ILSVRC for which the test set labels are available, so most of our experiments were performed on this data set. Table 5 compares the performance of different methods. Convolutional nets with dropout outperform other methods by a large margin. The architecture and implementation details are described in detail in Krizhevsky et al. (2012).

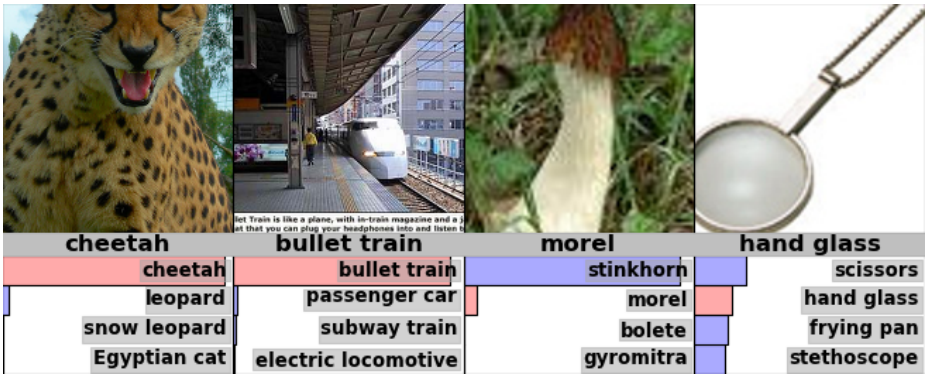


Figure 6: Some ImageNet test cases with the 4 most probable labels as predicted by our model. The length of the horizontal bars is proportional to the probability assigned to the labels by the model. Pink indicates ground truth.

Model	Top-1	Top-5
Sparse Coding (Lin et al., 2010)	47.1	28.2
SIFT + Fisher Vectors (Sanchez and Perronnin, 2011)	45.7	25.7
Conv Net + dropout (Krizhevsky et al., 2012)	37.5	17.0

Table 5: Results on the ILSVRC-2010 test set.

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SVM on Fisher Vectors of Dense SIFT and Color Statistics	-	-	27.3
Avg of classifiers over FVs of SIFT, LBP, GIST and CSIFT	-	-	26.2
Conv Net + dropout (Krizhevsky et al., 2012)	40.7	18.2	-
Avg of 5 Conv Nets + dropout (Krizhevsky et al., 2012)	38.1	16.4	16.4

Table 6: Results on the ILSVRC-2012 validation/test set.

Our model based on convolutional nets and dropout won the ILSVRC-2012 competition. Since the labels for the test set are not available, we report our results on the test set for the final submission and include the validation set results for different variations of our model. Table 6 shows the results from the competition. While the best methods based on standard vision features achieve a top-5 error rate of about 26%, convolutional nets with dropout achieve a test error of about 16% which is a staggering difference. Figure 6 shows some examples of predictions made by our model. We can see that the model makes very reasonable predictions, even when its best guess is not correct.

6.2 Results on TIMIT

Next, we applied dropout to a speech recognition task. We use the TIMIT data set which consists of recordings from 680 speakers covering 8 major dialects of American English reading ten phonetically-rich sentences in a controlled noise-free environment. Dropout neural networks were trained on windows of 21 log-filter bank frames to predict the label of the central frame. No speaker dependent operations were performed. Appendix B.4 describes the data preprocessing and training details. Table 7 compares dropout neural

nets with other models. A 6-layer net gives a phone error rate of 23.4%. Dropout further improves it to 21.8%. We also trained dropout nets starting from pretrained weights. A 4-layer net pretrained with a stack of RBMs get a phone error rate of 22.7%. With dropout, this reduces to 19.7%. Similarly, for an 8-layer net the error reduces from 20.5% to 19.7%.

Method	**Phone Error Rate%**
NN (6 layers) (Mohamed et al., 2010)	23.4
Dropout NN (6 layers)	21.8
DBN-pretrained NN (4 layers)	22.7
DBN-pretrained NN (6 layers) (Mohamed et al., 2010)	22.4
DBN-pretrained NN (8 layers) (Mohamed et al., 2010)	20.7
mcRBM-DBN-pretrained NN (5 layers) (Dahl et al., 2010)	20.5
DBN-pretrained NN (4 layers) + dropout	**19.7**
DBN-pretrained NN (8 layers) + dropout	**19.7**

Table 7: Phone error rate on the TIMIT core test set.

6.3 Results on a Text Data Set

To test the usefulness of dropout in the text domain, we used dropout networks to train a document classifier. We used a subset of the Reuters-RCV1 data set which is a collection of over 800,000 newswire articles from Reuters. These articles cover a variety of topics. The task is to take a bag of words representation of a document and classify it into 50 disjoint topics. Appendix B.5 describes the setup in more detail. Our best neural net which did not use dropout obtained an error rate of 31.05%. Adding dropout reduced the error to 29.62%. We found that the improvement was much smaller compared to that for the vision and speech data sets.

6.4 Comparison with Bayesian Neural Networks

Dropout can be seen as a way of doing an equally-weighted averaging of exponentially many models with shared weights. On the other hand, Bayesian neural networks (Neal, 1996) are the proper way of doing model averaging over the space of neural network structures and parameters. In dropout, each model is weighted equally, whereas in a Bayesian neural network each model is weighted taking into account the prior and how well the model fits the data, which is the more correct approach. Bayesian neural nets are extremely useful for solving problems in domains where data is scarce such as medical diagnosis, genetics, drug discovery and other computational biology applications. However, Bayesian neural nets are slow to train and difficult to scale to very large network sizes. Besides, it is expensive to get predictions from many large nets at test time. On the other hand, dropout neural nets are much faster to train and use at test time. In this section, we report experiments that compare Bayesian neural nets with dropout neural nets on a small data set where Bayesian neural networks are known to perform well and obtain state-of-the-art results. The aim is to analyze how much does dropout lose compared to Bayesian neural nets.

The data set that we use (Xiong et al., 2011) comes from the domain of genetics. The task is to predict the occurrence of alternative splicing based on RNA features. Alternative splicing is a significant cause of cellular diversity in mammalian tissues. Predicting the

Method	Code Quality (bits)
Neural Network (early stopping) (Xiong et al., 2011)	440
Regression, PCA (Xiong et al., 2011)	463
SVM, PCA (Xiong et al., 2011)	487
Neural Network with dropout	567
Bayesian Neural Network (Xiong et al., 2011)	**623**

Table 8: Results on the Alternative Splicing Data Set.

occurrence of alternate splicing in certain tissues under different conditions is important for understanding many human diseases. Given the RNA features, the task is to predict the probability of three splicing related events that biologists care about. The evaluation metric is Code Quality which is a measure of the negative KL divergence between the target and the predicted probability distributions (higher is better). Appendix B.6 includes a detailed description of the data set and this performance metric.

Table 8 summarizes the performance of different models on this data set. Xiong et al. (2011) used Bayesian neural nets for this task. As expected, we found that Bayesian neural nets perform better than dropout. However, we see that dropout improves significantly upon the performance of standard neural nets and outperforms all other methods. The challenge in this data set is to prevent overfitting since the size of the training set is small. One way to prevent overfitting is to reduce the input dimensionality using PCA. Thereafter, standard techniques such as SVMs or logistic regression can be used. However, with dropout we were able to prevent overfitting without the need to do dimensionality reduction. The dropout nets are very large (1000s of hidden units) compared to a few tens of units in the Bayesian network. This shows that dropout has a strong regularizing effect.

6.5 Comparison with Standard Regularizers

Several regularization methods have been proposed for preventing overfitting in neural networks. These include L2 weight decay (more generally Tikhonov regularization (Tikhonov, 1943)), lasso (Tibshirani, 1996), KL-sparsity and max-norm regularization. Dropout can be seen as another way of regularizing neural networks. In this section we compare dropout with some of these regularization methods using the MNIST data set.

The same network architecture (784-1024-1024-2048-10) with ReLUs was trained using stochastic gradient descent with different regularizations. Table 9 shows the results. The values of different hyperparameters associated with each kind of regularization (decay constants, target sparsity, dropout rate, max-norm upper bound) were obtained using a validation set. We found that dropout combined with max-norm regularization gives the lowest generalization error.

7. Salient Features

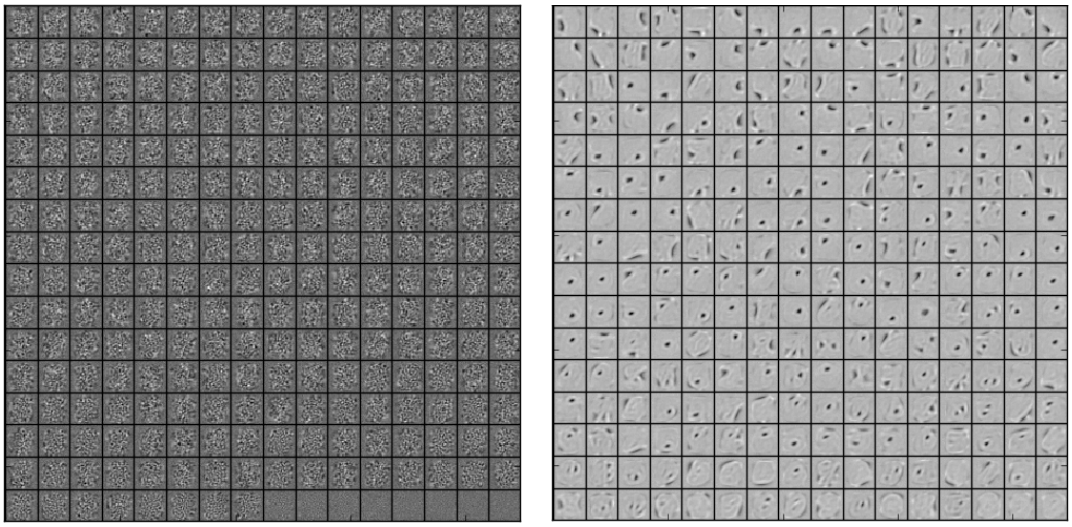
The experiments described in the previous section provide strong evidence that dropout is a useful technique for improving neural networks. In this section, we closely examine how dropout affects a neural network. We analyze the effect of dropout on the quality of features produced. We see how dropout affects the sparsity of hidden unit activations. We

Method	Test Classification error %
L2	1.62
L2 + L1 applied towards the end of training	1.60
L2 + KL-sparsity	1.55
Max-norm	1.35
Dropout + L2	1.25
Dropout + Max-norm	**1.05**

Table 9: Comparison of different regularization methods on MNIST.

also see how the advantages obtained from dropout vary with the probability of retaining units, size of the network and the size of the training set. These observations give some insight into why dropout works so well.

7.1 Effect on Features



(a) Without dropout

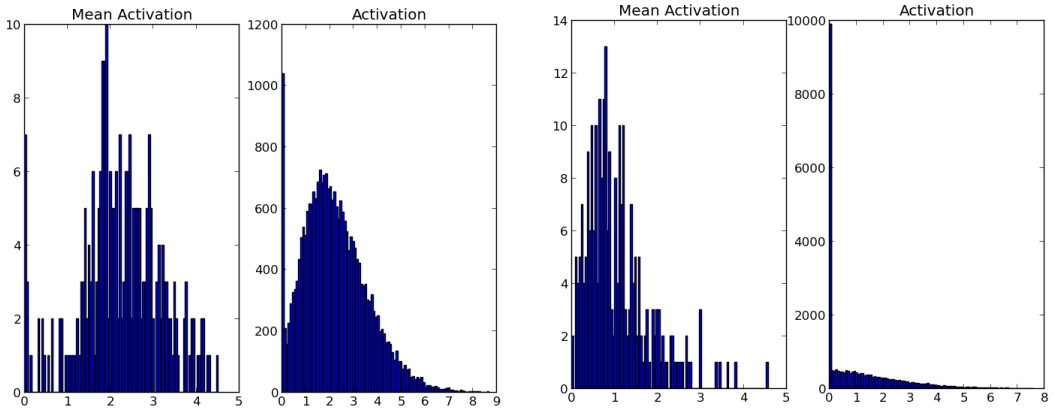
(b) Dropout with $p = 0.5$.

Figure 7: Features learned on MNIST with one hidden layer autoencoders having 256 rectified linear units.

In a standard neural network, the derivative received by each parameter tells it how it should change so the final loss function is reduced, *given* what all other units are doing. Therefore, units may change in a way that they fix up the mistakes of the other units. This may lead to complex co-adaptations. This in turn leads to overfitting because these co-adaptations do not generalize to unseen data. We hypothesize that for each hidden unit, dropout prevents co-adaptation by making the presence of other hidden units unreliable. Therefore, a hidden unit cannot rely on other specific units to correct its mistakes. It must perform well in a wide variety of different contexts provided by the other hidden units. To observe this effect directly, we look at the first level features learned by neural networks trained on visual tasks with and without dropout.

Figure 7a shows features learned by an autoencoder on MNIST with a single hidden layer of 256 rectified linear units without dropout. Figure 7b shows the features learned by an identical autoencoder which used dropout in the hidden layer with $p = 0.5$. Both autoencoders had similar test reconstruction errors. However, it is apparent that the features shown in Figure 7a have co-adapted in order to produce good reconstructions. Each hidden unit on its own does not seem to be detecting a meaningful feature. On the other hand, in Figure 7b, the hidden units seem to detect edges, strokes and spots in different parts of the image. This shows that dropout does break up co-adaptations, which is probably the main reason why it leads to lower generalization errors.

7.2 Effect on Sparsity



(a) Without dropout

(b) Dropout with $p = 0.5$.

Figure 8: Effect of dropout on sparsity. ReLUs were used for both models. **Left**: The histogram of mean activations shows that most units have a mean activation of about 2.0. The histogram of activations shows a huge mode away from zero. Clearly, a large fraction of units have high activation. **Right**: The histogram of mean activations shows that most units have a smaller mean mean activation of about 0.7. The histogram of activations shows a sharp peak at zero. Very few units have high activation.

We found that as a side-effect of doing dropout, the activations of the hidden units become sparse, even when no sparsity inducing regularizers are present. Thus, dropout automatically leads to sparse representations. To observe this effect, we take the autoencoders trained in the previous section and look at the sparsity of hidden unit activations on a random mini-batch taken from the test set. Figure 8a and Figure 8b compare the sparsity for the two models. In a good sparse model, there should only be a few highly activated units for any data case. Moreover, the average activation of any unit across data cases should be low. To assess both of these qualities, we plot two histograms for each model. For each model, the histogram on the left shows the distribution of mean activations of hidden units across the minibatch. The histogram on the right shows the distribution of activations of the hidden units.

Comparing the histograms of activations we can see that fewer hidden units have high activations in Figure 8b compared to Figure 8a, as seen by the significant mass away from

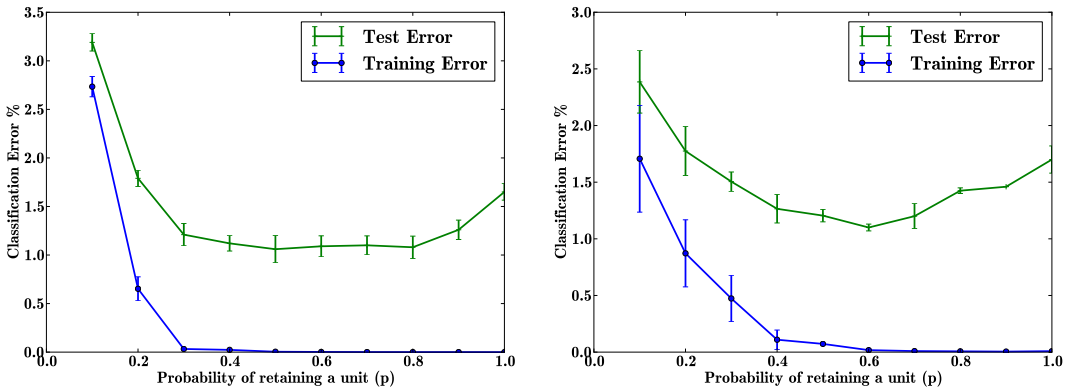
zero for the net that does not use dropout. The mean activations are also smaller for the dropout net. The overall mean activation of hidden units is close to 2.0 for the autoencoder without dropout but drops to around 0.7 when dropout is used.

7.3 Effect of Dropout Rate

Dropout has a tunable hyperparameter p (the probability of retaining a unit in the network). In this section, we explore the effect of varying this hyperparameter. The comparison is done in two situations.

1. The number of hidden units is held constant.
2. The number of hidden units is changed so that the expected number of hidden units that will be retained after dropout is held constant.

In the first case, we train the same network architecture with different amounts of dropout. We use a 784-2048-2048-2048-10 architecture. No input dropout was used. Figure 9a shows the test error obtained as a function of p. If the architecture is held constant, having a small p means very few units will turn on during training. It can be seen that this has led to underfitting since the training error is also high. We see that as p increases, the error goes down. It becomes flat when $0.4 \leq p \leq 0.8$ and then increases as p becomes close to 1.



(a) Keeping n fixed. (b) Keeping pn fixed.

Figure 9: Effect of changing dropout rates on MNIST.

Another interesting setting is the second case in which the quantity pn is held constant where n is the number of hidden units in any particular layer. This means that networks that have small p will have a large number of hidden units. Therefore, after applying dropout, the expected number of units that are present will be the same across different architectures. However, the test networks will be of different sizes. In our experiments, we set $pn = 256$ for the first two hidden layers and $pn = 512$ for the last hidden layer. Figure 9b shows the test error obtained as a function of p. We notice that the magnitude of errors for small values of p has reduced by a lot compared to Figure 9a (for $p = 0.1$ it fell from 2.7% to 1.7%). Values of p that are close to 0.6 seem to perform best for this choice of pn but our usual default value of 0.5 is close to optimal.

7.4 Effect of Data Set Size

One test of a good regularizer is that it should make it possible to get good generalization error from models with a large number of parameters trained on small data sets. This section explores the effect of changing the data set size when dropout is used with feed-forward networks. Huge neural networks trained in the standard way overfit massively on small data sets. To see if dropout can help, we run classification experiments on MNIST and vary the amount of data given to the network.

The results of these experiments are shown in Figure 10. The network was given data sets of size 100, 500, 1K, 5K, 10K and 50K chosen randomly from the MNIST training set. The same network architecture (784-1024-1024-2048-10) was used for all data sets. Dropout with $p = 0.5$ was performed at all the hidden layers and $p = 0.8$ at the input layer. It can be observed that for extremely small data sets (100, 500) dropout does not give any improvements. The model has enough parameters that it can overfit on the training data, even with all the noise coming from dropout. As the size of the data set is increased, the gain from doing dropout increases up to a point and then declines. This suggests that for any given architecture and dropout rate, there is a "sweet spot" corresponding to some amount of data that is large enough to not be memorized in spite of the noise but not so large that overfitting is not a problem anyways.

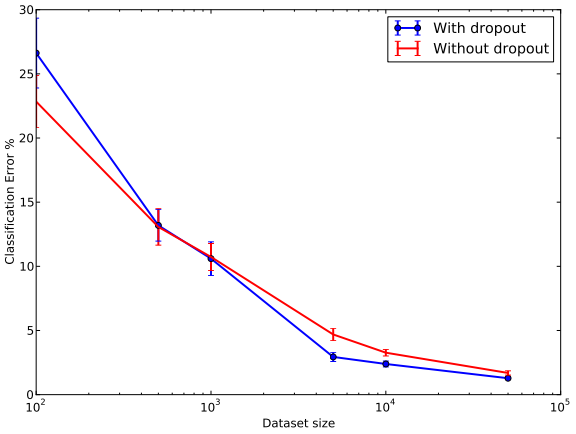


Figure 10: Effect of varying data set size.

7.5 Monte-Carlo Model Averaging vs. Weight Scaling

The efficient test time procedure that we propose is to do an approximate model combination by scaling down the weights of the trained neural network. An expensive but more correct way of averaging the models is to sample k neural nets using dropout for each test case and average their predictions. As $k \to \infty$, this Monte-Carlo model average gets close to the true model average. It is interesting to see empirically how many samples k are needed to match the performance of the approximate averaging method. By computing the error for different values of k we can see how quickly the error rate of the finite-sample average approaches the error rate of the true model average.

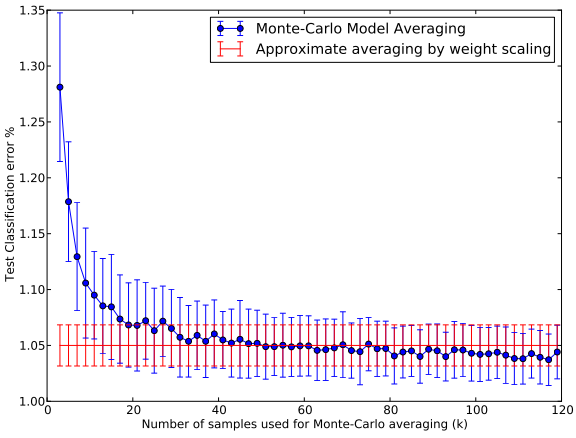


Figure 11: Monte-Carlo model averaging vs. weight scaling.

We again use the MNIST data set and do classification by averaging the predictions of k randomly sampled neural networks. Figure 11 shows the test error rate obtained for different values of k. This is compared with the error obtained using the weight scaling method (shown as a horizontal line). It can be seen that around $k = 50$, the Monte-Carlo method becomes as good as the approximate method. Thereafter, the Monte-Carlo method is slightly better than the approximate method but well within one standard deviation of it. This suggests that the weight scaling method is a fairly good approximation of the true model average.

8. Dropout Restricted Boltzmann Machines

Besides feed-forward neural networks, dropout can also be applied to Restricted Boltzmann Machines (RBM). In this section, we formally describe this model and show some results to illustrate its key properties.

8.1 Model Description

Consider an RBM with visible units $\mathbf{v} \in \{0,1\}^D$ and hidden units $\mathbf{h} \in \{0,1\}^F$. It defines the following probability distribution

$$P(\mathbf{h}, \mathbf{v}; \theta) = \frac{1}{\mathcal{Z}(\theta)} \exp(\mathbf{v}^\top W \mathbf{h} + \mathbf{a}^\top \mathbf{h} + \mathbf{b}^\top \mathbf{v}).$$

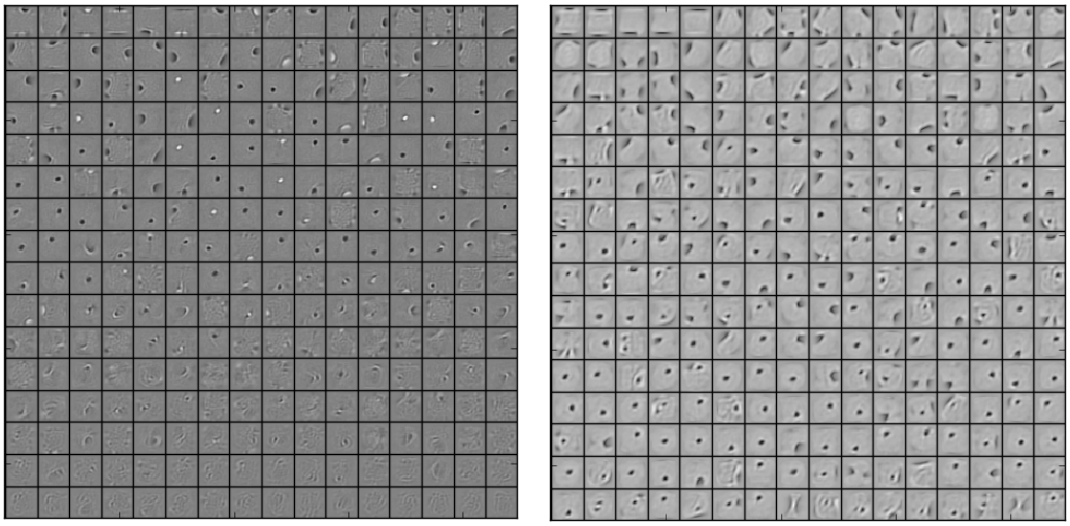
Where $\theta = \{W, \mathbf{a}, \mathbf{b}\}$ represents the model parameters and $\mathcal{Z}$ is the partition function.

Dropout RBMs are RBMs augmented with a vector of binary random variables $\mathbf{r} \in \{0,1\}^F$. Each random variable r_j takes the value 1 with probability p, independent of others. If r_j takes the value 1, the hidden unit h_j is retained, otherwise it is dropped from the model. The joint distribution defined by a Dropout RBM can be expressed as

$$\begin{aligned}
P(\mathbf{r}, \mathbf{h}, \mathbf{v}; p, \theta) &= P(\mathbf{r}; p) P(\mathbf{h}, \mathbf{v} | \mathbf{r}; \theta), \\
P(\mathbf{r}; p) &= \prod_{j=1}^{F} p^{r_j} (1-p)^{1-r_j}, \\
P(\mathbf{h}, \mathbf{v} | \mathbf{r}; \theta) &= \frac{1}{\mathcal{Z}'(\theta, \mathbf{r})} \exp(\mathbf{v}^\top W \mathbf{h} + \mathbf{a}^\top \mathbf{h} + \mathbf{b}^\top \mathbf{v}) \prod_{j=1}^{F} g(h_j, r_j), \\
g(h_j, r_j) &= \mathbb{1}(r_j = 1) + \mathbb{1}(r_j = 0)\mathbb{1}(h_j = 0).
\end{aligned}$$

$\mathcal{Z}'(\theta, \mathbf{r})$ is the normalization constant. $g(h_j, r_j)$ imposes the constraint that if $r_j = 0$, h_j must be 0. The distribution over $\mathbf{h}$, conditioned on $\mathbf{v}$ and $\mathbf{r}$ is factorial

$$\begin{aligned}
P(\mathbf{h} | \mathbf{r}, \mathbf{v}) &= \prod_{j=1}^{F} P(h_j | r_j, \mathbf{v}), \\
P(h_j = 1 | r_j, \mathbf{v}) &= \mathbb{1}(r_j = 1) \sigma\left(b_j + \sum_i W_{ij} v_i\right).
\end{aligned}$$



(a) Without dropout (b) Dropout with $p = 0.5$.

Figure 12: Features learned on MNIST by 256 hidden unit RBMs. The features are ordered by L2 norm.

The distribution over $\mathbf{v}$ conditioned on $\mathbf{h}$ is same as that of an RBM

$$P(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{D} P(v_i|\mathbf{h}),$$

$$P(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_j W_{ij} h_j\right).$$

Conditioned on $\mathbf{r}$, the distribution over $\{\mathbf{v}, \mathbf{h}\}$ is same as the distribution that an RBM would impose, except that the units for which $r_j = 0$ are dropped from $\mathbf{h}$. Therefore, the Dropout RBM model can be seen as a mixture of exponentially many RBMs with shared weights each using a different subset of $\mathbf{h}$.

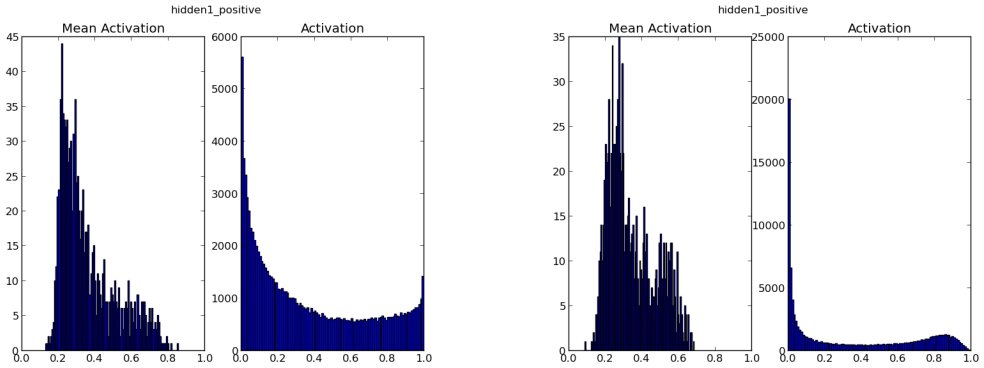
8.2 Learning Dropout RBMs

Learning algorithms developed for RBMs such as Contrastive Divergence (Hinton et al., 2006) can be directly applied for learning Dropout RBMs. The only difference is that $\mathbf{r}$ is first sampled and only the hidden units that are retained are used for training. Similar to dropout neural networks, a different $\mathbf{r}$ is sampled for each training case in every minibatch. In our experiments, we use CD-1 for training dropout RBMs.

8.3 Effect on Features

Dropout in feed-forward networks improved the quality of features by reducing co-adaptations. This section explores whether this effect transfers to Dropout RBMs as well.

Figure 12a shows features learned by a binary RBM with 256 hidden units. Figure 12b shows features learned by a dropout RBM with the same number of hidden units. Features



(a) Without dropout

(b) Dropout with $p = 0.5$.

Figure 13: Effect of dropout on sparsity. **Left**: The activation histogram shows that a large number of units have activations away from zero. **Right**: A large number of units have activations close to zero and very few units have high activation.

learned by the dropout RBM appear qualitatively different in the sense that they seem to capture features that are coarser compared to the sharply defined stroke-like features in the standard RBM. There seem to be very few dead units in the dropout RBM relative to the standard RBM.

8.4 Effect on Sparsity

Next, we investigate the effect of dropout RBM training on sparsity of the hidden unit activations. Figure 13a shows the histograms of hidden unit activations and their means on a test mini-batch after training an RBM. Figure 13b shows the same for dropout RBMs. The histograms clearly indicate that the dropout RBMs learn much sparser representations than standard RBMs even when no additional sparsity inducing regularizer is present.

9. Marginalizing Dropout

Dropout can be seen as a way of adding noise to the states of hidden units in a neural network. In this section, we explore the class of models that arise as a result of marginalizing this noise. These models can be seen as deterministic versions of dropout. In contrast to standard ("Monte-Carlo") dropout, these models do not need random bits and it is possible to get gradients for the marginalized loss functions. In this section, we briefly explore these models.

Deterministic algorithms have been proposed that try to learn models that are robust to feature deletion at test time (Globerson and Roweis, 2006). Marginalization in the context of denoising autoencoders has been explored previously (Chen et al., 2012). The marginalization of dropout noise in the context of linear regression was discussed in Srivastava (2013). Wang and Manning (2013) further explored the idea of marginalizing dropout to speed-up training. van der Maaten et al. (2013) investigated different input noise distributions and

the regularizers obtained by marginalizing this noise. Wager et al. (2013) describes how dropout can be seen as an adaptive regularizer.

9.1 Linear Regression

First we explore a very simple case of applying dropout to the classical problem of linear regression. Let $X \in \mathbb{R}^{N \times D}$ be a data matrix of N data points. $\mathbf{y} \in \mathbb{R}^N$ be a vector of targets. Linear regression tries to find a $\mathbf{w} \in \mathbb{R}^D$ that minimizes

$$||\mathbf{y} - X\mathbf{w}||^2.$$

When the input X is dropped out such that any input dimension is retained with probability p, the input can be expressed as $R * X$ where $R \in \{0, 1\}^{N \times D}$ is a random matrix with $R_{ij} \sim \text{Bernoulli}(p)$ and $*$ denotes an element-wise product. Marginalizing the noise, the objective function becomes

$$\underset{\mathbf{w}}{\text{minimize}} \quad \mathbb{E}_{R \sim \text{Bernoulli(p)}} \left[||\mathbf{y} - (R * X)\mathbf{w}||^2 \right].$$

This reduces to

$$\underset{\mathbf{w}}{\text{minimize}} \quad ||\mathbf{y} - pX\mathbf{w}||^2 + p(1-p)||\Gamma\mathbf{w}||^2,$$

where $\Gamma = (\text{diag}(X^\top X))^{1/2}$. Therefore, dropout with linear regression is equivalent, in expectation, to ridge regression with a particular form for Γ. This form of Γ essentially scales the weight cost for weight w_i by the standard deviation of the ith dimension of the data. If a particular data dimension varies a lot, the regularizer tries to squeeze its weight more.

Another interesting way to look at this objective is to absorb the factor of p into $\mathbf{w}$. This leads to the following form

$$\underset{\mathbf{w}}{\text{minimize}} \quad ||\mathbf{y} - X\widetilde{\mathbf{w}}||^2 + \frac{1-p}{p}||\Gamma\widetilde{\mathbf{w}}||^2,$$

where $\widetilde{\mathbf{w}} = p\mathbf{w}$. This makes the dependence of the regularization constant on p explicit. For p close to 1, all the inputs are retained and the regularization constant is small. As more dropout is done (by decreasing p), the regularization constant grows larger.

9.2 Logistic Regression and Deep Networks

For logistic regression and deep neural nets, it is hard to obtain a closed form marginalized model. However, Wang and Manning (2013) showed that in the context of dropout applied to logistic regression, the corresponding marginalized model can be trained approximately. Under reasonable assumptions, the distributions over the inputs to the logistic unit and over the gradients of the marginalized model are Gaussian. Their means and variances can be computed efficiently. This approximate marginalization outperforms Monte-Carlo dropout in terms of training time and generalization performance.

However, the assumptions involved in this technique become successively weaker as more layers are added. Therefore, the results are not directly applicable to deep networks.

Data Set	Architecture	Bernoulli dropout	Gaussian dropout
MNIST	2 layers, 1024 units each	1.08 ± 0.04	0.95 ± 0.04
CIFAR-10	3 conv + 2 fully connected layers	12.6 ± 0.1	12.5 ± 0.1

Table 10: Comparison of classification error % with Bernoulli and Gaussian dropout. For MNIST, the Bernoulli model uses $p = 0.5$ for the hidden units and $p = 0.8$ for the input units. For CIFAR-10, we use $p = (0.9, 0.75, 0.75, 0.5, 0.5, 0.5)$ going from the input layer to the top. The value of σ for the Gaussian dropout models was set to be $\sqrt{\frac{1-p}{p}}$. Results were averaged over 10 different random seeds.

10. Multiplicative Gaussian Noise

Dropout involves multiplying hidden activations by Bernoulli distributed random variables which take the value 1 with probability p and 0 otherwise. This idea can be generalized by multiplying the activations with random variables drawn from other distributions. We recently discovered that multiplying by a random variable drawn from $\mathcal{N}(1, 1)$ works just as well, or perhaps better than using Bernoulli noise. This new form of dropout amounts to adding a Gaussian distributed random variable with zero mean and standard deviation equal to the activation of the unit. That is, each hidden activation h_i is perturbed to $h_i + h_i r$ where $r \sim \mathcal{N}(0, 1)$, or equivalently $h_i r'$ where $r' \sim \mathcal{N}(1, 1)$. We can generalize this to $r' \sim \mathcal{N}(1, \sigma^2)$ where σ becomes an additional hyperparameter to tune, just like p was in the standard (Bernoulli) dropout. The expected value of the activations remains unchanged, therefore no weight scaling is required at test time.

In this paper, we described dropout as a method where we retain units with probability p at training time and scale down the weights by multiplying them by a factor of p at test time. Another way to achieve the same effect is to scale up the retained activations by multiplying by $1/p$ at training time and not modifying the weights at test time. These methods are equivalent with appropriate scaling of the learning rate and weight initializations at each layer.

Therefore, dropout can be seen as multiplying h_i by a Bernoulli random variable r_b that takes the value $1/p$ with probability p and 0 otherwise. $E[r_b] = 1$ and $Var[r_b] = (1 - p)/p$. For the Gaussian multiplicative noise, if we set $\sigma^2 = (1 - p)/p$, we end up multiplying h_i by a random variable r_g, where $E[r_g] = 1$ and $Var[r_g] = (1 - p)/p$. Therefore, both forms of dropout can be set up so that the random variable being multiplied by has the same mean and variance. However, given these first and second order moments, r_g has the highest entropy and r_b has the lowest. Both these extremes work well, although preliminary experimental results shown in Table 10 suggest that the high entropy case might work slightly better. For each layer, the value of σ in the Gaussian model was set to be $\sqrt{\frac{1-p}{p}}$ using the p from the corresponding layer in the Bernoulli model.

11. Conclusion

Dropout is a technique for improving neural networks by reducing overfitting. Standard backpropagation learning builds up brittle co-adaptations that work for the training data but do not generalize to unseen data. Random dropout breaks up these co-adaptations by

making the presence of any particular hidden unit unreliable. This technique was found to improve the performance of neural nets in a wide variety of application domains including object classification, digit recognition, speech recognition, document classification and analysis of computational biology data. This suggests that dropout is a general technique and is not specific to any domain. Methods that use dropout achieve state-of-the-art results on SVHN, ImageNet, CIFAR-100 and MNIST. Dropout considerably improved the performance of standard neural nets on other data sets as well.

This idea can be extended to Restricted Boltzmann Machines and other graphical models. The central idea of dropout is to take a large model that overfits easily and repeatedly sample and train smaller sub-models from it. RBMs easily fit into this framework. We developed Dropout RBMs and empirically showed that they have certain desirable properties.

One of the drawbacks of dropout is that it increases training time. A dropout network typically takes 2-3 times longer to train than a standard neural network of the same architecture. A major cause of this increase is that the parameter updates are very noisy. Each training case effectively tries to train a different random architecture. Therefore, the gradients that are being computed are not gradients of the final architecture that will be used at test time. Therefore, it is not surprising that training takes a long time. However, it is likely that this stochasticity prevents overfitting. This creates a trade-off between overfitting and training time. With more training time, one can use high dropout and suffer less overfitting. However, one way to obtain some of the benefits of dropout without stochasticity is to marginalize the noise to obtain a regularizer that does the same thing as the dropout procedure, in expectation. We showed that for linear regression this regularizer is a modified form of L2 regularization. For more complicated models, it is not obvious how to obtain an equivalent regularizer. Speeding up dropout is an interesting direction for future work.

Acknowledgments

This research was supported by OGS, NSERC and an Early Researcher Award.

Appendix A. A Practical Guide for Training Dropout Networks

Neural networks are infamous for requiring extensive hyperparameter tuning. Dropout networks are no exception. In this section, we describe heuristics that might be useful for applying dropout.

A.1 Network Size

It is to be expected that dropping units will reduce the capacity of a neural network. If n is the number of hidden units in any layer and p is the probability of retaining a unit, then instead of n hidden units, only pn units will be present after dropout, in expectation. Moreover, this set of pn units will be different each time and the units are not allowed to build co-adaptations freely. Therefore, if an n-sized layer is optimal for a standard neural net on any given task, a good dropout net should have at least n/p units. We found this to be a useful heuristic for setting the number of hidden units in both convolutional and fully connected networks.

A.2 Learning Rate and Momentum

Dropout introduces a significant amount of noise in the gradients compared to standard stochastic gradient descent. Therefore, a lot of gradients tend to cancel each other. In order to make up for this, a dropout net should typically use 10-100 times the learning rate that was optimal for a standard neural net. Another way to reduce the effect the noise is to use a high momentum. While momentum values of 0.9 are common for standard nets, with dropout we found that values around 0.95 to 0.99 work quite a lot better. Using high learning rate and/or momentum significantly speed up learning.

A.3 Max-norm Regularization

Though large momentum and learning rate speed up learning, they sometimes cause the network weights to grow very large. To prevent this, we can use max-norm regularization. This constrains the norm of the vector of incoming weights at each hidden unit to be bound by a constant c. Typical values of c range from 3 to 4.

A.4 Dropout Rate

Dropout introduces an extra hyperparameter—the probability of retaining a unit p. This hyperparameter controls the intensity of dropout. $p = 1$, implies no dropout and low values of p mean more dropout. Typical values of p for hidden units are in the range 0.5 to 0.8. For input layers, the choice depends on the kind of input. For real-valued inputs (image patches or speech frames), a typical value is 0.8. For hidden layers, the choice of p is coupled with the choice of number of hidden units n. Smaller p requires big n which slows down the training and leads to underfitting. Large p may not produce enough dropout to prevent overfitting.

Appendix B. Detailed Description of Experiments and Data Sets

.

This section describes the network architectures and training details for the experimental results reported in this paper. The code for reproducing these results can be obtained from `http://www.cs.toronto.edu/~nitish/dropout`. The implementation is GPU-based. We used the excellent CUDA libraries—cudamat (Mnih, 2009) and cuda-convnet (Krizhevsky et al., 2012) to implement our networks.

B.1 MNIST

The MNIST data set consists of 60,000 training and 10,000 test examples each representing a 28×28 digit image. We held out 10,000 random training images for validation. Hyperparameters were tuned on the validation set such that the best validation error was produced after 1 million weight updates. The validation set was then combined with the training set and training was done for 1 million weight updates. This net was used to evaluate the performance on the test set. This way of using the validation set was chosen because we found that it was easy to set up hyperparameters so that early stopping was not required at all. Therefore, once the hyperparameters were fixed, it made sense to combine the validation and training sets and train for a very long time.

The architectures shown in Figure 4 include all combinations of 2, 3, and 4 layer networks with 1024 and 2048 units in each layer. Thus, there are six architectures in all. For all the architectures (including the ones reported in Table 2), we used $p = 0.5$ in all hidden layers and $p = 0.8$ in the input layer. A final momentum of 0.95 and weight constraints with $c = 2$ was used in all the layers.

To test the limits of dropout's regularization power, we also experimented with 2 and 3 layer nets having 4096 and 8192 units. 2 layer nets gave improvements as shown in Table 2. However, the three layer nets performed slightly worse than 2 layer ones with the same level of dropout. When we increased dropout, performance improved but not enough to outperform the 2 layer nets.

B.2 SVHN

The SVHN data set consists of approximately 600,000 training images and 26,000 test images. The training set consists of two parts—A standard labeled training set and another set of labeled examples that are easy. A validation set was constructed by taking examples from both the parts. Two-thirds of it were taken from the standard set (400 per class) and one-third from the extra set (200 per class), a total of 6000 samples. This same process is used by Sermanet et al. (2012). The inputs were RGB pixels normalized to have zero mean and unit variance. Other preprocessing techniques such as global or local contrast normalization or ZCA whitening did not give any noticeable improvements.

The best architecture that we found uses three convolutional layers each followed by a max-pooling layer. The convolutional layers have 96, 128 and 256 filters respectively. Each convolutional layer has a 5×5 receptive field applied with a stride of 1 pixel. Each max pooling layer pools 3×3 regions at strides of 2 pixels. The convolutional layers are followed by two fully connected hidden layers having 2048 units each. All units use the rectified linear activation function. Dropout was applied to all the layers of the network with the probability of retaining the unit being $p = (0.9, 0.75, 0.75, 0.5, 0.5, 0.5)$ for the different layers of the network (going from input to convolutional layers to fully connected layers). In addition, the max-norm constraint with $c = 4$ was used for all the weights. A momentum of 0.95 was used in all the layers. These hyperparameters were tuned using a validation set. Since the training set was quite large, we did not combine the validation set with the training set for final training. We reported test error of the model that had smallest validation error.

B.3 CIFAR-10 and CIFAR-100

The CIFAR-10 and CIFAR-100 data sets consists of 50,000 training and 10,000 test images each. They have 10 and 100 image categories respectively. These are 32×32 color images. We used 5,000 of the training images for validation. We followed the procedure similar to MNIST, where we found the best hyperparameters using the validation set and then combined it with the training set. The images were preprocessed by doing global contrast normalization in each color channel followed by ZCA whitening. Global contrast normalization means that for image and each color channel in that image, we compute the mean of the pixel intensities and subtract it from the channel. ZCA whitening means that we mean center the data, rotate it onto its principle components, normalize each component

and then rotate it back. The network architecture and dropout rates are same as that for SVHN, except the learning rates for the input layer which had to be set to smaller values.

B.4 TIMIT

The open source Kaldi toolkit (Povey et al., 2011) was used to preprocess the data into log-filter banks. A monophone system was trained to do a forced alignment and to get labels for speech frames. Dropout neural networks were trained on windows of 21 consecutive frames to predict the label of the central frame. No speaker dependent operations were performed. The inputs were mean centered and normalized to have unit variance.

We used probability of retention $p = 0.8$ in the input layers and 0.5 in the hidden layers. Max-norm constraint with $c = 4$ was used in all the layers. A momentum of 0.95 with a high learning rate of 0.1 was used. The learning rate was decayed as $\epsilon_0(1 + t/T)^{-1}$. For DBN pretraining, we trained RBMs using CD-1. The variance of each input unit for the Gaussian RBM was fixed to 1. For finetuning the DBN with dropout, we found that in order to get the best results it was important to use a smaller learning rate (about 0.01). Adding max-norm constraints did not give any improvements.

B.5 Reuters

The Reuters RCV1 corpus contains more than 800,000 documents categorized into 103 classes. These classes are arranged in a tree hierarchy. We created a subset of this data set consisting of 402,738 articles and a vocabulary of 2000 words comprising of 50 categories in which each document belongs to exactly one class. The data was split into equal sized training and test sets. We tried many network architectures and found that dropout gave improvements in classification accuracy over all of them. However, the improvement was not as significant as that for the image and speech data sets. This might be explained by the fact that this data set is quite big (more than 200,000 training examples) and overfitting is not a very serious problem.

B.6 Alternative Splicing

The alternative splicing data set consists of data for 3665 cassette exons, 1014 RNA features and 4 tissue types derived from 27 mouse tissues. For each input, the target consists of 4 softmax units (one for tissue type). Each softmax unit has 3 states (*inc*, *exc*, *nc*) which are of the biological importance. For each softmax unit, the aim is to predict a distribution over these 3 states that matches the observed distribution from wet lab experiments as closely as possible. The evaluation metric is Code Quality which is defined as

$$\sum_{i=1}^{|\text{data points}|} \sum_{t \in \text{tissue types}} \sum_{s \in \{\text{inc, exc, nc}\}} p_{i,t}^s \log(\frac{q_t^s(r_i)}{\bar{p}^s}),$$

where, $p_{i,t}^s$ is the target probability for state s and tissue type t in input i; $q_t^s(r_i)$ is the predicted probability for state s in tissue type t for input r_i and $\bar{p}^s$ is the average of $p_{i,t}^s$ over i and t.

A two layer dropout network with 1024 units in each layer was trained on this data set. A value of $p = 0.5$ was used for the hidden layer and $p = 0.7$ for the input layer. Max-norm regularization with high decaying learning rates was used. Results were averaged across the same 5 folds used by Xiong et al. (2011).

References

M. Chen, Z. Xu, K. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning*, pages 767–774. ACM, 2012.

G. E. Dahl, M. Ranzato, A. Mohamed, and G. E. Hinton. Phone recognition with the mean-covariance restricted Boltzmann machine. In *Advances in Neural Information Processing Systems 23*, pages 469–477, 2010.

O. Dekel, O. Shamir, and L. Xiao. Learning to classify with missing and corrupted features. *Machine Learning*, 81(2):149–178, 2010.

A. Globerson and S. Roweis. Nightmare at test time: robust learning by feature deletion. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 353–360. ACM, 2006.

I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1319–1327. ACM, 2013.

G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *Proceedings of the International Conference on Computer Vision (ICCV'09)*. IEEE, 2009.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, Z. Li, M.-H. Tsai, X. Zhou, T. Huang, and T. Zhang. Imagenet classification: fast descriptor coding and large-scale svm training. Large scale visual recognition challenge, 2010.

A. Livnat, C. Papadimitriou, N. Pippenger, and M. W. Feldman. Sex, mixability, and modularity. *Proceedings of the National Academy of Sciences*, 107(4):1452–1457, 2010.

V. Mnih. CUDAMat: a CUDA-based matrix class for Python. Technical Report UTML TR 2009-004, Department of Computer Science, University of Toronto, November 2009.

A. Mohamed, G. E. Dahl, and G. E. Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 2010.

R. M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., 1996.

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*, 2011.

S. J. Nowlan and G. E. Hinton. Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4(4), 1992.

D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely. The Kaldi Speech Recognition Toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, 2011.

R. Salakhutdinov and G. Hinton. Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455, 2009.

R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning*. ACM, 2008.

J. Sanchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1665–1672, 2011.

P. Sermanet, S. Chintala, and Y. LeCun. Convolutional neural networks applied to house numbers digit classification. In *International Conference on Pattern Recognition (ICPR 2012)*, 2012.

P. Simard, D. Steinkraus, and J. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, volume 2, pages 958–962, 2003.

J. Snoek, H. Larochelle, and R. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, pages 2960–2968, 2012.

N. Srebro and A. Shraibman. Rank, trace-norm and max-norm. In *Proceedings of the 18th annual conference on Learning Theory*, COLT'05, pages 545–560. Springer-Verlag, 2005.

N. Srivastava. Improving Neural Networks with Dropout. Master's thesis, University of Toronto, January 2013.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B. Methodological*, 58(1):267–288, 1996.

A. N. Tikhonov. On the stability of inverse problems. *Doklady Akademii Nauk SSSR*, 39(5): 195–198, 1943.

L. van der Maaten, M. Chen, S. Tyree, and K. Q. Weinberger. Learning with marginalized corrupted features. In *Proceedings of the 30th International Conference on Machine Learning*, pages 410–418. ACM, 2013.

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. ACM, 2008.

P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. In *Proceedings of the 27th International Conference on Machine Learning*, pages 3371–3408. ACM, 2010.

S. Wager, S. Wang, and P. Liang. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems 26*, pages 351–359, 2013.

S. Wang and C. D. Manning. Fast dropout training. In *Proceedings of the 30th International Conference on Machine Learning*, pages 118–126. ACM, 2013.

H. Y. Xiong, Y. Barash, and B. J. Frey. Bayesian prediction of tissue-regulated splicing using RNA sequence and cellular context. *Bioinformatics*, 27(18):2554–2562, 2011.

M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *CoRR*, abs/1301.3557, 2013.

Journal of Machine Learning Research 15 (2014) 1959-2008 Submitted 10/12; Revised 7/13; Published 6/14

Sparse Factor Analysis for Learning and Content Analytics

Andrew S. Lan* MR.LAN@SPARFA.COM
Andrew E. Waters WATERS@SPARFA.COM
Dept. Electrical and Computer Engineering
Rice University
Houston, TX 77005, USA
Christoph Studer STUDER@SPARFA.COM
School of Electrical and Computer Engineering
Cornell University
Ithaca, NY 14853, USA
Richard G. Baraniuk RICHB@SPARFA.COM
Dept. Electrical and Computer Engineering
Rice University
Houston, TX 77005, USA

Editor: Francis Bach

Abstract

We develop a new model and algorithms for machine learning-based *learning analytics*, which estimate a learner's knowledge of the concepts underlying a domain, and *content analytics*, which estimate the relationships among a collection of questions and those concepts. Our model represents the probability that a learner provides the correct response to a question in terms of three factors: their understanding of a set of underlying concepts, the concepts involved in each question, and each question's intrinsic difficulty. We estimate these factors given the graded responses to a collection of questions. The underlying estimation problem is ill-posed in general, especially when only a subset of the questions are answered. The key observation that enables a well-posed solution is the fact that typical educational domains of interest involve only a small number of key concepts. Leveraging this observation, we develop both a bi-convex maximum-likelihood-based solution and a Bayesian solution to the resulting *SPARse Factor Analysis* (SPARFA) problem. We also incorporate user-defined tags on questions to facilitate the interpretability of the estimated factors. Experiments with synthetic and real-world data demonstrate the efficacy of our approach. Finally, we make a connection between SPARFA and noisy, binary-valued (1-bit) dictionary learning that is of independent interest.

Keywords: factor analysis, sparse probit regression, sparse logistic regression, Bayesian latent factor analysis, personalized learning

1. Introduction

Textbooks, lectures, and homework assignments were the answer to the main educational challenges of the 19th century, but they are the main bottleneck of the 21st century. Today's textbooks are static, linearly organized, time-consuming to develop, soon out-of-date,

*. First authorship determined by coin toss

and expensive. Lectures remain a primarily passive experience of copying down what an instructor says and writes on a board (or projects on a screen). Homework assignments that are not graded for weeks provide poor feedback to learners (e.g., students) on their learning progress. Even more importantly, today's courses provide only a "one-size-fits-all" learning experience that does not cater to the background, interests, and goals of individual learners.

1.1 The Promise of Personalized Learning

We envision a world where access to high-quality, personally tailored educational experiences is affordable to all of the world's learners. The key to reaching this goal is to integrate textbooks, lectures, and homework assignments into a *personalized learning system* (PLS) that closes the learning feedback loop by (i) continuously monitoring and analyzing learner interactions with learning resources in order to assess their learning progress and (ii) providing timely remediation, enrichment, or practice based on that analysis. See Linden and Glas (2000), Murray et al. (2004), Stamper et al. (2007), Rafferty et al. (2011), Li et al. (2011), and Knewton (2012) for various visions and examples.

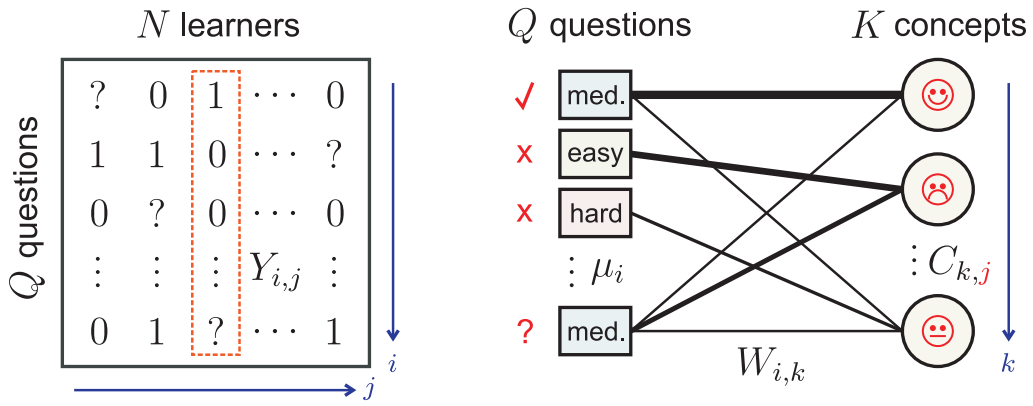
Some progress has been made over the past few decades on personalized learning; see, for example, the sizable literature on *intelligent tutoring systems* discussed in Psotka et al. (1988). To date, the lion's share of fielded, intelligent tutors have been rule-based systems that are hard-coded by domain experts to give learners feedback for pre-defined scenarios (e.g., Koedinger et al. 1997, Brusilovsky and Peylo 2003, VanLehn et al. 2005, and Butz et al. 2006). The specificity of such systems is counterbalanced by their high development cost in terms of both time and money, which has limited their scalability and impact in practice.

In a fresh direction, recent progress has been made on applying *machine learning* algorithms to mine learner interaction data and educational content (see the overview articles by Romero and Ventura 2007 and Baker and Yacef 2009). In contrast to rule-based approaches, machine learning-based PLSs promise to be rapid and inexpensive to deploy, which will enhance their scalability and impact. Indeed, the dawning age of "big data" provides new opportunities to build PLSs based on data rather than rules. We conceptualize the architecture of a generic machine learning-based PLS to have three interlocking components:

- *Learning analytics*: Algorithms that estimate what each learner does and does not understand based on data obtained from tracking their interactions with learning content.
- *Content analytics*: Algorithms that organize learning content such as text, video, simulations, questions, and feedback hints.
- *Scheduling*: Algorithms that use the results of learning and content analytics to suggest to each learner at each moment what they should be doing in order to maximize their learning outcomes, in effect closing the learning feedback loop.

1.2 Sparse Factor Analysis (SPARFA)

In this paper, we develop a new model and a suite of algorithms for joint machine learning-based *learning analytics* and *content analytics*. Our model (developed in Section 2) represents the probability that a learner provides the correct response to a given question in



(a) Graded learner–question responses.

(b) Inferred question–concept association graph.

Figure 1: (a) The SPARFA framework processes a (potentially incomplete) binary-valued data set of graded learner–question responses to (b) estimate the underlying questions-concept association graph and the abstract conceptual knowledge of each learner (illustrated here by smiley faces for learner $j = 3$, the column in (a) selected by the red dashed box).

terms of three factors: their knowledge of the underlying concepts, the concepts involved in each question, and each question's intrinsic difficulty.

Figure 1 provides a graphical depiction of our approach. As shown in Figure 1(a), we are provided with data related to the correctness of the learners' responses to a collection of questions. We encode these graded responses in a "gradebook," a source of information commonly used in the context of classical test theory (Norvick 1966). Specifically, the "gradebook" is a matrix with entry $Y_{i,j} = 1$ or 0 depending on whether learner j answers question i correctly or incorrectly, respectively. Question marks correspond to incomplete data due to unanswered or unassigned questions. Working left-to-right in Figure 1(b), we assume that the collection of questions (rectangles) is related to a small number of abstract concepts (circles) by a bipartite graph, where the edge weight $W_{i,k}$ indicates the degree to which question i involves concept k. We also assume that question i has intrinsic difficulty μ_i. Denoting learner j's knowledge of concept k by $C_{k,j}$, we calculate the probabilities that the learners answer the questions correctly in terms of $\mathbf{WC} + \mathbf{M}$, where $\mathbf{W}$ and $\mathbf{C}$ are matrix versions of $W_{i,k}$ and $C_{k,j}$, respectively, and $\mathbf{M}$ is a matrix containing the intrinsic question difficulty μ_i on row i. We transform the probability of a correct answer to an actual 1/0 correctness via a standard probit or logit link function (see Rasmussen and Williams 2006).

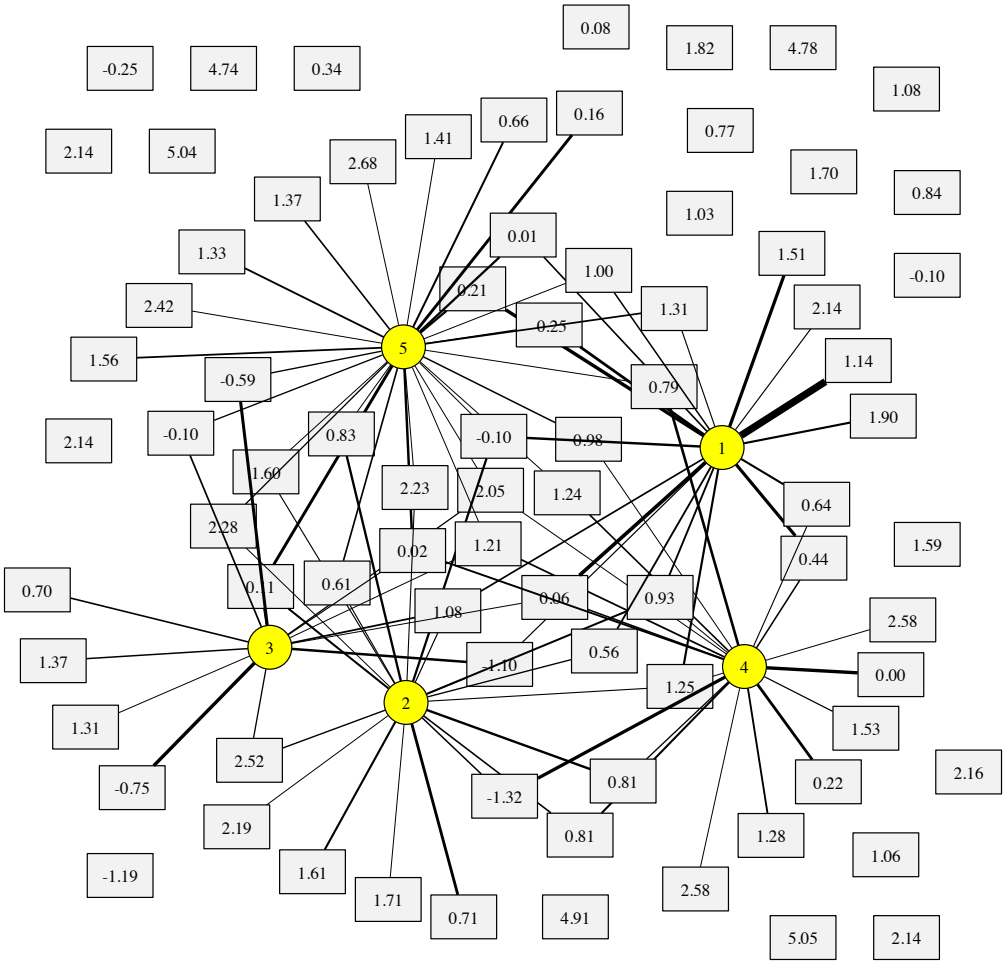
Armed with this model and given incomplete observations of the graded learner–question responses $Y_{i,j}$, our goal is to estimate the factors $\mathbf{W}$, $\mathbf{C}$, and $\mathbf{M}$. Such a *factor-analysis problem* is ill-posed in general, especially when each learner answers only a small subset of the collection of questions (see Harman 1976 for a factor analysis overview). Our first key observation that enables a well-posed solution is the fact that typical educational domains of interest involve only a small number of key concepts (i.e., we have $K \ll N, Q$ in Figure 1). Consequently, $\mathbf{W}$ becomes a tall, narrow $Q \times K$ matrix that relates the questions to a small set of abstract concepts, while $\mathbf{C}$ becomes a short, wide $K \times N$ matrix that relates learner knowledge to that same small set of abstract concepts. Note that the concepts are

"abstract" in that they will be estimated from the data rather than dictated by a subject matter expert. Our second key observation is that each question involves only a small subset of the abstract concepts. Consequently, the matrix $\mathbf{W}$ is sparsely populated. Our third observation is that the entries of $\mathbf{W}$ should be non-negative, since we postulate that having strong concept knowledge should never hurt a learner's chances to answer questions correctly. This constraint on $\mathbf{W}$ ensures that large positive values in $\mathbf{C}$ represent strong knowledge of the associated abstract concepts, which is crucial for a PLS to generate human-interpretable feedback to learners on their strengths and weaknesses.

Leveraging these observations, we propose a suite of new algorithms for solving the *SPARse Factor Analysis* (SPARFA) problem. Section 3 develops SPARFA-M, a matrix factorization method which uses an efficient bi-convex optimization approach to produce point estimates of the factors. Section 4 develops SPARFA-B, a Bayesian factor analysis method to produce posterior distributions of the factors. SPARFA-M is computationally efficient and scales to large-scale applications, while SPARFA-B is more computationally intensive but also provide richer statistical information on the latent factors. Since the concepts are abstract mathematical quantities estimated by the SPARFA algorithms, we develop a *post-processing* step in Section 5 to facilitate interpretation of the estimated latent concepts by associating user-defined tags for each question with each abstract concept.

In Section 6, we report on a range of experiments with a variety of synthetic and real-world data that demonstrate the wealth of information provided by the estimates of $\mathbf{W}$, $\mathbf{C}$ and $\mathbf{M}$. As an example, Figure 2 provides the results for a data set collected from learners using STEMscopes (2012), a science curriculum platform. The data set consists of 145 Grade 8 learners from a single school district answering a manually tagged set of 80 questions on Earth science; only 13.5% of all graded learner–question responses were observed. We applied the SPARFA-B algorithm to retrieve the factors $\mathbf{W}$, $\mathbf{C}$, and $\mathbf{M}$ using 5 latent concepts. The resulting sparse matrix $\mathbf{W}$ is displayed as a bipartite graph in Figure 2(a); circles denote the abstract concepts and boxes denote questions. Each question box is labeled with its estimated intrinsic difficulty μ_i, with large positive values denoting easy questions. Links between the concept and question nodes represent the active (non-zero) entries of $\mathbf{W}$, with thicker links denoting larger values $W_{i,k}$. Unconnected questions are those for which no concept explained the learners' answer pattern; such questions typically have either very low or very high intrinsic difficulty, resulting in nearly all learners answering them correctly or incorrectly. The tags provided in Figure 2(b) enable human-readable interpretability of the estimated abstract concepts.

We envision a range of potential learning and content analytics applications for the SPARFA framework that go far beyond the standard practice of merely forming column sums of the "gradebook" matrix (with entries $Y_{i,j}$) to arrive at a final scalar numerical score for each learner (which is then often further quantized to a letter grade on a 5-point scale). Each column of the estimated $\mathbf{C}$ matrix can be interpreted as a measure of the corresponding learner's knowledge about the abstract concepts. Low values indicate concepts ripe for remediation, while high values indicate concepts ripe for enrichment. The sparse graph stemming from the estimated $\mathbf{W}$ matrix automatically groups questions into similar types based on their concept association; this graph makes it straightforward to find a set of questions similar to a given target question. Finally, the estimated $\mathbf{M}$ matrix (with entries μ_i on each row) provides an estimate of each question's intrinsic difficulty. This



(a) Inferred question–concept association graph.

Concept 1		Concept 2		Concept 3	
Changes to land	(45%)	Evidence of the past	(74%)	Alternative energy	(76%)
Properties of soil	(28%)	Mixtures and solutions	(14%)	Environmental changes	(19%)
Uses of energy	(27%)	Environmental changes	(12%)	Changes from heat	(5%)
Concept 4		**Concept 5**			
Properties of soil	(77%)	Formulation of fossil fuels	(54%)		
Environmental changes	(17%)	Mixtures and solutions	(28%)		
Classifying matter	(6%)	Uses of energy	(18%)		

(b) Most important tags and relative weights for the estimated concepts.

Figure 2: (a) Sparse question–concept association graph and (b) most important tags associated with each concept for Grade 8 Earth science with $N = 135$ learners answering $Q = 80$ questions. Only 13.5% of all graded learner–question responses were observed.

property enables an instructor to assign questions in an orderly fashion as well as to prune out potentially problematic questions that are either too hard, too easy, too confusing, or unrelated to the concepts underlying the collection of questions.

In Section 7, we provide an overview of related work on machine learning-based personalized learning, and we conclude in Section 8. All proofs are relegated to three Appendices.

2. Statistical Model for Learning and Content Analytics

Our approach to learning and content analytics is based on a new statistical model that encodes the probability that a learner will answer a given question correctly in terms of three factors: (i) the learner's knowledge of a set of latent, abstract concepts, (ii) how the question is related to each concept, and (iii) the intrinsic difficulty of the question.

2.1 Model for Graded Learner Response Data

Let N denote the total number of learners, Q the total number of questions, and K the number of latent abstract concepts. We define $C_{k,j}$ as the *concept knowledge* of learner j on concept k, with large positive values of $C_{k,j}$ corresponding to a better chance of success on questions related to concept k. Stack these values into the column vector $\mathbf{c}_j \in \mathbb{R}^K$, $j \in \{1, \ldots, N\}$ and the $K \times N$ matrix $\mathbf{C} = [\,\mathbf{c}_1, \ldots, \mathbf{c}_N\,]$. We further define $W_{i,k}$ as the *question–concept association* of question i with respect to concept k, with larger values denoting stronger involvement of the concept. Stack these values into the column vector $\bar{\mathbf{w}}_i \in \mathbb{R}^K$, $i \in \{1, \ldots, Q\}$ and the $Q \times K$ matrix $\mathbf{W} = [\,\bar{\mathbf{w}}_1, \ldots, \bar{\mathbf{w}}_Q\,]^T$. Finally, we define the scalar $\mu_i \in \mathbb{R}$ as the *intrinsic difficulty* of question i, with larger values representing easier questions. Stack these values into the column vector $\boldsymbol{\mu}$ and form the $Q \times N$ matrix $\mathbf{M} = \boldsymbol{\mu}\,\mathbf{1}_{1\times N}$ as the product of $\boldsymbol{\mu} = [\,\mu_1, \ldots, \mu_Q\,]^T$ with the N-dimensional all-ones row vector $\mathbf{1}_{1\times N}$.

Given these definitions, we propose the following model for the binary-valued graded response variable $Y_{i,j} \in \{0, 1\}$ for learner j on question i, with 1 representing a correct response and 0 an incorrect response:

$$\begin{aligned} Z_{i,j} &= \bar{\mathbf{w}}_i^T \mathbf{c}_j + \mu_i, \qquad \forall i, j, \\ Y_{i,j} &\sim \mathit{Ber}(\Phi(Z_{i,j})), \quad (i,j) \in \Omega_{\text{obs}}. \end{aligned} \tag{1}$$

Here, $\mathit{Ber}(z)$ designates a Bernoulli distribution with success probability z, and $\Phi(z)$ denotes an *inverse link function*[1] that maps a real value z to the success probability of a binary random variable. Thus, the slack variable $\Phi(Z_{i,j}) \in [0, 1]$ governs the probability of learner j answering question i correctly.

The set $\Omega_{\text{obs}} \subseteq \{1, \ldots, Q\} \times \{1, \ldots, N\}$ in (1) contains the indices associated with the observed graded learner response data. Hence, our framework is able to handle the case of incomplete or missing data (e.g., when the learners do not answer all of the questions).[2]

1. Inverse link functions are often called *response functions* in the generalized linear models literature (see, e.g., Guisan et al. 2002).
2. Two common situations lead to missing learner response data. First, a learner might not attempt a question because it was not assigned or available to them. In this case, we simply exclude their response from Ω_{obs}. Second, a learner might not attempt a question because it was assigned to them but was too difficult. In this case, we treat their response as incorrect, as is typical in standard testing settings.

Stack the values $Y_{i,j}$ and $Z_{i,j}$ into the $Q \times N$ matrices $\mathbf{Y}$ and $\mathbf{Z}$, respectively. We can conveniently rewrite (1) in matrix form as

$$Y_{i,j} \sim Ber(\Phi(Z_{i,j})),\ (i,j) \in \Omega_{\text{obs}} \quad \text{with} \quad \mathbf{Z} = \mathbf{WC} + \mathbf{M}. \tag{2}$$

In this paper, we focus on the two most commonly used link functions in the machine learning literature. The *inverse probit* function is defined as

$$\Phi_{\text{pro}}(x) = \int_{-\infty}^{x} \mathcal{N}(t)\,\mathrm{d}t = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2}\,\mathrm{d}t, \tag{3}$$

where $\mathcal{N}(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}$ is the probability density function (PDF) of the standard normal distribution (with mean zero and variance one). The *inverse logit* link function is defined as

$$\Phi_{\log}(x) = \frac{1}{1 + e^{-x}}. \tag{4}$$

As noted in the Introduction, $\mathbf{C}$, $\mathbf{W}$, and $\boldsymbol{\mu}$ (or equivalently, $\mathbf{M}$) have natural interpretations in real education settings. Column j of $\mathbf{C}$ can be interpreted as a measure of learner j's knowledge about the abstract concepts, with larger $C_{k,j}$ values implying more knowledge. The non-zero entries in $\mathbf{W}$ can be used to visualize the connectivity between concepts and questions (see Figure 1(b) for an example), with larger $W_{i,k}$ values implying stronger ties between question i and concept k. The values of $\boldsymbol{\mu}$ contains estimates of each question's intrinsic difficulty.

2.2 Joint Estimation of Concept Knowledge and Question–Concept Association

Given a (possibly partially observed) matrix of graded learner response data $\mathbf{Y}$, we aim to estimate the learner concept knowledge matrix $\mathbf{C}$, the question–concept association matrix $\mathbf{W}$, and the question intrinsic difficulty vector $\boldsymbol{\mu}$. In practice, the latent factors $\mathbf{W}$ and $\mathbf{C}$, and the vector $\boldsymbol{\mu}$ will contain many more unknowns than we have observations in $\mathbf{Y}$; hence, estimating $\mathbf{W}$, $\mathbf{C}$, and $\boldsymbol{\mu}$ is, in general, an ill-posed inverse problem. The situation is further exacerbated if many entries in $\mathbf{Y}$ are unobserved.

To regularize this inverse problem, prevent over-fitting, improve identifiability,[3] and enhance interpretability of the entries in $\mathbf{C}$ and $\mathbf{W}$, we appeal to the following three observations regarding education that are reasonable for typical exam, homework, and practice questions at all levels. We will exploit these observations extensively in the sequel as fundamental assumptions:

(A1) *Low-dimensionality*: The number of latent, abstract concepts K is small relative to both the number of learners N and the number of questions Q. This implies that the questions are redundant and that the learners' graded responses live in a low-dimensional space. The parameter K dictates the concept *granularity*. Small K extracts just a few general, broad concepts, whereas large K extracts more specific and detailed concepts.[4]

3. If $\mathbf{Z} = \mathbf{WC}$, then for any orthonormal matrix $\mathbf{H}$ with $\mathbf{H}^T\mathbf{H} = \mathbf{I}$, we have $\mathbf{Z} = \mathbf{WH}^T\mathbf{HC} = \widetilde{\mathbf{W}}\widetilde{\mathbf{C}}$. Hence, the estimation of $\mathbf{W}$ and $\mathbf{C}$ is, in general, non-unique up to a unitary matrix rotation.
4. Standard techniques like cross-validation (Hastie et al. 2010) can be used to select K. We provide the corresponding details in Section 6.3.

(A2) *Sparsity*: Each question should be associated with only a small subset of the concepts in the domain of the course/assessment. In other words, we assume that the matrix $\mathbf{W}$ is sparsely populated, i.e., contains mostly zero entries.

(A3) *Non-negativity*: A learner's knowledge of a given concept does not negatively affect their probability of correctly answering a given question, i.e., knowledge of a concept is not "harmful." In other words, the entries of $\mathbf{W}$ are non-negative, which provides a natural interpretation for the entries in $\mathbf{C}$: Large values $C_{k,j}$ indicate strong knowledge of the corresponding concept, whereas negative values indicate weak knowledge.

In practice, N can be larger than Q and vice versa, and hence, we do not impose any additional assumptions on their values. Assumptions (A2) and (A3) impose sparsity and non-negativity constraints on $\mathbf{W}$. Since these assumptions are likely to be violated under arbitrary unitary transforms of the factors, they help alleviate several well-known identifiability problems that arise in factor analysis.

We will refer to the problem of estimating $\mathbf{W}$, $\mathbf{C}$, and $\boldsymbol{\mu}$, given the observations $\mathbf{Y}$, under the assumptions (A1)–(A3) as the *SPARse Factor Analysis* (SPARFA) problem. We now develop two complementary algorithms to solve the SPARFA problem. Next, we detail SPARFA-M and SPARFA-B, a matrix-factorization approach and a Bayesian approach to estimate the quantities of interest.

3. SPARFA-M: Maximum Likelihood-based Sparse Factor Analysis

Our first algorithm, SPARFA-M, solves the SPARFA problem using maximum-likelihood-based probit or logistic regression.

3.1 Problem Formulation

To estimate $\mathbf{W}$, $\mathbf{C}$, and $\boldsymbol{\mu}$, we maximize the likelihood of the observed data $Y_{i,j}$, $(i,j) \in \Omega_{\text{obs}}$

$$p(Y_{i,j}|\bar{\mathbf{w}}_i, \mathbf{c}_j) = \Phi\big(\bar{\mathbf{w}}_i^T \mathbf{c}_j\big)^{Y_{i,j}} \, \big(1 - \Phi(\bar{\mathbf{w}}_i^T \mathbf{c}_j)\big)^{1-Y_{i,j}}$$

given $\mathbf{W}$, $\mathbf{C}$, and $\boldsymbol{\mu}$ and subject to the assumptions (A1), (A2), and (A3) from Section 2.2. This likelihood yields the following optimization problem:

$$\text{(P*)} \quad \begin{cases} \underset{\mathbf{W},\mathbf{C}}{\text{maximize}} & \sum_{(i,j)\in\Omega_{\text{obs}}} \log p(Y_{i,j}|\bar{\mathbf{w}}_i, \mathbf{c}_j) \\ \text{subject to} & \|\bar{\mathbf{w}}_i\|_0 \le s \;\forall i, \;\; \|\bar{\mathbf{w}}_i\|_2 \le \kappa \;\forall i, \;\; W_{i,k} \ge 0 \;\forall i,k, \;\; \|\mathbf{C}\|_{\text{F}} = \xi. \end{cases}$$

Let us take a quick tour of the problem (P*) and its constraints. The intrinsic difficulty vector $\boldsymbol{\mu}$ is incorporated as an additional column of $\mathbf{W}$, and $\mathbf{C}$ is augmented with an all-ones row accordingly. We impose sparsity on each vector $\bar{\mathbf{w}}_i$ to comply with (A2) by limiting its maximum number of nonzero coefficients using the constraint $\|\bar{\mathbf{w}}_i\|_0 \le s$; here $\|\mathbf{a}\|_0$ counts the number of non-zero entries in the vector $\mathbf{a}$. The ℓ_2-norm constraint on each vector $\bar{\mathbf{w}}_i$ with $\kappa > 0$ is required for our convergence proof below. We enforce non-negativity on each entry $W_{i,k}$ to comply with (A3). Finally, we normalize the Frobenius norm of the concept knowledge matrix $\mathbf{C}$ to a given $\xi > 0$ to suppress arbitrary scalings between the entries in both matrices $\mathbf{W}$ and $\mathbf{C}$.

Unfortunately, optimizing over the sparsity constraints $\|\bar{\mathbf{w}}_i\|_0 \leq s$ requires a combinatorial search over all K-dimensional support sets having no more than s non-zero entries. Hence, (P*) cannot be solved efficiently in practice for the typically large problem sizes of interest. In order to arrive at an optimization problem that can be solved with a reasonable computational complexity, we *relax* the sparsity constraints $\|\bar{\mathbf{w}}_i\|_0 \leq s$ in (P*) to ℓ_1-norm constraints as in Chen et al. (1998) and move them, the ℓ_2-norm constraints, and the Frobenius norm constraint, into the objective function via Lagrange multipliers:

$$\text{(P)} \quad \underset{\mathbf{W},\mathbf{C}\colon W_{i,k}\geq 0\ \forall i,k}{\text{minimize}} \ \textstyle\sum_{(i,j)\in\Omega_{\text{obs}}} -\log p(Y_{i,j}|\bar{\mathbf{w}}_i,\mathbf{c}_j) + \lambda \sum_i \|\bar{\mathbf{w}}_i\|_1 + \frac{\mu}{2}\sum_i \|\bar{\mathbf{w}}_i\|_2^2 + \frac{\gamma}{2}\|\mathbf{C}\|_{\text{F}}^2 .$$

The first regularization term $\lambda \sum_i \|\bar{\mathbf{w}}_i\|_1$ induces sparsity on each vector $\bar{\mathbf{w}}_i$, with the single parameter $\lambda > 0$ controlling the sparsity level. Since one can arbitrarily increase the scale of the vectors $\bar{\mathbf{w}}_i$ while decreasing the scale of the vectors $\mathbf{c}_j$ accordingly (and vice versa) without changing the likelihood, we gauge these vectors using the second and third regularization terms $\frac{\mu}{2}\sum_i \|\bar{\mathbf{w}}_i\|_2^2$ and $\frac{\gamma}{2}\|\mathbf{C}\|_{\text{F}}^2$ with the regularization parameters $\mu > 0$ and $\gamma > 0$, respectively.[5] We emphasize that since $\|\mathbf{C}\|_{\text{F}}^2 = \sum_j \|\mathbf{c}_j\|_2^2$, we can impose a regularizer on each column rather than the entire matrix $\mathbf{C}$, which facilitates the development of the efficient algorithm detailed below.

3.2 The SPARFA-M Algorithm

Since the first negative log-likelihood term in the objective function of (P) is convex in the product $\mathbf{WC}$ for both the probit and the logit functions (see, e.g., Hastie et al. 2010), and since the rest of the regularization terms are convex in either $\mathbf{W}$ or $\mathbf{C}$ while the nonnegativity constraints on $W_{i,k}$ are with respect to a convex set, the problem (P) is *biconvex* in the individual factors $\mathbf{W}$ and $\mathbf{C}$. More importantly, with respect to blocks of variables $\bar{\mathbf{w}}_i$, $\mathbf{c}_j$, the problem (P) is *block multi-convex* in the sense of Xu and Yin (2012).

SPARFA-M is an alternating optimization approach to (approximately) *solve* (P) that proceeds as follows. We initialize $\mathbf{W}$ and $\mathbf{C}$ with random entries and then iteratively optimize the objective function of (P) for both factors in an alternating fashion. Each outer iteration involves solving two kinds of inner subproblems. In the first subproblem, we hold $\mathbf{W}$ constant and separately optimize each block of variables in $\mathbf{c}_j$; in the second subproblem, we hold $\mathbf{C}$ constant and separately optimize each block of variables $\bar{\mathbf{w}}_i$. Each subproblem is solved using an iterative method; see Section 3.3 for the respective algorithms. The outer loop is terminated whenever a maximum number of outer iterations $I_{\max}$ is reached, or if the decrease in the objective function of (P) is smaller than a certain threshold.

The two subproblems constituting the inner iterations of SPARFA-M correspond to the following convex ℓ_1/ℓ_2-norm and ℓ_2-norm regularized regression (RR) problems:

$$\text{(RR}_1^+\text{)} \quad \underset{\bar{\mathbf{w}}_i\colon W_{i,k}\geq 0\ \forall k}{\text{minimize}} \ \textstyle\sum_{j\colon (i,j)\in\Omega_{\text{obs}}} -\log p(Y_{i,j}|\bar{\mathbf{w}}_i,\mathbf{c}_j) + \lambda\|\bar{\mathbf{w}}_i\|_1 + \frac{\mu}{2}\|\bar{\mathbf{w}}_i\|_2^2 ,$$

$$\text{(RR}_2\text{)} \quad \underset{\mathbf{c}_j}{\text{minimize}} \ \textstyle\sum_{i\colon (i,j)\in\Omega_{\text{obs}}} -\log p(Y_{i,j}|\bar{\mathbf{w}}_i,\mathbf{c}_j) + \frac{\gamma}{2}\|\mathbf{c}_j\|_2^2 .$$

5. The first ℓ_1-norm regularization term in (RR$_1^+$) already gauges the norm of the $\bar{\mathbf{w}}_i$. The ℓ_2-norm regularizer $\frac{\mu}{2}\sum_i \|\bar{\mathbf{w}}_i\|_2^2$ is included only to aid in establishing the convergence results for SPARFA-M as detailed in Section 3.4.

We develop two novel first-order methods that efficiently solve (RR_1^+) and (RR_2) for both probit and logistic regression. These methods scale well to high-dimensional problems, in contrast to existing second-order methods. In addition, the probit link function makes the explicit computation of the Hessian difficult, which is only required for second-order methods. Therefore, we build our algorithm on the fast iterative soft-thresholding algorithm (FISTA) framework developed in Beck and Teboulle (2009), which enables the development of efficient first-order methods with accelerated convergence.

3.3 Accelerated First-Order Methods for Regularized Probit/Logistic Regression

The FISTA framework (Beck and Teboulle 2009) iteratively solves optimization problems whose objective function is given by $f(\cdot) + g(\cdot)$, where $f(\cdot)$ is a continuously differentiable convex function and $g(\cdot)$ is convex but potentially non-smooth. This approach is particularly well-suited to the inner subproblem (RR_1^+) due to the presence of the non-smooth ℓ_1-norm regularizer and the non-negativity constraint. Concretely, we associate the log-likelihood function plus the ℓ_2-norm regularizer $\frac{\mu}{2}\|\bar{\mathbf{w}}_i\|_2^2$ with $f(\cdot)$ and the ℓ_1-norm regularization term with $g(\cdot)$. For the inner subproblem (RR_2), we associate the log-likelihood function with $f(\cdot)$ and the ℓ_2-norm regularization term with $g(\cdot)$.[6]

Each FISTA iteration consists of two steps: (i) a gradient-descent step in $f(\cdot)$ and (ii) a shrinkage step determined by $g(\cdot)$. For simplicity of exposition, we consider the case where all entries in $\mathbf{Y}$ are observed, i.e., $\Omega_{\mathrm{obs}} = \{1, \ldots, Q\} \times \{1, \ldots, N\}$; the extension to the case with missing entries in $\mathbf{Y}$ is straightforward. We will derive the algorithm for the case of probit regression first and then point out the departures for logistic regression.

For (RR_1^+), the gradients of $f(\bar{\mathbf{w}}_i)$ with respect to the i^{th} block of regression coefficients $\bar{\mathbf{w}}_i$ are given by

$$\nabla f_{\mathrm{pro}}^i = \nabla_{\bar{\mathbf{w}}_i}^{\mathrm{pro}}(-\textstyle\sum_j \log p_{\mathrm{pro}}(Y_{i,j}|\bar{\mathbf{w}}_i, \mathbf{c}_j) + \frac{\mu}{2}\|\bar{\mathbf{w}}_i\|_2^2) = -\mathbf{C}\mathbf{D}^i(\bar{\mathbf{y}}^i - \mathbf{p}_{\mathrm{pro}}^i) + \mu\bar{\mathbf{w}}_i, \qquad (5)$$

where $\bar{\mathbf{y}}^i$ is an $N \times 1$ column vector corresponding to the transpose of the i^{th} row of $\mathbf{Y}$. $\mathbf{p}_{\mathrm{pro}}^i$ is an $N \times 1$ vector whose j^{th} element equals the probability of $Y_{i,j}$ being 1; that is, $p_{\mathrm{pro}}(Y_{i,j} = 1|\bar{\mathbf{w}}_i, \mathbf{c}_j) = \Phi_{\mathrm{pro}}(\bar{\mathbf{w}}_i^T\mathbf{c}_j)$. The entries of the $N \times N$ diagonal matrix $\mathbf{D}$ are given by

$$D_{j,j}^i = \frac{\mathcal{N}(\bar{\mathbf{w}}_i^T\mathbf{c}_j)}{\Phi_{\mathrm{pro}}(\bar{\mathbf{w}}_i^T\mathbf{c}_j)(1 - \Phi_{\mathrm{pro}}(\bar{\mathbf{w}}_i^T\mathbf{c}_j))}.$$

The gradient step in each FISTA iteration $\ell = 1, 2, \ldots$ corresponds to

$$\hat{\bar{\mathbf{w}}}_i^{\ell+1} \leftarrow \bar{\mathbf{w}}_i^{\ell} - t_\ell \nabla f_{\mathrm{pro}}^i, \qquad (6)$$

where t_ℓ is a suitable step-size. To comply with (A3), the shrinkage step in (RR_1^+) corresponds to a non-negative soft-thresholding operation

$$\bar{\mathbf{w}}_i^{\ell+1} \leftarrow \max\{\hat{\bar{\mathbf{w}}}_i^{\ell+1} - \lambda t_\ell, 0\}. \qquad (7)$$

6. Of course, both $f(\cdot)$ and $g(\cdot)$ are smooth for (RR_2). Hence, we could also apply an accelerated gradient-descent approach instead, e.g., as described in Nesterov (2007).

For (RR_2), the gradient step becomes

$$\hat{\mathbf{c}}_j^{\ell+1} \leftarrow \mathbf{c}_j^{\ell} - t_\ell \nabla f_{\text{pro}}^i,$$

which is the same as (5) and (6) after replacing $\mathbf{C}$ with $\mathbf{W}^T$ and μ with γ. The shrinkage step for (RR_2) is the simple re-scaling

$$\mathbf{c}_j^{\ell+1} \leftarrow \frac{1}{1+\gamma t_\ell} \hat{\mathbf{c}}_j^{\ell+1}. \tag{8}$$

In the logistic regression case, the steps (6), (7), and (8) remain the same but the gradient changes to

$$\nabla f_{\text{log}}^i = \nabla_{\bar{\mathbf{w}}_i}^{\text{log}} (-\textstyle\sum_j \log p_{\text{log}}(Y_{i,j}|\bar{\mathbf{w}}_i, \mathbf{c}_j) + \frac{\mu}{2}\|\bar{\mathbf{w}}_i\|_2^2) = -\mathbf{C}(\bar{\mathbf{y}}^i - \mathbf{p}_{\text{log}}^i) + \mu \bar{\mathbf{w}}_i, \tag{9}$$

where the $N \times 1$ vector $\mathbf{p}_{\text{log}}^i$ has elements $\mathbf{p}_{\text{log}}(Y_{i,j} = 1|\bar{\mathbf{w}}_i, \mathbf{c}_j) = \Phi_{\text{log}}(\bar{\mathbf{w}}_i^T \mathbf{c}_j)$.

The above steps require a suitable step-size t_ℓ to ensure convergence to the optimal solution. A common approach that guarantees convergence is to set $t_\ell = 1/L$, where L is the Lipschitz constant of $f(\cdot)$ (see Beck and Teboulle 2009 for the details). The Lipschitz constants for both the probit and logit cases are analyzed in Theorem 1 below. Alternatively, one can also perform backtracking, which—under certain circumstances—can be more efficient; see (Beck and Teboulle, 2009, p. 194) for more details.

3.4 Convergence Analysis of SPARFA-M

While the SPARFA-M objective function is guaranteed to be non-increasing over the outer iterations (Boyd and Vandenberghe 2004), the factors $\mathbf{W}$ and $\mathbf{C}$ do not necessarily converge to a global or local optimum due to its biconvex (or more generally, block multi-convex) nature. It is difficult, in general, to develop rigorous statements for the convergence behavior of block multi-convex problems. Nevertheless, we can establish the global convergence of SPARFA-M from any starting point to a critical point of the objective function using recent results developed in Xu and Yin (2012). The convergence results below appear to be novel for both sparse matrix factorization as well as dictionary learning.

3.4.1 Convergence Analysis of Regularized Regression using FISTA

In order to establish the SPARFA-M convergence result, we first adapt the convergence results for FISTA in Beck and Teboulle (2009) to prove convergence on the two subproblems (RR_1^+) and (RR_2). The following theorem is a consequence of (Beck and Teboulle, 2009, Thm. 4.4) combined with Lemmata 4 and 5 in Appendix A. If back-tracking is used to select step-size t_ℓ (Beck and Teboulle, 2009, p. 194), then let α correspond to the backtracking parameter. Otherwise set $\alpha = 1$ and for (RR_1^+) let $t_\ell = 1/L_1$ and for (RR_2) let $t_\ell = 1/L_2$. In Lemma 5, we compute that $L_1 = \sigma_{\max}^2(\mathbf{C}) + \mu$ and $L_2 = \sigma_{\max}^2(\mathbf{W})$ for the probit case, and $L_1 = \frac{1}{4}\sigma_{\max}^2(\mathbf{C}) + \mu$ and $L_2 = \frac{1}{4}\sigma_{\max}^2(\mathbf{W})$ for the logit case.

Theorem 1 (Linear convergence of RR using FISTA) *Given i and j, let*

$$F_1(\bar{\mathbf{w}}_i) = \sum\nolimits_{j\,:\,(i,j)\in\Omega_{obs}} -\log p(Y_{i,j}|\bar{\mathbf{w}}_i, \mathbf{c}_j) + \lambda\|\bar{\mathbf{w}}_i\|_1 + \frac{\mu}{2}\|\bar{\mathbf{w}}_i\|_2^2, \quad W_{i,k} \geq 0\ \forall k,$$
$$F_2(\mathbf{c}_j) = \sum\nolimits_{i\,:\,(i,j)\in\Omega_{obs}} -\log p(Y_{i,j}|\bar{\mathbf{w}}_i, \mathbf{c}_j) + \frac{\gamma}{2}\|\mathbf{c}_j\|_2^2$$

be the cost functions of (RR_1^+) *and* (RR_2)*, respectively. Then, we have*

$$F_1(\bar{\mathbf{w}}_i^\ell) - F_1(\bar{\mathbf{w}}_i^*) \leq \frac{2\alpha L_1 \|\bar{\mathbf{w}}_i^0 - \bar{\mathbf{w}}_i^*\|^2}{(\ell+1)^2},$$
$$F_2(\mathbf{c}_j^\ell) - F_2(\mathbf{c}_j^*) \leq \frac{2\alpha L_2 \|\mathbf{c}_j^0 - \mathbf{c}_j^*\|^2}{(\ell+1)^2},$$

where $\bar{\mathbf{w}}_i^0$ *and* $\mathbf{c}_j^0$ *are the initialization points of* (RR_1^+) *and* (RR_2)*,* $\bar{\mathbf{w}}_i^\ell$ *and* $\mathbf{c}_j^\ell$ *designate the solution estimates at the* ℓ^{th} *inner iteration, and* $\bar{\mathbf{w}}_i^*$ *and* $\mathbf{c}_j^*$ *denote the optimal solutions.*

In addition to establishing convergence, Theorem 1 reveals that the difference between the cost functions at the current estimates and the optimal solution points, $F_1(\bar{\mathbf{w}}_i^\ell) - F_1(\bar{\mathbf{w}}_i^*)$ and $F_2(\mathbf{c}_j^\ell) - F_2(\mathbf{c}_j^*)$, decrease as $O(\ell^{-2})$.

3.4.2 Convergence Analysis of SPARFA-M

We are now ready to establish global convergence of SPARFA-M to a critical point. To this end, we first define $\mathbf{x} = [\bar{\mathbf{w}}_1^T, \ldots, \bar{\mathbf{w}}_Q^T, \mathbf{c}_1^T, \ldots, \mathbf{c}_N^T]^T \in \mathbb{R}^{(N+Q)K}$ and rewrite the objective function (P) of SPARFA-M as follows:

$$F(\mathbf{x}) = \sum_{(i,j)\in\Omega_{\text{obs}}} -\log p(Y_{i,j}|\bar{\mathbf{w}}_i, \mathbf{c}_j) + \frac{\mu}{2}\sum_i \|\bar{\mathbf{w}}_i\|_2^2 + \lambda \sum_i \|\bar{\mathbf{w}}_i\|_1 + \sum_{i,k} \delta(W_{i,k} < 0) + \frac{\gamma}{2}\sum_j \|\mathbf{c}_j\|_2^2$$

with the indicator function $\delta(z < 0) = \infty$ if $z < 0$ and 0 otherwise. Note that we have re-formulated the non-negativity constraint as a set indicator function and added it to the objective function of (P). Since minimizing $F(\mathbf{x})$ is equivalent to solving (P), we can now use the results developed in Xu and Yin (2012) to establish the following convergence result for the SPARFA-M algorithm. The proof can be found in Appendix B.

Theorem 2 (Global convergence of SPARFA-M) *From any starting point* $\mathbf{x}^0$*, let* $\{\mathbf{x}^t\}$ *be the sequence of estimates generated by the SPARFA-M algorithm with* $t = 1, 2, \ldots$ *as the outer iteration number. Then, the sequence* $\{\mathbf{x}^t\}$ *converges to the finite limit point* $\hat{\mathbf{x}}$*, which is a critical point of (P). Moreover, if the starting point* $\mathbf{x}^0$ *is within a close neighborhood of a global optimum of (P), then SPARFA-M converges to this global optimum.*

Since the problem (P) is bi-convex in nature, we cannot guarantee that SPARFA-M always converges to a *global optimum* from an *arbitrary* starting point. Nevertheless, the use of multiple randomized initialization points can be used to increase the chance of being in the close vicinity of a global optimum, which improves the (empirical) performance of SPARFA-M (see Section 3.5 for details). Note that we do not provide the convergence rate of SPARFA-M, since the associated parameters in (Xu and Yin, 2012, Thm. 2.9) are difficult to determine for the model at hand; a detailed analysis of the convergence rate for SPARFA-M is part of ongoing work.

3.5 Algorithmic Details and Improvements for SPARFA-M

In this section, we outline a toolbox of techniques that improve the empirical performance of SPARFA-M and provide guidelines for choosing the key algorithm parameters.

3.5.1 Reducing Computational Complexity in Practice

To reduce the computational complexity of SPARFA-M in practice, we can improve the convergence rates of (RR_1^+) and (RR_2). In particular, the regularizer $\frac{\mu}{2}\|\bar{\mathbf{w}}_i\|_2^2$ in (RR_1^+) has been added to (P) to facilitate the proof for Theorem 2. This term, however, typically slows down the (empirical) convergence of FISTA, especially for large values of μ. We therefore set μ to a small positive value (e.g., $\mu = 10^{-4}$), which leads to fast convergence of (RR_1^+) while still guaranteeing convergence of SPARFA-M.

Selecting the appropriate (i.e., preferably large) step-sizes t_ℓ in (6), (7), and (8) is also crucial for fast convergence. In Lemmata 4 and 5, we derive the Lipschitz constants L for (RR_1^+) and (RR_2), which enables us to set the step-sizes t_ℓ to the constant value $t = 1/L$. In all of our experiments below, we exclusively use constant step-sizes, since we observed that backtracking (Beck and Teboulle 2009, p. 194) provided no advantage in terms of computational complexity for SPARFA-M.

To further reduce the computational complexity of SPARFA-M without degrading its empirical performance noticeably, we have found that instead of running the large number of inner iterations it typically takes to converge, we can run just a few (e.g., 10) inner iterations per outer iteration.

3.5.2 Reducing the Chance of Getting Stuck in Local Minima

The performance of SPARFA-M strongly depends on the initialization of $\mathbf{W}$ and $\mathbf{C}$, due to the bi-convex nature of (P). We have found that running SPARFA-M multiple times with different starting points and picking the solution with the smallest overall objective function delivers excellent performance. In addition, we can deploy the standard heuristics used in the dictionary-learning literature (Aharon et al., 2006, Section IV-E) to further improve the convergence towards a global optimum. For example, every few outer iterations, we can evaluate the current $\mathbf{W}$ and $\mathbf{C}$. If two rows of $\mathbf{C}$ are similar (as measured by the absolute value of the inner product between them), then we re-initialize one of them as an i.i.d. Gaussian vector. Moreover, if some columns in $\mathbf{W}$ contain only zero entries, then we re-initialize them with i.i.d. Gaussian vectors. Note that the convergence proof in Section 3.4 does not apply to implementations employing such trickery.

3.5.3 Parameter Selection

The input parameters to SPARFA-M include the number of concepts K and the regularization parameters γ and λ. The number of concepts K is a user-specified value. In practice, cross-validation can be used to select K if the task is to predict missing entries of $\mathbf{Y}$, (see Section 6.3). The sparsity parameter λ and the ℓ_2-norm penalty parameter γ strongly affect the output of SPARFA-M; they can be selected using any of a number of criteria, including the Bayesian information criterion (BIC) or cross-validation, as detailed in Hastie et al. (2010). Both criteria resulted in similar performance in all of the experiments reported in Section 6.

3.6 Related Work on Maximum Likelihood-based Sparse Factor Analysis

Sparse logistic factor analysis has previously been studied in Lee et al. (2010) in the context of principal components analysis. There are three major differences with the SPARFA framework. First, Lee et al. (2010) do not impose the non-negativity constraint on $\mathbf{W}$ that is critical for the interpretation of the estimated factors. Second, they impose an orthonormality constraint on $\mathbf{C}$ that does not make sense in educational scenarios. Third, they optimize an upper bound on the negative log-likelihood function in each outer iteration, in contrast to SPARFA-M, which optimizes the exact cost functions in (RR_1^+) and (RR_2).

The problem (P) shares some similarities with the method for missing data imputation outlined in (Mohamed et al., 2012, Eq. 7). However, the problem (P) studied here includes an additional non-negativity constraint on $\mathbf{W}$ and the regularization term $\frac{\mu}{2}\sum_i \|\bar{\mathbf{w}}_i\|_2^2$ that are important for the interpretation of the estimated factors and the convergence analysis. Moreover, SPARFA-M utilizes the accelerated FISTA framework as opposed to the more straightforward but less efficient gradient descent method in Mohamed et al. (2012).

SPARFA-M is capable of handling both the inverse logit and inverse probit link functions. For the inverse logit link function, one could solve (RR_1^+) and (RR_2) using an iteratively reweighted second-order algorithm as in Hastie et al. (2010), Minka (2003), Lee et al. (2006), Park and Hastie (2008), or an interior-point method as in Koh et al. (2007). However, none of these techniques extend naturally to the inverse probit link function, which is essential for some applications, e.g., in noisy compressive sensing recovery from 1-bit measurements (e.g., Jacques et al. 2013 or Plan and Vershynin 2012, submitted). Moreover, second-order techniques typically do not scale well to high-dimensional problems due to the necessary computation of the Hessian. In contrast, SPARFA-M scales favorably due to the fact that it utilizes the accelerated first-order FISTA method, avoiding the computation of the Hessian.

4. SPARFA-B: Bayesian Sparse Factor Analysis

Our second algorithm, SPARFA-B, solves the SPARFA problem using a Bayesian method based on Markov chain Monte-Carlo (MCMC) sampling. In contrast to SPARFA-M, which computes point estimates for each of the parameters of interest, SPARFA-B computes full posterior distributions for $\mathbf{W}, \mathbf{C}$, and $\boldsymbol{\mu}$.

While SPARFA-B has a higher computational complexity than SPARFA-M, it has several notable benefits in the context of learning and content analytics. First, the full posterior distributions enable the computation of informative quantities such as credible intervals and posterior modes for all parameters of interest. Second, since MCMC methods explore the full posterior space, they are not subject to being trapped indefinitely in local minima, which is possible with SPARFA-M. Third, the hyperparameters used in Bayesian methods generally have intuitive meanings, in contrary to the regularization parameters of optimization-based methods like SPARFA-M. These hyperparameters can also be specially chosen to incorporate additional prior information about the problem.

4.1 Problem Formulation

As discussed in Section 2.2, we require the matrix $\mathbf{W}$ to be both sparse (A2) and non-negative (A3). We enforce these assumptions through the following prior distributions that

are a variant of the well-studied spike-slab model (West, 2003; Ishwaran and Rao, 2005) adapted for non-negative factor loadings:

$$W_{i,k} \sim r_k \, Exp(\lambda_k) + (1 - r_k)\, \delta_0, \;\; \lambda_k \sim Ga(\alpha, \beta), \;\; \text{and} \;\; r_k \sim Beta(e, f). \tag{10}$$

Here, $Exp(x|\lambda) \sim \lambda e^{-\lambda x}$, $x \geq 0$, and $Ga(x|\alpha, \beta) \sim \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}$, $x \geq 0$, δ_0 is the Dirac delta function, and α, β, e, f are hyperparameters. The model (10) uses the latent random variable r_k to control the sparsity via the hyperparameters e and f. This set of priors induces a conjugate form on the posterior that enables efficient sampling. We note that both the exponential rate parameters λ_k as well as the inclusion probabilities r_k are grouped per factor. The remaining priors used in the proposed Bayesian model are summarized as

$$\mathbf{c}_j \sim \mathcal{N}(0, \mathbf{V}), \;\; \mathbf{V} \sim IW(\mathbf{V}_0, h), \;\; \text{and} \;\; \mu_i \sim \mathcal{N}(\mu_0, v_{\boldsymbol{\mu}}), \tag{11}$$

where $\mathbf{V}_0$, h, μ_0, and $v_{\boldsymbol{\mu}}$ are hyperparameters.

4.2 The SPARFA-B Algorithm

We obtain posterior distribution estimates for the parameters of interest through an MCMC method based on the Gibbs' sampler. To implement this, we must derive the conditional posteriors for each of the parameters of interest. We note again that the graded learner-response matrix $\mathbf{Y}$ will not be fully observed, in general. Thus, our sampling method must be equipped to handle missing data.

The majority of the posterior distributions follow from standard results in Bayesian analysis and will not be derived in detail here. The exception is the posterior distribution of $W_{i,k} \;\forall i, k$. The spike-slab model that enforces sparsity in $\mathbf{W}$ requires first sampling $W_{i,k} \neq 0 | \mathbf{Z}, \mathbf{C}, \boldsymbol{\mu}$ and then sampling $W_{i,k} | \mathbf{Z}, \mathbf{C}, \boldsymbol{\mu}$, for all $W_{i,k} \neq 0$. These posterior distributions differ from previous results in the literature due to our assumption of an exponential (rather than a normal) prior on $W_{i,k}$. We next derive these two results in detail.

4.2.1 Derivation of Posterior Distribution of $W_{i,k}$

We seek both the probability that an entry $W_{i,k}$ is active (non-zero) and the distribution of $W_{i,k}$ when active given our observations. The following theorem states the final sampling results; the proof is given in Appendix C.

Theorem 3 (Posterior distributions for W) *For all $i = 1, \ldots, Q$ and all $k = 1, \ldots, K$, the posterior sampling results for $W_{i,k} = 0 | \mathbf{Z}, \mathbf{C}, \boldsymbol{\mu}$ and $W_{i,k} | \mathbf{Z}, \mathbf{C}, \boldsymbol{\mu}, W_{i,k} \neq 0$ are given by*

$$\widehat{R}_{i,k} = p(W_{i,k} = 0 | \mathbf{Z}, \mathbf{C}, \boldsymbol{\mu}) = \frac{\frac{\mathcal{N}^r(0|\widehat{M}_{i,k}, \widehat{S}_{i,k}, \lambda_k)}{Exp(0|\lambda_k)}(1-r_k)}{\frac{\mathcal{N}^r(0|\widehat{M}_{i,k}, \widehat{S}_{i,k}, \lambda_k)}{Exp(0|\lambda_k)}(1-r_k)+r_k},$$

$$W_{i,k} | \mathbf{Z}, \mathbf{C}, \boldsymbol{\mu}, W_{i,k} \neq 0 \sim \mathcal{N}^r(\widehat{M}_{i,k}, \widehat{S}_{i,k}, \lambda_k),$$

$$\widehat{M}_{i,k} = \frac{\sum_{\{j:(i,j)\in\Omega_{obs}\}} \big((Z_{i,j} - \mu_i) - \sum_{k' \neq k} W_{i,k'} C_{k',j}\big) C_{k,j}}{\sum_{\{j:(i,j)\in\Omega_{obs}\}} C_{k,j}^2},$$

$$\widehat{S}_{i,k} = \Big(\sum_{\{j:(i,j)\in\Omega_{obs}\}} C_{k,j}^2\Big)^{-1},$$

where $\mathcal{N}^r(x|m,s,\lambda) = \frac{e^{\lambda m-\lambda^2 s/2}}{\sqrt{2\pi s}\Phi\left(\frac{m-\lambda s}{\sqrt{s}}\right)} e^{-(x-m)^2/2s-\lambda m}$ *represents a rectified normal distribution (see Schmidt et al. 2009).*

4.2.2 Sampling Methodology

SPARFA-B carries out the following MCMC steps to compute posterior distributions for all parameters of interest:

1. For all $(i,j) \in \Omega_{\text{obs}}$, draw $Z_{i,j} \sim \mathcal{N}\big((\mathbf{WC})_{i,j} + \mu_i, 1\big)$, truncating above 0 if $Y_{i,j} = 1$, and truncating below 0 if $Y_{i,j} = 0$.

2. For all $i = 1,\dots,Q$, draw $\mu_i \sim \mathcal{N}(m_i, v)$ with $v = (v_{\boldsymbol{\mu}}^{-1} + n')^{-1}$, $m_i = \mu_0 + v\sum_{\{j:(i,j)\in\Omega_{\text{obs}}\}} \left(Z_{i,j} - \bar{\mathbf{w}}_i^T \mathbf{c}_j\right)$, and n' the number of learners responding to question i.

3. For all $j = 1,\dots,N$, draw $\mathbf{c}_j \sim \mathcal{N}(\mathbf{m}_j, \mathbf{M}_j)$ with $\mathbf{M}_j = (\mathbf{V}^{-1} + \widetilde{\mathbf{W}}^T\widetilde{\mathbf{W}})^{-1}$, and $\mathbf{m}_j = \mathbf{M}_j\widetilde{\mathbf{W}}^T(\widetilde{\mathbf{z}}_j - \widetilde{\boldsymbol{\mu}})$. The notation $\widetilde{(\cdot)}$ denotes the restriction of the vector or matrix to the set of rows $i : (i,j) \in \Omega_{\text{obs}}$.

4. Draw $\mathbf{V} \sim IW(\mathbf{V}_0 + \mathbf{CC}^T, N + h)$.

5. For all $i = 1,\dots,Q$ and $k = 1,\dots,K$, draw $W_{i,k} \sim \widehat{R}_{i,k}\mathcal{N}^r(\widehat{M}_{i,k}, \widehat{S}_{i,k}) + (1 - \widehat{R}_{i,k})\delta_0$, where $\widehat{R}_{i,k}, \widehat{M}_{i,k}$, and $\widehat{S}_{i,k}$ are as stated in Theorem 3.

6. For all $k = 1,\dots,K$, let b_k define the number of active (i.e., non-zero) entries of $\bar{\mathbf{w}}_k$. Draw $\lambda_k \sim Ga(\alpha + b_k, \beta + \sum_{i=1}^{Q} W_{i,k})$.

7. For all $k = 1,\dots,K$, draw $r_k \sim Beta(e + b_k, f + Q - b_k)$, with b_k defined as in Step 6.

4.3 Algorithmic Details and Improvements for SPARFA-B

Here we discuss some several practical issues for efficiently implementing SPARFA-B, selecting the hyperparameters, and techniques for easy visualization of the SPARFA-B results.

4.3.1 Improving Computational Efficiency

The Gibbs sampling scheme of SPARFA-B enables efficient implementation in several ways. First, draws from the truncated normal in Step 1 of Section 4.2.2 are decoupled from one another, allowing them to be performed independently and, potentially, in parallel. Second, sampling of the elements in each column of $\mathbf{W}$ can be carried out in parallel by computing the relevant factors of Step 5 in matrix form. Since $K \ll Q, N$ by assumption (A1), the relevant parameters are recomputed only a relatively small number of times. One taxing computation is the calculation of the covariance matrix $\mathbf{M}_j$ for each $j = 1,\dots,N$ in Step 3. This computation is necessary, since we do not constrain each learner to answer the same set of questions which, in turn, changes the nature of the covariance calculation for each individual learner. For data sets where all learners answer the same set of questions, this covariance matrix is the same for all learners and, hence, can be carried out once per MCMC iteration.

4.3.2 Parameter Selection

The selection of the hyperparameters is performed at the discretion of the user. As is typical for Bayesian methods, non-informative (broad) hyperparameters can be used to avoid biasing results and to allow for adequate exploration of the posterior space. Tighter hyperparameters can be used when additional side information is available. For example, prior information from subject matter experts might indicate which concepts are related to which questions or might indicate the intrinsic difficulty of the questions.

Since SPARFA-M has a substantial speed advantage over SPARFA-B, it may be advantageous to first run SPARFA-M and then use its output to help in determining the hyperparameters or to initialize the SPARFA-B variables directly.

4.3.3 Post-Processing for Data Visualization

As discussed above, the generation of posterior statistics is one of the primary advantages of SPARFA-B. However, for many tasks, such as visualization of the retrieved knowledge base, it is often convenient to post-process the output of SPARFA-B to obtain point estimates for each parameter. For many Bayesian methods, simply computing the posterior mean is often sufficient. This is the case for most parameters computed by SPARFA-B, including $\mathbf{C}$ and $\boldsymbol{\mu}$. The posterior mean of $\mathbf{W}$, however, is generally non-sparse, since the MCMC will generally explore the possibility of including each entry of $\mathbf{W}$. Nevertheless, we can easily generate a sparse $\mathbf{W}$ by examining the posterior mean of the inclusion statistics contained in $\widehat{R}_{i,k}, \forall i, k$. Concretely, if the posterior mean of $\widehat{R}_{i,k}$ is small, then we set the corresponding entry of $W_{i,k}$ to zero. Otherwise, we set $W_{i,k}$ to its posterior mean. We will make use of this method throughout the experiments presented in Section 6.

4.4 Related Work on Bayesian Sparse Factor Analysis

Sparsity models for Bayesian factor analysis have been well-explored in the statistical literature (West, 2003; Tipping, 2001; Ishwaran and Rao, 2005). One popular avenue for promoting sparsity is to place a prior on the variance of each component in $\mathbf{W}$ (see, e.g., Tipping 2001, Fokoue 2004, and Pournara and Wernisch 2007). In such a model, large variance values indicate active components, while small variance values indicate inactive components. Another approach is to model active and inactive components directly using a form of a spike-slab model due to West (2003) and used in Goodfellow et al. (2012), Mohamed et al. (2012), and Hahn et al. (2012):

$$W_{i,k} \sim r_k \mathcal{N}(0, v_k) + (1 - r_k)\, \delta_0, \;\; v_k \sim IG(\alpha, \beta), \;\; \text{and} \;\; r_k \sim Beta(e, f).$$

The approach employed in (10) utilizes a spike-slab prior with an exponential distribution, rather than a normal distribution, for the active components of $\mathbf{W}$. We chose this prior for several reasons. First, it enforces the non-negativity assumption (A3). Second, it induces a posterior distribution that can be both computed in closed form and sampled efficiently. Third, its tail is slightly heavier than that of a standard normal distribution, which improves the exploration of quantities further away from zero.

A sparse factor analysis model with non-negativity constraints that is related to the one proposed here was discussed in Meng et al. (2010), although their methodology is quite

different from ours. Specifically, they impose non-negativity on the (dense) matrix $\mathbf{C}$ rather than on the sparse factor loading matrix $\mathbf{W}$. Furthermore, they enforce non-negativity using a truncated normal[7] rather than an exponential prior.

5. Tag Analysis: Post-Processing to Interpret the Estimated Concepts

So far we have developed SPARFA-M and SPARFA-B to estimate $\mathbf{W}$, $\mathbf{C}$, and $\boldsymbol{\mu}$ (or equivalently, $\mathbf{M}$) in (2) given the partial binary observations in $\mathbf{Y}$. Both $\mathbf{W}$ and $\mathbf{C}$ encode a small number of latent concepts. As we initially noted, the concepts are "*abstract*" in that they are estimated from the data rather than dictated by a subject matter expert. In this section we develop a principled post-processing approach to interpret the meaning of the abstract concepts after they have been estimated from learner responses, which is important if our results are to be usable for learning analytics and content analytics in practice. Our approach applies when the questions come with a set of user-generated "tags" or "labels" that describe in a free-form manner what ideas underlie each question.

We develop a post-processing algorithm for the estimated matrices $\mathbf{W}$ and $\mathbf{C}$ that estimates the association between the latent concepts and the user-generated tags, enabling concepts to be interpreted as a "bag of tags." Additionally, we show how to extract a personalized tag knowledge profile for each learner. The efficacy of our tag-analysis framework will be demonstrated in the real-world experiments in Section 6.2.

5.1 Incorporating Question–Tag Information

Suppose that a set of tags has been generated for each question that represent the topic(s) or theme(s) of each question. The tags could be generated by the course instructors, subject matter experts, learners, or, more broadly, by crowd-sourcing. In general, the tags provide a redundant representation of the true knowledge components, i.e., concepts are associated to a "bag of tags."

Assume that there is a total number of M tags associated with the Q questions. We form a $Q \times M$ matrix $\mathbf{T}$, where each column of $\mathbf{T}$ is associated to one of the M pre-defined tags. We set $T_{i,m} = 1$ if tag $m \in \{1, \ldots, M\}$ is present in question i and 0 otherwise. Now, we postulate that the question association matrix $\mathbf{W}$ extracted by SPARFA can be further factorized as $\mathbf{W} = \mathbf{TA}$, where $\mathbf{A}$ is an $M \times K$ matrix representing the tags-to-concept mapping. This leads to the following additional assumptions:

(A4) *Non-negativity:* The matrix $\mathbf{A}$ is non-negative. This increases the interpretability of the result, since concepts should not be negatively correlated with any tags, in general.

(A5) *Sparsity:* Each column of $\mathbf{A}$ is sparse. This ensures that the estimated concepts relate to only a few tags.

7. One could alternatively employ a truncated normal distribution on the support $[0, \infty)$ for the active entries in $\mathbf{W}$. In experiments with this model, we found a slight, though noticeable, improvement in prediction performance on real-data experiments using the exponential prior.

5.2 Estimating the Concept–Tag Associations and Learner–Tag Knowledge

The assumptions (A4) and (A5) enable us to extract $\mathbf{A}$ using ℓ_1-norm regularized non-negative least-squares as described in Hastie et al. (2010) and Chen et al. (1998). Specifically, to obtain each column $\mathbf{a}_k$ of $\mathbf{A}$, $k = 1, \ldots, K$, we solve the following convex optimization problem, a non-negative variant of *basis pursuit denoising*:

$$(\text{BPDN}_+) \quad \underset{\mathbf{a}_k \,:\, A_{m,k} \geq 0 \;\forall m}{\text{minimize}} \;\; \tfrac{1}{2}\|\mathbf{w}_k - \mathbf{T}\mathbf{a}_k\|_2^2 + \eta\|\mathbf{a}_k\|_1 \,.$$

Here, $\mathbf{w}_k$ represents the k^{th} column of $\mathbf{W}$, and the parameter η controls the sparsity level of the solution $\mathbf{a}_k$.

We propose a first-order method derived from the FISTA framework in Beck and Teboulle (2009) to solve (BPDN_+). The algorithm consists of two steps: A gradient step with respect to the ℓ_2-norm penalty function, and a projection step with respect to the ℓ_1-norm regularizer subject to the non-negative constraints on $\mathbf{a}_k$. By solving (BPDN_+) for $k = 1, \ldots, K$, and building $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_K]$, we can (i) assign tags to each concept based on the non-zero entries in $\mathbf{A}$ and (ii) estimate a tag-knowledge profile for each learner.

5.2.1 Associating Tags to Each Concept

Using the concept–tag association matrix $\mathbf{A}$ we can directly associate tags to each concept estimated by the SPARFA algorithms. We first normalize the entries in $\mathbf{a}_k$ such that they sum to one. With this normalization, we can then calculate percentages that show the proportion of each tag that contributes to concept k corresponding to the non-zero entries of $\mathbf{a}_k$. This concept tagging method typically will assign multiple tags to each concept, thus, enabling one to identify the coarse meaning of each concept (see Section 6.2 for examples using real-world data).

5.2.2 Learner Tag Knowledge Profiles

Using the concept–tag association matrix $\mathbf{A}$, we can assess each learner's knowledge of each tag. To this end, we form an $M \times N$ matrix $\mathbf{U} = \mathbf{AC}$, where the $U_{m,j}$ characterizes the knowledge of learner j of tag m. This information could be used, for example, by a PLS to automatically inform each learner which tags they have strong knowledge of and which tags they do not. Course instructors can use the information contained in $\mathbf{U}$ to extract measures representing the knowledge of all learners on a given tag, e.g., to identify the tags for which the entire class lacks strong knowledge. This information would enable the course instructor to select future learning content that deals with those specific tags. A real-world example demonstrating the efficacy of this framework is shown below in Section 6.2.1.

6. Experiments

In this section, we validate SPARFA-M and SPARFA-B on both synthetic and real-world educational data sets. First, using synthetic data, we validate that both algorithms can accurately estimate the underlying factors from binary-valued observations and characterize their performance under different circumstances. Specifically, we benchmark the factor estimation performance of SPARFA-M and SPARFA-B against a variant of the well-established

K-SVD algorithm (Aharon et al. 2006) used in dictionary-learning applications. Second, using real-world graded learner-response data we demonstrate the efficacy SPARFA-M (both probit and logit variants) and of SPARFA-B for learning and content analytics. Specifically, we showcase how the estimated learner concept knowledge, question–concept association, and intrinsic question difficulty can support machine learning-based personalized learning. Finally, we compare SPARFA-M against the recently proposed binary-valued collaborative filtering algorithm CF-IRT (Bergner et al. 2012) that predicts unobserved learner responses.

6.1 Synthetic Data Experiments

We first characterize the estimation performance of SPARFA-M and SPARFA-B using synthetic test data generated from a known ground truth model. We generate instances of $\mathbf{W}$, $\mathbf{C}$, and $\boldsymbol{\mu}$ under pre-defined distributions and then generate the binary-valued observations $\mathbf{Y}$ according to (2).

Our report on the synthetic experiments is organized as follows. In Section 6.1.1, we outline K-SVD$_+$, a variant of the well-established K-SVD dictionary-learning (DL) algorithm originally proposed in Aharon et al. (2006); we use it as a baseline method for comparison to both SPARFA algorithms. In Section 6.1.2 we detail the performance metrics. We compare SPARFA-M, SPARFA-B, and K-SVD$_+$ as we vary the problem size and number of concepts (Section 6.1.3), observation incompleteness (Section 6.1.4), and the sparsity of $\mathbf{W}$ (Section 6.1.5). In the above-referenced experiments, we simulate the observation matrix $\mathbf{Y}$ via the inverse probit link function and use only the probit variant of SPARFA-M in order to make a fair comparison with SPARFA-B. In a real-world situation, however, the link function is generally unknown. In Section 6.1.6 we conduct model-mismatch experiments, where we generate data from one link function but analyze assuming the other.

In all synthetic experiments, we average the results of all performance measures over 25 Monte-Carlo trials, limited primarily by the computational complexity of SPARFA-B, for each instance of the model parameters we control.

6.1.1 Baseline Algorithm: K-SVD$_+$

Since we are not aware of any existing algorithms to solve (2) subject to the assumptions (A1)–(A3), we deploy a novel baseline algorithm based on the well-known K-SVD algorithm of Aharon et al. (2006), which is widely used in various dictionary learning settings but ignores the inverse probit or logit link functions. Since the standard K-SVD algorithm also ignores the non-negativity constraint used in the SPARFA model, we develop a variant of the non-negative K-SVD algorithm proposed in Aharon et al. (2005) that we refer to as K-SVD$_+$. In the sparse coding stage of K-SVD$_+$, we use the non-negative variant of orthogonal matching pursuit (OMP) outlined in Bruckstein et al. (2008); that is, we enforce the non-negativity constraint by iteratively picking the entry corresponding to the maximum inner product without taking its absolute value. We also solve a non-negative least-squares problem to determine the residual error for the next iteration. In the dictionary update stage of K-SVD$_+$, we use a variant of the rank-one approximation algorithm detailed in (Aharon et al., 2005, Figure 4), where we impose non-negativity on the elements in $\mathbf{W}$ but not on the elements of $\mathbf{C}$.

K-SVD$_+$ has as input parameters the sparsity level of each row of $\mathbf{W}$. In what follows, we provide K-SVD$_+$ with the known *ground truth* for the number of non-zero components in order to obtain its best-possible performance. This will favor K-SVD$_+$ over both SPARFA algorithms, since, in practice, such oracle information is not available.

6.1.2 Performance Measures

In each simulation, we evaluate the performance of SPARFA-M, SPARFA-B, and K-SVD$_+$ by comparing the fidelity of the estimates $\widehat{\mathbf{W}}$, $\widehat{\mathbf{C}}$, and $\hat{\boldsymbol{\mu}}$ to the ground truth $\mathbf{W}$, $\mathbf{C}$, and $\boldsymbol{\mu}$. Performance evaluation is complicated by the facts that (i) SPARFA-B outputs posterior distributions rather than simple point estimates of the parameters and (ii) factor-analysis methods are generally susceptible to permutation of the latent factors. We address the first concern by post-processing the output of SPARFA-B to obtain point estimates for $\mathbf{W}$, $\mathbf{C}$, and $\boldsymbol{\mu}$ as detailed in Section 4.3.3 using $\widehat{R}_{i,k} < 0.35$ for the threshold value. We address the second concern by normalizing the columns of $\mathbf{W}$, $\widehat{\mathbf{W}}$ and the rows of $\mathbf{C}$, $\widehat{\mathbf{C}}$ to unit ℓ_2-norm, permuting the columns of $\widehat{\mathbf{W}}$ and $\widehat{\mathbf{C}}$ to best match the ground truth, and then compare $\mathbf{W}$ and $\mathbf{C}$ with the estimates $\widehat{\mathbf{W}}$ and $\widehat{\mathbf{C}}$. We also compute the Hamming distance between the support set of $\mathbf{W}$ and that of the (column-permuted) estimate $\widehat{\mathbf{W}}$. To summarize, the performance measures used in the sequel are

$$E_{\mathbf{W}} = \|\mathbf{W} - \widehat{\mathbf{W}}\|_F^2 / \|\mathbf{W}\|_F^2, \qquad E_{\mathbf{C}} = \|\mathbf{C} - \widehat{\mathbf{C}}\|_F^2 / \|\mathbf{C}\|_F^2,$$
$$E_{\boldsymbol{\mu}} = \|\boldsymbol{\mu} - \hat{\boldsymbol{\mu}}\|_2^2 / \|\boldsymbol{\mu}\|_2^2, \qquad E_{\mathbf{H}} = \|\mathbf{H} - \widehat{\mathbf{H}}\|_F^2 / \|\mathbf{H}\|_F^2,$$

where $\mathbf{H} \in \{0,1\}^{Q \times K}$ with $H_{i,k} = 1$ if $W_{i,k} > 0$ and $H_{i,k} = 0$ otherwise. The $Q \times K$ matrix $\widehat{\mathbf{H}}$ is defined analogously using $\widehat{\mathbf{W}}$.

6.1.3 Impact of Problem Size and Number of Concepts

In this experiment, we study the performance of SPARFA vs. KSVD$_+$ as we vary the number of learners N, the number of questions Q, and the number of concepts K.

Experimental setup We vary the number of learners N and the number of questions $Q \in \{50, 100, 200\}$, and the number of concepts $K \in \{5, 10\}$. For each combination of (N, Q, K), we generate $\mathbf{W}$, $\mathbf{C}$, $\boldsymbol{\mu}$, and $\mathbf{Y}$ according to (10) and (11) with $v_\mu = 1$, $\lambda_k = 2/3$ $\forall k$, and $\mathbf{V}_0 = \mathbf{I}_K$. For each instance, we choose the number of non-zero entries in each row of $\mathbf{W}$ as $DU(1,3)$ where $DU(a,b)$ denotes the discrete uniform distribution in the range a to b. For each trial, we run the probit version of SPARFA-M, SPARFA-B, and K-SVD$_+$ to obtain the estimates $\widehat{\mathbf{W}}$, $\widehat{\mathbf{C}}$, $\hat{\boldsymbol{\mu}}$, and calculate $\widehat{\mathbf{H}}$. For all of the synthetic experiments with SPARFA-M, we set the regularization parameters $\gamma = 0.1$ and select λ using the BIC (Hastie et al. 2010). For SPARFA-B, we set the hyperparameters to $h = K + 1$, $v_\mu = 1$, $\alpha = 1$, $\beta = 1.5$, $e = 1$, and $f = 1.5$; moreover, we burn-in the MCMC for 30,000 iterations and take output samples over the next 30,000 iterations.

Results and discussion Figure 3 shows box-and-whisker plots for the three algorithms and the four performance measures. We observe that the performance of all of the algorithms generally improves as the problem size increases. Moreover, SPARFA-B has superior performance for $E_{\mathbf{W}}$, $E_{\mathbf{C}}$, and $E_{\boldsymbol{\mu}}$. We furthermore see that both SPARFA-B and SPARFA-M outperform K-SVD$_+$ on $E_{\mathbf{W}}$, $E_{\mathbf{C}}$, and especially $E_{\boldsymbol{\mu}}$. K-SVD$_+$ performs very well in terms of

$E_{\mathbf{H}}$ (slightly better than both SPARFA-M and SPARFA-B) due to the fact that we provide it with the oracle sparsity level, which is, of course, not available in practice. SPARFA-B's improved estimation accuracy over SPARFA-M comes at the price of significantly higher computational complexity. For example, for $N = Q = 200$ and $K = 5$, SPARFA-B requires roughly 10 minutes on a 3.2 GHz quad-core desktop PC, while SPARFA-M and K-SVD$_+$ require only 6 s.

In summary, SPARFA-B is well-suited to small problems where solution accuracy or the need for confidence statistics are the key factors; SPARFA-M, in contrast, is destined for analyzing large-scale problems where low computational complexity (e.g., to generate immediate learner feedback) is important.

6.1.4 Impact of the Number of Incomplete Observations

In this experiment, we study the impact of the number of observations in $\mathbf{Y}$ on the performance of the probit version of SPARFA-M, SPARFA-B, and K-SVD$_+$.

Experimental setup We set $N = Q = 100$, $K = 5$, and all other parameters as in Section 6.1.3. We then vary the percentage P_{obs} of entries in $\mathbf{Y}$ that are observed as 100%, 80%, 60%, 40%, and 20%. The locations of missing entries are generated i.i.d. and uniformly over the entire matrix.

Results and discussion Figure 4 shows that the estimation performance of all methods degrades gracefully as the percentage of missing observations increases. Again, SPARFA-B outperforms the other algorithms on $E_{\mathbf{W}}$, $E_{\mathbf{C}}$, and $E_{\boldsymbol{\mu}}$. K-SVD$_+$ performs worse than both SPARFA algorithms except on $E_{\mathbf{H}}$, where it achieves comparable performance. We conclude that SPARFA-M and SPARFA-B can both reliably estimate the underlying factors, even in cases of highly incomplete data.

6.1.5 Impact of Sparsity Level

In this experiment, we study the impact of the sparsity level in $\mathbf{W}$ on the performance of the probit version of SPARFA-M, SPARFA-B, and K-SVD$_+$.

Experimental setup We choose the active entries of $\mathbf{W}$ i.i.d. $Ber(q)$ and vary $q \in \{0.2, 0.4, 0.6, 0.8\}$ to control the number of non-zero entries in each row of $\mathbf{W}$. All other parameters are set as in Section 6.1.3. This data-generation method allows for scenarios in which some rows of $\mathbf{W}$ contain no active entries, as well as scenarios with all active entries. We set the hyperparameters for SPARFA-B to $h = K + 1 = 6$, $v_\mu = 1$, and $e = 1$, and $f = 1.5$. For $q = 0.2$ we set $\alpha = 2$ and $\beta = 5$. For $q = 0.8$ we set $\alpha = 5$ and $\beta = 2$. For all other cases, we set $\alpha = \beta = 2$.

Results and discussion Figure 5 shows that sparser $\mathbf{W}$ lead to lower estimation errors. This demonstrates that the SPARFA algorithms are well-suited to applications where the underlying factors have a high level of sparsity. SPARFA-B outperforms SPARFA-M across all metrics. The performance of K-SVD$_+$ is worse than both SPARFA algorithms except on the support estimation error $E_{\mathbf{H}}$, which is due to the fact that K-SVD$_+$ is aware of the oracle sparsity level.

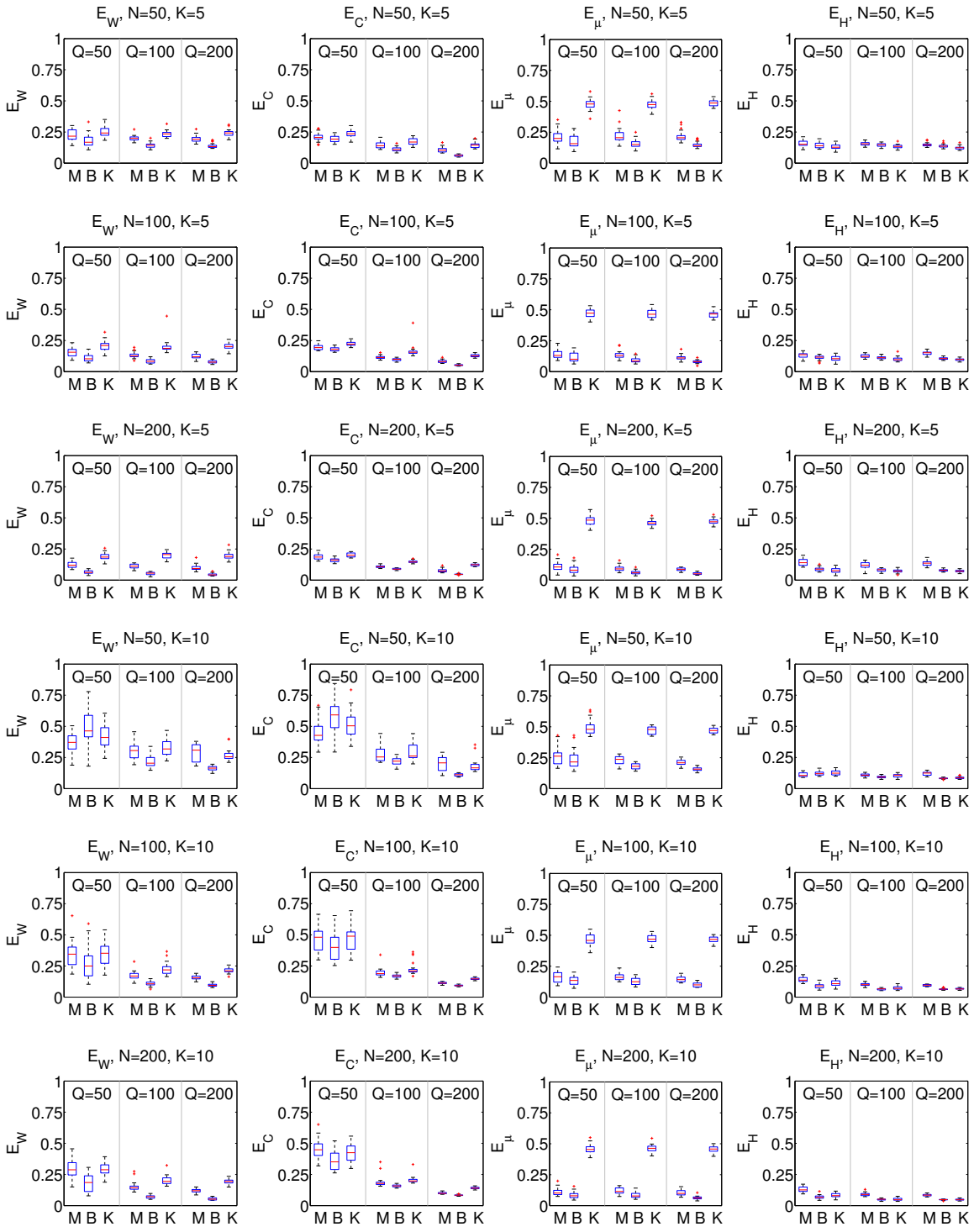


Figure 3: Performance comparison of SPARFA-M, SPARFA-B, and K-SVD$_+$ for different problem sizes $Q \times N$ and number of concepts K. The performance naturally improves as the problem size increases, while both SPARFA algorithms outperform K-SVD$_+$. M denotes SPARFA-M, B denotes SPARFA-B, and K denotes KSVD$_+$.

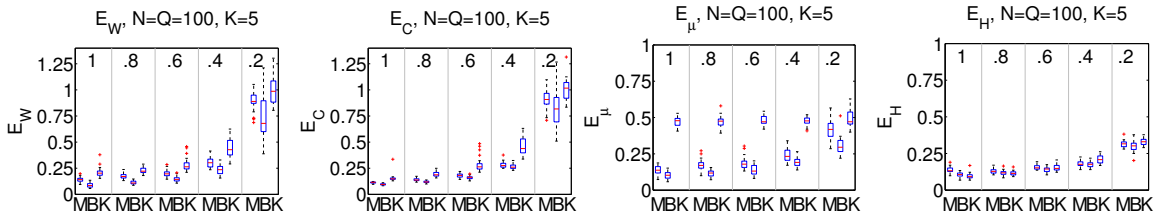


Figure 4: Performance comparison of SPARFA-M, SPARFA-B, and K-SVD$_+$ for different percentages of observed entries in $\mathbf{Y}$. The performance degrades gracefully as the number of observations decreases, while the SPARFA algorithms outperform K-SVD$_+$.

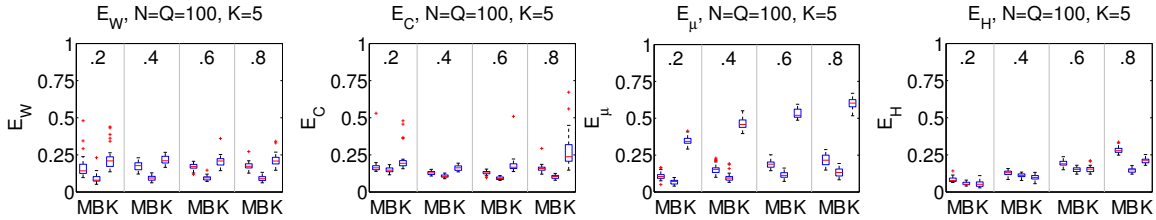


Figure 5: Performance comparison of SPARFA-M, SPARFA-B, and K-SVD$_+$ for different sparsity levels in the rows in $\mathbf{W}$. The performance degrades gracefully as the sparsity level increases, while the SPARFA algorithms outperform K-SVD$_+$.

6.1.6 Impact of Model Mismatch

In this experiment, we examine the impact of model mismatch by using a link function for estimation that does not match the true link function from which the data is generated.

Experimental setup We fix $N = Q = 100$ and $K = 5$, and set all other parameters as in Section 6.1.3. Then, for each generated instance of $\mathbf{W}$, $\mathbf{C}$ and $\boldsymbol{\mu}$, we generate $\mathbf{Y}_{\text{pro}}$ and $\mathbf{Y}_{\text{log}}$ according to both the inverse probit link and the inverse logit link, respectively. We then run SPARFA-M (both the probit and logit variants), SPARFA-B (which uses only the probit link function), and K-SVD$_+$ on both $\mathbf{Y}_{\text{pro}}$ and $\mathbf{Y}_{\text{log}}$.

Results and discussion Figure 6 shows that model mismatch does not severely affect $E_{\mathbf{W}}$, $E_{\mathbf{C}}$, and $E_{\mathbf{H}}$ for both SPARFA-M and SPARFA-B. However, due to the difference in the functional forms between the probit and logit link functions, model mismatch does lead to an increase in $E_{\boldsymbol{\mu}}$ for both SPARFA algorithms. We also see that K-SVD$_+$ performs worse than both SPARFA methods, because it ignores the link function.

6.2 Real Data Experiments

We next test the SPARFA algorithms on three real-world educational data sets. Since all variants of SPARFA-M and SPARFA-B obtained similar results in the synthetic data experiments in Section 6.1, for the sake of brevity, we will show the results for only one of the algorithms for each data set. In what follows, we select the sparsity penalty param-

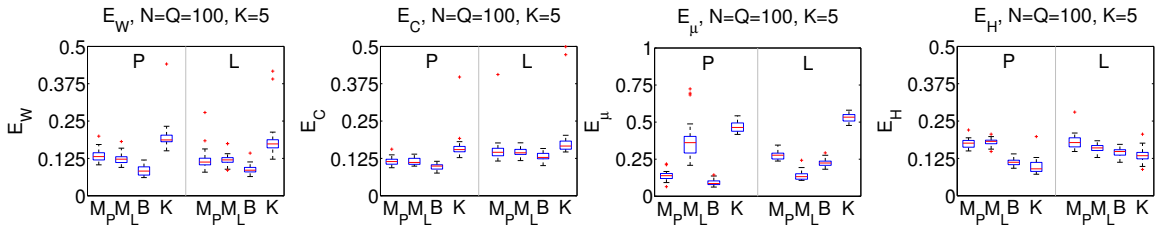


Figure 6: Performance comparison of SPARFA-M, SPARFA-B, and K-SVD$_+$ with probit/logit model mismatch; $\mathsf{M_P}$ and $\mathsf{M_L}$ indicate probit and logit SPARFA-M, respectively. In the left/right halves of each box plot, we generate $\mathbf{Y}$ according to the inverse probit/logit link functions. The performance degrades only slightly with mismatch, while both SPARFA algorithms outperform K-SVD$_+$.

eter λ in SPARFA-M using the BIC as described in Hastie et al. (2010) and choose the hyperparameters for SPARFA-B to be largely non-informative.

6.2.1 Undergraduate DSP Course

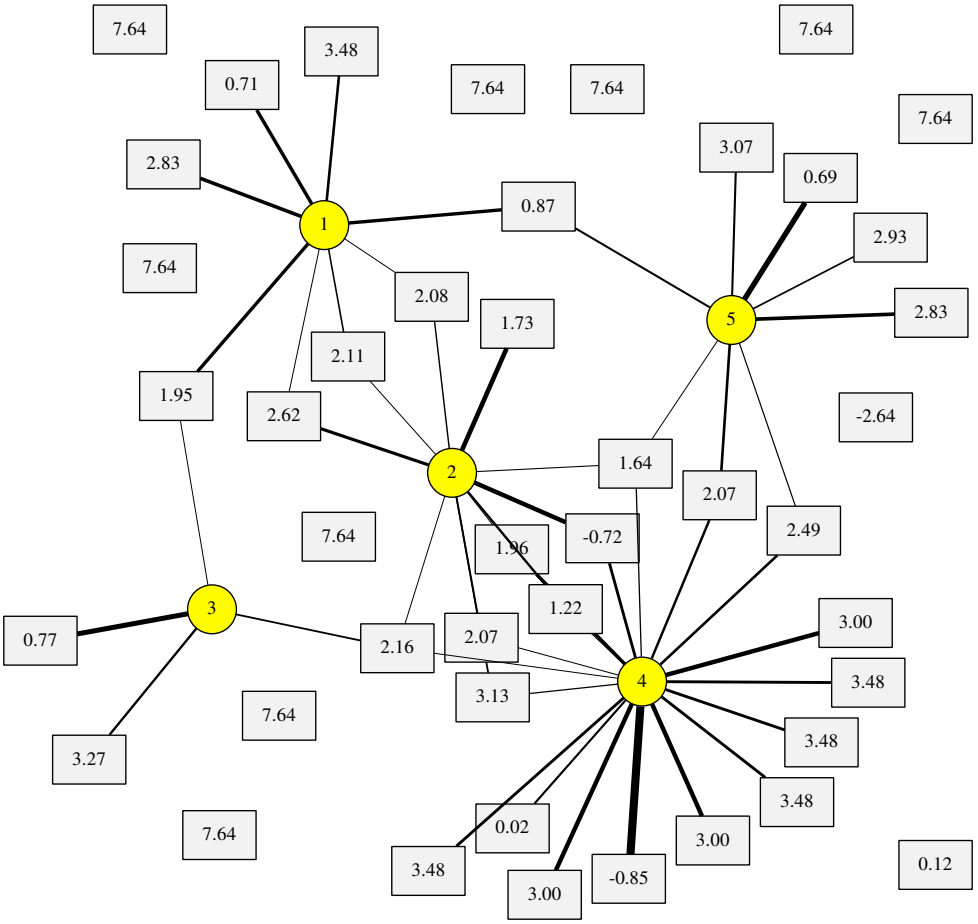
Data set We analyze a very small data set consisting of $N = 15$ learners answering $Q = 44$ questions taken from the final exam of an introductory course on digital signal processing (DSP) taught at Rice University in Fall 2011 (ELEC 301, Rice University 2011). There is no missing data in the matrix $\mathbf{Y}$.

Analysis We estimate $\mathbf{W}$, $\mathbf{C}$, and $\boldsymbol{\mu}$ from $\mathbf{Y}$ using the logit version of SPARFA-M assuming $K = 5$ concepts to achieve a concept granularity that matches the complexity of the analyzed data set. Since the questions had been manually tagged by the course instructor, we deploy the tag-analysis approach proposed in Section 5. Specifically, we form a 44×12 matrix $\mathbf{T}$ using the $M = 12$ available tags and estimate the 12×5 concept–tag association matrix $\mathbf{A}$ in order to interpret the meaning of each retrieved concept. For each concept, we only show the top 3 tags and their relative contributions. We also compute the 12×15 learner tag knowledge profile matrix $\mathbf{U}$.

Results and discussion Figure 7(a) visualizes the estimated question–concept association matrix $\widehat{\mathbf{W}}$ as a bipartite graph consisting of question and concept nodes.[8] In the graph, circles represent the estimated concepts and squares represent questions, with thicker edges indicating stronger question–concept associations (i.e., larger entries $\widehat{W}_{i,k}$). Questions are also labeled with their estimated intrinsic difficulty μ_i, with larger positive values of μ_i indicating easier questions. Note that ten questions are not linked to any concept. All $Q = 15$ learners answered these questions correctly; as a result nothing can be estimated about their underlying concept structure. Figure 7(b) provides the concept–tag association (top 3 tags) for each of the 5 estimated concepts.

Table 1 provides Learner 1's knowledge of the various tags relative to other learners. Large positive values mean that Learner 1 has strong knowledge of the tag, while large negative values indicate a deficiency in knowledge of the tag. Table 2 shows the average tag

8. To avoid the scaling identifiability problem that is typical in factor analysis, we normalize each row of $\mathbf{C}$ to unit ℓ_2-norm and scale each column of $\mathbf{W}$ accordingly prior to visualizing the bipartite graph. This enables us to compare the strength of question–concept associations across different concepts.



(a) Question–concept association graph. Circles correspond to concepts and rectangles to questions; the values in each rectangle corresponds to that question's intrinsic difficulty.

Concept 1		Concept 2		Concept 3	
Frequency response	(46%)	Fourier transform	(40%)	z-transform	(66%)
Sampling rate	(23%)	Laplace transform	(36%)	Pole/zero plot	(22%)
Aliasing	(21%)	z-transform	(24%)	Laplace transform	(12%)
Concept 4		**Concept 5**			
Fourier transform	(43%)	Impulse response	(74%)		
Systems/circuits	(31%)	Transfer function	(15%)		
Transfer function	(26%)	Fourier transform	(11%)		

(b) Most important tags and relative weights for the estimated concepts.

Figure 7: (a) Question–concept association graph and (b) most important tags associated with each concept for an undergraduate DSP course with $N = 15$ learners answering $Q = 44$ questions.

z-transform	Impulse response	Transfer function	Fourier transform	Laplace transform
1.09	−1.80	−0.50	0.99	−0.77

Table 1: Selected tag knowledge of Learner 1.

z-transform	Impulse response	Transfer function	Fourier transform	Laplace transform
0.04	−0.03	−0.10	0.11	0.03

Table 2: Average tag knowledge of all learners.

knowledge of the entire class, computed by averaging the entries of each row in the learner tag knowledge matrix $\mathbf{U}$ as described in Section 5.2.2. Table 1 indicates that Learner 1 has particularly weak knowledge of the tag "Impulse response." Armed with this information, a PLS could automatically suggest remediation about this concept to Learner 1. Table 2 indicates that the entire class has (on average) weak knowledge of the tag "Transfer function." With this information, a PLS could suggest to the class instructor that they provide remediation about this concept to the entire class.

6.2.2 Grade 8 Science Course

Data set The STEMscopes data set was introduced in Section 1.2. There is substantial missing data in the matrix $\mathbf{Y}$, with only 13.5% of its entries observed.

Analysis We compare the results of SPARFA-M and SPARFA-B on this data set to highlight the pros and cons of each approach. For both algorithms, we select $K = 5$ concepts. For SPARFA-B, we fix reasonably broad (non-informative) values for all hyperparameters. For μ_0 we calculate the average rate of correct answers p_s on observed graded responses of all learners to all questions and use $\mu_0 = \Phi_{\text{pro}}^{-1}(p_s)$. The variance $v_{\boldsymbol{\mu}}$ is left sufficiently broad to enable adequate exploration of the intrinsic difficulty for each questions. Point estimates of $\mathbf{W}$, $\mathbf{C}$ and $\boldsymbol{\mu}$ are generated from the SPARFA-B posterior distributions using the methods described in Section 4.3.3. Specifically, an entry $\widehat{W}_{i,k}$ that has a corresponding active probability $\widehat{R}_{i,k} < 0.55$ is thresholded to 0. Otherwise, we set $\widehat{W}_{i,k}$ to its posterior mean. On a 3.2 GHz quad-core desktop PC, SPARFA-M converged to its final estimates in 4 s, while SPARFA-B required 10 minutes.

Results and discussion Both SPARFA-M and SPARFA-B deliver comparable factorizations. The estimated question–concept association graph for SPARFA-B is shown in Figure 2(a), with the accompanying concept–tag association in Figure 2(b). Again we see a sparse relationship between questions and concepts. The few outlier questions that are not associated with any concept are generally those questions with very low intrinsic difficulty or those questions with very few responses.

One advantage of SPARFA-B over SPARFA-M is its ability to provide not only point estimates of the parameters of interest but also reliability information for those estimates. This reliability information can be useful for decision making, since it enables one to tailor

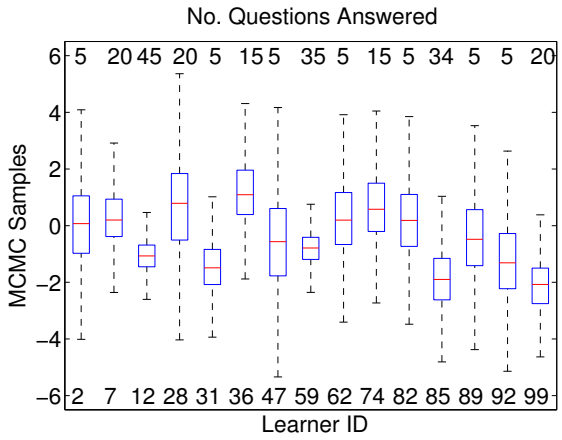


Figure 8: Concept 5 knowledge estimates generated by SPARFA-B for the STEMscopes data for a randomly selected subset of learners. The box-whisker plot shows the posterior variance of the MCMC samples, with each box-whisker plot corresponding to a different learner in the data set. Anonymized learner IDs are shown on the bottom, while the number of relevant questions answered by each learner answered is indicated on the top of the plot.

actions according to the associated uncertainty. If there is considerable uncertainty regarding learner mastery of a particular concept, for example, it may be a more appropriate use of time of the learner to ask additional questions that reduce the uncertainty, rather than assigning new material for which the learner may not be adequately prepared.

We demonstrate the utility of SPARFA-B's posterior distribution information on the learner concept knowledge matrix $\mathbf{C}$. Figure 8 shows box-whisker plots of the MCMC output samples over 30,000 iterations (after a burn-in period of 30,000 iterations) for a set of learners for Concept 5. Each box-whisker plot corresponds to the posterior distribution for a different learner. These plots enable us to visualize both the posterior mean and variance associated with the concept knowledge estimates $\hat{\mathbf{c}}_j$. As one would expect, the estimation variance tends to decrease as the number of answered questions increases (shown in the top portion of Figure 8).

The exact set of questions answered by a learner also affects the posterior variance of our estimate, as different questions convey different levels of information regarding a learner's concept mastery. An example of this phenomenon is observed by comparing Learners 7 and 28. Each of these two learners answered 20 questions and had a nearly equal number of correct answers (16 and 17, respectively). A conventional analysis that looked only at the percentage of correct answers would conclude that both learners have similar concept mastery. However, the actual set of questions answered by each learner is not the same, due to their respective instructors assigning different questions. While SPARFA-B finds a similar posterior mean for Learner 7 and Learner 28, it finds very different posterior variances, with considerably more variance for Learner 28. The SPARFA-B posterior samples shed additional light on the situation at hand. Most of the questions answered by Learner 28 are deemed easy (defined as having intrinsic difficulties $\hat{\mu}_i$ larger than one). Moreover, the remaining, more difficult questions answered by Learner 28 show stronger affinity to concepts

		C1	C2	C3	C4	C5
Q3 (27 responses)	M	Yes	No	No	No	Yes
	B	94%	36%	48%	18%	80%
Q56 (5 responses)	M	No	No	No	No	No
	B	30%	30%	26%	31%	31%
Q72 (6 responses)	M	No	No	No	No	Yes
	B	61%	34%	29%	36%	58%

Table 3: Comparison of SPARFA-M and SPARFA-B for three selected questions and the $K = 5$ estimated concepts in the STEMscopes data set. For SPARFA-M, the labels "Yes" and "No" indicate whether a particular concept was detected in the question. For SPARFA-B, we show the posterior inclusion probability (in percent), which indicates the percentage of iterations in which a particular concept was sampled.

other than Concept 5. In contrast, roughly half of the questions answered by Learner 7 are deemed hard and all of these questions have stronger affinity to Concept 5. Thus, the questions answered by Learner 28 convey only weak information about the knowledge of Concept 5, while those answered by Learner 7 convey strong information. Thus, we cannot determine from Learner 28's responses whether they have mastered Concept 5 well or not. Such SPARFA-B posterior data would enable a PLS to quickly assess this scenario and tailor the presentation of future questions to Learner 28—in this case, presenting more difficult questions related to Concept 5 would reduce the estimation variance on their concept knowledge and allow a PLS to better plan future educational tasks for this particular learner.

Second, we demonstrate the utility of SPARFA-B's posterior distribution information on the question–concept association matrix $\mathbf{W}$. Accurate estimation of $\mathbf{W}$ enables course instructors and content authors to validate the extent to which problems measure knowledge across various concepts. In general, there is a strong degree of commonality between the results of SPARFA-M and SPARFA-B, especially as the number of learners answering a question grow. We present some illustrative examples of support estimation on $\mathbf{W}$ for both SPARFA algorithms in Table 3. We use the labels "Yes"/"No" to indicate inclusion of a concept by SPARFA-M and show the posterior inclusion probabilities for each concept by SPARFA-B. Here, both SPARFA-M and SPARFA-B agree strongly on both Question 3 and Question 56. Question 72 is answered by only 6 learners, and SPARFA-M discovers a link between this question and Concept 5. SPARFA-B proposes Concept 5 in 58% of all MCMC iterations, but also Concept 1 in 60% of all MCMC iterations. Furthermore, the proposals of Concept 1 and Concept 5 are nearly mutually exclusive; in most iterations only one of the two concepts is proposed, but both are rarely proposed jointly. This behavior implies that SPARFA-B has found two competing models that explain the data associated with Question 72. To resolve this ambiguity, a PLS would need to gather more learner responses.

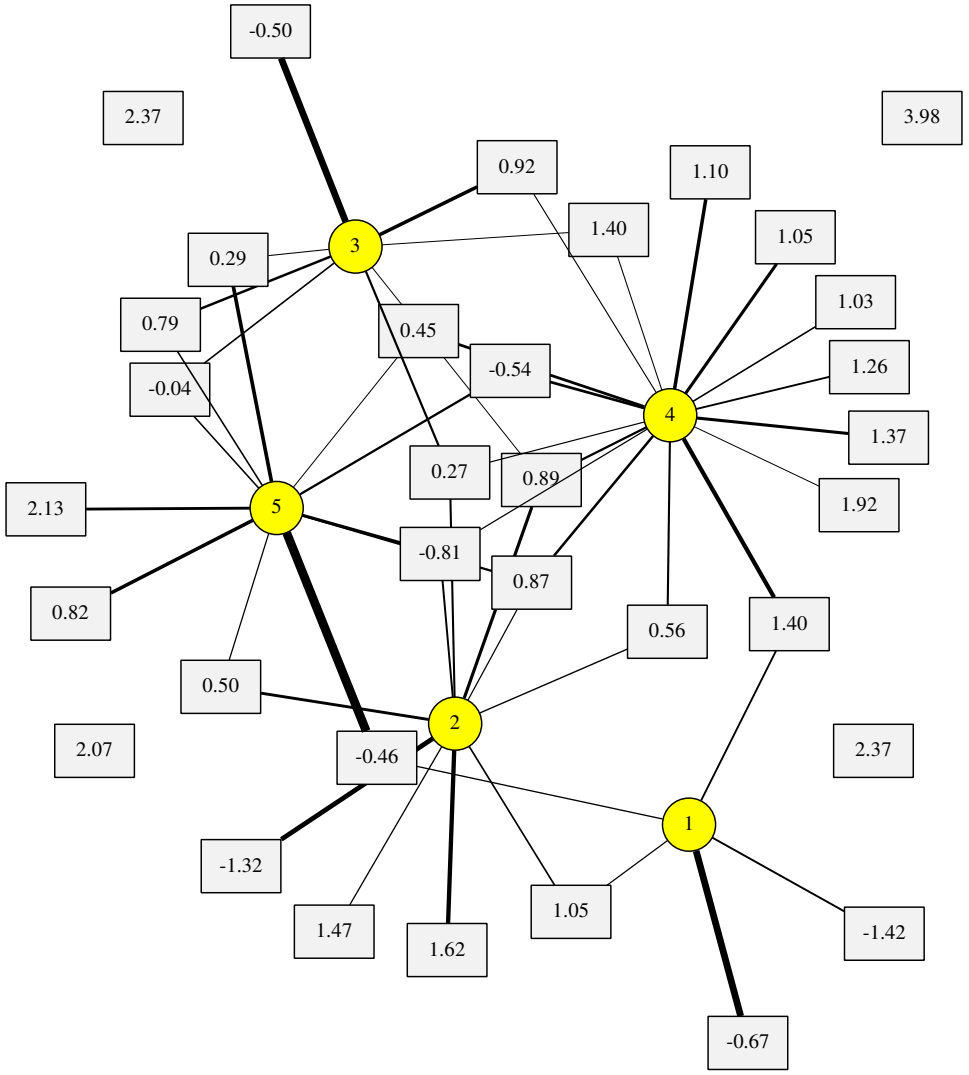
6.2.3 Algebra Test Administered on Amazon Mechanical Turk

For a final demonstration of the capabilities the SPARFA algorithms, we analyze a data set from a high school algebra test carried out by Daniel Calderón of Rice University on Amazon Mechanical Turk, a crowd-sourcing marketplace (Amazon Mechanical Turk 2012).
Data set The data set consists of $N = 99$ learners answering $Q = 34$ questions covering topics such as geometry, equation solving, and visualizing function graphs. Calderón manually labeled the questions from a set of $M = 10$ tags. The data set is fully populated, with no missing entries.
Analysis We estimate $\mathbf{W}$, $\mathbf{C}$, and $\boldsymbol{\mu}$ from the fully populated 34×99 binary-valued matrix $\mathbf{Y}$ using the logit version of SPARFA-M assuming $K = 5$ concepts. We deploy the tag-analysis approach proposed in Section 5 to interpret each concept. Additionally, we calculate the likelihoods of the responses using (1) and the estimates $\widehat{\mathbf{W}}$, $\widehat{\mathbf{C}}$ and $\hat{\boldsymbol{\mu}}$. The results from SPARFA-M are summarized in Figure 9. We detail the results of our analysis for Questions 19–26 in Table 4 and for Learner 1 in Table 5.
Results and discussion With the aid of SPARFA, we can analyze the strengths and weaknesses of each learner's concept knowledge both individually and relative to other users. We can also detect outlier responses that are due to guessing, cheating, or carelessness. The values in the estimated concept knowledge matrix $\widehat{\mathbf{C}}$ measure each learner's concept knowledge relative to all other learners. The estimated intrinsic difficulties of the questions $\hat{\boldsymbol{\mu}}$ provide a relative measure that summarizes how all users perform on each question.

Let us now consider an example in detail; see Table 4 and Table 5. Learner 1 incorrectly answered Questions 21 and 26 (see Table 4), which involve Concepts 1 and 2. Their knowledge of these concepts is not heavily penalized, however (see Table 5), due to the high intrinsic difficulty of these two questions, which means that most other users also incorrectly answered them. User 1 also incorrectly answered Questions 24 and 25, which involve Concepts 2 and 4. Their knowledge of these concepts is penalized, due to the low intrinsic difficulty of these two questions, which means that most other users correctly answered them. Finally, Learner 1 correctly answered Questions 19 and 20, which involve Concepts 1 and 5. Their knowledge of these concepts is boosted, due to the high intrinsic difficulty of these two questions.

SPARFA can also be used to identify each user's individual strengths and weaknesses. Continuing the example, Learner 1 needs to improve their knowledge of Concept 4 (associated with the tags "Simplifying expressions", "Trigonometry," and "Plotting functions") significantly, while their deficiencies on Concepts 2 and 3 are relatively minor.

Finally, by investigating the likelihoods of the graded responses, we can detect outlier responses, which would enables a PLS to detect guessing and cheating. By inspecting the concept knowledge of Learner 1 in Table 5, we can identify insufficient knowledge of Concept 4. Hence, Learner 1's correct answer to Question 22 is likely due to a random guess, since the predicted likelihood of providing the correct answer is estimated at only 0.21.



(a) Question–concept association graph.

Concept 1		Concept 2		Concept 3	
Fractions	(57%)	Plotting functions	(64%)	Geometry	(63%)
Solving equations	(42%)	System of equations	(27%)	Simplifying expressions	(27%)
Arithmetic	(1%)	Simplifying expressions	(9%)	Trigonometry	(10%)
Concept 4		**Concept 5**			
Simplifying expressions	(64%)	Trigonometry	(53%)		
Trigonometry	(21%)	Slope	(40%)		
Plotting Functions	(15%)	Solving equations	(7%)		

(b) Most important tags and relative weights for the estimated concepts.

Figure 9: (a) Question–concept association graph and (b) most important tags associated with each concept for a high-school algebra test carried out on Amazon Mechanical Turk with $N = 99$ users answering $Q = 34$ questions.

Question number	19	20	21	22	23	24	25	26
Learner's graded response $Y_{i,j}$	1	1	0	1	1	0	0	0
Correct answer likelihood $p(Y_{i,j} = 1 \mid \bar{\mathbf{w}}_i, \mathbf{c}_j, \mu_i)$	0.79	0.71	0.11	**0.21**	0.93	0.23	0.43	0.00
Underlying concepts	1	1, 5	1	2, 3, 4	3, 5	2, 4	1, 4	2, 4
Intrinsic difficulty μ_i	−1.42	−0.46	−0.67	0.27	0.79	0.56	1.40	−0.81

Table 4: Graded responses and their underlying concepts for Learner 1 (1 designates a correct response and 0 an incorrect response).

Concept number	1	2	3	4	5
Concept knowledge	0.46	−0.35	0.72	−1.67	0.61

Table 5: Estimated concept knowledge for Learner 1.

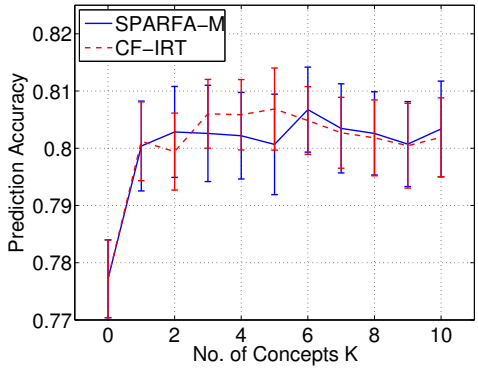
6.3 Predicting Unobserved Learner Responses

We now compare SPARFA-M against the recently proposed binary-valued collaborative filtering algorithm CF-IRT (Bergner et al. 2012) in an experiment to predict unobserved learner responses.

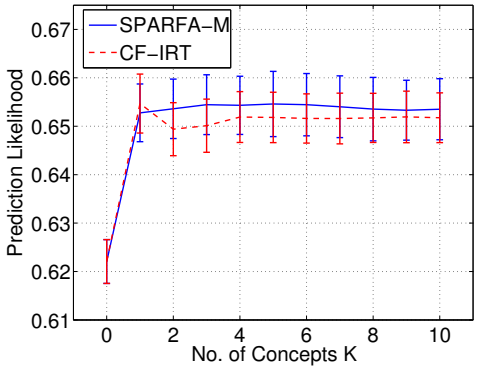
Data set and experimental setup In this section, we study both the Mechanical Turk algebra test data set and a portion of the ASSISTment data set (Pardos and Heffernan 2010). The ASSISTment data set consists of $N = 403$ learners answering $Q = 219$ questions, with 25% of the responses observed (see Vats et al. 2013 for additional details on the data set). In each of the 25 trials we run for both data sets, we hold out 20% of the observed learner responses as a test set, and train both the logistic variant of SPARFA-M[9] and CF-IRT on the rest. The regularization parameters of both algorithms are selected using 4-fold cross-validation on the training set. We use two performance metrics to evaluate the performance of these algorithms, namely (i) the prediction accuracy, which corresponds to the percentage of correctly predicted unobserved responses, and (ii) the average prediction likelihood $\frac{1}{|\bar{\Omega}_{\text{obs}}|} \sum_{i,j:(i,j)\in\bar{\Omega}_{\text{obs}}} p(Y_{i,j} | \bar{\mathbf{w}}_i, \mathbf{c}_j)$ of the unobserved responses, as proposed in González-Brenes and Mostow (2012), for example.

Results and discussion Figure 10 shows the prediction accuracy and prediction likelihood for both the Mechanical Turk algebra test data set and the ASSISTment data set. We see that SPARFA-M delivers comparable (sometimes slightly superior) prediction performance to CF-IRT in predicting unobserved learner responses.

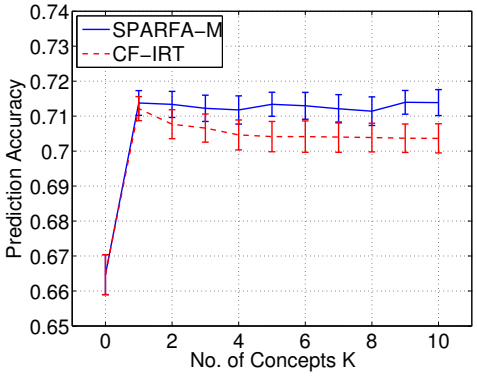
9. In order to arrive at a fair comparison, we choose to use the logistic variant of SPARFA-M, since CF-IRT also relies on a logistic model.



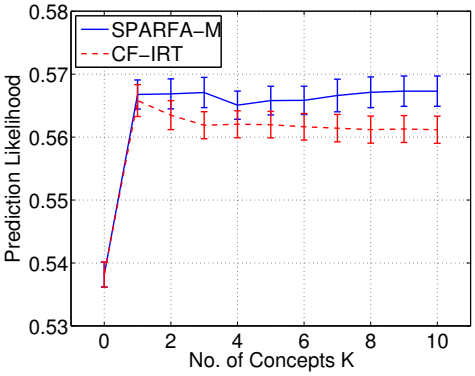
(a) Prediction accuracy for the Mechanical Turk algebra test data set.



(b) Average prediction likelihood for the Mechanical Turk algebra test data set.



(c) Prediction accuracy for the ASSISTment data set.



(d) Average prediction likelihood for the ASSISTment data set.

Figure 10: Performance comparison of SPARFA-M and CF-IRT on (a) prediction accuracy and (b) average prediction likelihood for the Mechanical Turk algebra test data set, (c) prediction accuracy and (d) average prediction likelihood for the ASSISTment data set. SPARFA-M achieves comparable or better performance than CF-IRT while enabling interpretability of the estimated latent concepts

Furthermore, we see from Figure 10 that the prediction performance varies little over different values of K, meaning that the specific choice of K has little influence on the prediction performance within a certain range. This phenomenon agrees with other collaborative filtering results (see, e.g., Koren et al. 2009; Koren and Sill 2011). Consequently, the choice of K essentially dictates the granularity of the abstract concepts we wish to estimate. We choose $K = 5$ in the real data experiments of Section 6.2 when we visualize the question–concept associations as bi-partite graphs, as it provides a desirable granularity of the estimated concepts in the data sets. We emphasize that SPARFA-M is able to provide interpretable estimated factors while achieving comparable (or slightly superior) prediction performance than that achieved by CF-IRT, which does not provide interpretability. This

feature of SPARFA is key for the development of PLSs, as it enables an automated way of generating interpretable feedback to learners in a purely data-driven fashion.

7. Related Work on Machine Learning-based Personalized Learning

A range of different machine learning algorithms have been applied in educational contexts. Bayesian belief networks have been successfully used to probabilistically model and analyze learner response data (e.g., Krudysz et al. 2006; Woolf 2008; Krudysz and McClellan 2011). Such models, however, rely on predefined question–concept dependencies (that are not necessarily the true dependencies governing learner responses) and primarily only work for a single concept. In contrast, SPARFA discovers question–concept dependencies from solely the graded learner responses to questions and naturally estimates multi-concept question dependencies.

Modeling question–concept associations has been studied in Barnes (2005), Thai-Nghe et al. (2011a), Thai-Nghe et al. (2011b), and Desmarais (2011). The approach in Barnes (2005) characterizes the underlying question–concept associations using binary values, which ignore the relative strengths of the question–concept associations. In contrast, SPARFA differentiates between strong and weak relationships through the real-valued weights $W_{i,k}$. The matrix and tensor factorization methods proposed in Barnes (2005), Thai-Nghe et al. (2011a), and Thai-Nghe et al. (2011b) treat graded learner responses as real but deterministic values. In contrast, the probabilistic framework underlying SPARFA provides a statistically principled model for graded responses; the likelihood of the observed graded responses provides even more explanatory power.

Existing intelligent tutoring systems capable of modeling question–concept relations probabilistically include Khan Academy (Dijksman and Khan 2011; Hu 2011) and the system of Bachrach et al. (2012). Both approaches, however, are limited to dealing with a single concept. In contrast, SPARFA is built from the ground up to deal with multiple latent concepts.

A probit model for graded learner responses is used in Desmarais (2011) without exploiting the idea of low-dimensional latent concepts. In contrast, SPARFA leverages multiple latent concepts and therefore can create learner concept knowledge profiles for personalized feedback. Moreover, SPARFA-M is compatible with the popular logit model.

The recent results developed in Beheshti et al. (2012) and Bergner et al. (2012) address the problem of *predicting* the missing entries in a binary-valued graded learner response matrix. Both papers use low-dimensional latent factor techniques specifically developed for collaborative filtering, as, e.g., discussed in Linden et al. (2003) and Herlocker et al. (2004). While predicting missing correctness values is an important task, these methods do not take into account the sparsity and non-negativity of the matrix $\mathbf{W}$; this inhibits the interpretation of the relationships among questions and concepts. In contrast, SPARFA accounts for both the sparsity and non-negativity of $\mathbf{W}$, which enables the interpretation of the value $C_{k,j}$ as learner j's knowledge of concept k.

There is a large body of work on item response theory (IRT), which uses statistical models to analyze and score graded question response data (see, e.g., Lord 1980, Baker and Kim 2004, and Reckase 2009 for overview articles). The main body of the IRT literature builds on the model developed by Rasch (1993) and has been applied mainly in the

context of adaptive testing, e.g., in the graduate record examination (GRE) and graduate management (GMAT) tests (Chang and Ying 2009, Thompson 2009, and Linacre 1999). While the SPARFA model shares some similarity to the model in Rasch (1993) by modeling question–concept association strengths and intrinsic difficulties of questions, it also models each learner in terms of a multi-dimensional concept knowledge vector. This capability of SPARFA is in stark contrast to the Rasch model, where each learner is characterized by a single, scalar ability parameter. Consequently, the SPARFA framework is able to provide stronger explanatory power in the estimated factors compared to that of the conventional Rasch model. We finally note that multi-dimensional variants of IRT have been proposed in McDonald (2000), Yao (2003), and Reckase (2009). We emphasize, however, that the design of these algorithms leads to poor interpretability of the resulting parameter estimates.

8. Conclusions

In this paper, we have formulated a new approach to learning and content analytics, which is based on a new statistical model that encodes the probability that a learner will answer a given question correctly in terms of three factors: (i) the learner's knowledge of a set of latent concepts, (ii) how the question relates to each concept, and (iii) the intrinsic difficulty of the question. We have proposed two algorithms, SPARFA-M and SPARFA-B, to estimate the above three factors given incomplete observations of graded learner question responses. SPARFA-M uses an efficient Maximum Likelihood-based bi-convex optimization approach to produce point estimates of the factors, while SPARFA-B uses Bayesian factor analysis to produce posterior distributions of the factors. In practice, SPARFA-M is beneficial in applications where timely results are required; SPARFA-B is favored in situations where posterior statistics are required. We have also introduced a novel method for incorporating user-defined tags on questions to facilitate the interpretability of the estimated factors. Experiments with both synthetic and real world education data sets have demonstrated both the efficacy and robustness of the SPARFA algorithms.

The quantities estimated by SPARFA can be used directly in a range of PLS functions. For instance, we can identify the knowledge level of learners on particular concepts and diagnose why a given learner has incorrectly answered a particular question or type of question. Moreover, we can discover the hidden relationships among questions and latent concepts, which is useful for identifying questions that do and do not aid in measuring a learner's conceptual knowledge. Outlier responses that are either due to guessing or cheating can also be detected. In concert, these functions can enable a PLS to generate personalized feedback and recommendation of study materials, thereby enhancing overall learning efficiency.

Various extensions and refinements to the SPARFA framework developed here have been proposed recently. Most of these results aim at improving interpretability of the SPARFA model parameters. In particular, a variant of SPARFA-M that analyzes ordinal rather than binary-valued responses and directly utilizes tag information in the probabilistic model has been detailed in Lan et al. (2013a). Another variant of SPARFA-M that further improves the interpretability of the underlying concepts via the joint analysis of graded learner responses and question/response text has been proposed in Lan et al. (2013b). A nonparametric Bayesian variant of SPARFA-B that estimates both the number of concepts K as well as the

reliability of each learner from data has been developed in Fronczyk et al. (2013, submitted). The results of this nonparametric method confirm our choice of $K = 5$ concepts for the real-world educational data sets considered in Section 6.2.

Before closing, we would like to point out a connection between SPARFA and *dictionary learning* that is of independent interest. This connection can be seen by noting that (2) for both the probit and inverse logit functions is statistically equivalent to (see Rasmussen and Williams 2006):

$$Y_{i,j} = [\text{sign}(\mathbf{WC} + \mathbf{M} + \mathbf{N})]_{i,j}, \ (i,j) \in \Omega_{\text{obs}},$$

where $\text{sign}(\cdot)$ denotes the entry-wise sign function and the entries of $\mathbf{N}$ are i.i.d. and drawn from either a standard Gaussian or standard logistic distribution. Hence, estimating $\mathbf{W}$, $\mathbf{C}$, and $\mathbf{M}$ (or equivalently, $\boldsymbol{\mu}$) is equivalent to learning a (possibly overcomplete) dictionary from the data $\mathbf{Y}$. The key departures from the dictionary-learning literature (Aharon et al. 2006; Mairal et al. 2010) and algorithm variants capable of handling missing observations (Studer and Baraniuk 2012) are the binary-valued observations and the non-negativity constraint on $\mathbf{W}$. Note that the algorithms developed in Section 3 to solve the sub-problems by holding one of the factors $\mathbf{W}$ or $\mathbf{C}$ fixed and solving for the other variable can be used to solve noisy binary-valued (or 1-bit) compressive sensing or sparse signal recovery problems, e.g., as studied in Boufounos and Baraniuk (2008), Jacques et al. (2013), and Plan and Vershynin (2012, submitted). Thus, the proposed SPARFA algorithms can be applied to a wide range of applications beyond education, including the analysis of survey data, voting patterns, gene expression, and signal recovery from noisy 1-bit compressive measurements.

Acknowledgments

Thanks to Wotao Yin and Yangyang Xu for helpful discussions on the convergence proof of SPARFA-M, Genevera Allen for insights into probit regression, Marina Vannucci for helpful discussions regarding Bayesian factor analysis, Daniel Calderón for organizing and administering the Amazon Mechanical Turk experiments in Section 6.2.3, and Carlos Monroy and Reid Whitaker for providing the STEMscopes data. We furthermore thank the anonymous reviewers for their valuable comments which improved the exposition of our results.

This work was supported by the National Science Foundation under Cyberlearning grant IIS-1124535, the Air Force Office of Scientific Research under grant FA9550-09-1-0432, the Google Faculty Research Award program.

Please see our website `www.sparfa.com`, where you can learn more about the project and purchase SPARFA t-shirts and other merchandise.

Appendix A. Proof of Theorem 1

We now establish the convergence of the FISTA algorithms that solve the SPARFA-M sub-problems $(\text{RR})_1^+$ and $(\text{RR})_2$. We start by deriving the relevant Lipschitz constants.

Lemma 4 (Scalar Lipschitz constants) *Let* $g_{pro}(x) = \frac{\Phi'_{pro}(x)}{\Phi_{pro}(x)}$ *and* $g_{log}(x) = \frac{\Phi'_{log}(x)}{\Phi_{log}(x)}$, $x \in \mathbb{R}$, *where* $\Phi_{pro}(x)$ *and* $\Phi_{log}(x)$ *are the inverse probit and logit link functions defined*

in (3) and (4), respectively. Then, for $y, z \in \mathbb{R}$ we have

$$|g_{pro}(y) - g_{pro}(z)| \leq L_{pro}|y - z|\,, \tag{12}$$
$$|g_{log}(y) - g_{log}(z)| \leq L_{log}|y - z|\,, \tag{13}$$

with the constants $L_{pro} = 1$ for the probit case and $L_{log} = 1/4$ for the logit case.

Proof For simplicity of exposition, we omit the subscripts designating the probit and logit cases in what follows. We first derive L_{pro} for the probit case by computing the derivative of $g(x)$ and bounding its derivative from below and above. The derivative of $g(x)$ is given by

$$g'(x) = -\frac{\mathcal{N}(x)}{\Phi(x)}\left(x + \frac{\mathcal{N}(x)}{\Phi(x)}\right). \tag{14}$$

where $\mathcal{N}(t) = \frac{1}{\sqrt{2\pi}}e^{-t^2/2}$ is the PDF of the standard normal distribution.

We first bound this derivative for $x \leq 0$. To this end, we individually bound the first and second factor in (14) using the following bounds listed in Olver (2010):

$$-\frac{x}{2} + \sqrt{\frac{x^2}{4} + \frac{2}{\pi}} \leq \frac{\mathcal{N}(x)}{\Phi(x)} \leq -\frac{x}{2} + \sqrt{\frac{x^2}{4} + 1}, \quad x \leq 0$$

and

$$\frac{x}{2} + \sqrt{\frac{x^2}{4} + \frac{2}{\pi}} \leq x + \frac{\mathcal{N}(x)}{\Phi(x)} \leq \frac{x}{2} + \sqrt{\frac{x^2}{4} + 1}, \quad x \leq 0.$$

Multiplying the above inequalities leads to the bounds

$$-1 \leq g'(x) \leq -\frac{2}{\pi}, \qquad x \leq 0. \tag{15}$$

We next bound the derivative of (14) for $x > 0$. For $x > 0$, $\mathcal{N}(x)$ is a positive decreasing function and $\Phi(x)$ is a positive increasing function; hence $\frac{\mathcal{N}(x)}{\Phi(x)}$ is a decreasing function and $\frac{\mathcal{N}(x)}{\Phi(x)} \leq \frac{\mathcal{N}(0)}{\Phi(0)} = \sqrt{2/\pi}$. Thus, we arrive at

$$g'(x) = -\frac{\mathcal{N}(x)}{\Phi(x)}\left(x + \frac{\mathcal{N}(x)}{\Phi(x)}\right) \geq -\frac{\mathcal{N}(x)}{\Phi(x)}\left(x + \sqrt{2/\pi}\right),$$

where we have used the facts that $\Phi(x) \geq 1/2$ and $\mathcal{N}(x) \leq \frac{1}{\sqrt{2\pi}}$ for $x > 0$. According to (3) and the bound of Chu (1955), we have

$$\Phi(x) = \frac{1}{2} + \int_0^x \mathcal{N}(t \mid 0, 1)\mathrm{d}t \geq \frac{1}{2} + \frac{1}{2}\sqrt{1 - e^{-x^2/2}} \geq 1 - \frac{1}{2}e^{-x^2/2}, \tag{16}$$

where the second inequality follows from the fact that $(1 - e^{-x^2/2}) \in [0, 1]$. Using (16) we can further bound $g'(x)$ from below as

$$g'(x) \geq -\frac{\mathcal{N}(x)}{1 - \frac{1}{2}e^{-x^2/2}}\left(x + \sqrt{2/\pi}\right).$$

Let us now assume that

$$-\frac{\mathcal{N}(x)}{1-\frac{1}{2}e^{-x^2/2}}\left(x+\sqrt{2/\pi}\right)\geq -1.$$

In order to prove that this assumption is true, we rearrange terms to obtain

$$\left(\frac{x}{\sqrt{2\pi}}+(1/\pi+1/2)\right)e^{-x^2/2}\leq 1. \tag{17}$$

Now we find the maximum of the LHS of (17) for $x>0$. To this end, we observe that $\frac{x}{\sqrt{2\pi}}+(1/\pi+1/2)$ is monotonically increasing and that $e^{-x^2/2}$ monotonically decreasing for $x>0$; hence, this function has a unique maximum in this region. By taking its derivative and setting it to zero, we obtain

$$x^2+\sqrt{2/\pi}+\sqrt{\pi/2}-1=0$$

Substituting the result of this equation, i.e., $\hat{x}\approx 0.4068$, into (17) leads to

$$\left(\frac{\hat{x}}{\sqrt{2\pi}}+(1/\pi+1/2)\right)e^{-\hat{x}^2/2}\approx 0.9027\leq 1,$$

which certifies our assumption. Hence, we have

$$-1\leq g'(x)\leq 0,\quad x>0.$$

Combining this result with the one for $x\leq 0$ in (15) yields

$$-1\leq g'(x)\leq 0,\quad x\in\mathbb{R}.$$

We finally obtain the following bound on the scalar Lipschitz constant (12):

$$|g_{\text{pro}}(y)-g_{\text{pro}}(z)|\leq\left|\int_y^z\left|g'_{\text{pro}}(x)\right|dx\right|\leq\left|\int_y^z 1\,dx\right|=|y-z|\,,$$

which concludes the proof for the probit case.

We now develop the bound $L_{\log}$ for the logit case. To this end, we bound the derivative of $g_{\log}(x)=\frac{1}{1+e^x}$ as follows:

$$0\geq g'_{\log}(x)=-\frac{e^x}{(1+e^x)^2}=-\frac{1}{e^x+e^{-x}+2}\geq -\frac{1}{4}.$$

where we used the inequality of arithmetic and geometric means. Consequently, we have the following bound on the scalar Lipschitz constant (13):

$$|g_{\log}(y)-g_{\log}(z)|\leq\left|\int_y^z\left|g'_{\log}(x)\right|dx\right|\leq|\int_y^z\frac{1}{4}\,dx|=\frac{1}{4}|y-z|,$$

which concludes the proof for the logit case. ■

The following lemma establishes a bound on the (vector) Lipschitz constants for the individual regularized regression problems (RR_1^+) and (RR_2) for both the probit and the logit case, using the results in Lemma 4. We work out in detail the analysis of (RR_1^+) for $\bar{\mathbf{w}}_i$, i.e., the transpose of the i^{th} row of $\mathbf{W}$. The proofs for the remaining subproblems for other rows of $\mathbf{W}$ and all columns of $\mathbf{C}$ follow analogously.

Lemma 5 (Lipschitz constants) *For a given i and j, let*

$$f_w(\bar{\mathbf{w}}_i) = -\sum_j \log\, p(Y_{i,j}|\bar{\mathbf{w}}_i, \mathbf{c}_j) + \frac{\mu}{2}\|\bar{\mathbf{w}}_i\|_2^2 = -\sum_j \log\, \Phi((2\,Y_{i,j}-1)\bar{\mathbf{w}}_i^T\mathbf{c}_j) + \frac{\mu}{2}\|\bar{\mathbf{w}}_i\|_2^2\,,$$
$$f_c(\mathbf{c}_j) = -\sum_i \log\, p(Y_{i,j}|\bar{\mathbf{w}}_i, \mathbf{c}_j) = -\sum_i \log\, \Phi((2\,Y_{i,j}-1)\bar{\mathbf{w}}_i^T\mathbf{c}_j),$$

where $Y_{i,j}$, $\bar{\mathbf{w}}_i$, and $\mathbf{c}_j$ are defined as in Section 2.1. Here, $\Phi(x)$ designates the inverse link function, which can either be (3) or (4). Then, for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^K$, we have

$$\|\nabla f_w(\mathbf{x}) - \nabla f_w(\mathbf{y})\|_2 \leq (L\sigma_{max}^2(\mathbf{C}) + \mu)\|\mathbf{x}-\mathbf{y}\|_2,$$
$$\|\nabla f_c(\mathbf{x}) - \nabla f_c(\mathbf{y})\|_2 \leq L\sigma_{max}^2(\mathbf{W})\|\mathbf{x}-\mathbf{y}\|_2,$$

where $L = L_{pro} = 1$ and $L = L_{log} = 1/4$ are the scalar Lipschitz constants for the probit and logit cases from Lemma 4, respectively.

Proof For the sake of brevity, we only show the proof for $f_w(\mathbf{x})$ in the probit case. The logit cases and the cases for $f_c(\mathbf{x})$ follow analogously. In what follows, the PDF $\mathcal{N}(x)$ and CDF $\Phi(x)$ of the standard normal density (the inverse probit link function) defined in (3) are assumed to operate element-wise on the vector $\mathbf{x} \in \mathbb{R}^K$.

In order to simplify the derivation of the proof, we define the following effective matrix associated to $\bar{\mathbf{w}}_i$ as

$$\mathbf{C}_{\text{eff},i} = [\,(2\,Y_{i,1}-1)\mathbf{c}_1, \ldots, (2\,Y_{i,N}-1)\mathbf{c}_N\,],$$

which is equivalent to a right-multiplication $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_N]$ with a diagonal matrix containing the binary-valued response variables $(2\,Y_{i,j}-1) \in \{-1,+1\}\ \forall j$. We can now establish an upper bound of the ℓ_2-norm of the difference between the gradients at two arbitrary points $\mathbf{x}$ and $\mathbf{y}$ as follows:

$$\begin{aligned}
\|\nabla f_w(\mathbf{x}) - \nabla f_w(\mathbf{y})\|_2 &= \left\| \mathbf{C}_{\text{eff},i}\frac{\mathcal{N}(\mathbf{C}_{\text{eff},i}^T\mathbf{x})}{\Phi(\mathbf{C}_{\text{eff},i}^T\mathbf{x})} - \mathbf{C}_{\text{eff},i}\frac{\mathcal{N}(\mathbf{C}_{\text{eff},i}^T\mathbf{y})}{\Phi(\mathbf{C}_{\text{eff},i}^T\mathbf{y})} + \mu\mathbf{x} - \mu\mathbf{y} \right\|_2 \\
&\leq \sigma_{\max}(\mathbf{C}_{\text{eff},i}) \left\| \frac{\mathcal{N}(\mathbf{C}_{\text{eff},i}^T\mathbf{x})}{\Phi(\mathbf{C}_{\text{eff},i}^T\mathbf{x})} - \frac{\mathcal{N}(\mathbf{C}_{\text{eff},i}^T\mathbf{y})}{\Phi(\mathbf{C}_{\text{eff},i}^T\mathbf{y})} \right\|_2 + \mu\|\mathbf{x}-\mathbf{y}\|_2 \qquad (18)\\
&\leq L\sigma_{\max}(\mathbf{C}_{\text{eff},i})\|\mathbf{C}_{\text{eff},i}^T\mathbf{x} - \mathbf{C}_{\text{eff},i}^T\mathbf{y}\|_2 + \mu\|\mathbf{x}-\mathbf{y}\|_2 \qquad (19)\\
&\leq L\sigma_{\max}^2(\mathbf{C}_{\text{eff},i})\|\mathbf{x}-\mathbf{y}\|_2 + \mu\|\mathbf{x}-\mathbf{y}\|_2 \qquad (20)\\
&= (L\sigma_{\max}^2(\mathbf{C}) + \mu)\|\mathbf{x}-\mathbf{y}\|_2. \qquad (21)
\end{aligned}$$

Here, (18) uses the triangle inequality and the Rayleigh-Ritz theorem of Horn and Johnson (1991), where $\sigma_{\max}(\mathbf{C}_{\text{eff},i})$ denotes the principal singular value of $\mathbf{C}_{\text{eff},i}$. The bound (19) follows from Lemma 4, and (20) is, once more, a consequence of the Rayleigh-Ritz theorem. The final equality (21) follows from the fact that flipping the signs of the columns of a matrix (as we did to arrive at $\mathbf{C}_{\text{eff},i}$) does not affect its singular values, which concludes the proof. Note that the proof for $f_c(\cdot)$ follows by omitting μ and substitute $\mathbf{C}$ by $\mathbf{W}$ in (21). ■

Note that in all of the above proofs we only considered the case where the observation matrix $\mathbf{Y}$ is fully populated. Our proofs easily adapt to the case of missing entries in $\mathbf{Y}$, by replacing the matrix $\mathbf{C}$ to $\mathbf{C}_{\mathcal{I}}$, where $\mathbf{C}_{\mathcal{I}}$ corresponds to the matrix containing the columns of $\mathbf{C}$ corresponding to the observed entries indexed by the set $\mathcal{I} = \{j : (i,j) \in \Omega_{\text{obs}}\}$. We omit the details for the sake of brevity.

Appendix B. Proof of Theorem 2

Minimizing $F(\mathbf{x})$ as defined in Theorem 2 using SPARFA-M corresponds to a multi-block coordinate descent problem, where the subproblems $(\text{RR})_1^+$ and $(\text{RR})_2$ correspond to (Xu and Yin, 2012, Problem. 1.2b and 1.2a), respectively. Hence, we can use the results of (Xu and Yin, 2012, Lemma 2.6, Corrollary 2.7, and Theorem 2.8) to establish global convergence of SPARFA-M. To this end, we must verify that the problem (P) satisfies all of the assumptions in (Xu and Yin, 2012, Assumption 1, Assumption 2, and Lemma 2.6).

B.1 Prerequisites

We first show that the smooth part of the cost function in (P), i.e., the negative log-likelihood plus both ℓ_2-norm regularization terms, is Lipschitz continuous on any bounded set in $\mathbb{R}^{(N+Q)K}$. Then, we show that the probit log-likelihood function is real analytic. Note that the logit log-likelihood function is real analytic as shown in (Xu and Yin, 2012, Section 2.3). Finally, we combine both results to prove Theorem 2, which establishes the global convergence of SPARFA-M.

Lemma 6 (Lipschitz continuity) *Define* $\mathbf{x} = [\bar{\mathbf{w}}_1^T, \ldots, \bar{\mathbf{w}}_Q^T,\ \mathbf{c}_1^T, \ldots, \mathbf{c}_N^T]^T$, *and let*

$$f(\mathbf{x}) = -\sum_{(i,j)\in\Omega_{obs}} \log p(Y_{i,j}|\bar{\mathbf{w}}_i, \mathbf{c}_j) + \frac{\mu}{2}\sum_i \|\bar{\mathbf{w}}_i\|_2^2 + \frac{\gamma}{2}\sum_j \|\mathbf{c}_j\|_2^2.$$

Then, $f(\mathbf{x})$ *is Lipschitz continuous on any bounded set* $\mathcal{D} = \{\mathbf{x} : \|\mathbf{x}\|_2 \leq D\}$.

Proof Let $\mathbf{y}, \mathbf{z} \in \mathcal{D}$, recall the notation of Lemma 5, and let $\bar{\mathbf{w}}_i^y$, $\bar{\mathbf{w}}_i^z$, $\mathbf{c}_j^y$, and $\mathbf{c}_j^z$ denote the blocks of variables $\bar{\mathbf{w}}_i$ and $\mathbf{c}_j$ in $\mathbf{y}$ and $\mathbf{z}$, respectively. We now have

$$\begin{aligned}
\|\nabla f(\mathbf{y}) - \nabla f(\mathbf{z})\|_2 &= \Big(\sum_{i,j}\big((\nabla f_w(\bar{\mathbf{w}}_i^y) - \nabla f_w(\bar{\mathbf{w}}_i^z))^2 + (\nabla f_c(\mathbf{c}_j^y) - \nabla f_c(\mathbf{c}_j^z))^2 + \gamma^2\|\mathbf{c}_j^y - \mathbf{c}_j^z\|_2^2\big)\Big)^{\frac{1}{2}} \\
&\leq \Big(\sum_{i,j}\big(\big(L\sigma_{\max}^2(\mathbf{C}) + \mu\big)^2\|\bar{\mathbf{w}}_i^y - \bar{\mathbf{w}}_i^z\|_2^2 + \big(L^2\sigma_{\max}^4(\mathbf{W}) + \gamma^2\big)\|\mathbf{c}_j^y - \mathbf{c}_j^z\|_2^2\big)\Big)^{\frac{1}{2}} \qquad (22) \\
&\leq \big(L\left(\|\mathbf{W}\|_F^2 + \|\mathbf{C}\|_F^2\right) + \max\{\mu,\gamma\}\big)\|\mathbf{y} - \mathbf{z}\|_2 \qquad (23) \\
&\leq \big(LD^2 + \max\{\mu,\gamma\}\big)\|\mathbf{y} - \mathbf{z}\|_2,
\end{aligned}$$

where (22) follows from Lemma 5, and (23) follows from the fact that the maximum singular value of a matrix is no greater than its Frobenius norm (Horn and Johnson 1991), which is bounded by D for $\mathbf{y}, \mathbf{z} \in \mathcal{D}$. We furthermore have $L = 1$ for the probit case and $L = 1/4$

for the logit case, shown in Lemma 4. Thus, $f(\mathbf{x})$ is Lipschitz continuous on any bounded set. ■

Lemma 7 (Real analyticity) *Define* $\mathbf{x} = [\bar{\mathbf{w}}_1^T, \ldots, \bar{\mathbf{w}}_Q^T, \mathbf{c}_1^T, \ldots, \mathbf{c}_N^T]^T$, *and let*

$$g(\mathbf{x}) = -\sum_{(i,j)\in\Omega_{obs}} \log p(Y_{i,j}|\bar{\mathbf{w}}_i, \mathbf{c}_j) = -\sum_{(i,j)\in\Omega_{obs}} \log \Phi_{pro}((2Y_{i,j}-1)\bar{\mathbf{w}}_i^T\mathbf{c}_j),$$

where $\Phi_{pro}(\cdot)$ *is the inverse probit link function defined in (3). Then,* $g(\mathbf{x})$ *is real analytic.*

Proof Recall the important property established by Krantz and Parks (2002) that compositions of real analytic functions are real analytic. Therefore, the standard normal density $\mathcal{N}(x)$ is real analytic, since the exponential function and x^2 are both real analytical functions. Consequently, let $\mathcal{N}^{(k)}(x)$ denote the k^{th} derivative of $\mathcal{N}(x)$, then $\left(\frac{\mathcal{N}^{(k)}(x)}{k!}\right)^{\frac{1}{k}}$ is bounded for all k, according to the definition of real analytic functions.

Now we show that $\Phi_{\text{pro}}(x) = \int_{-\infty}^{x} \mathcal{N}(t)\mathrm{d}t$ is also real analytic. Its k^{th} derivative is given by $\Phi_{\text{pro}}^{(k)}(x) = \mathcal{N}^{(k-1)}(x)$, and therefore $\left(\frac{\Phi_{\text{pro}}^{(k)}(x)}{k!}\right)^{\frac{1}{k}}$ is obviously bounded for all k, since $\left(\frac{\mathcal{N}^{(k)}(x)}{k!}\right)^{\frac{1}{k}}$ is bounded for all k as we have just shown. Thus, $\Phi_{\text{pro}}(x)$ is real analytic.

Given that $\Phi_{\text{pro}}(x)$ is real-analytic, it follows that the negative log-probit-likelihood $-\log\Phi_{\text{pro}}(x)$ is real analytic, since both the logarithm function and the inverse probit link function are real analytic. Finally, extending the proof from scalar functions to vector functions preserves analyticity according to (Xu and Yin, 2012, Section 2.2). ■

B.2 Proof of Theorem 2

We are finally armed to prove Theorem 2. We begin by showing that our problem (P) meets (Xu and Yin, 2012, Assumptions 1 and 2). Then, we show that (P) meets all the additional assumptions needed for the convergence results in (Xu and Yin, 2012, Lemma 2.6), through which we can establish convergence of the sequence $\{\mathbf{x}^t\}$ from certain starting points to some finite limit point. Finally, we use (Xu and Yin, 2012, Theorem 2.8) to show global convergence of SPARFA-M from any starting point.

B.2.1 Assumption 1

We start by showing that (P) meets (Xu and Yin, 2012, Assumption 1). Since every term in our objective function in (P) is non-negative, we have $F(\mathbf{x}) > -\infty$. It is easy to verify that (P) is also block multi-convex in the variable $\mathbf{x}$, with the rows of $\mathbf{W}$ and columns of $\mathbf{C}$ forming the blocks. Consequently, the problem (P) has at least one critical point, since $F(\mathbf{x})$ is lower bounded by 0. Therefore, Assumption 1 is met.

B.2.2 Assumption 2

Problem (P) also meets (Xu and Yin, 2012, Assumption 2) regarding the strong convexity of the individual subproblems. Due to the presence of the quadratic terms $\frac{\mu}{2}\|\bar{\mathbf{w}}_i\|_2^2$ and $\frac{\gamma}{2}\|\mathbf{c}_j\|_2^2$, the smooth part of the objective functions of the individual subproblems (RR_1^+) and (RR_2) are strongly convex with parameters μ and γ, respectively. Consequently, Assumption 2 is satisfied.

B.2.3 Assumptions in (Xu and Yin, 2012, Lem. 2.6)

Problem (P) also meets the assumptions in (Xu and Yin, 2012, Lem. 2.6) regarding the Lipschitz continuity of the subproblems and the Kurdyka-Łojasiewicz inequality. Lemma 6 shows that $f(\mathbf{x}) = -\sum_{(i,j)\in\Omega_{\text{obs}}} \log p(Y_{i,j}|\bar{\mathbf{w}}_i, \mathbf{c}_j) + \frac{\mu}{2}\sum_i \|\bar{\mathbf{w}}_i\|_2^2 + \frac{\gamma}{2}\sum_j \|\mathbf{c}_j\|_2^2$, satisfies the Lipschitz continuous requirement in (Xu and Yin, 2012, Lemma 2.6). As shown in Lemma 7 for the probit case and as shown in (Xu and Yin, 2012, Section 2.2) for the logit case, the negative log-likelihood term in (P) is real analytic, therefore also sub-analytic. All the regularizer functions in $F(\mathbf{x})$ defined in Theorem 2 are semi-algebraic and therefore sub-analytic, a consequence of (Bolte et al., 2006, Section 2.1) and (Xu and Yin, 2012, Section 2.2). Using (Fischer, 2008, Theorems 1.1 and 1.2), the objective function $F(\mathbf{x})$ is also sub-analytic, since all of its parts are sub-analytic and bounded below (non-negative), therefore satisfying the Kurdyka-Łojasiewicz inequality at any point $\mathbf{x}$, as shown in (Bolte et al., 2006, Theorem 3.1). Finally, the SPARFA-M algorithm uses $\omega_i^{k-1} \equiv 0$ and $\ell = \min\{\mu, \gamma\}$ where ω_i^{k-1} and ℓ as defined in (Xu and Yin, 2012, Lemma 2.6).

Up to this point, we have shown that (P) satisfies all assumptions and requirements in (Xu and Yin, 2012, Lemma 2.6). Now, SPARFA-M follows (Xu and Yin, 2012, Lemma 2.6) in the sense that, if $\mathbf{x}^0$ is sufficiently close to some critical point $\hat{\mathbf{x}}$ of (P), (more specifically, $\mathbf{x}^0 \in \mathcal{B}$ for some $\mathcal{B} \subset \mathcal{U}$ where $\mathcal{U}$ is a neighborhood of $\hat{\mathbf{x}}$ in which Kurdyka-Łojasiewicz inequality holds), then $\{\mathbf{x}^t\}$ converges to a point in $\mathcal{B}$. This establishes the convergence of SPARFA-M to a local minimum point from certain starting points.

B.2.4 Global Convergence

Finally, we can use (Xu and Yin, 2012, Lemma 2.6) to establish global convergence of SPARFA-M. It is obvious that the objective function (P) is bounded on any bounded set. Hence, the sequence $\{\mathbf{x}^k\}$ will always have a finite limit point and meet the assumptions in (Xu and Yin, 2012, Theorem 2.8). The final statement of Theorem 2 now directly follows from (Xu and Yin, 2012, Theorem 2.8). Moreover, if the starting point is in close proximity to a global minimum, then SPARFA-M is guaranteed to converge to a global minimum. This is a consequence of (Xu and Yin, 2012, Corollary 2.7).

Appendix C. Proof of Theorem 3

Proof To prove Theorem 3, we first define some notation. Let $\mathcal{N}(x|m,s) = \frac{1}{\sqrt{2\pi s}} e^{-(x-m)^2/2s}$ define the normal PDF with mean m and variance s. Furthermore, let $Exp(m|\lambda) = \lambda e^{-\lambda m}$, $m \geq 0$ define the PDF of the exponential distribution with rate parameter λ.

We are ultimately concerned with identifying whether the factor $W_{i,k}$ is active given our current beliefs about all other parameters in the model. Given the probit model, this

is equivalent to determining whether or not an exponential random variable is present in Gaussian noise. Let $x|m,s \sim \mathcal{N}(0|m,s)$ with $m \sim r\,Exp(m|\lambda) + (1-r)\,\delta_0$ and δ_0 the Dirac delta function located at 0. The posterior distribution $p(m=0|x)$ can be derived via Bayes' rule as follows:

$$\begin{aligned} p(m=0|x) &= \frac{\mathcal{N}(x|m=0,s)(1-r)}{\mathcal{N}(x|m=0,s)(1-r) + r\int \mathcal{N}(x|m,s)\,Exp(m|\lambda)\mathrm{d}m}, \\ &= \frac{\frac{\mathcal{N}(x|0,s)}{\int \mathcal{N}(x|m,s)\,Exp(m|\lambda)dm}(1-r)}{\frac{\mathcal{N}(x|0,s)}{\int \mathcal{N}(x|m,s)\,Exp(m|\lambda)\mathrm{d}m}(1-r) + r}, \\ &= \frac{\frac{Exp(0|\lambda)\mathcal{N}(x|0,s)}{\int \mathcal{N}(x|m,s)\,Exp(m|\lambda)\mathrm{d}m}(1-r)}{\frac{Exp(0|\lambda)\mathcal{N}(x|0,s)}{\int \mathcal{N}(x|m,s)\,Exp(m|\lambda)\mathrm{d}m}(1-r) + r\,Exp(0|\lambda)}. \end{aligned} \tag{24}$$

Here, it is important to recognize that $\frac{Exp(m|\lambda)\mathcal{N}(x|m,s)}{\int \mathcal{N}(x|m,s)\,Exp(m|\lambda)dm}$ denotes the posterior under the continuous portion of the prior (i.e., $m \neq 0$). Since the exponential prior we have chosen is not conjugate to the normal likelihood, we must compute this distribution in closed form. To this end, let $\mathcal{N}^r(x|m,s,\lambda) \propto \mathcal{N}(x|m,s)Exp(m|\lambda) = C_0 e^{-(x-m)^2/2s-\lambda m}$ denote a rectified normal distribution with normalization constant C_0. Completing the square and carrying out the integration, we find $C_0 = \frac{e^{\lambda m - \lambda^2 s/2}}{\sqrt{2\pi s}\Phi\left(\frac{m-\lambda s}{\sqrt{s}}\right)}$, which leads to

$$\mathcal{N}^r(x|m,s,\lambda) = \frac{e^{\lambda m - \lambda^2 s/2}}{\sqrt{2\pi s}\Phi\left(\frac{m-\lambda s}{\sqrt{s}}\right)} e^{-(x-m)^2/2s-\lambda m}.$$

We can now rewrite (24) as

$$p(m=0|x) = \frac{\frac{\mathcal{N}^r(0|\hat{m},\hat{s},\lambda)}{Exp(0|\lambda)}(1-r)}{\frac{\mathcal{N}^r(0|\hat{m},\hat{s},\lambda)}{Exp(0|\lambda)}(1-r) + r}$$

or, alternatively, as

$$\hat{r} = p(m \neq 0|x) = 1 - p(m=0|x) = \frac{\frac{Exp(0|\lambda)}{\mathcal{N}^r(0|\hat{m},\hat{s},\lambda)}}{\frac{Exp(0|\lambda)}{\mathcal{N}^r(0|\hat{m},\hat{s},\lambda)} + \frac{1-r}{r}}. \tag{25}$$

All that remains now is to determine $\hat{m}$ and $\hat{s}$ in (25) for our full factor analysis scenario. Recall that our probabilistic model corresponds to $\mathbf{Z} = \mathbf{WC} + \mathbf{M}$. Further recall our definition of the observation set $\Omega_{\text{obs}} = \{(i,j) : Y_{i,j} \text{ is observed}\}$. We can now calculate the

posterior on each coefficient $W_{i,k} \neq 0$ as follows:

$$\begin{aligned} p(W_{i,k}|\mathbf{Z},\mathbf{C},\boldsymbol{\mu}) &\propto p(W_{i,k})\, p(\mathbf{Z}|\mathbf{W}_{-(i,k)},\mathbf{C},\boldsymbol{\mu}) \\ &\propto e^{-\lambda W_{i,k}}\, e^{-\frac{1}{2\sigma^2}\sum_{\{j:(i,j)\in\Omega_{\text{obs}}\}}\left((Z_{i,j}-\mu_i)-\sum_{k=1}^{K} W_{i,k}C_{k,j}\right)^2} \\ &= e^{-\lambda W_{i,k}}\, e^{-\frac{1}{2\sigma^2}\sum_{\{j:(i,j)\in\Omega_{\text{obs}}\}}\left((Z_{i,j}-\mu_i)-\sum_{k'\neq k} W_{i,k'}C_{k',j}-W_{i,k}C_{k,j}\right)^2} \\ &\propto e^{-\lambda W_{i,k}}\, e^{-\frac{1}{2\sigma^2}\sum_{\{j:(i,j)\in\Omega_{\text{obs}}\}}\left(W_{i,k}^2C_{k,j}^2-2((Z_{i,j}-\mu_i)-\sum_{k'\neq k} W_{i,k'}C_{k',j})W_{i,k}C_{k,j}\right)} \\ &\propto e^{-\lambda W_{i,k}} e^{-\frac{\sum_{\{j:(i,j)\in\Omega_{\text{obs}}\}}C_{k,j}^2}{2\sigma^2}\left(W_{i,k}-\frac{\sum_{\{j:(i,j)\in\Omega_{\text{obs}}\}}\left((Z_{i,j}-\mu_i)-\sum_{k'\neq k}W_{i,k'}C_{k',j}\right)C_{k,j}}{\sum_{\{j:(i,j)\in\Omega_{\text{obs}}\}}C_{k,j}^2}\right)^2}, \end{aligned} \tag{26}$$

where the last step is obtained by completing the square in $W_{i,k}$.

The final result in (26) implies that $W_{i,k} \sim \mathcal{N}^r(\hat{m}, \hat{s}, \lambda)$, where

$$\hat{m} = \frac{\sum_{\{j:(i,j)\in\Omega_{\text{obs}}\}}\left((Z_{i,j}-\mu_i)-\sum_{k'\neq k} W_{i,k'}C_{k',j}\right)C_{k,j}}{\sum_{\{j:(i,j)\in\Omega_{\text{obs}}\}} C_{k,j}^2}$$

and $\hat{s} = \frac{\sigma^2}{\sum_{\{j:(i,j)\in\Omega_{\text{obs}}\}} C_{k,j}^2}$. Combining the results of (25) and (26), recognizing that $\sigma^2 = 1$ in the standard probit model, and adopting the notation $\widehat{R}_{i,k}$, $\widehat{M}_{i,k}$ and $\widehat{S}_{i,k}$ for the values of $\hat{r}$, $\hat{m}$ and $\hat{s}$ corresponding to each $\widehat{W}_{i,k}$, furnishes the final sampling result. ■

References

M. Aharon, M. Elad, and A. M. Bruckstein. K-SVD and its non-negative variant for dictionary design. In *Proc. SPIE Conf. on Wavelets*, volume 5914, pages 327–339, July 2005.

M. Aharon, M. Elad, and A. M. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54 (11):4311–4322, Dec. 2006.

Amazon Mechanical Turk, Sep. 2012. URL `https://www.mturk.com/mturk/welcome`.

Y. Bachrach, T. P. Minka, J. Guiver, and T. Graepel. How to grade a test without knowing the answers – a Bayesian graphical model for adaptive crowdsourcing and aptitude testing. In *Proc. 29th Intl. Conf. on Machine Learning*, pages 1183–1190, June 2012.

F. B. Baker and S. H. Kim. *Item Response Theory: Parameter Estimation Techniques*. Marcel Dekker Inc., 2nd edition, 2004.

R. Baker and K. Yacef. The state of educational data mining in 2009: A review and future visions. *Journal of Educational Data Mining*, 1(1):3–17, Oct. 2009.

T. Barnes. The Q-matrix method: Mining student response data for knowledge. In *Proc. AAAI Workshop Educational Data Mining*, July 2005.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Science*, 2(1):183–202, Mar. 2009.

B. Beheshti, M. Desmarais, and R. Naceur. Methods to find the number of latent skills. In *Proc. 5th Intl. Conf. on Educational Data Mining*, pages 81–86, June 2012.

Y. Bergner, S. Droschler, G. Kortemeyer, S. Rayyan, D. Seaton, and D. Pritchard. Model-based collaborative filtering analysis of student response data: Machine-learning item response theory. In *Proc. 5th Intl. Conf. on Educational Data Mining*, pages 95–102, June 2012.

J. Bolte, A. Daniilidis, and A. Lewis. The Łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems. *SIAM Journal on Optimization*, 17(4):1205–1223, Dec. 2006.

P. T. Boufounos and R. G. Baraniuk. 1-bit compressive sensing. In *Proc. Conf. on Information Science and Systems (CISS)*, Mar. 2008.

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

A. M. Bruckstein, M. Elad, and M. Zibulevsky. On the uniqueness of nonnegative sparse solutions to underdetermined systems of equations. *IEEE Transactions on Information Theory*, 54(11):4813–4820, Nov. 2008.

P. Brusilovsky and C. Peylo. Adaptive and intelligent web-based educational systems. *Intl. Journal of Artificial Intelligence in Education*, 13(2-4):159–172, Apr. 2003.

C. J. Butz, S. Hua, and R. B. Maguire. A web-based Bayesian intelligent tutoring system for computer programming. *Web Intelligence and Agent Systems*, 4(1):77–97, Nov. 2006.

H. Chang and Z. Ying. Nonlinear sequential designs for logistic item response theory models with applications to computerized adaptive tests. *Annals of Statistics*, 37(3):1466–1488, June 2009.

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, Mar. 1998.

J. T. Chu. On bounds for the normal integral. *IEEE Transactions on Signal Processing*, 42 (1/2):263–265, June 1955.

M. Desmarais. Conditions for effectively deriving a Q-matrix from data with non-negative matrix factorization. In *Proc. 4th Intl. Conf. on Educational Data Mining*, pages 41–50, July 2011.

J. A. Dijksman and S. Khan. Khan Academy: the world's free virtual school. In *APS Meeting Abstracts*, page 14006, Mar. 2011.

ELEC 301, Rice University. Introduction to signals and systems, May 2011. URL `http://dsp.rice.edu/courses/elec301`.

A. Fischer. On sums of subanalytic functions. *Preprint*, 2008.

E. Fokoue. Stochastic determination of the intrinsic structure in Bayesian factor analysis. Technical report, Statistical and Applied Mathematical Sciences Institute, June 2004.

K. Fronczyk, A. E. Waters, M. Guindani, R. G. Baraniuk, and M. Vannucci. A Bayesian infinite factor model for learning and content analytics. *Computational Statistics and Data Analysis*, June 2013, submitted.

J. P. González-Brenes and J. Mostow. Dynamic cognitive tracing: Towards unified discovery of student and cognitive models. In *Proc. 5th Intl. Conf. on Educational Data Mining*, pages 49–56, June 2012.

I. Goodfellow, A. Courville, and Y. Bengio. Large-scale feature learning with spike-and-slab sparse coding. In *Proc. 29th Intl. Conf. on Machine Learning*, pages 1439–1446, July 2012.

A. Guisan, T. C. Edwards Jr, and T. Hastie. Generalized linear and generalized additive models in studies of species distributions: setting the scene. *Ecological Modelling*, 157 (2–3):89–100, Nov. 2002.

P. R. Hahn, C. M. Carvalho, and J. G. Scott. A sparse factor-analytic probit model for congressional voting patterns. *Journal of the Royal Statistical Society*, 61(4):619–635, Aug. 2012.

H. H. Harman. *Modern Factor Analysis*. The University of Chicago Press, 1976.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2010.

J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, Jan. 2004.

R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

D. Hu. How Khan academy is using machine learning to assess student mastery. *online: http://david-hu.com/*, Nov. 2011.

H. Ishwaran and J. S. Rao. Spike and slab variable selection: frequentist and Bayesian strategies. *Annals of Statistics*, 33(2):730–773, Apr. 2005.

L. Jacques, J. N. Laska, P. T. Boufounos, and R. G. Baraniuk. Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors. *IEEE Transaction on Info Theory*, 59(4), Apr. 2013.

Knewton. Knewton adaptive learning: Building the world's most powerful recommendation engine for education. *online: http://www.knewton.com/adaptive-learning-white-paper/*, June 2012.

K. R. Koedinger, J. R. Anderson, W. H. Hadley, and M. A. Mark. Intelligent tutoring goes to school in the big city. *Intl. Journal of Artificial Intelligence in Education*, 8(1):30–43, 1997.

K. Koh, S. Kim, and S. Boyd. An interior-point method for large-scale ℓ_1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

Y. Koren and J. Sill. OrdRec: an ordinal model for predicting personalized item rating distributions. In *Proc. of the 5th ACM Conf. on Recommender Systems*, pages 117–124, Oct. 2011.

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

S. G. Krantz and H. R. Parks. *A Primer of Real Analytic Functions*. Birkhauser, 2002.

G. A. Krudysz and J. H. McClellan. Collaborative system for signal processing education. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, pages 2904–2907, May 2011.

G. A. Krudysz, J. S. Li, and J. H. McClellan. Web-based Bayesian tutoring system. In *12th Digital Signal Processing Workshop - 4th Signal Processing Education Workshop*, pages 129–134, Sep. 2006.

A. S. Lan, C. Studer, A. E. Waters, and R. G. Baraniuk. Tag-aware ordinal sparse factor analysis for learning and content analytics. In *Proc. 6th Intl. Conf. on Educational Data Mining*, pages 90–97, July 2013a.

A. S. Lan, C. Studer, A. E. Waters, and R. G. Baraniuk. Joint topic modeling and factor analysis of textual information and graded response data. In *Proc. 6th Intl. Conf. on Educational Data Mining*, pages 324–325, July 2013b.

S. Lee, H. Lee, P. Abbeel, and A. Y. Ng. Efficient ℓ_1 regularized logistic regression. In *Proc. National Conf. on Artificial Intelligence*, volume 21, pages 401–408, 2006.

S. Lee, J. Z. Huang, and J. Hu. Sparse logistic principal components analysis for binary data. *Annals of Applied Statistics*, 4(3):1579–1601, Sept. 2010.

N. Li, W. W. Cohen, and K. R. Koedinger. A machine learning approach for automatic student model discovery. In *Proc. 4th Intl. Conf. on Educational Data Mining*, pages 31–40, July 2011.

J. M. Linacre. Understanding Rasch measurement: Estimation methods for Rasch measures. *Journal of Outcome Measurement*, 3(4):382–405, 1999.

G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, Jan. 2003.

W. J. V. D. Linden and editors Glas, C. A. W. *Computerized Adaptive Testing: Theory and Practice*. Kluwer Academic Publishers, 2000.

F. M. Lord. *Applications of Item Response Theory to Practical Testing Problems*. Erlbaum Associates, 1980.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

R. P. McDonald. A basis for multidimensional item response theory. *Applied Psychological Measurement*, 247(2):99–114, June 2000.

J. Meng, J. Zhang, Y. Qi, Y. Chen, and Y. Huang. Uncovering transcriptional regulatory networks by sparse Bayesian factor model. *EURASIP Journal on Advances in Signal Processing*, 2010(3):1–18, Mar. 2010.

T. P. Minka. A comparison of numerical optimizers for logistic regression. Technical report, 2003. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.85.7017&rep=rep1&type=pdf.

S. Mohamed, K. Heller, and Z. Ghahramani. Bayesian and ℓ_1 approaches for sparse unsupervised learning. In *Proc. 29th Intl. Conf. on Machine Learning*, pages 751–758, July 2012.

R. C. Murray, K. VanLehn, and J. Mostow. Looking ahead to select tutorial actions: A decision-theoretic approach. *Intl. Journal of Artificial Intelligence in Education*, 14(3–4): 235–278, Dec. 2004.

Y. Nesterov. Gradient methods for minimizing composite objective function. Technical report, Université catholique de Louvain, Sep. 2007.

M. R. Norvick. The axioms and principal results of classical test theory. *Journal of Mathematical Psychology*, 3(1):1–18, Feb. 1966.

F. W. J. Olver, editor. *NIST Handbook of Mathematical Functions*. Cambridge University Press, 2010.

Z. A. Pardos and N. T. Heffernan. Modeling individualization in a Bayesian networks implementation of knowledge tracing. In *User Modeling, Adaptation, and Personalization*, volume 6075, pages 255–266. Springer, June 2010.

M. Y. Park and T. Hastie. Penalized logistic regression for detecting gene interactions. *Biostatistics*, 9(1):30–50, Jan. 2008.

Y. Plan and R. Vershynin. Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach. 59(1), Feb. 2012, submitted.

I. Pournara and L. Wernisch. Factor analysis for gene regulatory networks and transcription factor activity profiles. *BMC Bioinformatics*, 8(1):61, Feb. 2007.

J. Psotka, L. D. Massey, and editors Mutter, S. A. *Intelligent Tutoring Systems: Lessons Learned*. Lawrence Erlbaum Associates, 1988.

A. N. Rafferty, E. Brunskill, T. L. Griffiths, and P. Shafto. Faster teaching by POMDP planning. In *Proc. 15th Intl. Conf. on Artificial Intelligence in Education*, pages 280–287, June 2011.

G. Rasch. *Probabilistic Models for Some Intelligence and Attainment Tests.* MESA Press, 1993.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Process for Machine Learning.* MIT Press, 2006.

M. D. Reckase. *Multidimensional Item Response Theory.* Springer Publishing Company, Incorporated, 1st edition, 2009.

C. Romero and S. Ventura. Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications*, 33(1):135–146, July 2007.

M. N. Schmidt, O. Winther, and L. K. Hansen. Bayesian non-negative matrix factorization. In *Independent Component Analysis and Signal Separation*, volume 5441, pages 540–547, Mar. 2009.

J. C. Stamper, T. Barnes, and M. Croy. Extracting student models for intelligent tutoring systems. In *Proc. National Conf. on Artificial Intelligence*, volume 22, pages 113–147, July 2007.

STEMscopes. STEMscopes science education, Sep. 2012. URL `http://stemscopes.com/`.

C. Studer and R. G. Baraniuk. Dictionary learning from sparsely corrupted or compressed signals. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, pages 3341–3344, Mar. 2012.

N. Thai-Nghe, L. Drumond, T. Horvath, and L. Schmidt-Thieme. Multi-relational factorization models for predicting student performance. *KDD Workshop on Knowledge Discovery in Educational Data (KDDinED)*, Aug. 2011a.

N. Thai-Nghe, T. Horvath, and L. Schmidt-Thieme. Factorization models for forecasting student performance. In *Proc. 4th Intl. Conf. on Educational Data Mining*, pages 11–20, July 2011b.

N. A. Thompson. Item selection in computerized classification testing. *Educational and Psychological Measurement*, 69(5):778–793, Oct. 2009.

M. E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

K. VanLehn, C. Lynch, K. Schulze, J. A. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein, and M. Wintersgill. The Andes physics tutoring system: Lessons learned. *Intl. Journal of Artificial Intelligence in Education*, 15(3):147–204, Sep. 2005.

D. Vats, C. Studer, A. S. Lan, L. Carin, and R. G. Baraniuk. Test size reduction for concept estimation. In *Proc. 6th Intl. Conf. on Educational Data Mining*, pages 292–295, July 2013.

M. West. Bayesian factor regression models in the "large p, small n" paradigm. *Bayesian Statistics*, 7:723–732, Sep. 2003.

B. P. Woolf. *Building Intelligent Interactive Tutors: Student-centered Strategies for Revolutionizing E-learning*. Morgan Kaufman Publishers, 2008.

Y. Xu and W. Yin. A block coordinate descent method for multi-convex optimization with applications to nonnegative tensor factorization and completion. Technical report, Rice University CAAM, Sep. 2012.

L. Yao. *BMIRT: Bayesian Multivariate Item Response Theory*. CTb/McGraw-Hill, 2003.

Journal of Machine Learning Research 15 (2014) 2009-2053 Submitted 7/13; Revised 4/14; Published 6/14

Causal Discovery with Continuous Additive Noise Models

Jonas Peters* PETERS@STAT.MATH.ETHZ.CH
Seminar for Statistics, ETH Zürich
Rämistrasse 101, 8092 Zürich
Switzerland

Joris M. Mooij* J.M.MOOIJ@UVA.NL
Institute for Informatics, University of Amsterdam
Postbox 94323, 1090 GH Amsterdam
The Netherlands

Institute for Computing and Information Sciences, Radboud University Nijmegen
Postbox 9010, 6500 GL Nijmegen
The Netherlands

Dominik Janzing JANZING@TUEBINGEN.MPG.DE
Bernhard Schölkopf BS@TUEBINGEN.MPG.DE
Max Planck Institute for Intelligent Systems
Spemannstraße 38, 72076 Tübingen
Germany

Editor: Aapo Hyvärinen

Abstract

We consider the problem of learning causal directed acyclic graphs from an observational joint distribution. One can use these graphs to predict the outcome of interventional experiments, from which data are often not available. We show that if the observational distribution follows a structural equation model with an additive noise structure, the directed acyclic graph becomes identifiable from the distribution under mild conditions. This constitutes an interesting alternative to traditional methods that assume faithfulness and identify only the Markov equivalence class of the graph, thus leaving some edges undirected. We provide practical algorithms for finitely many samples, RESIT (regression with subsequent independence test) and two methods based on an independence score. We prove that RESIT is correct in the population setting and provide an empirical evaluation.

Keywords: causal inference, structural equation models, additive noise, identifiability, causal minimality, Bayesian networks

1. Introduction

Many scientific questions deal with the causal structure of a data-generating process. E.g., if we know the reasons why an individual is more susceptible to a disease than others, we can hope to develop new drugs in order to cure this disease or prevent its outbreak. Recent results indicate that knowing the causal structure is also useful for classical machine learning tasks. In the two variable case, for example, knowing which is cause and which is effect has

∗. Part of this work was done while JP and JMM were with the MPI Tübingen.

implications for semi-supervised learning and covariate shift adaptation (Schölkopf et al., 2012).

We consider a p-dimensional random vector $\mathbf{X} = (X_1, \ldots, X_p)$ with a joint distribution $\mathcal{L}(\mathbf{X})$ and assume that there is a true acyclic causal graph $\mathcal{G}$ that describes the data generating process (see Section 1.3). In this work we address the following problem of causal inference: given the distribution $\mathcal{L}(\mathbf{X})$ we try to infer the graph $\mathcal{G}$. A priori, the causal graph contains information about the physical process that cannot be found in properties of the joint distribution. One therefore requires assumptions connecting these two worlds. While traditional methods like PC, FCI (Spirtes et al., 2000) or score-based approaches (e.g. Chickering, 2002), that are explained in more detail in Section 2, make assumptions that enable us to recover the graph up to the Markov equivalence class, we investigate a different set of assumptions. If the data have been generated by an additive noise model (see Section 3), we will generically be able to recover the correct graph from the joint distribution.

In the remainder of this section we set up the required notation and definitions for graphs (Section 1.1), briefly introduce Judea Pearl's do-notation (Section 1.2) and use it to define our object of interest, a true causal graph (Section 1.3). We introduce structural equation models (SEMs) in Section 1.4. After discussing existing methods in Section 2, we provide the main results of this work in Section 3. We prove that for restricted additive noise models, a special class of SEMs, one can identify the graph from the joint distribution. This is possible not only for additive noise models (ANMs) but for all classes of SEMs that are able to identify graphs from a bivariate distribution, meaning they can distinguish between cause and effect. Section 4 proposes algorithms that can be used in practice, when instead of the joint distribution, we are only given i.i.d. samples. These algorithms are tested in Section 5.

This paper builds on the conference papers of Hoyer et al. (2009), Peters et al. (2011b) and Mooij et al. (2009)[1] but extends the material in several aspects. All deliberations in Section 1.3 about the true causal graph and Example 10 are novel. The presentation of the theoretical results in Section 3 is improved. In particular, we added the motivating Example 26 and Propositions 4 and 29. Example 25 provides a non-identifiable case different from the linear Gaussian example. Proposition 23 is based on Zhang and Hyvärinen (2009) and contains important necessary conditions for the failure of identifiability. In Corollary 31 we present a novel identifiability result for a class of nonlinear functions and Gaussian noise variables. Proposition 17 proves that causal minimality is satisfied if the structural equations do not contain constant functions. Section 3.3 contains results that guarantee to find the set of correct topological orderings when the assumption of causal minimality is dropped. Theorem 34 proves a conjecture from Mooij et al. (2009) by showing that given a regression and independence oracle the algorithm provided in Mooij et al. (2009) is correct. We propose a new score function for estimating the true directed acyclic graph in Section 4.2 and present two corresponding score-based methods. We provide an extended section on simulation experiments and discuss experiments on real data.

1. Parts of Sections 1 and 2 have been taken and modified from the PhD thesis of Peters (2012).

1.1 Directed Acyclic Graphs

We start with some basic notation for graphs. Consider a finite family of random variables $\mathbf{X} = (X_1, \ldots, X_p)$ with index set $\mathbf{V} := \{1, \ldots, p\}$ (we use capital letters for random variables and bold letters for sets and vectors). We denote their joint distribution by $\mathcal{L}(\mathbf{X})$. We write $p_{X_1}(x)$ or simply $p(x)$ for the Radon-Nikodym derivative of $\mathcal{L}(X_1)$ either with respect to the Lebesgue or the counting measure and (sometimes implicitly) assume its existence. A graph $\mathcal{G} = (\mathbf{V}, \mathcal{E})$ consists of nodes $\mathbf{V}$ and edges $\mathcal{E} \subseteq \mathbf{V}^2$ with $(v, v) \notin \mathcal{E}$ for any $v \in \mathbf{V}$. In a slight abuse of notation we identify the nodes (or vertices) $j \in \mathbf{V}$ with the variables X_j, the context should clarify the meaning. We also consider sets of variables $\mathbf{S} \subseteq \mathbf{X}$ as a single multivariate variable. We now introduce graph terminology that we require later. Most of the definitions can be found in Spirtes et al. (2000); Koller and Friedman (2009); Lauritzen (1996), for example.

Let $\mathcal{G} = (\mathbf{V}, \mathcal{E})$ be a graph with $\mathbf{V} := \{1, \ldots, p\}$ and corresponding random variables $\mathbf{X} = (X_1, \ldots, X_p)$. A graph $\mathcal{G}_1 = (\mathbf{V}_1, \mathcal{E}_1)$ is called a **subgraph** of $\mathcal{G}$ if $\mathbf{V}_1 = \mathbf{V}$ and $\mathcal{E}_1 \subseteq \mathcal{E}$; we then write $\mathcal{G}_1 \leq \mathcal{G}$. If additionally, $\mathcal{E}_1 \neq \mathcal{E}$, we call $\mathcal{G}_1$ a **proper subgraph** of $\mathcal{G}$.

A node i is called a **parent** of j if $(i, j) \in \mathcal{E}$ and a **child** if $(j, i) \in \mathcal{E}$. The set of parents of j is denoted by $\mathbf{PA}_j^{\mathcal{G}}$, the set of its children by $\mathbf{CH}_j^{\mathcal{G}}$. Two nodes i and j are **adjacent** if either $(i, j) \in \mathcal{E}$ or $(j, i) \in \mathcal{E}$. We call $\mathcal{G}$ **fully connected** if all pairs of nodes are adjacent. We say that there is an undirected edge between two adjacent nodes i and j if $(i, j) \in \mathcal{E}$ and $(j, i) \in \mathcal{E}$. An edge between two adjacent nodes is directed if it is not undirected. We then write $i \to j$ for $(i, j) \in \mathcal{E}$. Three nodes are called an **immorality** or a **v-structure** if one node is a child of the two others that themselves are not adjacent. The **skeleton** of $\mathcal{G}$ is the set of all edges without taking the direction into account, that is all (i, j), such that $(i, j) \in \mathcal{E}$ or $(j, i) \in \mathcal{E}$.

A **path** in $\mathcal{G}$ is a sequence of (at least two) distinct vertices $i_1, \ldots, i_n$, such that there is an edge between i_k and i_{k+1} for all $k = 1, \ldots, n-1$. If $i_k \to i_{k+1}$ for all k we speak of a **directed path** from i_1 to i_n and call i_n a **descendant** of i_1. We denote all descendants of i by $\mathbf{DE}_i^{\mathcal{G}}$ and all non-descendants of i, excluding i, by $\mathbf{ND}_i^{\mathcal{G}}$. In this work, i is neither a descendant nor a non-descendant of itself. If $i_{k-1} \to i_k$ and $i_{k+1} \to i_k$, i_k is called a **collider** on this path. $\mathcal{G}$ is called a **partially directed acyclic graph (PDAG)** if there is no directed cycle, i.e., if there is no pair (j, k) such that there are directed paths from j to k and from k to j. $\mathcal{G}$ is called a **directed acyclic graph (DAG)** if it is a PDAG and all edges are directed.

In a DAG, a path between i_1 and i_n is **blocked by a set $\mathbf{S}$** (with neither i_1 nor i_n in this set) whenever there is a node i_k, such that one of the following two possibilities hold: 1. $i_k \in \mathbf{S}$ and $i_{k-1} \to i_k \to i_{k+1}$ or $i_{k-1} \leftarrow i_k \leftarrow i_{k+1}$ or $i_{k-1} \leftarrow i_k \to i_{k+1}$ Or 2., $i_{k-1} \to i_k \leftarrow i_{k+1}$ and neither i_k nor any of its descendants is in $\mathbf{S}$. We say that two disjoint subsets of vertices $\mathbf{A}$ and $\mathbf{B}$ are **d-separated** by a third (also disjoint) subset $\mathbf{S}$ if every path between nodes in $\mathbf{A}$ and $\mathbf{B}$ is blocked by $\mathbf{S}$. Throughout this work, $\perp\!\!\!\perp$ denotes (conditional) independence. The joint distribution $\mathcal{L}(\mathbf{X})$ is said to be **Markov with respect to the DAG** $\mathcal{G}$ if

$$\mathbf{A}, \mathbf{B} \text{ } d\text{-sep. by } \mathbf{C} \;\Rightarrow\; \mathbf{A} \perp\!\!\!\perp \mathbf{B} \,|\, \mathbf{C}$$

for all disjoint sets $\mathbf{A}, \mathbf{B}, \mathbf{C}$. $\mathcal{L}(\mathbf{X})$ is said to be **faithful to the DAG** $\mathcal{G}$ if

$$\mathbf{A}, \mathbf{B} \text{ } d\text{-sep. by } \mathbf{C} \;\Leftarrow\; \mathbf{A} \perp\!\!\!\perp \mathbf{B} \,|\, \mathbf{C}$$

Figure 1: After fine-tuning the parameters for the two graphs, both models generate the same joint distribution.

for all disjoint sets $\mathbf{A}, \mathbf{B}, \mathbf{C}$. A distribution satisfies **causal minimality** with respect to $\mathcal{G}$ if it is Markov with respect to $\mathcal{G}$, but not to any proper subgraph of $\mathcal{G}$. We denote by $\mathcal{M}(\mathcal{G})$ the set of distributions that are Markov with respect to $\mathcal{G}$: $\mathcal{M}(\mathcal{G}) := \{\mathcal{L}(\mathbf{X}) : \mathcal{L}(\mathbf{X}) \text{ is Markov w.r.t. } \mathcal{G}\}$. Two DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ are **Markov equivalent** if $\mathcal{M}(\mathcal{G}_1) = \mathcal{M}(\mathcal{G}_2)$. This is the case if and only if $\mathcal{G}_1$ and $\mathcal{G}_2$ satisfy the same set of d-separations, that means the Markov condition entails the same set of (conditional) independence conditions. The set of all DAGs that are Markov equivalent to some DAG (a so-called Markov equivalence class) can be represented by a **completed PDAG**. This graph satisfies $(i, j) \in \mathcal{E}$ if and only if one member of the Markov equivalence class does. Verma and Pearl (1991) showed that:

Lemma 1 *Two DAGs are Markov equivalent if and only if they have the same skeleton and the same immoralities.*

Faithfulness is not very intuitive at first glance. We now give an example of a distribution that is Markov but not faithful with respect to some DAG $\mathcal{G}_1$. This is achieved by making two paths cancel each other and creating an independence that is not implied by the graph structure.

Example 2 *Consider the two graphs in Figure 1. Corresponding to the left graph we generate a joint distribution by the following equations.* $X = N_X, Y = aX + N_Y, Z = bY + cX + N_Z$, *with normally distributed noise variables* $N_X \sim \mathcal{N}(0, \sigma_X^2)$, $N_Y \sim \mathcal{N}(0, \sigma_Y^2)$ *and* $N_Z \sim \mathcal{N}(0, \sigma_Z^2)$ *that are jointly independent. This is an example of a linear Gaussian structural equation model with graph* $\mathcal{G}_1$ *that we formally define in Section 1.4. Now, if* $a \cdot b + c = 0$, *the distribution is not faithful with respect to* $\mathcal{G}_1$ *since we obtain* $X \perp\!\!\!\perp Z$; *more precisely, it is not triangle-faithful (Zhang and Spirtes, 2008).*

Correspondingly, we generate a distribution related to graph $\mathcal{G}_2$: $X = \tilde{N}_X, Y = \tilde{a}X + \tilde{b}Z + \tilde{N}_Y, Z = \tilde{N}_Z$, *with all* $\tilde{N}_\cdot \sim \mathcal{N}(0, \tau_\cdot^2)$ *jointly independent. If we choose* $\tau_X^2 = \sigma_X^2$, $\tilde{a} = a$, $\tau_Z^2 = b^2\sigma_Y^2 + \sigma_Z^2$, $\tilde{b} = (b\sigma_Y^2)/(b^2\sigma_Y^2 + \sigma_Z^2)$ *and* $\tau_Y^2 = \sigma_Y^2 - (b^2\sigma_Y^4)/(b^2\sigma_Y^2 + \sigma_Z^2)$, *both models lead to the covariance matrix*

$$\Sigma = \begin{pmatrix} \sigma_X^2 & a\sigma_X^2 & 0 \\ a\sigma_X^2 & a^2\sigma_X^2 + \sigma_Y^2 & b\sigma_Y^2 \\ 0 & b\sigma_Y^2 & b^2\sigma_Y^2 + \sigma_Z^2 \end{pmatrix}$$

and thus to the same distribution. It can be checked that the distribution is faithful with respect to $\mathcal{G}_2$ *if* $\tilde{a}, \tilde{b} \neq 0$ *and all* $\tilde{\tau}_\cdot > 0$.

The distribution from Example 2 is faithful with respect to $\mathcal{G}_2$, but not with respect to $\mathcal{G}_1$. Nevertheless, for both models, causal minimality is satisfied if none of the parameters vanishes: the distribution is not Markov to any proper subgraph of $\mathcal{G}_1$ or $\mathcal{G}_2$ since removing an arrow would correspond to a new (conditional) independence that does not hold in the distribution. Note that $\mathcal{G}_2$ is not a proper subgraph of $\mathcal{G}_1$. In general, causal minimality is weaker than faithfulness:

Remark 3 *If $\mathcal{L}(\mathbf{X})$ is faithful with respect to $\mathcal{G}$, then causal minimality is satisfied.*

This is due to the fact that any two nodes that are not directly connected by an edge can be d-separated. Another, equivalent formulation of causal minimality reads as follows:

Proposition 4 *Consider the random vector $\mathbf{X} = (X_1, \ldots, X_p)$ and assume that the joint distribution has a density with respect to a product measure. Suppose that $\mathcal{L}(\mathbf{X})$ is Markov with respect to $\mathcal{G}$. Then $\mathcal{L}(\mathbf{X})$ satisfies causal minimality with respect to $\mathcal{G}$ if and only if $\forall X_j \, \forall Y \in \mathbf{PA}_j^{\mathcal{G}}$ we have that $X_j \not\perp\!\!\!\perp Y \,|\, \mathbf{PA}_j^{\mathcal{G}} \setminus \{Y\}$.*

Proof See Appendix A.1. ■

1.2 Intervention Distributions[2]

Given a directed acyclic graph (DAG) $\mathcal{G}$, Pearl (2009) introduces the *do*-notation as a mathematical description of interventional experiments. More precisely, $do(X_j = \tilde{p}(x_j))$ stands for setting the variable X_j randomly according to the distribution $\tilde{p}(x_j)$, irrespective of its parents, while not interfering with any other variable. Formally:

Definition 5 *Let $\mathbf{X} = (X_1, \ldots, X_p)$ be a collection of variables with joint distribution $\mathcal{L}(\mathbf{X})$ that we assume to be absolutely continuous with respect to the Lebesgue measure or the counting measure (i.e., there exists a probability density function or a probability mass function). Given a DAG $\mathcal{G}$ over $\mathbf{X}$, we define the intervention distribution $do(X_j = \tilde{p}(x_j))$ of $X_1, \ldots, X_p$ by*

$$p\big(x_1, \ldots, x_p \,|\, do(X_j = \tilde{p}(x_j))\big) := \prod_{i \neq j}^{p} p(x_i | x_{\mathbf{PA}_i}) \cdot \tilde{p}(x_j)$$

if $p(x_1, \ldots, x_p) > 0$ and zero otherwise. Here $\tilde{p}(x_j)$ is either a probability density function or a probability mass function. Similarly, we can intervene at different nodes at the same time by defining the intervention distribution $do(X_j = \tilde{p}(x_j), j \in \mathbf{J})$ for $\mathbf{J} \subseteq \mathbf{V}$ as

$$p\big(x_1, \ldots, x_p \,|\, do(X_j = \tilde{p}(x_j), j \in \mathbf{J})\big) := \prod_{i \notin \mathbf{J}} p(x_i | x_{\mathbf{PA}_i}) \cdot \prod_{j \in \mathbf{J}} \tilde{p}(x_j)$$

if $p(x_1, \ldots, x_p) > 0$ and zero otherwise.

2. Sections 1.2 and 1.3 are not essential for understanding the rest of the paper and can be skipped on first reading.

Here, $x_{\mathbf{PA}_i}$ denotes the tuple of all x_j for X_j being a parent of X_i in $\mathcal{G}$. Pearl (2009) introduces Definition 5 with the special case of $\tilde{p}(x_j) = \delta_{x_j,\tilde{x}_j}$, where $\delta_{x_j,\tilde{x}_j} = 1$ if $x_j = \tilde{x}_j$ and $\delta_{x_j,\tilde{x}_j} = 0$ otherwise; this corresponds to a point mass at $\tilde{x_j}$. For more details on *soft* interventions, see Eberhardt and Scheines (2007). Note that in general:

$$p(x_1, \ldots, x_p \,|\, do(X_j = \tilde{x}_j)) \neq p(x_1, \ldots, x_p \,|\, X_j = \tilde{x}_j)\,.$$

The expression $p\big(x_1, \ldots, x_p \,|\, do(X_j = \tilde{x}_j, j \in \mathbf{J})\big)$ yields a distribution over $X_1, \ldots, X_p$. If we are only interested in computing the marginal $p\big(x_i \,|\, do(X_j = \tilde{x}_j)\big)$, where X_i is not a parent of X_j, we can use the parent adjustment formula (Pearl, 2009, Theorem 3.2.2)

$$p(x_i \,|\, do(X_j = \tilde{x}_j)) = \sum_{x_{\mathbf{PA}_j}} p(x_i \,|\, \tilde{x}_j, x_{\mathbf{PA}_j})\, p(x_{\mathbf{PA}_j})\,. \tag{1}$$

1.3 True Causal Graphs[2]

In this section we clarify what we mean by a true causal graph $\mathcal{G}_c$. In short, we use this term if the results of randomized studies are determined by $\mathcal{G}_c$ and the observational joint distribution. This means that the graph and the observational joint distribution lead to causal effects that one observes in practice. Two important restrictive assumptions that we make throughout this work are *acyclicity* (the absence of directed cycles, in other words, no causal feedback loops are allowed) and *causal sufficiency* (the absence of hidden variables that are a common cause of at least two observed variables).

Definition 6 *Assume we are given a distribution $\mathcal{L}(\mathbf{X})$ over $X_1, \ldots, X_p$ and distributions $\mathcal{L}_{do(X_j=\tilde{p}(x_j),j\in\mathbf{J})}(\mathbf{X})$ for all $\mathbf{J} \subseteq \mathbf{V} = \{1, \ldots, p\}$ (think of the variables X_j having been randomized). We then call the graph $\mathcal{G}_c$ a* true causal graph *for these distributions if*

- *$\mathcal{G}_c$ is a directed acyclic graph;*
- *the distribution $\mathcal{L}(\mathbf{X})$ is Markov with respect to $\mathcal{G}_c$;*
- *for all $\mathbf{J} \subseteq \mathbf{V}$ and $\tilde{p}(x_j)$ with $j \in \mathbf{J}$ the distribution $\mathcal{L}_{do(X_j=\tilde{p}(x_j),j\in\mathbf{J})}(\mathbf{X})$ coincides with $p\big(x_1, \ldots, x_p \,|\, do(X_j = \tilde{p}(x_j), j \in \mathbf{J})\big)$, computed from $\mathcal{G}_c$ as in Definition 5.*

Definition 6 is purely mathematical if one considers $\mathcal{L}_{do(X_j=\tilde{p}(x_j),j\in\mathbf{J})}(\mathbf{X})$ as an abstract family of given distributions. But it is a small step to make the relation to the "real world". We call $\mathcal{G}_c$ the true causal graph *of a data generating process* if it is the true causal graph for the distributions $\mathcal{L}(\mathbf{X})$ and $\mathcal{L}_{do(X_j=\tilde{p}(x_j),j\in\mathbf{J})}(\mathbf{X})$, where the latter are obtained by randomizing X_j according to $\tilde{p}(x_j)$. In some situations, the precise design of a randomized experiment may not be obvious. While most people would agree on how to randomize over medical treatment procedures, there is probably less agreement how to randomize over the tolerance of a person (does this include other changes of his personality, too?). Only sometimes, this problem can be resolved by including more variables and taking a less coarse-grained point of view. We do not go into further detail since we believe that this would require philosophical deliberations, which lie beyond the scope of this work. Instead, we may explicitly add the requirement that "most people agree on what a randomized experiment should look like in this context".

In general, there can be more than one true causal DAG. If one requires causal minimality, the true causal DAG is unique.

Proposition 7 *Assume $\mathcal{L}(X_1, \ldots, X_p)$ has a density and consider all true causal DAGs $\mathbb{G} := \{\mathcal{G}_{c,1}, \ldots, \mathcal{G}_{c,m}\}$ of $X_1, \ldots, X_p$. Then there is a partial order on $\mathbb{G}$ using the subgraph property $\leq$ as an ordering. This ordering has a least element $\mathcal{G}_c$, i.e., $\mathcal{G}_c \leq \mathcal{G}_{c,i}$ for all i. This element $\mathcal{G}_c$ is the unique true causal DAG such that $\mathcal{L}(\mathbf{X})$ satisfies causal minimality with respect to $\mathcal{G}_c$.*

Proof See Appendix A.2 ■

We now briefly comment on a true causal graph's behavior when some of the variables from the joint distribution are marginalized out.

Example 8 *(i) If $X \leftarrow Z \rightarrow Y$ is the only true causal graph for X, Y and Z, there is no true causal graph for the variables X and Y (the do-statements do not coincide).*

(ii) Assume that the graph $X \rightarrow Y \rightarrow Z$ with additional $X \rightarrow Z$ is the only true causal graph for X, Y and Z and assume that $\mathcal{L}(X, Y, Z)$ is faithful with respect to this graph. Then, the only true causal graph for the variables X and Z is $X \rightarrow Z$.

(iii) If the situation is the same as in (ii) with the difference that $X \perp\!\!\!\perp Z$ (i.e., $\mathcal{L}(X, Y, Z)$ is not faithful with respect to the true causal graph), the empty graph and $Z \leftarrow X$ are also true causal graphs for X and Z.

Latent projections (Verma and Pearl, 1991) provide a formal way to obtain a true causal graph for marginalization. Cases (ii) and (iii) show that there are no purely graphical criteria that provide the *minimal* true causal graph described in Proposition 7.

The results presented in the remainder of this paper can be understood without causal interpretation. Using these techniques to infer a true causal graph, however, requires the assumption that such a true causal DAG $\mathcal{G}_c$ for the observed distribution of $X_1, \ldots, X_p$ exists. This includes the assumption that all "relevant" variables have been observed, sometimes called causal sufficiency, and that there are no feedback loops.

Richardson and Spirtes (2002) introduce a representation of graphs (so-called Maximal Ancestral Graphs, or MAGs) with hidden variables that is closed under marginalization and conditioning. The FCI algorithm (Spirtes et al., 2000) exploits the conditional independences in the data to partially reconstruct the graph. Other work concentrates on hidden variables in structural equation models (e.g., Hoyer et al., 2008; Janzing et al., 2009; Silva and Ghahramani, 2009).

1.4 Structural Equation Models

A structural equation model (SEM) (also called a functional model) is defined as a tuple $(\mathcal{S}, \mathcal{L}(\mathbf{N}))$, where $\mathcal{S} = (S_1, \ldots, S_p)$ is a collection of p equations

$$S_j : \quad X_j = f_j(\mathbf{PA}_j, N_j)\,, \qquad j = 1, \ldots, p \tag{2}$$

and $\mathcal{L}(\mathbf{N}) = \mathcal{L}(N_1, \ldots, N_p)$ is the joint distribution of the noise variables, which we require to be jointly independent, i.e., $\mathcal{L}(\mathbf{N})$ is a product distribution. We consider SEMs only for real-valued random variables $X_1, \ldots, X_p$. The graph of a structural equation model is obtained simply by drawing direct edges from each parent to its direct effects, i.e., from

each variable X_k occurring on the right-hand side of equation (2) to X_j. We henceforth assume this graph to be acyclic. According to the notation defined in Section 1.1, $\mathbf{PA}_j$ are the parents of X_j.

The $\mathbf{PA}_j$ can be considered as the direct causes of X_j. An SEM specifies how the $\mathbf{PA}_j$ affect X_j. Note that in physics (chemistry, biology, ...), we would usually expect that such causal relationships occur in time, and are governed by sets of coupled differential equations. Under certain assumptions such as stable equilibria, one can derive an SEM that describes how the equilibrium states of such a dynamical system will react to physical interventions on the observables involved (Mooij et al., 2013). We do not deal with these issues in the present paper but take the SEM as our starting point instead. We formulate the identifiability results without the notion of causality.

Pearl (2009) shows in Theorem 1.4.1 that the law $\mathcal{L}(\mathbf{X})$ generated by an SEM is Markov with respect to its graph. Reversely, there always exists an SEM that models a given distribution.[3]

Proposition 9 *Consider $X_1, \ldots, X_p$ and let $\mathcal{L}(\mathbf{X})$ have a strictly positive density with respect to the Lebesgue measure and assume it is Markov with respect to $\mathcal{G}$. Then there exists an SEM $(\mathcal{S}, \mathcal{L}(\mathbf{N}))$ with graph $\mathcal{G}$ that generates the distribution $\mathcal{L}(\mathbf{X})$.*

Proof See Appendix A.3. ■

Structural equation models contain strictly more information than their corresponding graph and law and hence also more information than the family of all intervention distributions together with the observational distribution. This information sometimes helps to answer counterfactual questions, as shown in the following example.

Example 10 *Let $N_1, N_2 \sim Ber(0.5)$ and $N_3 \sim U(\{0,1,2\})$, such that the three variables are jointly independent. That is, N_1, N_2 have a Bernoulli distribution with parameter 0.5 and N_3 is uniformly distributed on $\{0,1,2\}$. We define two different SEMs, first consider:*

$$\mathcal{S}_A = \begin{cases} X_1 = N_1 \\ X_2 = N_2 \\ X_3 = (1_{N_3>0} \cdot X_1 + 1_{N_3=0} \cdot X_2) \cdot 1_{X_1 \neq X_2} + N_3 \cdot 1_{X_1 = X_2} \,. \end{cases}$$

If X_1 and X_2 have different values, depending on N_3 we either choose $X_3 = X_1$ or $X_3 = X_2$. Otherwise $X_3 = N_3$. Now, $\mathcal{S}_B$ differs from $\mathcal{S}_A$ only in the latter case:

$$\mathcal{S}_B = \begin{cases} X_1 = N_1 \\ X_2 = N_2 \\ X_3 = (1_{N_3>0} \cdot X_1 + 1_{N_3=0} \cdot X_2) \cdot 1_{X_1 \neq X_2} + (2 - N_3) \cdot 1_{X_1 = X_2} \,. \end{cases}$$

It can be checked that both SEMs generate the same observational distribution, which satisfies causal minimality with respect to the graph $X_1 \to X_3 \leftarrow X_2$. They also generate the same intervention distributions, for any possible intervention. But the two models differ in a counterfactual statement.[4] *Suppose, we have seen a sample $(X_1, X_2, X_3) = (1, 0, 0)$ and*

3. A similar but weaker statement than Proposition 9 can be found in Druzdzel and van Leijen (2001); Janzing and Schölkopf (2010).
4. Here, we make use of Judea Pearl's definition of counterfactuals (Pearl, 2009).

we are interested in the counterfactual question, what X_3 would have been if X_1 had been 0. From both $\mathcal{S}_A$ and $\mathcal{S}_B$ it follows that $N_3 = 0$, and thus the two SEMs "predict" different values for X_3 under a counterfactual change of X_1.

If we want to use an estimated SEM to predict counterfactual questions, this example shows that we require assumptions that let us distinguish between $\mathcal{S}_A$ or $\mathcal{S}_B$. In this work we exploit the additive noise assumption to infer the *structure* of an SEM. We do not claim that we can predict counterfactual statements.

Structural equation models have been used for a long time in fields like agriculture or social sciences (e.g., Wright, 1921; Bollen, 1989). Model selection, for example, was done by fitting different structures that were considered as reasonable given the prior knowledge about the system. These candidate structures were then compared using goodness of fit tests. In this work we instead consider the question of identifiability, which has not been addressed until more recently.

Problem 11 (population case) *We are given a distribution $\mathcal{L}(\mathbf{X}) = \mathcal{L}(X_1, \ldots, X_p)$ that has been generated by an (unknown) structural equation model with graph $\mathcal{G}_0$; in particular, $\mathcal{L}(\mathbf{X})$ is Markov with respect to $\mathcal{G}_0$. Can the (observational) distribution $\mathcal{L}(\mathbf{X})$ be generated by a structural equation model with a different graph $\mathcal{G} \neq \mathcal{G}_0$? If not, we call $\mathcal{G}_0$ identifiable from $\mathcal{L}(\mathbf{X})$.*

In general, $\mathcal{G}_0$ is not identifiable from $\mathcal{L}(\mathbf{X})$: the joint distribution $\mathcal{L}(\mathbf{X})$ is certainly Markov with respect to a lot of different graphs, e.g., to all fully connected acyclic graphs. Proposition 9 states the existence of corresponding SEMs. What can be done to overcome this indeterminacy? The hope is that by using additional assumptions one obtains restricted models, in which we can identify the graph from the joint distribution. Considering graphical models, we see in Section 2.1 how the assumption that $\mathcal{L}(\mathbf{X})$ is Markov and faithful with respect to $\mathcal{G}_0$ leads to identifiability of the Markov equivalence class of $\mathcal{G}_0$. Considering SEMs, we see in Section 3 that additive noise models as a special case of restricted SEMs even lead to identifiability of the correct DAG. Also Section 2.3 contains such a restriction based on SEMs.

2. Alternative Methods

We briefly describe some existing methods and provide references for more details.

2.1 Estimating the Markov Equivalence Class: Independence-Based Methods

Conditional independence-based methods like the PC algorithm and the FCI algorithm (Spirtes et al., 2000) assume that $\mathcal{L}(\mathbf{X})$ is Markov and faithful with respect to the correct graph $\mathcal{G}_0$ (that means *all* conditional independences in the joint distribution are entailed by the Markov condition, cf. Section 1.1). Since both assumptions put restrictions only on the conditional independences in the joint distribution, these methods are not able to distinguish between two graphs that entail exactly the same set of (conditional) independences, i.e., between Markov equivalent graphs. Since many Markov equivalence classes contain more than one graph, conditional independence-based methods thus usually leave some arrows undirected and cannot uniquely identify the correct graph.

The first step of the PC algorithm determines the variables that are adjacent. One therefore has to test whether two variables are dependent given *any* other subset of variables. The PC algorithm exploits a very clever procedure to reduce the size of the condition set. In the worst case, however, one has to perform conditional independence tests with conditioning sets of up to $p-2$ variables (where p is the number of variables in the graph). Although there is recent work on kernel-based conditional independence tests (Fukumizu et al., 2008; Zhang et al., 2011), such tests are difficult to perform in practice if one does not restrict the variables to follow a Gaussian distribution, for example (e.g., Bergsma, 2004).

To prove consistency of the PC algorithm one does not only require faithfulness, but strong faithfulness (Zhang and Spirtes, 2003; Kalisch and Bühlmann, 2007). Uhler et al. (2013) argue that this is a restrictive condition. Since parts of faithfulness can be tested given the data (Zhang and Spirtes, 2008), the condition may be weakened.

From our perspective independence-based methods face the following challenges: (1) We can identify the correct DAG only up to Markov equivalence classes. (2) Conditional independence testing, especially with a large conditioning set, is difficult in practice. (3) Simulation experiments suggest, that in many cases, the distribution is close to unfaithfulness. In these cases there is no guarantee that the inferred graph(s) will be close to the original one.

2.2 Estimating the Markov Equivalence Class: Score-Based Methods

Although the roots for score-based methods for causal inference may date back even further, we mainly refer to Geiger and Heckerman (1994), Heckerman (1997) and Chickering (2002) and references therein. Given the data $\mathcal{D}$ from a vector $\mathbf{X}$ of variables, i.e., n i.i.d. samples, the idea is to assign a score $S(\mathcal{D},\mathcal{G})$ to each graph $\mathcal{G}$ and search over the space of DAGs for the best scoring graph:

$$\hat{\mathcal{G}} := \operatorname*{argmax}_{\mathcal{G} \text{ DAG over } \mathbf{X}} S(\mathcal{D},\mathcal{G})\,. \tag{3}$$

There are several possibilities to define such a scoring function. Often a parametric model is assumed (e.g., linear Gaussian equations or multinomial distributions), which introduces a set of parameters $\theta \in \mathbf{\Theta}$.

From a Bayesian point of view, we may define priors $p_{pr}(\mathcal{G})$ and $p_{pr}(\theta)$ over DAGs and parameters and consider the log posterior as a score function, or equivalently (note that $p(\mathcal{D})$ is constant over all DAGs):

$$S(\mathcal{D},\mathcal{G}) := \log p_{pr}(\mathcal{G}) + \log p(\mathcal{D}|\mathcal{G})\,,$$

where $p(\mathcal{D}|\mathcal{G})$ is the marginal likelihood

$$p(\mathcal{D}|\mathcal{G}) = \int_{\mathbf{\Theta}} p(\mathcal{D}|\mathcal{G},\theta)\cdot p_{pr}(\theta)\,d\theta.$$

In this case, $\hat{\mathcal{G}}$ defined in (3) is the mode of the posterior distribution, which is usually called the *maximum a posteriori* (or MAP) estimator. Instead of a MAP estimator, one may be interested in the full posterior distribution over DAGs. This distribution can subsequently

be averaged over all graphs to get a posterior of the hypothesis about the existence of a specific edge, for example.

In the case of parametric models, we call two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ *distribution equivalent* if for each parameter $\theta_1 \in \mathbf{\Theta}_1$ there is a corresponding parameter $\theta_2 \in \mathbf{\Theta_2}$, such that the distribution obtained from $\mathcal{G}_1$ in combination with θ_1 is the same as the distribution obtained from graph $\mathcal{G}_2$ with θ_2, and vice versa. It is known that in the linear Gaussian case (or for unconstrained multinomial distributions) two graphs are distribution-equivalent if and only if they are Markov equivalent. One may therefore argue that $p(\mathcal{D}|\mathcal{G}_1)$ and $p(\mathcal{D}|\mathcal{G}_2)$ should be the same for Markov equivalent graphs $\mathcal{G}_1$ and $\mathcal{G}_2$. Heckerman and Geiger (1995) discuss how to choose the prior over parameters accordingly.

Instead, we may consider the maximum likelihood estimator $\hat{\theta}$ in each graph and define a score function by using a penalty, e.g., the Bayesian Information Criterion (BIC):

$$S(\mathcal{D}, \mathcal{G}) = \log p(\mathcal{D}|\hat{\theta}, \mathcal{G}) - \frac{d}{2} \log n \,,$$

where n is the sample size and d the dimensionality of the parameter θ.

Since the search space of all DAGs is growing super-exponentially in the number of variables (e.g., Chickering, 2002), greedy search algorithms are applied to solve equation (3): at each step there is a candidate graph and a set of neighboring graphs. For all these neighbors one computes the score and considers the best-scoring graph as the new candidate. If none of the neighbors obtains a better score, the search procedure terminates (not knowing whether one obtained only a local optimum). Clearly, one therefore has to define a neighborhood relation. Starting from a graph $\mathcal{G}$, we may define all graphs as neighbors from $\mathcal{G}$ that can be obtained by removing, adding or reversing one edge. In the linear Gaussian case, for example, one cannot distinguish between Markov equivalent graphs. It turns out that in those cases it is beneficial to change the search space to Markov equivalence classes instead of DAGs. The greedy equivalence search (GES) (Meek, 1997; Chickering, 2002) starts with the empty graph and consists of two-phases. In the first phase, edges are added until a local maximum is reached; in the second phase, edges are removed until a local maximum is reached, which is then given as an output of the algorithm. Chickering (2002) proves consistency of this method by using consistency of the BIC (Haughton, 1988).

2.3 Estimating the DAG: LiNGAM

Kano and Shimizu (2003) and Shimizu et al. (2006) propose an inspiring method exploiting non-Gaussianity of the data.[5] Although their work covers the general case, the idea is maybe best understood in the case of two variables:

Example 12 *Suppose*

$$Y = \phi X + N, \quad N \perp\!\!\!\perp X \,,$$

where X and N are normally distributed. It is easy to check that

$$X = \tilde{\phi} Y + \tilde{N}, \quad \tilde{N} \perp\!\!\!\perp Y \,.$$

with $\tilde{\phi} = \frac{\phi \mathbf{var}(X)}{\phi^2 \mathbf{var}(X) + \sigma^2} \neq \frac{1}{\phi}$ and $\tilde{N} = X - \tilde{\phi} Y$.

5. A more detailed tutorial can be found on `http://www.ar.sanken.osaka-u.ac.jp/~sshimizu/papers/Shimizu13BHMK.pdf`.

If we consider non-Gaussian noise, however, the structural equation model becomes identifiable.

Proposition 13 *Let X and Y be two random variables, for which*

$$Y = \phi X + N, \quad N \perp\!\!\!\perp X,\ \phi \neq 0$$

holds. Then we can reverse the process, i.e., there exists $\psi \in \mathbb{R}$ and a noise $\tilde{N}$, such that

$$X = \psi Y + \tilde{N}, \quad \tilde{N} \perp\!\!\!\perp Y\,,$$

if and only if X and N are Gaussian distributed.

Shimizu et al. (2006) were the first to report this result. They prove it even for more than two variables using Independent Component Analysis (ICA) (Comon, 1994, Theorem 11), which itself is proved using the Darmois-Skitovič theorem (Skitovič, 1954, 1962; Darmois, 1953). Alternatively, Proposition 13 can be proved directly using the Darmois-Skitovič theorem (e.g., Peters, 2008, Theorem 2.10).

Theorem 14 (Shimizu et al., 2006) *Assume a linear SEM with graph $\mathcal{G}_0$*

$$X_j = \sum_{k \in \mathbf{PA}_j^{\mathcal{G}_0}} \beta_{jk} X_k + N_j\,, \qquad j = 1, \ldots, p\,, \tag{4}$$

where all N_j are jointly independent and non-Gaussian distributed. Additionally, for each $j \in \{1, \ldots, p\}$ we require $\beta_{jk} \neq 0$ for all $k \in \mathbf{PA}_j^{\mathcal{G}_0}$. Then, the graph $\mathcal{G}_0$ is identifiable from the joint distribution.

The authors call this model a linear non-Gaussian acyclic model (LiNGAM) and provide a practical method based on ICA that can be applied to a finite amount of data. Later, improved versions of this method have been proposed in Shimizu et al. (2011); Hyvärinen and Smith (2013).

2.4 Estimating the DAG: Gaussian SEMs with Equal Error Variances

There is another deviation from linear Gaussian SEMs that makes the graph identifiable. Peters and Bühlmann (2014) show that restricting the noise variables to have the same variance is sufficient to recover the graph structure.

Theorem 15 (Peters and Bühlmann, 2014) *Assume an SEM with graph $\mathcal{G}_0$*

$$X_j = \sum_{k \in \mathbf{PA}_j^{\mathcal{G}_0}} \beta_{jk} X_k + N_j\,, \qquad j = 1, \ldots, p\,, \tag{5}$$

where all N_j are i.i.d. and follow a Gaussian distribution. Additionally, for each $j \in \{1, \ldots, p\}$ we require $\beta_{jk} \neq 0$ for all $k \in \mathbf{PA}_j^{\mathcal{G}_0}$. Then, the graph $\mathcal{G}_0$ is identifiable from the joint distribution.

For estimating the coefficients β_{jk} and the error variance σ^2, Peters and Bühlmann (2014) propose to use a penalized maximum likelihood method (BIC). For optimization they propose a greedy search algorithm in the space of DAGs. Rescaling the variables changes the variance of the error terms. Therefore, in many applications model (5) cannot be sensibly applied. The BIC criterion, however, always allows us to compare the method's score with the score of a linear Gaussian SEM that uses more parameters and does not make the assumption of equal error variances.

3. Identifiability of Continuous Additive Noise Models

Recall that equation (2) defines the general form of an SEM: $X_j = f_j(\mathbf{PA}_j, N_j)\,, j = 1, \ldots, p$ with jointly independent variables N_i. We have seen that these models are too general to identify the graph (Proposition 9). It turns out, however, that constraining the function class leads to identifiability. As a first step we restrict the form of the function to be additive with respect to the noise variable. (Throughout this section we assume that all random variables are absolutely continuous with respect to the Lebesgue measure. Peters et al. (2011a) provide an extension for variables that are absolutely continuous with respect to the counting measure.)

Definition 16 *We define a continuous additive noise model (ANM) as a tuple* $(\mathcal{S}, \mathcal{L}(\mathbf{N}))$*, where* $\mathcal{S} = (S_1, \ldots, S_p)$ *is a collection of* p *equations*

$$S_j : \quad X_j = f_j(\mathbf{PA}_j) + N_j\,, \qquad j = 1, \ldots, p\,, \tag{6}$$

where the $\mathbf{PA}_j$ *correspond to the direct parents of* X_j*, and the noise variables* N_j *have a strictly positive density (with respect to the Lebesgue measure) and are jointly independent. Furthermore, we assume that the corresponding graph is acyclic.*

For these models causal minimality reduces to the condition that each function f_j is not constant in any of its arguments:

Proposition 17 *Consider a distribution generated by a model* (6) *and assume that the functions* f_j *are not constant in any of its arguments, i.e., for all* j *and* $i \in \mathbf{PA}_j$ *there are some* $x_{\mathbf{PA}_j \setminus \{i\}}$ *and some* $x_i \neq x_i'$ *such that*

$$f_j(x_{\mathbf{PA}_j \setminus \{i\}}, x_i) \neq f_j(x_{\mathbf{PA}_j \setminus \{i\}}, x_i')\,.$$

Then the joint distribution satisfies causal minimality with respect to the corresponding graph. Conversely, if there is a j *and* i *such that* $f_j(x_{\mathbf{PA}_j \setminus \{i\}}, \cdot)$ *is constant, causal minimality is violated.*

Proof See Appendix A.4 ■

Linear functions and Gaussian variables identify only the correct Markov equivalence class and not necessarily the correct graph. In the remainder of this section we establish results showing that this is an exceptional case. We develop conditions that guarantee the identifiability of the DAG. Proposition 21 indicates that this condition is rather weak.

3.1 Bivariate Additive Noise Models

We now add another assumption about the form of the structural equations.

Definition 18 *Consider an additive noise model* (6) *with two variables, i.e., the two equations* $X_i = N_i$ *and* $X_j = f_j(X_i) + N_j$ *with* $\{i, j\} = \{1, 2\}$. *We call this SEM an* identifiable bivariate additive noise model *if the triple* $(f_j, \mathcal{L}(X_i), \mathcal{L}(N_j))$ *satisfies Condition 19.*

Condition 19 *The triple* $(f_j, \mathcal{L}(X_i), \mathcal{L}(N_j))$ *does not solve the following differential equation for all* x_i, x_j *with* $\nu''(x_j - f(x_i))f'(x_i) \neq 0$*:*

$$\xi''' = \xi'' \left(-\frac{\nu''' f'}{\nu''} + \frac{f''}{f'} \right) - 2\nu'' f'' f' + \nu' f''' + \frac{\nu' \nu''' f'' f'}{\nu''} - \frac{\nu'(f'')^2}{f'}\,, \tag{7}$$

Here, $f := f_j$, *and* $\xi := \log p_{X_i}$ *and* $\nu := \log p_{N_j}$ *are the logarithms of the strictly positive densities. To improve readability, we have skipped the arguments* $x_j - f(x_i)$, x_i, *and* x_i *for* ν, ξ, *and* f *and their derivatives, respectively.*

Zhang and Hyvärinen (2009) even allow for a bijective transformation of the data, i.e., $X_j = g_j(f_j(X_i) + N_j)$ and obtain a similar differential equation as (7). As the name in Definition 18 already suggests, we have identifiability for this class of SEMs.

Theorem 20 *Let* $\mathcal{L}(\mathbf{X}) = \mathcal{L}(X_1, X_2)$ *be generated by an identifiable bivariate additive noise model with graph* $\mathcal{G}_0$ *and assume causal minimality, i.e., a non-constant function* f_j *(Proposition 17). Then,* $\mathcal{G}_0$ *is identifiable from the joint distribution.*

Proof The proof of Hoyer et al. (2009) is reproduced in Appendix A.5. ■

Intuitively speaking, we expect a "generic" triple $(f_j, \mathcal{L}(X_i), \mathcal{L}(N_j))$ to satisfy Condition 19. The following proposition presents one possible formalization. After fixing $(f_j, \mathcal{L}(N_j))$ we consider the space of all distributions $\mathcal{L}(X_i)$ such that Condition 19 is violated. This space is contained in a three dimensional space. Since the space of continuous distributions is infinite dimensional, we can therefore say that Condition 19 is satisfied for "most distributions" $\mathcal{L}(X_i)$.

Proposition 21 *If for a fixed pair* $(f_j, \mathcal{L}(N_j))$ *there exists* $x_j \in \mathbb{R}$ *such that* $\nu''(x_j - f(x_i))f'(x_i) \neq 0$ *for all but a countable set of points* $x_i \in \mathbb{R}$, *the set of all* $\mathcal{L}(X_i)$ *for which* $(f_j, \mathcal{L}(X_i), \mathcal{L}(N_j))$ *does not satisfy Condition 19 is contained in a* 3*-dimensional space.*

Proof See Appendix A.6. ■

The condition $\nu''(x_j - f(x_i))f'(x_i) \neq 0$ holds for all x_i if there is no interval where f is constant and the logarithm of the noise density is not linear, for example. In the case of Gaussian variables, the differential equation (7) simplifies. We thus have the following result.

Corollary 22 *If* X_i *and* N_j *follow a Gaussian distribution and* $(f_j, \mathcal{L}(X_i), \mathcal{L}(N_j))$ *does not satisfy Condition 19, then* f_j *is linear.*

Proof See Appendix A.7. ■

Although non-identifiable cases are rare, the question remains when identifiability is violated. Zhang and Hyvärinen (2009) prove that non-identifiable additive noise models necessarily fall into one out of five classes.

Proposition 23 (Zhang and Hyvärinen, 2009) *Consider $X_j = f_j(X_i) + N_j$ with fully supported noise variable N_j that is independent of X_i and three times differentiable function f_j. Let further $\frac{d}{dx_i} f_j(x_i) \frac{d^2}{dx_j^2} \log p_{N_j}(x_j) = 0$ only at finitely many points (x_i, x_j). If there is a backward model, i.e., we can write $X_i = g_i(X_j) + M_i$ with M_i independent of X_j, then one of the following must hold.*

I. X_i is Gaussian, N_j is Gaussian and f_j is linear.

II. X_i is log-mix-lin-exp, N_j is log-mix-lin-exp and f_j is linear.

III. X_i is log-mix-lin-exp, N_j is one-sided asymptotically exponential and f_j is strictly monotonic with $f_j'(x_i) \to 0$ as $x_i \to \infty$ or as $x_i \to -\infty$.

IV. X_i is log-mix-lin-exp, N_j is generalized mixture of two exponentials and f_j is strictly monotonic with $f_j'(x_i) \to 0$ as $x_i \to \infty$ or as $x_i \to -\infty$.

V. X_i is generalized mixture of two exponentials, N_j is two-sided asymptotically exponential and f_j is strictly monotonic with $f_j'(x_i) \to 0$ as $x_i \to \infty$ or as $x_i \to -\infty$.

Precise definitions can be found in Appendix A.8. In particular, we obtain identifiability whenever the function f_j is not injective. Proposition 23 states that belonging to one of these classes is a necessary condition for non-identifiability. We now show sufficiency for two classes. The linear Gaussian case is well-known and easy to prove.

Example 24 *Let $X_2 = aX_1 + N_2$ with independent $N_2 \sim \mathcal{N}(0, \sigma^2)$ and $X_1 \sim \mathcal{N}(0, \tau^2)$. We can then consider all variables in $\mathcal{L}_2$ and project X_1 onto X_2. This leads to an orthogonal decomposition $X_1 = \tilde{a}X_2 + \tilde{N}_1$. Since for jointly Gaussian variables uncorrelatedness implies independence, we obtain a backward additive noise model. Figure 2 (left) shows the joint density and the functions for the forward and backward model.*

We also give an example of a nonidentifiable additive noise model with non-Gaussian distributions; the forward model is described by case II, and the backward model by case IV:

Example 25 *Let $X_2 = aX_1 + b + N_2$ with independent log-mix-lin-exp N_2 and X_1, i.e., we have the log-densities*

$$\xi(x) = \log p_{X_1}(x) = c_1 \exp(c_2 x) + c_3 x + c_4$$

and

$$\nu(n) = \log p_{N_2}(n) = \gamma_1 \exp(\gamma_2 n) + \gamma_3 n + \gamma_4 \,.$$

Then X_2 is a generalized mixture of exponential distributions. If and only if $c_2 = -a\gamma_2$ and $c_3 \neq a\gamma_3$ we obtain a valid backward model $X_1 = \tilde{f}_1(X_2) + \tilde{N}_1$ with log-mix-lin-exp $\tilde{N}_1$. Again, Figure 2 (right) shows the joint distribution over X_1 and X_2 and forward and backward functions.

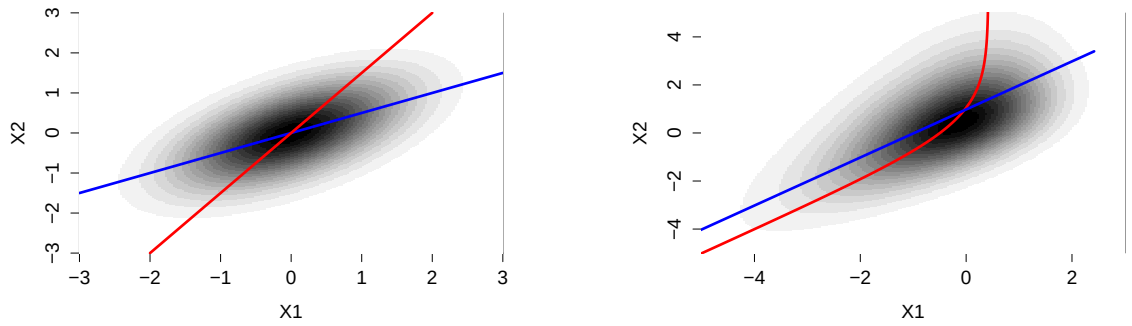


Figure 2: Joint density over X_1 and X_2 for two non-identifiable examples. The left panel shows Example 24 (linear Gaussian case) and the right panel shows Example 25 (the latter plot is based on kernel density estimation). The blue function corresponds to the forward model $X_2 = f_2(X_1) + N_2$, the red function to the backward model $X_1 = \tilde{f}_1(X_2) + \tilde{N}_1$.

Proof See Appendix A.9. ■

Example 25 shows how parameters of function, input and noise distribution have to be "fine-tuned" to yield non-identifiability (Janzing and Steudel, 2010).

It can be shown that bivariate identifiability even holds generically when feedback is allowed (i.e., if both $X \to Y$ *and* $Y \to X$), at least when assuming noise and input distributions to be Gaussian (Mooij et al., 2011).

3.2 From Bivariate to Multivariate Models

It turns out that Condition 19 also suffices to prove identifiability in the multivariate case. Assume we are given p structural equations $X_j = f_j(\mathbf{PA}_j) + N_j$ as in (6). If we fix all arguments of the functions f_j except for one parent and the noise variable, we obtain a bivariate model. One may expect that it suffices to put restrictions like Condition 19 on this triple of function, input and noise distribution. This is not the case.

Example 26 *Consider the following SEM*

$$X_1 = N_1, \quad X_2 = f_2(X_1) + N_2, \quad X_3 = f_3(X_1) + a \cdot X_2 + N_3$$

with $N_1 \sim t_{\nu=3}$, $N_2 \sim \mathcal{N}(0, \sigma_2^2)$ *and* $N_3 \sim \mathcal{N}(0, \sigma_3^2)$, *i.e.,* N_1 *is* t*-distributed with* 3 *degrees of freedom and* N_2 *and* N_3 *are normally distributed.* X_2 *and* X_3 *are non-Gaussian but*

$$X_3 \,|\, _{X_1=x_1} = c + a \cdot X_2 \,|\, _{X_1=x_1} + N_3$$

is a linear Gaussian equation for all x_1. *We can revert this equation and obtain the same joint distribution by an SEM of the form*

$$X_1 = M_1, \quad X_2 = g_2(X_1) + b \cdot X_3 + M_2, \quad X_3 = g_3(X_1) + M_3\,,$$

for some g_1, g_2 and $M_1 \sim t_{\nu=3}$, $M_2 \sim \mathcal{N}(0, \tilde{\sigma}_2^2)$ and $M_3 \sim \mathcal{N}(0, \tilde{\sigma}_3^2)$. Thus, the DAG is not identifiable from the joint distribution.

Instead, we need to put restrictions on conditional distributions.

Definition 27 *Consider an additive noise model* (6) *with p variables. We call this SEM a* restricted additive noise model *if for all $j \in \mathbf{V}$, $i \in \mathbf{PA}_j$ and all sets $\mathbf{S} \subseteq \mathbf{V}$ with $\mathbf{PA}_j \setminus \{i\} \subseteq \mathbf{S} \subseteq \mathbf{ND}_j \setminus \{i, j\}$, there is an $x_{\mathbf{S}}$ with $p_{\mathbf{S}}(x_{\mathbf{S}}) > 0$, s.t.*

$$\Big(f_j(x_{\mathbf{PA}_j \setminus \{i\}}, \underbrace{\cdot}_{X_i}), \mathcal{L}(X_i \,|\, X_{\mathbf{S}} = x_{\mathbf{S}}), \mathcal{L}(N_j)\Big)$$

satisfies Condition 19. Here, the underbrace indicates the input component of f_j for variable X_i. In particular, we require the noise variables to have non-vanishing densities and the functions f_j to be continuous and three times continuously differentiable.

Assuming causal minimality, we can identify the structure of the SEM from the distribution.

Theorem 28 *Let $\mathcal{L}(\mathbf{X}) = \mathcal{L}(X_1, \ldots, X_p)$ be generated by a restricted additive noise model with graph $\mathcal{G}_0$ and let $\mathcal{L}(\mathbf{X})$ satisfy causal minimality with respect to $\mathcal{G}_0$, i.e., the functions f_j are not constant (Proposition 17). Then, $\mathcal{G}_0$ is identifiable from the joint distribution.*

Proof See Appendix A.11. ■

Our proof of Theorem 28 contains a graphical statement that turns out to be a main argument for proving identifiability for Gaussian models with equal error variances (Peters and Bühlmann, 2014). We thus state it explicitly as a proposition.

Proposition 29 *Let $\mathcal{G}$ and $\mathcal{G}'$ be two different DAGs over variables $\mathbf{X}$.*

(i) Assume that $\mathcal{L}(\mathbf{X})$ has a strictly positive density and satisfies the Markov condition and causal minimality with respect to $\mathcal{G}$ and $\mathcal{G}'$. Then there are variables $L, Y \in \mathbf{X}$ such that for the sets $\mathbf{Q} := \mathbf{PA}_L^{\mathcal{G}} \setminus \{Y\}$, $\mathbf{R} := \mathbf{PA}_Y^{\mathcal{G}'} \setminus \{L\}$ and $\mathbf{S} := \mathbf{Q} \cup \mathbf{R}$ we have

- *$Y \to L$ in $\mathcal{G}$ and $L \to Y$ in $\mathcal{G}'$*
- *$\mathbf{S} \subseteq \mathbf{ND}_L^{\mathcal{G}} \setminus \{Y\}$ and $\mathbf{S} \subseteq \mathbf{ND}_Y^{\mathcal{G}'} \setminus \{L\}$*

(ii) In particular, if $\mathcal{L}(\mathbf{X})$ is Markov and faithful with respect to $\mathcal{G}$ and $\mathcal{G}'$ (i.e., both graphs belong to the same Markov equivalence class), there are variables L, Y such that

- *$Y \to L$ in $\mathcal{G}$ and $L \to Y$ in $\mathcal{G}'$*
- *$\mathbf{PA}_L^{\mathcal{G}} \setminus \{Y\} = \mathbf{PA}_Y^{\mathcal{G}'} \setminus \{L\}$*

Proof See Appendix A.12. ■

If the distribution is Markov and faithful with respect to the underlying graph it is known that we can recover the correct Markov equivalence class. Chickering (1995) proves that two graphs within this Markov equivalence class can be transformed into each other by a sequence of so-called covered edge reversals. This result implies part (ii) of the proposition. Part (i) establishes a similar statement when replacing faithfulness by causal minimality.

Although Theorem 28 is stated for additive noise models, it can be seen as an example of a more general principle.

Remark 30 *Theorem 28 is not limited to restricted additive noise models. Whenever we have a restriction like Condition 19 that ensures identifiability in the bivariate case (Theorem 20), the multivariate version (Theorem 28) remains valid. The proof we provide in the appendix stays exactly the same. The algorithms in Section 4, however, use standard regression methods and therefore rely on the additive noise assumption.*

The result can therefore be used to prove identifiability of SEMs that are restricted to discrete additive noise models (Peters et al., 2011a) or post-nonlinear additive noise models (Zhang and Hyvärinen, 2009). In the latter model class we allow a bijective nonlinear distortion: $X_j = g_j\big(f_j(\mathbf{PA}_j) + N_j\big)$. These models allow for more complicated functional relationships but are harder to fit from empirical data than the additive noise models considered in this work.

We now state a specific identifiability result for Gaussian noise that we believe to constitute an important model class for applications. Tamada et al. (2011b) have already used this result for structure learning without giving an identifiability result (see also Tamada et al., 2011a). More recently, Bühlmann et al. (2013) investigate model (8) in a high-dimensional context. A bivariate version of the following corollary can be found as Lemma 6 in Zhang and Hyvärinen (2009).

Corollary 31 *(i) Let $\mathcal{L}(\mathbf{X}) = \mathcal{L}(X_1, \ldots, X_p)$ be generated by an SEM with*

$$X_j = f_j(\mathbf{PA}_j) + N_j\,,$$

with normally distributed noise variables $N_j \sim \mathcal{N}(0, \sigma_j^2)$ and three times differentiable functions f_j that are not linear in any component: denote the parents $\mathbf{PA}_j$ of X_j by $X_{k_1}, \ldots, X_{k_\ell}$, then the function $f_j(x_{k_1}, \ldots, x_{k_{a-1}}, \cdot, x_{k_{a+1}}, \ldots, x_{k_\ell})$ is assumed to be nonlinear for all a and some $x_{k_1}, \ldots, x_{k_{a-1}}, x_{k_{a+1}}, \ldots, x_{k_\ell} \in \mathbb{R}^{\ell-1}$.

(ii) As a special case, let $\mathcal{L}(\mathbf{X}) = \mathcal{L}(X_1, \ldots, X_p)$ be generated by an SEM with

$$X_j = \sum_{k \in \mathbf{PA}_j} f_{j,k}(X_k) + N_j\,, \tag{8}$$

with normally distributed noise variables $N_j \sim \mathcal{N}(0, \sigma_j^2)$ and three times differentiable, nonlinear functions $f_{j,k}$.

In both cases (i) and (ii), we can identify the corresponding graph $\mathcal{G}_0$ from the distribution $\mathcal{L}(\mathbf{X})$. The statements remain true if the noise distributions for source nodes, i.e., nodes with no parents, are allowed to have a non-Gaussian density with full support on the real line $\mathbb{R}$ (the proof remains identical).

Proof See Appendix A.13. ■

Additive noise models as in (8), for which the structural equations are additive in the parents (but the noise does not need to be Gaussian) are called causal additive models (CAMs), see Bühlmann et al. (2013).

Theorem 28 requires the positivity of densities in order to make use of the intersection property of conditional independence. Peters (2014) shows that the intersection property still holds under weaker assumptions and discusses fundamental limits of causal inference when positivity is violated.

3.3 Estimating the Topological Order

We now investigate the case when we drop the assumption of causal minimality. Assume therefore that we are given a distribution $\mathcal{L}(\mathbf{X})$ from an additive noise model with graph $\mathcal{G}_0$. We cannot recover the correct graph $\mathcal{G}_0$ because we can always add edges $i \to j$ or remove edges that "do not have any effect" without changing the distribution. This is formalized by the following lemma. (This lemma may be useful in more general contexts, other than additive noise models, too.)

Lemma 32 *Let $\mathcal{L}(\mathbf{X})$ be generated by an additive noise model with graph $\mathcal{G}_0$.*

(a) For each supergraph $\mathcal{G} \geq \mathcal{G}_0$ there is an additive noise model that leads to the distribution $\mathcal{L}(\mathbf{X})$.

(b) For each subgraph $\mathcal{G} \leq \mathcal{G}_0$ such that $\mathcal{L}(\mathbf{X})$ is Markov with respect to $\mathcal{G}$ there is an additive noise model that leads to the distribution $\mathcal{L}(\mathbf{X})$. Furthermore, there is an additive noise model with unique graph $\mathcal{G}_0^{min} \leq \mathcal{G}_0$ that leads to $\mathcal{L}(\mathbf{X})$ and satisfies causal minimality.

Proof See Appendix A.14. ■

Despite this indeterminacy we can still recover the correct order of the variables. Given a permutation $\pi \in S_p$ on $\{1, \ldots, p\}$ we therefore define the fully connected DAG $\mathcal{G}_\pi^{\text{full}}$ by the DAG that contains all edges $\pi(i) \to \pi(j)$ for $i < j$.

As a direct consequence of Theorem 28 and Lemma 32 we have the following result.

Corollary 33 *Let $\mathcal{L}(\mathbf{X}) = \mathcal{L}(X_1, \ldots, X_p)$ be generated by an additive noise model with graph $\mathcal{G}_0$. Assume that the SEM corresponding to the minimal graph $\mathcal{G}_0^{min}$ defined as in Lemma 32 (b) is a restricted additive noise model. Consider an ordering π and a restricted ANM with corresponding graph $\mathcal{G}_\pi^{full,\, min}$ (Lemma 32 (b)) that generates the distribution $\mathcal{L}(\mathbf{X})$. Theorem 28 implies that $\mathcal{G}_\pi^{full,\, min} = \mathcal{G}_0^{min}$. In this sense the set of true orderings*

$$\Pi^0 := \{\pi \in S_p \mid \mathcal{G}_\pi^{full} \geq \mathcal{G}_0^{min}\}$$

is identifiable from $\mathcal{L}(\mathbf{X})$.

This result is useful, for example, if the search over structures is performed in the space of permutations rather than in the space of DAGs (e.g. Friedman and Koller, 2003; Teyssier and Koller, 2005; Bühlmann et al., 2013).

4. Algorithms

The theoretical results do not imply an algorithm for finitely many data that is either computationally or statistically efficient. In this section we propose an algorithm called RESIT that is based on independence-tests and two simple algorithms that make use of an independence score. We prove correctness of RESIT in the population case.

4.1 Regression with Subsequent Independence Test (RESIT)

In practice, we are given i.i.d. data from the joint distribution and try to estimate the corresponding DAG. The following method is based on the fact that for each node X_i the corresponding noise variable N_i is independent of all non-descendants of X_i. In particular, for each sink node X_i we have that N_i is independent of $\mathbf{X} \setminus \{X_i\}$. We therefore propose an iterative procedure: in each step we identify and disregard a sink node. This is done by regressing each of the remaining variables on all other remaining variables and measuring the independence between the residuals and those other variables. The variable leading to the least dependent residuals is considered the sink node (Algorithm 1, lines $4-13$). This first phase of the procedure yields a topological ordering or a fully connected DAG (see Section 3.3). In the second phase we visit every node and eliminate incoming edges until the residuals are not independent anymore, see Algorithm 1, lines $15-22$.

The procedure can make use of any regression method and dependence measure, in this work we choose the p-value of the HSIC independence test (Gretton et al., 2008) as a dependence measure. Under independence, Gretton et al. (2008) provide an asymptotically correct null distribution for the test statistic times sample size. (We use moment matching to approximate this distribution by a gamma distribution.) Since under dependence the test statistic is guaranteed to converge to a value different from zero, we know that the p-value converges to zero only for dependence. As a regression method we choose linear regression, gam regression (R package `mgcv`) or Gaussian process regression (R package `gptk`).

Algorithm 1 is a slightly modified version of the one proposed in Mooij et al. (2009). In this work, we always want to obtain a graph estimate; we thus consider the node with the least dependent residuals as being the sink node, instead of stopping the search when no independence hypothesis is accepted as in Mooij et al. (2009).

Given that we have infinite data, a consistent non-parametric regression method and a perfect independence test ("independence oracle"), RESIT is correct.

Theorem 34 *Assume $\mathcal{L}(\mathbf{X}) = \mathcal{L}(X_1, \ldots, X_p)$ is generated by a restricted additive noise model with graph $\mathcal{G}_0$ and assume that $\mathcal{L}(\mathbf{X})$ satisfies causal minimality with respect to $\mathcal{G}_0$. Then, RESIT used with a consistent non-parametric regression method and an independence oracle is guaranteed to find the correct graph $\mathcal{G}_0$ from the joint distribution $\mathcal{L}(\mathbf{X})$.*

Proof See Appendix A.15 ■

RESIT performs $\mathcal{O}(p^2)$ independence tests, which is polynomial in the number of nodes. In phase 2 of the algorithm, the removal of superfluous edges costs $\mathcal{O}(p)$. Both the independence test and the variable selection method may scale with the sample size, of course. RESIT's polynomial behavior in p may come as a surprise since problems in Bayesian network learning are often NP-hard (e.g. Chickering, 1996).

Despite this theoretical guarantee, RESIT does not scale well to a high number of nodes. Since we cannot make use of an independence oracle in practice, we have to detect dependence between a random variable and a random vector from finitely many data. The order in which the independence tests are performed (phase 2, line 16 in Algorithm 1) may lead to different results. Also, type one errors in phase 2 lead to superfluous edges in the output of the method. If the significance level of the independence test is independent of the number of variables (in the experiments we choose 5%), this effect may lead to a

Algorithm 1 Regression with subsequent independence test (RESIT)

Input: I.i.d. samples of a p-dimensional distribution on $(X_1, \ldots, X_p)$
$S := \{1, \ldots, p\}, \pi := [\,]$
PHASE 1: Determine topological order.
repeat
  for $k \in S$ **do**
    Regress X_k on $\{X_i\}_{i \in S \setminus \{k\}}$.
    Measure dependence between residuals and $\{X_i\}_{i \in S \setminus \{k\}}$.
  end for
  Let k^* be the k with the weakest dependence.
  $S := S \setminus \{k^*\}$
  $\mathrm{pa}(k^*) := S$
  $\pi := [k^*, \pi]$ (π will be the topological order, its last component being a sink)
until $\#S = 0$
PHASE 2: Remove superfluous edges.
for $k \in \{2, \ldots, p\}$ **do**
  for $\ell \in \mathrm{pa}(\pi(k))$ **do**
    Regress $X_{\pi(k)}$ on $\{X_i\}_{i \in \mathrm{pa}(\pi(k)) \setminus \{\ell\}}$.
    if residuals are independent of $\{X_i\}_{i \in \{\pi(1), \ldots, \pi(k-1)\}}$ **then**
      $\mathrm{pa}(\pi(k)) := \mathrm{pa}(\pi(k)) \setminus \{\ell\}$
    end if
  end for
end for
Output: $(\mathrm{pa}(1), \ldots, \mathrm{pa}(p))$

high structural Hamming distance between true and estimated graph for a large number of variables. Furthermore, we have to perform nonparametric regression with possibly many covariates. For high dimensions, these are both statistically hard problems that require huge sample sizes.

4.2 Independence-Based Score

Searching for sink nodes makes the method described in Section 4.1 inherently asymmetric. Mistakes made in the first iterations propagate through the whole procedure. We therefore investigate the performance of independence-based score methods. Theorem 28 ensures that if the data come from a restricted additive noise model we can fit only one (minimal) structure to the data. In order to estimate the graph structure we can test all possible DAGs and determine which DAG yields the most independent residuals. But even in the limit of infinitely many data we may find more than one DAG satisfying this constraint, some of which may not satisfy causal minimality. We therefore propose to take a penalized independence score

$$\hat{\mathcal{G}} = \operatorname*{argmin}_{\mathcal{G}} \sum_{i=1}^{p} \mathrm{DM}(\mathrm{res}_i^{\mathcal{G},\mathrm{RM}}, \mathrm{res}_{-i}^{\mathcal{G},\mathrm{RM}}) + \lambda \,\#(\mathrm{edges})\,. \tag{9}$$

Here, res_i are the residuals of node X_i, when regressing it on its parents; they depend on the graph $\mathcal{G}$ and on the regression method RM. We denote the residuals of all variables except for X_i by res_{-i} and DM denotes a measure of dependence. Note that variables $\mathbf{N} = (N_1, \ldots, N_p)$ are jointly independent if and only if each N_i is independent of $\mathbf{N} \setminus \{N_i\}$, $i = 1, \ldots, p$. We do not prove (or claim) that the minimizer of (9) is a consistent estimator for the correct DAG; we expect this to depend on the choice of DM and RM and λ.

As dependence measure we use minus the logarithm of the p-values of an independence test based on the Hilbert Schmidt Independence Criterion HSIC (Gretton et al., 2008). As regression methods we use linear regression, generalized additive models (gam) or Gaussian process regression. For the regularization parameter λ we propose to use $\log(0.05) - \log(0.01)$. This is a heuristic choice based on the following idea: we only allow for an additional edge if it allows the p-value to increase from 0.01 to 0.05 or, equivalently, by a factor of five. In practice, p-values estimated by bootstrap techniques or p-values that are smaller than computer precision can become zero and the logarithm becomes minus infinity. We therefore always consider the maximum of the computed p-value and 10^{-350}. Although our choices seem to work well in practice, we do not claim that they are optimal.

4.2.1 Brute-Force

For small graphs, we can solve equation (9) by computing the score for all possible DAGs and choose the DAG with the lowest score. Since the number of DAGs grows hyper-exponentially in the number of nodes, this method becomes quickly computationally intractable; e.g., for $p = 7$, there are $1,138,779,265$ DAGs (OEIS Foundation Inc., 2011). Nevertheless, we use this algorithm up to $p = 4$ for comparison.

4.2.2 Greedy DAG Search (GDS)

A strategy to circumvent the computational complexity of equation (9) is to use greedy search algorithms (e.g., Chickering, 2002). At each step we are given a current DAG and score neighboring DAGs that are arranged in some order (see below). Here, all DAGs are called neighbors that can be reached by an edge reversal, addition or removal. Whenever a DAG has a better score than the current DAG, we stop scoring other neighbors and exchange the latter by the former. To obtain "better" steps, in each step we consider at least p neighbors. In order to reduce the running time of the algorithm, we do not score neighboring DAGs in a completely random order but start by adding or removing edges into nodes whose residuals are highly dependent on the other residuals instead. More precisely, we are randomly sorting the nodes, choosing each node one by one with a probability proportional to the reciprocal dependence measure of its residuals. If all neighboring DAGs have a worse score than the current graph G, we nevertheless consider the best neighbor H. If H has a neighbor with a better score than G, we continue with this graph. Otherwise we stop and output G as the optimal graph. This is a simple version of tabu search (e.g. Koller and Friedman, 2009) that is used to avoid local optima. This method is not guaranteed to find the best scoring graph.

Code for the proposed methods is provided on the first and second author's homepage.

5. Experiments

The following subsections report some empirical performance of the described methods.

5.1 Experiments on Synthetic Data

For varying sample size n and number of variables p we compare the described methods. Given a value of p, we randomly choose an ordering of the variables with respect to the uniform distribution and include each of the $p(p-1)/2$ possible edges with a probability of $2/(p-1)$. This results in an expected number of p edges and can be considered as a (modestly) sparse setting. For a linear and a nonlinear setting we report the average structural Hamming distance (Acid and de Campos, 2003; Tsamardinos et al., 2006) to the true directed acyclic graph and to the true completed partially directed acyclic graph over 100 simulations. The structural Hamming distance (SHD) between two partially directed acyclic graphs counts how many edge types do not coincide. Estimating a non-edge or a directed edge instead of an undirected edge, for example, contributes an error of one to the overall distance. We also report analogous results for the structural intervention distance (SID), which has recently been proposed (Peters and Bühlmann, 2013). Given the estimated graph we can infer the intervention distribution $p(X_j \,|\, do(X_i = x_i))$ by parent adjustment (1). We call a pair of nodes (X_i, X_j) *good* if the intervention distribution $p(X_j \,|\, do(X_i = x_i))$ inferred from the estimated DAG using (1) coincides with the intervention distribution inferred from the correct DAG for all observational distributions $\mathcal{L}(\mathbf{X})$. The SID counts the number of pairs that are not good. Some methods output a Markov equivalence class instead of a single DAG. Different DAGs within such a class lead to different intervention distribution and thus different SIDs. In that case, we therefore provide the smallest and largest SID attained by members within the Markov equivalence class. As the SHD, the SID is a purely structural measure that is independent of any distribution. The rationale behind the new measure is that a reversed edge in the estimated DAG leads to more false causal effects than an additional edge does. The SHD, however, weights both errors equally.

We compare the greedy DAG search (GDS), brute-force (BF), regression with subsequent independence test (RESIT), linear non-Gaussian additive models (LINGAM), the PC algorithm (PC) with partial correlation and significance level 0.01 and greedy equivalence search (GES), see Sections 4.2.2, 4.2.1, 4.1, 2.3, 2.1 and 2.2, respectively. We also compare them with the conservative PC algorithm (CPC), suggested by Ramsey et al. (2006), and random guessing (RAND). The latter chooses a random DAG with edge inclusion probability uniformly chosen between zero and one. Its estimate does not depend on the data.

5.1.1 Linear Structural Equation Models

We first consider a linear setting as in equation (4), where the coefficients β_{jk} are uniformly chosen from $[-2, -0.1] \cup [0.1, 2]$ and the noise variables N_j are independent and distributed according to $K_j \cdot \text{sign}(M_j) \cdot |M_j|^{\alpha_j}$ with $M_j \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$, $K_j \overset{\text{iid}}{\sim} \mathcal{U}([0.1, 0.5])$ and $\alpha_j \overset{\text{iid}}{\sim} \mathcal{U}([2,4])$. The top box plot in Figure 3 compares the SHD of the estimated structure to the correct DAG for $p = 4$ and $n = 100$. All methods make use of the linear structure of the data, e.g., by performing linear regression. The brute-force method performs best, which indicates that the score function in equation (9) is a sensible choice for small graphs. Greedy DAG

search performs almost equally well, it does not encounter many local optima in this setting. The constraint-based methods and greedy equivalent search perform worse. Comparing SID leads to the same conclusion (Figure 3, bottom).

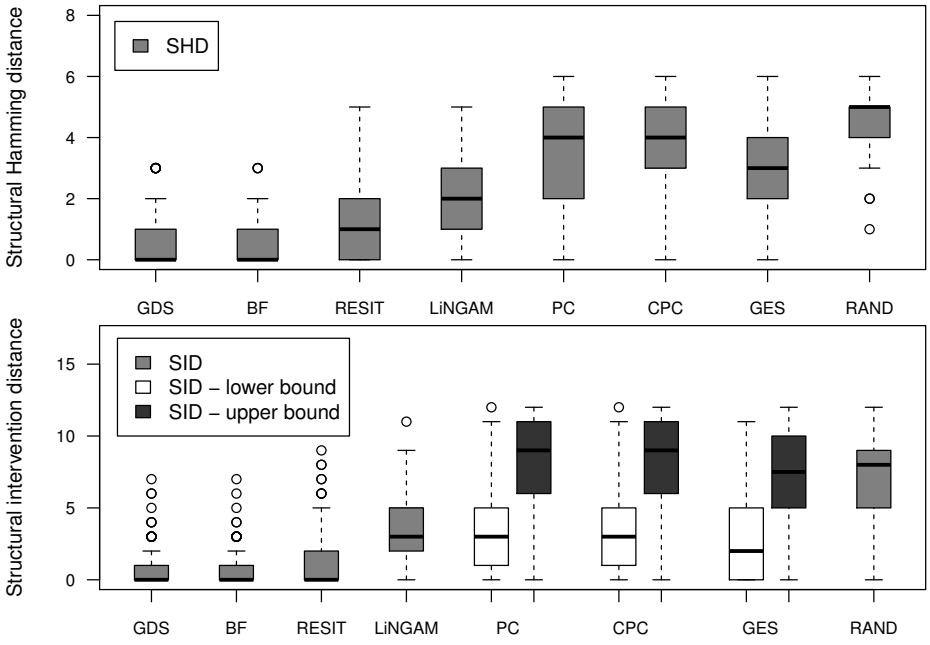


Figure 3: Box plots of the SHD between the estimated structure (either DAG or CPDAG) and the correct DAG for $p = 4$ and $n = 100$ for linear non-Gaussian SEMs (top). The SID is computed between the correct DAG and the estimated DAG (bottom). Some methods estimate only the Markov equivalence class. We then compute the SID to the "best" and to the "worst" DAG within the equivalence class; therefore a lower and an upper bound are shown.

Tables 1 and 2 provide summaries for $p \in \{4, 15\}$ and $n \in \{100, 500\}$. We additionally show distances of the estimated CPDAGs to the true CPDAGs. Therefore, if methods output a DAG instead of a CPDAG, this DAG is transformed into the CPDAG of the corresponding Markov equivalence class. For $p = 4$ and $n = 500$, GDS and brute force find almost always the correct graph (86 and 90 out of 100). RESIT and LiNGAM still perform much better than the PC methods and GES. For $p = 15$, the performance of RESIT (and GES) in relation to the other methods seems to be better when evaluating SID compared to evaluating the SHD. This indicates that the pruning (and penalization of the number of edges) does not work perfectly. Especially for RESIT with high sample size and fixed significance level, making mistakes in phase 1 leads to many edges that cannot be removed in phase 2 (and thus a large SHD). The brute-force method is not applicable to $p = 15$.

Table 1: Linear SEMs: SHD between the estimated structure and the correct DAG and SHD between the estimated CPDAG to the correct CPDAG; for both the average and the standard deviation over 100 experiments are shown (best averages are highlighted).

	GDS	BF	RESIT	LiNGAM	PC	CPC	GES	RAND
	$p = 4, n = 100$							
DAG	0.7 ± 0.9	0.6 ± 0.8	1.2 ± 1.3	1.9 ± 1.2	3.5 ± 1.5	3.6 ± 1.4	3.1 ± 1.7	4.4 ± 1.0
CPDAG	1.1 ± 1.5	0.9 ± 1.4	1.5 ± 1.7	2.4 ± 1.5	2.4 ± 1.7	2.3 ± 1.6	2.0 ± 2.0	4.3 ± 1.4
	$p = 4, n = 500$							
DAG	0.2 ± 0.6	0.1 ± 0.3	0.6 ± 0.8	0.5 ± 0.8	3.1 ± 1.4	3.2 ± 1.4	2.9 ± 1.6	4.1 ± 1.2
CPDAG	0.3 ± 0.9	0.2 ± 0.5	0.9 ± 1.3	0.8 ± 1.2	1.9 ± 1.8	1.6 ± 1.7	1.6 ± 1.9	3.9 ± 1.4
	$p = 15, n = 100$							
DAG	12.2 ± 5.3	–	25.2 ± 8.3	11.1 ± 3.7	13.0 ± 3.6	13.7 ± 3.7	12.7 ± 4.2	57.4 ± 26.4
CPDAG	13.2 ± 5.4	–	27.0 ± 8.5	12.4 ± 3.9	10.7 ± 3.5	10.8 ± 3.8	12.4 ± 4.9	58.5 ± 27.1
	$p = 15, n = 500$							
DAG	6.1 ± 6.4	–	51.2 ± 17.8	3.4 ± 2.8	10.2 ± 3.8	10.8 ± 4.2	8.7 ± 4.6	57.6 ± 24.2
CPDAG	6.8 ± 6.9	–	54.5 ± 18.5	4.5 ± 3.8	8.2 ± 4.6	7.5 ± 4.4	7.1 ± 5.6	58.9 ± 25.0

Table 2: Linear SEMs: SID to the correct DAG; the table shows average and standard deviation over 100 experiments.

GDS	BF	RESIT	LiNGAM	PC	CPC	GES	RAND
$p = 4, n = 100$							
1.0 ± 1.5	0.8 ± 1.4	1.5 ± 2.2	3.3 ± 2.1	3.4 ± 2.9 8.0 ± 3.2	3.2 ± 2.7 8.5 ± 3.2	2.9 ± 3.3 7.2 ± 3.5	7.0 ± 2.8
$p = 4, n = 500$							
0.2 ± 0.7	0.1 ± 0.4	0.3 ± 1.0	0.9 ± 1.4	2.8 ± 3.1 7.4 ± 3.4	2.3 ± 2.7 7.6 ± 3.3	2.1 ± 2.9 6.9 ± 3.6	6.3 ± 2.8
$p = 15, n = 100$							
32.3 ± 24.1	–	35.3 ± 21.2	45.1 ± 24.1	36.5 ± 21.3 63.7 ± 30.3	32.5 ± 20.2 66.4 ± 31.5	26.5 ± 18.3 37.6 ± 20.6	55.6 ± 27.1
$p = 15, n = 500$							
12.6 ± 16.3	–	18.1 ± 13.8	14.2 ± 14.6	33.6 ± 29.5 55.0 ± 32.9	23.2 ± 19.8 55.6 ± 32.4	18.1 ± 21.4 31.6 ± 22.2	57.5 ± 34.1

5.1.2 Nonlinear Structural Equation Models

We also sample data from nonlinear SEMs. We choose an additive structure as in equation (8) and sample the functions from a Gaussian process with bandwidth one. The noise variables N_j are independent and normally distributed with a uniformly chosen variance. Tables 3 and 4 show summaries for $p \in \{4, 15\}$ and $n \in \{100, 500\}$. We cannot run the brute-force method on data sets with $p = 15$. For $p = 4$, we have a similar situation as in Figure 3 with GDS and the BF method outperforming all others (RESIT performing a bit worse). Remarkably, for $p = 15$ and $n = 100$, a lot of the methods do not perform much better than random guessing when comparing the SID. The estimated CPDAG of the constraint-based methods can have very different lower and upper bounds for SID. This

Table 3: Nonlinear SEMs: SHD between the estimated structure and the correct DAG and SHD between the estimated CPDAG to the correct CPDAG; for both the average and the standard deviation over 100 experiments are shown.

	GDS	BF	RESIT	LiNGAM	PC	CPC	GES	RAND
	$p = 4, n = 100$							
DAG	1.5 ± 1.4	1.0 ± 1.0	1.7 ± 1.3	3.5 ± 1.2	3.5 ± 1.5	3.8 ± 1.4	3.5 ± 1.3	4.0 ± 1.3
CPDAG	1.7 ± 1.7	1.2 ± 1.4	2.0 ± 1.6	3.0 ± 1.4	2.9 ± 1.5	2.7 ± 1.4	3.4 ± 1.7	3.9 ± 1.4
	$p = 4, n = 500$							
DAG	0.5 ± 0.9	0.3 ± 0.5	0.8 ± 0.9	3.7 ± 1.2	3.5 ± 1.5	3.8 ± 1.5	3.3 ± 1.5	4.1 ± 1.2
CPDAG	0.6 ± 1.1	0.6 ± 1.0	1.0 ± 1.3	3.0 ± 1.7	3.1 ± 1.9	2.8 ± 1.8	3.4 ± 1.9	3.8 ± 1.6
	$p = 15, n = 100$							
DAG	14.3 ± 4.9	–	15.4 ± 5.7	15.4 ± 3.6	14.2 ± 3.5	15.5 ± 3.6	24.8 ± 6.3	56.8 ± 24.1
CPDAG	15.1 ± 5.4	–	16.5 ± 5.9	15.3 ± 4.0	13.3 ± 3.6	13.3 ± 4.0	26.4 ± 6.5	58.0 ± 24.7
	$p = 15, n = 500$							
DAG	13.0 ± 8.4	–	10.1 ± 5.7	21.4 ± 6.9	13.9 ± 4.5	15.1 ± 4.8	26.8 ± 8.5	56.1 ± 26.8
CPDAG	14.2 ± 9.2	–	11.3 ± 6.3	21.1 ± 7.3	13.7 ± 4.9	13.4 ± 5.1	28.6 ± 8.8	57.0 ± 27.3

Table 4: Nonlinear SEMs: SID to the correct DAG; the table shows average and standard deviation over 100 experiments.

GDS	BF	RESIT	LiNGAM	PC	CPC	GES	RAND
$p = 4, n = 100$							
2.0 ± 2.5	1.4 ± 1.7	2.0 ± 1.9	8.2 ± 2.8	4.7 ± 3.2 7.8 ± 3.4	4.3 ± 2.7 8.5 ± 3.2	4.7 ± 3.2 7.2 ± 3.2	6.3 ± 3.1
$p = 4, n = 500$							
0.6 ± 1.8	0.2 ± 0.8	0.9 ± 1.3	8.0 ± 2.8	4.3 ± 3.7 7.3 ± 3.2	3.7 ± 3.3 8.1 ± 3.2	3.6 ± 3.0 6.5 ± 3.3	6.6 ± 3.4
$p = 15, n = 100$							
50.6 ± 25.3	–	44.4 ± 23.9	65.0 ± 28.3	49.7 ± 24.6 68.6 ± 31.5	40.4 ± 21.6 76.7 ± 32.8	49.0 ± 27.3 53.6 ± 28.9	60.0 ± 29.9
$p = 15, n = 500$							
35.9 ± 26.8	–	24.6 ± 18.6	67.3 ± 28.1	49.9 ± 29.0 60.3 ± 31.0	36.4 ± 22.1 70.3 ± 34.6	40.2 ± 23.3 44.6 ± 24.0	58.9 ± 27.8

means that some DAGs within the equivalence class perform much better than others. (The methods do not propose any particular DAG, they treat all DAGs within the class equally.)

Figure 4 shows box plots of SHD and SID for the special case $p = 15$ and $n = 500$. This time, RESIT perform slightly better than all other methods. It makes use of the nonlinearity of the structural equations. Again, the high SHD for GES indicates that the estimate probably contains too many edges (since its SID is better than the one for the PC methods).

In conclusion, for $p = 4$, the brute force method works best for both linear and nonlinear data. Roughly speaking, for $p = 15$, LiNGAM and GDS work best in the linear non-Gaussian setting and RESIT works best for nonlinear data. If one does not know whether

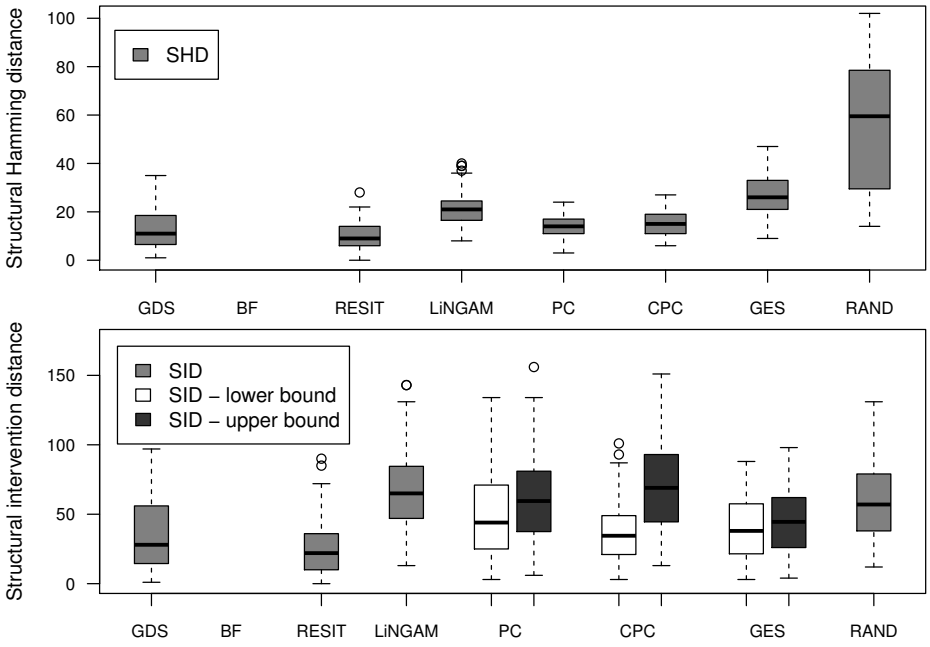


Figure 4: Similar to Figure 3: box plots of the SHD between estimated structure and correct DAG (top) and box plots of the SID to the correct DAG (bottom) for $p = 15$, $n = 500$ and nonlinear Gaussian SEMs.

the data are linear or nonlinear, GDS provides an alternative that works reasonably well in both settings.

5.2 Altitude, Temperature and Duration of Sunshine

We consider recordings of average temperature T, average duration of sunshine DS and the altitude A at 349 German weather stations (Deutscher Wetterdienst, 2008). Figure 5 shows scatter plots of all pairs. LiNGAM estimates $T \to A$, PC and CPC estimate $T \to A \leftarrow DS$,

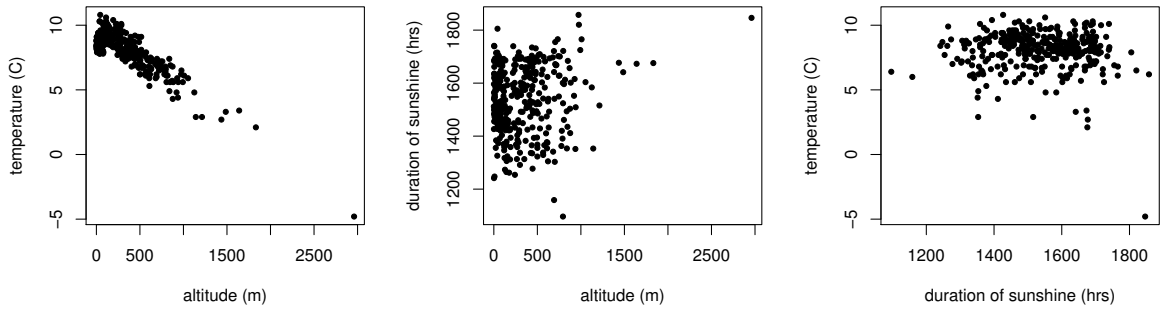


Figure 5: Scatter plots of the three pairs, altitude, temperature and duration of sunshine.

GES estimates a fully connected DAG. The brute-force estimate with linear regression obtains a score of 103.6. Since we are taking the logarithm to base 10 in equation (9), we see that the model does not fit the data well. More sensible seems the gam regression, for which both GDS and brute-force output the DAG $T \leftarrow A \rightarrow DS$ and $T \rightarrow DS$, which receives a score of 5.9. Also RESIT outputs this DAG. Although there might be a feedback between duration of sunshine and temperature through the generation of clouds, we believe that the link from sunshine to temperature should be stronger. In fact, the corresponding DAG $T \leftarrow A \rightarrow DS$ with $T \leftarrow DS$ receives the second best score. Furthermore, these data may be confounded by geographical location. Together with the possible feedback loop and a possible deviation from additive noise models this might be the reason why we do not obtain clear independence of the residuals: the HSIC between the residuals of temperature and the two others leads to a p-value of 0.012 (the other two p-values are both about 0.12). In practice, we often expect some violations of the model assumptions. This example, however, indicates that it may still possible to obtain reasonable estimates of the underlying causal structure if the violations are not too strong.

5.3 Cause-Effect Pairs

We have tested the performance of additive noise models on a collection of various cause-effect pairs, an extended version of the "Cause-effect pairs" data set described in Mooij and Janzing (2010). We used version 0.8 of this data set, which consists of observations of 86 different pairs of variables from various domains. The task is to infer which variable is the cause and which variable the effect, for each of the pairs. For example, one of the pairs consists of 349 measurements of altitude and temperature taken at different weather stations in Germany (Deutscher Wetterdienst, 2008), the same data as considered in the previous subsection. It should be obvious that here the altitude is the cause, and the temperature is the effect. The complete data set and a more detailed description of each pair can be obtained from `http://webdav.tuebingen.mpg.de/cause-effect`.

For each pair of variables (X_i, Y_i), with $i = 1, \ldots, 86$, we test the two possible additive noise models that correspond with the two different possible causal directions, $X_i \rightarrow Y_i$ and $Y_i \rightarrow X_i$. For both directions, we estimate the functional relationship by performing Gaussian Process regression using the GPML toolbox (Rasmussen and Nickisch, 2010). We use the expected value of the Gaussian Process given the observations as an estimate of the functional dependence between the cause and the effect. The goodness-of-fit is then evaluated by testing independence of the residuals and the inputs. Here, we use the HSIC as an independence test and approximate the null distribution with a gamma distribution in order to obtain p-values (Gretton et al., 2005). We thus obtain two p-values for each pair, one for each possible causal direction (where a high p-value corresponds to not rejecting independence, i.e., not rejecting the causal model). We then rank the pairs according to the highest of the two p-values of the pair. Using this ranking, we can make decisions for only a subset of the pairs, starting with the pair for which the highest of the two p-values is the largest among all pairs (we say these pairs have a high rank). In this way we trade off accuracy, i.e., percentage of correct decisions, versus the amount of decisions taken.

Five of the pairs have multivariate X_i or Y_i, and we did not include those in the analysis for convenience. Furthermore, not all the pairs are independent; for example, life expectancy

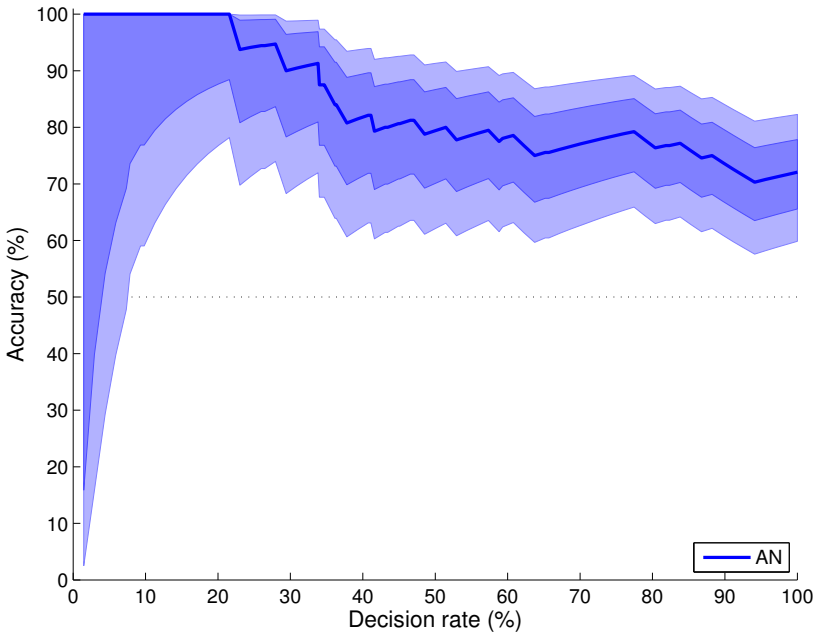


Figure 6: Results of the additive noise method on version 0.8 of the cause-effect pairs data set. After weighting, effectively 68 out of 86 pairs remained. The plot shows estimated accuracy, 68% and 95% confidence intervals for each decision rate.

versus latitude occurs more than once, but measurements were done in different years and for different gender. We therefore assigned weights to the cause-effect pairs to compensate for this when calculating the accuracy and decision rate. For example, the pair life expectancy versus latitude appears eight times (for different combinations of gender and year), hence each of these pairs is weighted down with the factor 1/8; on the other hand, the pair altitude vs. temperature at weather stations occurs only once, and therefore gets weight 1. Denoting the weight of each pair with w_i, the "effective" number of pairs becomes $\sum_{i=1}^{86} w_i = 68$. The five pairs with multivariate X_i or Y_i were given zero weight. If the set of highest-ranked pairs is denoted $\mathcal{I}$, and the set of correct decisions is denoted $\mathcal{C}$, then the *accuracy* (fraction of correct decisions) and and the *decision rate* (fraction of decisions taken) are defined as

$$\text{accuracy} = \frac{\sum_{i\in\mathcal{I}\cap\mathcal{C}} w_i}{\sum_{i\in\mathcal{I}} w_i}, \qquad \text{decision rate} = \frac{\sum_{i\in\mathcal{I}} w_i}{\sum_{i=1}^{86} w_i}.$$

The results are plotted in Figure 6. It shows the accuracy (dark blue line) as a function of the decision rate, together with confidence intervals (light blue regions). The amount of cause-effect pairs from which the accuracy can be estimated decreases proportionally to the decision rate; the accuracies reported for low decision rates therefore have higher uncertainty than the accuracies reported for high decision rates. For each decision rate, we have plotted the 68% and 95% confidence intervals for the estimated success probability assuming a binomial distribution using the Clopper-Pearson method. If for a given decision rate, the 95% confidence region lies above the line at 50%, the method performs significantly

better than random guessing (for that decision rate). For example, if we take a decision for all pairs, $72 \pm 6\%$ of the decisions are correct, significantly more than random guessing. If we only take the 20% most confident decisions, all of them are correct, again significantly more than random guessing.

6. Discussion and Future Work

Apart from a few exceptions we can identify the directed acyclic graph from a bivariate distribution that has been generated by a structural equation model with continuous additive noise. Such an identifiability in the bivariate case generalizes under mild assumptions to identifiability in the multivariate case (i.e., graphs with more than two variables). This can be beneficial for the field of causal inference: if the true data generating process can be represented by a restricted structural equation model like additive noise models, the causal graph can be inferred from the joint distribution. We believe that formulating the problem using structural equation models rather than graphical models made it easier to state and exploit the assumption of additive noise. While the language of graphical models allow us to define some notion connecting a graph to the distribution (e.g., faithfulness), SEMs allow us to impose specific restrictions on the possible functional relationships between nodes and its children. This is closer in spirit to a machine learning approach where properties of function classes play a crucial role in the estimation. Both artificial and real data sets indicate that methods based on restricted structural equation models can outperform traditional constraint-based methods.

We have proposed two methods for estimating the graph from finitely many data. RESIT iteratively identifies sink nodes. Another method optimizes a score that reflects the independence of residuals. Although the score seems to be suitable to detect the correct graph structure, it remains unclear how to find the best scoring DAG when an exhaustive search is infeasible. We investigated the possibility to search this space by greedily choosing best-scoring neighbors (GDS). Multiple random initializations may decrease the chance that the greedy DAG search gets stuck in local optima by the additional cost of computational complexity. We further believe that the proposed score may benefit from an extended version of HSIC that is able to estimate mutual independence instead of pairwise independence. Recently, Nowzohour and Bühlmann (2013) have suggested a penalized likelihood based score for bivariate models. They estimate the noise distribution and use the BIC for penalization. In principle this idea can again be combined with a brute-force search as in Section 4.2.1 or a greedy DAG search as in Section 4.2.2.

Making the methods applicable to larger graphs ($p > 20$) remains a major challenge. Also, studying the statistical properties of the methods (for example, establishing consistency) is an important task for future research.

Acknowledgements

We thank Peter Bühlmann and Markus Kalisch for helpful discussions and Patrik Hoyer for the collaboration initiating the idea of nonlinear additive noise models (Hoyer et al., 2009). The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-

2013) under REA grant agreement no 326496. JM was supported by NWO, the Netherlands Organization for Scientific Research (VENI grant 639.031.036). We thank the anonymous reviewers for their insightful comments.

Appendix A. Proofs

We now provide all proofs that have been omitted in the main text.

A.1 Proof of Proposition 4

Proof "if": Assume that causal minimality is not satisfied. Then, there is an X_j and a $Y \in \mathbf{PA}_j^{\mathcal{G}}$, such that $\mathcal{L}(\mathbf{X})$ is also Markov with respect to the graph obtained when removing the edge $Y \to X_j$ from $\mathcal{G}$.
"only if": If $\mathcal{L}(\mathbf{X})$ has a density, the Markov condition is equivalent to the Markov factorization (Lauritzen, 1996, Theorem 3.27). Assume that $Y \in \mathbf{PA}_j^{\mathcal{G}}$ and $X_j \perp\!\!\!\perp Y \,|\, \mathbf{PA}_j^{\mathcal{G}} \setminus \{Y\}$. Then $P(\mathbf{X}) = P(X_j|\mathbf{PA}_j^{\mathcal{G}} \setminus \{Y\}) \prod_{k \neq j} P(X_k|\mathbf{PA}_k^{\mathcal{G}})$, which implies that $\mathcal{L}(\mathbf{X})$ is Markov w.r.t. $\mathcal{G}$ without $Y \to X_j$. ■

A.2 Proof of Proposition 7

Proof We will prove that for all $\mathcal{G}_1$ and $\mathcal{G}_2$ in $\mathbb{G}$ there is DAG $\mathcal{G} \in \mathbb{G}$ such that $\mathcal{G} \leq \mathcal{G}_1$ and $\mathcal{G} \leq \mathcal{G}_2$. This implies the existence of a least element since the set $\mathbb{G}$ is finite. Consider any node X_i and denote the $\mathcal{G}_1$-parents by $X_{j_1}, \ldots, X_{j_r}, X_{k_{r+1}}, \ldots, X_{k_{r+s}}$ and the $\mathcal{G}_2$-parents by $X_{j_1}, \ldots, X_{j_r}, X_{\ell_{r+1}}, \ldots, X_{\ell_{r+t}}$, such that $\{k_{r+1}, \ldots, k_{r+s}\}$ and $\{\ell_{r+1}, \ldots, \ell_{r+t}\}$ are disjoint sets. Here, $X_{j_1}, \ldots, X_{j_r}$ are the joint parents in $\mathcal{G}_1$ and $\mathcal{G}_2$. We have for all $x_{j_1}, \ldots, x_{j_r}$, $x_{k_{r+1}}, \ldots, x_{k_{r+s}}$ and $x_{\ell_{r+1}}, \ldots, x_{\ell_{r+t}}$ (at which the density p is strictly positive) that

$$\begin{aligned}
&p(X_i \,|\, X_{j_1} = x_{j_1}, \ldots, X_{j_r} = x_{j_r}, X_{k_{r+1}} = x_{k_{r+1}}, \ldots, X_{k_{r+s}} = x_{k_{r+s}}) \\
&= p\big(X_i \,|\, do(X_{j_1} = x_{j_1}, \ldots, X_{j_r} = x_{j_r}, X_{k_{r+1}} = x_{k_{r+1}}, \ldots, X_{k_{r+s}} = x_{k_{r+s}}, \\
&\qquad\qquad X_{\ell_{r+1}} = x_{\ell_{r+1}}, \ldots, X_{\ell_{r+t}} = x_{\ell_{r+t}})\big) \\
&= p(X_i \,|\, X_{j_1} = x_{j_1}, \ldots, X_{j_r} = x_{j_r}, X_{\ell_{r+1}} = x_{\ell_{r+1}}, \ldots, X_{\ell_{r+t}} = x_{\ell_{r+t}}) =: (*)\,.
\end{aligned}$$

This implies

$$(*) = p(X_i \,|\, X_{j_1} = x_{j_1}, \ldots, X_{j_r} = x_{j_r})\,.$$

Set the variables $X_{j_1}, \ldots, X_{j_r}$ to be the $\mathcal{G}$-parents of node X_i and repeat for all nodes X_i. The distribution $\mathcal{L}(\mathbf{X})$ is Markov w.r.t. graph $\mathcal{G}$ by its construction. Note that all proper subgraphs of a true causal DAG with respect to which $\mathcal{L}(\mathbf{X})$ is Markov are again true causal DAGs. This proves the statement about causal minimality. ■

A.3 Proof of Proposition 9

Proof Let $N_1, \cdots, N_p$ be independent and uniformly distributed between 0 and 1. We then define $X_j = f_j(\mathbf{PA}_j, N_j)$ with

$$f_j(x_{\mathbf{PA}_j}, n_j) = F^{-1}_{X_j|\mathbf{PA}_j = x_{\mathbf{PA}_j}}(n_j)\,,$$

where $F^{-1}_{X_j|\mathbf{PA}_j = x_{\mathbf{PA}_j}}$ is the inverse cdf of X_j given $\mathbf{PA}_j = x_{\mathbf{PA}_j}$. ■

A.4 Proof of Proposition 17

Proof Assume causal minimality is not satisfied. We can then find a j and $i \in \mathbf{PA}_j$ with $X_j = f_j(X_{\mathbf{PA}_j \setminus \{i\}}, X_i) + N_j$ that does not depend on X_i if we condition on all other parents $\mathbf{PA}_j \setminus \{i\}$ (Proposition 4). Let us denote $\mathbf{PA}_j \setminus \{X_i\}$ by X_A. For the function f_j it follows that $f_j(x_A, x_i) = c_{x_A}$ for $\mathcal{L}(X_A, X_i)$-almost all (x_A, x_i). Indeed, assume without loss of generality that $\mathbf{E}N_j = 0$, take the mean of $X_j \,|\, \mathbf{PA}_j^{\mathcal{G}_0} = (x_A, x_i)$ and use e.g. (2b) from Dawid (1979). The continuity of f_j implies that f_j is constant in its last argument.

The converse statement follows from Proposition 4, too. ■

A.5 Proof of Theorem 20

Proof To simplify notation we write $X := X_i$ and $Y := X_j$ (see Definition 18). If $\mathcal{G}_0$ is the empty graph, $X \perp\!\!\!\perp Y$. On the other hand, if the graph is not empty, $X \perp\!\!\!\perp Y$ would be a violation of causal minimality. We can therefore now assume that the graph is not empty and $X \not\perp\!\!\!\perp Y$. Let us assume that the graph is not identifiable and we have

$$p_n(y - f(x))p_x(x) = p(x, y) = p_{\tilde{n}}(x - g(y))p_y(y)\,. \tag{10}$$

Set

$$\pi(x, y) := \log p(x, y) = \nu(y - f(x)) + \xi(x)\,, \tag{11}$$

and $\tilde{\nu} := \log p_{\tilde{n}}$, $\eta := \log p_y$. From the r.h.s. of Equation (10) we find $\pi(x, y) = \tilde{\nu}(x - g(y)) + \eta(y)$, implying

$$\frac{\partial^2 \pi}{\partial x \partial y} = -\tilde{\nu}''(x - g(y))g'(y) \quad \text{and} \quad \frac{\partial^2 \pi}{\partial x^2} = \tilde{\nu}''(x - g(y))\,.$$

We conclude

$$\frac{\partial}{\partial x}\left(\frac{\partial^2 \pi / \partial x^2}{\partial^2 \pi / (\partial x \partial y)}\right) = 0\,. \tag{12}$$

Using Equation (11) we obtain

$$\frac{\partial^2 \pi}{\partial x \partial y} = -\nu''(y - f(x))f'(x)\,, \tag{13}$$

and

$$\frac{\partial^2 \pi}{\partial x^2} = \frac{\partial}{\partial x}\left(-\nu'(y - f(x))f'(x) + \xi'(x)\right) = \nu''(f')^2 - \nu' f'' + \xi''\,, \tag{14}$$

where we have dropped the arguments for convenience. Combining Equations (13) and (14) yields

$$\frac{\partial}{\partial x}\left(\frac{\frac{\partial^2\pi}{\partial x^2}}{\frac{\partial^2\pi}{\partial x\partial y}}\right) = -2f'' + \frac{\nu' f'''}{\nu'' f'} - \xi'''\frac{1}{\nu'' f'} + \frac{\nu'\nu''' f''}{(\nu'')^2} - \frac{\nu'(f'')^2}{\nu''(f')^2} - \xi''\frac{\nu'''}{(\nu'')^2} + \xi''\frac{f''}{\nu''(f')^2}\,.$$

Due to equation (12) this expression must vanish and we obtain the differential equation (7)

$$\xi''' = \xi''\left(-\frac{\nu''' f'}{\nu''} + \frac{f''}{f'}\right) - 2\nu'' f'' f' + \nu' f''' + \frac{\nu'\nu''' f'' f'}{\nu''} - \frac{\nu'(f'')^2}{f'}$$

by term reordering. This contradicts the assumption that the distribution is generated by an identifiable bivariate additive noise model, see Condition 19. ■

A.6 Proof of Proposition 21

Proof Let the notation be as in Theorem 20 and let y be fixed such that $\nu''(y-f(x))f'(x) \neq 0$ holds for all but countably many x. Given f, ν, we obtain a linear inhomogeneous differential equation (DE) for ξ:

$$\xi'''(x) = \xi''(x)G(x,y) + H(x,y)\,, \tag{15}$$

where G and H are defined by

$$G := -\frac{\nu''' f'}{\nu''} + \frac{f''}{f'}$$

and

$$H := -2\nu'' f'' f' + \nu' f''' + \frac{\nu'\nu''' f'' f'}{\nu''} - \frac{\nu'(f'')^2}{f'}\,,$$

see proof of Theorem 20. Setting $z := \xi''$ we have $z'(x) = z(x)G(x,y) + H(x,y)$. Given that such a function z exists, it is given by

$$z(x) = z(x_0)e^{\int_{x_0}^{x} G(\tilde{x},y)d\tilde{x}} + \int_{x_0}^{x} e^{\int_{\hat{x}}^{x} G(\tilde{x},y)d\tilde{x}} H(\hat{x},y)d\hat{x}\,. \tag{16}$$

Then z is determined by $z(x_0)$ since we can extend Equation (16) to the remaining points. The set of all functions ξ satisfying the linear inhomogenous DE (15) is a 3-dimensional affine space: Once we have fixed $\xi(x_0), \xi'(x_0), \xi''(x_0)$ for some arbitrary point x_0, ξ is completely determined. Given fixed f and ν, the set of all ξ admitting a backward model is contained in this subspace. ■

A.7 Proof of Corollary 22

Proof Similarly to how (12) was derived, under the assumption of the existence of a reverse model one can derive

$$\frac{\partial^2\pi}{\partial x\partial y}\cdot\frac{\partial}{\partial x}\left(\frac{\partial^2\pi}{\partial x^2}\right) = \frac{\partial^2\pi}{\partial x^2}\cdot\frac{\partial}{\partial x}\left(\frac{\partial^2\pi}{\partial x\partial y}\right)\,.$$

Now using (13) and (14), we obtain

$$(-\nu''f')\cdot\frac{\partial}{\partial x}\left(\nu''(f')^2-\nu'f''+\xi''\right)=(\nu''(f')^2-\nu'f''+\xi'')\cdot\frac{\partial}{\partial x}\left(-\nu''f'\right),$$

which reduces to

$$-2(\nu''f')^2f''+\nu''f'\nu'f'''-\nu''f'\xi'''=-\nu'f''\nu'''(f')^2+\xi''\nu'''(f')^2+\nu''\nu'(f'')^2-\nu''f''\xi''.$$

Substituting the assumptions $\xi'''=0$ and $\nu'''=0$ (and hence $\nu''=C$ everywhere with $C\neq 0$ since otherwise ν cannot be a proper log-density) yields

$$\nu'\big(y-f(x)\big)\cdot\left(f'f'''-(f'')^2\right)=2C(f')^2f''-f''\xi''.$$

Since $C\neq 0$ there exists an α such that $\nu'(\alpha)=0$. Then, restricting ourselves to the submanifold $\{(x,y)\in\mathbb{R}^2 : y-f(x)=\alpha\}$ on which $\nu'=0$, we have

$$0=f''(2C(f')^2-\xi'').$$

Therefore, for all x in the open set $[f''\neq 0]$, we have $(f'(x))^2=\xi''/(2C)$, which is a constant, so $f''=0$ on $[f''\neq 0]$: a contradiction. Therefore, $f''=0$ everywhere. ■

A.8 Definitions of Proposition 23

Definition 35 *(Zhang and Hyvärinen, 2009) A one-dimensional distribution that is absolutely continuous with respect to the Lebesgue measure and has positive density p is called:*

- log-mix-lin-exp *if there are c_1,c_2,c_3,c_4 with $c_1<0$ and $c_2c_3>0$ such that*

$$\log p(x)=c_1\exp(c_2x)+c_3x+c_4,$$

- one-sided asymptotically exponential *if there is $c\neq 0$ such that*

$$\frac{d}{dx}\log p(x)\to c$$

  as $x\to-\infty$ or $x\to\infty$,

- two-sided asymptotically exponential *if there are $c_1\neq 0$ and $c_2\neq 0$ such that*

$$\frac{d}{dx}\log p(x)\to c_1$$

  as $x\to-\infty$ and

$$\frac{d}{dx}\log p(x)\to c_2$$

  as $x\to\infty$

- *and a* generalized mixture of two exponentials *if there are d_1,d_2,d_3,d_4,d_5,d_6 with $d_4>0$, $d_3>0$, $d_1d_5>0$ and $d_2<-\frac{d_1}{d_5}$ such that*

$$\log p(x)=d_1x+d_2\log(d_3+d_4\exp(d_5x))+d_6.$$

A.9 Proof of Example 25

Proof Our starting point is the assumption of nonidentifiability. In other words, we can describe the joint distribution of x and y both as an additive noise model where X causes Y, and as an additive noise model where Y causes X. Using the same notation as in Theorem 20, this means that:

$$\xi(x) + \nu\big(y - f(x)\big) = \eta(y) + \tilde{\nu}\big(x - g(y)\big) \qquad \forall x, y \in \mathbb{R}\,. \tag{17}$$

Case II in Proposition 23 (reproduced from Table 1 in Zhang and Hyvärinen, 2009) states that if both ξ and ν are log-mix-lin-exp and f is affine, then there could be an unidentifiable model. Let us verify whether that is indeed the case. We take

$$\begin{aligned}\xi(x) &= c_1 \exp(c_2 x) + c_3 x + c_4\\ \nu(n) &= \gamma_1 \exp(\gamma_2 n) + \gamma_3 n + \gamma_4\\ f(x) &= ax + b\end{aligned}$$

with $a \neq 0$ ($a = 0$ is the degenerate case with X and Y independent).

We can rewrite (17) as follows, by substituting x with $x + g(y)$:

$$c_1 e^{c_2(x+g(y))} + c_3(x+g(y)) + c_4 + \gamma_1 e^{\gamma_2(y-ax-ag(y)-b)} + \gamma_3(y-ax-ag(y)-b) + \gamma_4 = \eta(y) + \tilde{\nu}(x)\,. \tag{18}$$

Differentiating with respect to x yields

$$c_1 c_2 e^{c_2(x+g(y))} + c_3 - a\gamma_1\gamma_2 e^{\gamma_2(y-ax-ag(y)-b)} - \gamma_3 a = \tilde{\nu}'(x)\,. \tag{19}$$

Differentiating with respect to y yields

$$c_1 c_2^2 e^{c_2(x+g(y))} g'(y) - a\gamma_1\gamma_2^2 e^{\gamma_2(y-ax-ag(y)-b)}(1 - ag'(y)) = 0\,.$$

This can only be satisfied for all x if $c_2 = -a\gamma_2$. In that case:

$$-ac_1 g'(y) + \gamma_1 e^{\gamma_2(y-b)}(1 - ag'(y)) = 0\,.$$

Rewriting:

$$ag'(y) = \frac{\gamma_1 e^{\gamma_2(y-b)}}{c_1 + \gamma_1 e^{\gamma_2(y-b)}}\,.$$

Integrating:

$$g(y) = -\frac{1}{c_2}\ln(-c_1 - \gamma_1 e^{\gamma_2(y-b)}) + \frac{C}{c_2}\,.$$

Note that

$$e^{c_2 g(y)} = -\frac{1}{c_1 + \gamma_1 e^{\gamma_2(y-b)}} e^{-C}\,.$$

Substituting into (19) yields

$$-c_2 e^{-C} e^{c_2 x} + c_3 - \gamma_3 a = \tilde{\nu}'(x)\,.$$

Integrating yields

$$-e^{-C} e^{c_2 x} + (c_3 - \gamma_3 a)x + \delta_4 = \tilde{\nu}(x)\,,$$

which is also log-mix-lin-exp with parameters $\delta_1 = -e^{-C}$, $\delta_2 = c_2$, $\delta_3 = c_3 - \gamma_3 a$, δ_4. Substituting into (18):

$$g(y)(c_3 - \gamma_3 a) + \gamma_3 y + c_4 - \gamma_3 b + \gamma_4 - \delta_4 = \eta(y)\,,$$

i.e.,

$$\eta(y) = \left(-\frac{1}{c_2}\ln(-c_1 - \gamma_1 e^{\gamma_2(y-b)}) + \frac{C}{c_2}\right)(c_3 - \gamma_3 a) + \gamma_3 y + c_4 - \gamma_3 b + \gamma_4 - \delta_4\,.$$

This gives an inequality constraint: $c_3 \neq a\gamma_3$. $\eta(y)$ is a generalized mixture of exponentials distribution with parameters $d_1 = \gamma_3$, $d_2 = -\frac{c_3 - a\gamma_3}{c_2}$, $d_3 = -c_1$, $d_4 = -\gamma_1 e^{-\gamma_2 b}$, $d_5 = \gamma_2$, $d_6 = C\frac{c_3 - a\gamma_3}{c_2} + c_4 - \gamma_3 b + \gamma_4 - \delta_4$. One can check that all constraints on the parameters of the generalized mixture of exponentials are satisfied. Choosing C appropriately allows for normalizing the log-density. One can also easily verify that with these choices of $\tilde{\nu}(x)$ and $\eta(y)$, equation (17) holds, and therefore this gives an example of a nonidentifiable additive noise model. ■

A.10 Some Lemmata

The following four statements are all plausible and their proof is mostly about technicalities. The reader may skip to the next section and use the lemmata whenever needed. For random variables A and B we use $A\,|_{B=b}$ to denote the random variable A after conditioning on $B = b$ (assuming densities exist and B has positive density at b).

Lemma 36 *Let $Y \in \mathcal{Y}, N \in \mathcal{N}, \mathbf{Q} \in \mathcal{Q}, \mathbf{R} \in \mathcal{R}$ be random variables whose joint distribution is absolutely continuous with respect to some product measure ($\mathbf{Q}$ and $\mathbf{R}$ can be multivariate) and with density $p_{Y,\mathbf{Q},\mathbf{R},N}(y,\mathbf{q},\mathbf{r},n)$. Let $f : \mathcal{Y} \times \mathcal{Q} \times \mathcal{N} \to \mathbb{R}$ be a measurable function. If $N \perp\!\!\!\perp (Y, \mathbf{Q}, \mathbf{R})$ then for all $\mathbf{q} \in \mathcal{Q}, \mathbf{r} \in \mathcal{R}$ with $p_{\mathbf{Q},\mathbf{R}}(\mathbf{q},\mathbf{r}) > 0$:*

$$f(Y, \mathbf{Q}, N)\,|_{\mathbf{Q}=\mathbf{q},\mathbf{R}=\mathbf{r}} \stackrel{\mathcal{L}}{=} f(Y\,|_{\mathbf{Q}=\mathbf{q},\mathbf{R}=\mathbf{r}}, \mathbf{q}, N)\,.$$

A formal proof of this statement can be found in Peters et al. (2011b, Lemma 2).

Lemma 37 *Let $\mathcal{L}(\mathbf{X})$ be generated according to a SEM as in (2) with corresponding DAG $\mathcal{G}$ and consider a variable $X \in \mathbf{X}$. If $\mathbf{S} \subseteq \mathbf{ND}_X^{\mathcal{G}}$ then $N_X \perp\!\!\!\perp \mathbf{S}$.*

Proof Write $\mathbf{S} = \{S_1, \ldots, S_k\}$. Then

$$\mathbf{S} = \left(f_{S_1}(\mathbf{PA}_{S_1}^{\mathcal{G}}, N_{S_1}), \ldots, f_{S_k}(\mathbf{PA}_{S_k}^{\mathcal{G}}, N_{S_k})\right).$$

Again, one can substitute the parents of S_i by the corresponding functional equations and proceed recursively. After finitely many steps one obtains $\mathbf{S} = f(N_{T_1}, \ldots, N_{T_l})$, where $\{T_1, \ldots, T_l\}$ is the set of *all* ancestors of nodes in $\mathbf{S}$, which does not contain X. Since all noise variables are jointly independent we have $N_X \perp\!\!\!\perp \mathbf{S}$. ■

With the intersection property of conditional independence (e.g., 1.1.5 in Pearl, 2009), Proposition 4 has the following corollary that we formalize as a lemma.

Lemma 38 *Consider the random vector* $\mathbf{X}$ *and assume that the joint distribution has a (strictly) positive density. Then* $\mathcal{L}(\mathbf{X})$ *satisfies causal minimality with respect to* $\mathcal{G}$ *if and only if* $\forall B \in \mathbf{X}\ \forall A \in \mathbf{PA}_B^{\mathcal{G}}$ *and* $\forall \mathbf{S} \subset \mathbf{X}$ *with* $\mathbf{PA}_B^{\mathcal{G}} \setminus \{A\} \subseteq \mathbf{S} \subseteq \mathbf{ND}_B^{\mathcal{G}} \setminus \{A\}$ *we have that*

$$B \not\perp\!\!\!\perp A \mid \mathbf{S}\,.$$

Proof The "if" part is immediate. For the "only if" let us denote $\mathbf{P} := \mathbf{PA}_B^{\mathcal{G}} \setminus \{A\}$ and $\mathbf{Q} := \mathbf{S} \setminus (\mathbf{PA}_B^{\mathcal{G}} \setminus \{A\})$, such that $\mathbf{S} = \mathbf{P} \cup \mathbf{Q}$. Observe that $B \not\perp\!\!\!\perp A \,|\, \mathbf{P}$ (see Proposition 4) implies $B \not\perp\!\!\!\perp (\{A\} \cup \mathbf{Q}) \,|\, \mathbf{P}$. From the Markov condition we have $B \perp\!\!\!\perp \mathbf{Q} \,|\, (\mathbf{P} \cup \{A\})$. The intersection property of conditional independence yields $B \not\perp\!\!\!\perp A \,|\, (\mathbf{P} \cup \mathbf{Q})$. ■

A.11 Proof of Theorem 28

Proof We assume that there are two restricted additive noise models (see Definition 27) that both induce $\mathcal{L}(\mathbf{X})$, one with graph $\mathcal{G}$, the other with graph $\mathcal{G}'$. We will show that $\mathcal{G} = \mathcal{G}'$. Consider the variables L, Y from Proposition 29 (i) and define the sets $\mathbf{Q} := \mathbf{PA}_L^{\mathcal{G}} \setminus \{Y\}$, $\mathbf{R} := \mathbf{PA}_Y^{\mathcal{G}'} \setminus \{L\}$ and $\mathbf{S} := \mathbf{Q} \cup \mathbf{R}$. At first, we consider any $\mathbf{s} = (\mathbf{q}, \mathbf{r})$ and write $L^* := L \,|_{\mathbf{S}=\mathbf{s}}$ and $Y^* := Y \,|_{\mathbf{S}=\mathbf{s}}$. Lemma 37 gives us $N_L \perp\!\!\!\perp (Y, \mathbf{S})$ and $N_Y \perp\!\!\!\perp (L, \mathbf{S})$ and we can thus apply Lemma 36. From $\mathcal{G}$ we find

$$L^* = f_L(\mathbf{q}, Y^*) + N_L, \qquad N_L \perp\!\!\!\perp Y^*$$

and from $\mathcal{G}'$ we have

$$Y^* = g_Y(\mathbf{r}, L^*) + N_Y, \qquad N_Y \perp\!\!\!\perp L^*\,.$$

This contradicts Theorem 20 since according to Definition 27 we can choose $\mathbf{s} = (\mathbf{q}, \mathbf{r})$ such that $(f_L(\mathbf{q}, \cdot), \mathcal{L}(Y^*), \mathcal{L}(N_L))$ and $(g_Y(\mathbf{r}, \cdot), \mathcal{L}(L^*), \mathcal{L}(N_Y))$ satisfy Condition 19.

■

A.12 Proof of Proposition 29

Proof Since DAGs do not contain any cycles, we always find nodes that have no descendants (start a directed path at some node: after at most $\#\mathbf{X} - 1$ steps we reach a node without a child). Eliminating such a node from the graph leads to a DAG, again; we can discard further nodes without children in the new graph. We repeat this process for all nodes that have no children in both $\mathcal{G}$ and $\mathcal{G}'$ and have the same parents in both graphs. If we end up with no nodes left, the two graphs are identical which violates the assumption of the proposition. Otherwise, we end up with a smaller set of variables that we again call $\mathbf{X}$, two smaller graphs that we again call $\mathcal{G}$ and $\mathcal{G}'$ and a node L that has no children in $\mathcal{G}$ and either $\mathbf{PA}_L^{\mathcal{G}} \neq \mathbf{PA}_L^{\mathcal{G}'}$ or $\mathbf{CH}_L^{\mathcal{G}'} \neq \emptyset$. We will show that this leads to a contradiction. Importantly, because of the Markov property of the distribution with respect to $\mathcal{G}$, all other nodes are independent of L given $\mathbf{PA}_L^{\mathcal{G}}$:

$$L \perp\!\!\!\perp \mathbf{X} \setminus (\mathbf{PA}_L^{\mathcal{G}} \cup \{L\}) \mid \mathbf{PA}_L^{\mathcal{G}}\,. \tag{20}$$

To make the arguments easier to understand, we introduce the following notation (see also Fig. 7): we partition $\mathcal{G}$-parents of L into $\mathbf{Y}, \mathbf{Z}$ and $\mathbf{W}$. Here, $\mathbf{Z}$ are also $\mathcal{G}'$-parents of L, $\mathbf{Y}$ are $\mathcal{G}'$-children of L and $\mathbf{W}$ are not adjacent to L in $\mathcal{G}'$. We denote with $\mathbf{D}$ the $\mathcal{G}'$-parents of L that are not adjacent to L in $\mathcal{G}$ and by $\mathbf{E}$ the $\mathcal{G}'$-children of L that are not adjacent to L in $\mathcal{G}$. Thus: $\mathbf{PA}_L^{\mathcal{G}} = \mathbf{Y} \cup \mathbf{Z} \cup \mathbf{W}$, $\mathbf{CH}_L^{\mathcal{G}} = \emptyset$, $\mathbf{PA}_L^{\mathcal{G}'} = \mathbf{Z} \cup \mathbf{D}$, $\mathbf{CH}_L^{\mathcal{G}'} = \mathbf{Y} \cup \mathbf{E}$.

Figure 7: Nodes adjacent to L in $\mathcal{G}$ and $\mathcal{G}'$

Consider $\mathbf{T} := \mathbf{W} \cup \mathbf{Y}$. We distinguish two cases:

Case (i): $\mathbf{T} = \emptyset$.
Then there must be a node $D \in \mathbf{D}$ or a node $E \in \mathbf{E}$, otherwise L would have been discarded.

1. If there is a $D \in \mathbf{D}$ then (20) implies $L \perp\!\!\!\perp D \,|\, \mathbf{S}$ for $\mathbf{S} := \mathbf{Z} \cup \mathbf{D} \setminus \{D\}$, which contradicts Lemma 38 (applied to $\mathcal{G}'$).

2. If $\mathbf{D} = \emptyset$ and there is $E \in \mathbf{E}$ then $E \perp\!\!\!\perp L \,|\, \mathbf{S}$ holds for $\mathbf{S} := \mathbf{Z} \cup \mathbf{PA}_E^{\mathcal{G}'} \setminus \{L\}$ (see graph $\mathcal{G}$), which also contradicts Lemma 38 (note that $\mathbf{Z} \subseteq \mathbf{ND}_E^{\mathcal{G}'}$ to avoid cycles).

Case (ii): $\mathbf{T} \neq \emptyset$.
Then $\mathbf{T}$ contains a "$\mathcal{G}'$-youngest" node with the property that there is no directed $\mathcal{G}'$-path from this node to any other node in $\mathbf{T}$. This node may not be unique.

1. Suppose that some $W \in \mathbf{W}$ is such a youngest node. Consider the DAG $\tilde{\mathcal{G}}'$ that equals $\mathcal{G}'$ with additional edges $Y \to W$ and $W' \to W$ for all $Y \in \mathbf{Y}$ and $W' \in \mathbf{W} \setminus \{W\}$. In $\tilde{\mathcal{G}}'$, L and W are not adjacent. Thus we find a set $\tilde{\mathbf{S}}$ such that $\tilde{\mathbf{S}}$ d-separates L and W in $\tilde{\mathcal{G}}'$; indeed, one can take $\tilde{\mathbf{S}} = \mathbf{PA}_L^{\tilde{\mathcal{G}}'}$ if $W \notin \mathbf{DE}_L^{\tilde{\mathcal{G}}'}$ and $\tilde{\mathbf{S}} := \mathbf{PA}_W^{\tilde{\mathcal{G}}'}$ if $L \notin \mathbf{DE}_W^{\tilde{\mathcal{G}}'}$. Then also $\mathbf{S} = \tilde{\mathbf{S}} \cup \{\mathbf{Y}, \mathbf{Z}, \mathbf{W} \setminus \{W\}\}$ d-separates L and W in $\tilde{\mathcal{G}}'$.

   Indeed, all $Y \in \mathbf{Y}$ are already in $\tilde{\mathbf{S}}$ in order to block $L \to Y \to W$. Suppose there is a $\tilde{\mathcal{G}}'$-path that is blocked by $\tilde{\mathbf{S}}$ and unblocked if we add Z and W' nodes to $\tilde{\mathbf{S}}$. How can we unblock a path by including more nodes? The path ($L \cdots V_1 \cdots U_1 \cdots W$ in Fig. 8) must contain a collider V_1 that is an ancestor of a Z with $V_1, \ldots, V_m, Z \notin \tilde{\mathbf{S}}$ and corresponding nodes U_i for a W' node. Choose V_1 and U_1 on the given path so close to each other such that there is no such collider in between. If there is no V_1, choose U_1 closest to L, if there is no U_1, choose V_1 closest to W. Now the path $L \leftarrow Z \cdots V_1 \cdots U_1 \cdots W' \to W$ is unblocked given $\tilde{\mathbf{S}}$, which is a contradiction to the assumption that $\tilde{\mathbf{S}}$ d-separates L and W.

   But then $\mathbf{S}$ d-separates L and W in $\mathcal{G}'$, too (there are less paths), and we have $L \perp\!\!\!\perp W \mid \mathbf{S}$, which contradicts Lemma 38 (applied to $\mathcal{G}$).

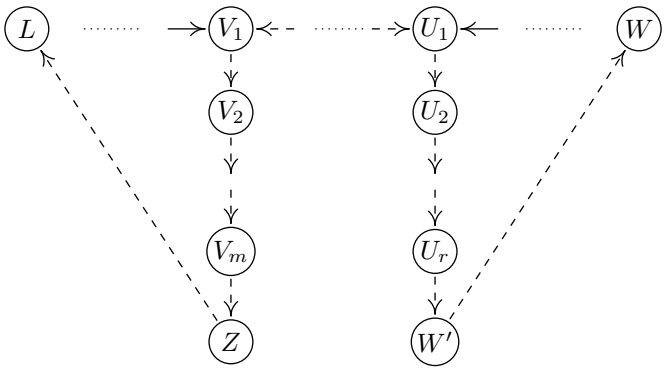


Figure 8: Assume the path $L \cdots V_1 \cdots U_1 \cdots W$ is blocked by $\tilde{\mathbf{S}}$, but unblocked if we include Z and W'. Then the dashed path is unblocked given $\tilde{\mathbf{S}}$.

2. Therefore, the $\mathcal{G}'$-youngest node in $\mathbf{T}$ must be some $Y \in \mathbf{Y}$.
   Define $\mathbf{Q} := \mathbf{PA}_L^{\mathcal{G}} \setminus \{Y\}$, $\mathbf{R} := \mathbf{PA}_Y^{\mathcal{G}'} \setminus \{L\}$ and $\mathbf{S} := \mathbf{Q} \cup \mathbf{R}$. Clearly, $\mathbf{S} \subseteq \mathbf{ND}_L^{\mathcal{G}} \setminus \{Y\}$ since L does not have any descendants in $\mathcal{G}$. Further, $\mathbf{S} \subseteq \mathbf{ND}_Y^{\mathcal{G}'} \setminus \{L\}$ because Y is the $\mathcal{G}'$-youngest under all $\mathbf{W}$ and $\mathbf{Y} \setminus \{Y\}$ by construction and any directed path from Y to $Z \in \mathbf{Z}$ would introduce a cycle in $\mathcal{G}'$. Ergo, $\{Y\} \cup \mathbf{S} \subseteq \mathbf{ND}_L^{\mathcal{G}}$ and $\{L\} \cup \mathbf{S} \subseteq \mathbf{ND}_Y^{\mathcal{G}'}$.

The variables L and Y and the sets $\mathbf{Q}, \mathbf{R}$ and $\mathbf{S}$ satisfy the conditions required in statement (i) of Proposition 29.

Statement (ii) follows as a special case since for Markov equivalent graphs, $\mathbf{W}, \mathbf{D}$ and $\mathbf{E}$ are all empty. Consider the $\mathcal{G}'$-youngest node Y. In order to avoid v-structures appearing in $\mathcal{G}$ and not in $\mathcal{G}'$ all nodes $Z \in \mathbf{Z}$ are directly connected to the $\mathcal{G}'$-youngest Y. And to avoid cycles, those nodes $Z \in \mathbf{Z}$ are $\mathcal{G}'$-parents of Y. The node Y cannot have other parents except for the ones in $\mathbf{Y}$ and $\mathbf{Z}$ since this would introduce v-structures in $\mathcal{G}'$ (with collider Y) that do not appear in $\mathcal{G}$. ■

A.13 Proof of Corollary 31

Proof We only prove (i) since (ii) is a special case. Causal minimality is satisfied because of Proposition 17. We can then assume that the statement is false and apply the same argument as in Theorem 28. This yields the two equations

$$\begin{aligned} L^* &= f_L(\mathbf{q}, Y^*) + N_L, \qquad N_L \perp\!\!\!\perp Y^* \qquad \text{and} \\ Y^* &= g_Y(\mathbf{r}, L^*) + N_Y, \qquad N_Y \perp\!\!\!\perp L^* . \end{aligned}$$

Let us define $f := f_L(\mathbf{q}, \cdot)$ and $g := g_Y(\mathbf{r}, \cdot)$. Because of independence of N_Y and L^* we have

$$0 = \frac{\partial^2 \log p(\ell^*, n_y)}{\partial n_y \, \partial \ell^*} = \frac{\partial^2 \log p(y^*, n_\ell)}{\partial n_y \, \partial \ell^*} = \frac{\partial^2 \log p(y^*) + \partial^2 \log p(n_\ell)}{\partial n_y \, \partial \ell^*}$$

with $y^* = g(\ell^*) + n_y$ and $n_\ell = \ell^* - f(g(\ell^*) + n_y)$. Ergo, for all ℓ^* and y^* we have

$$0 = \frac{\partial^2 \log p_{Y^*}(y^*)}{(\partial y^*)^2} g'(\ell^*) - \frac{1}{\sigma_{N_L}^2} f'(y^*)^2 g'(\ell^*) + \frac{1}{\sigma_{N_L}^2} f'(y^*) + \frac{\ell^* - f(y^*)}{\sigma_{N_L}^2} f''(y^*) g'(\ell^*) \,. \quad (21)$$

If there is a ℓ^* with $g'(\ell^*) = 0$, (21) implies that f' is constantly zero which is not the case. Exchanging the role of L^* and Y^* yields $f'(y^*) \neq 0$. Since N_L is Gaussian, Proposition 23 implies that f is linear. This contradicts the assumption of nonlinearity. For completeness, however, we give a direct proof that is similar to Lemma 6 in Zhang and Hyvärinen (2009). Dividing (21) by $g'(\ell^*) f'(y^*)$ yields

$$0 = \frac{\partial^2 \log p_{Y^*}(y^*)}{(\partial y^*)^2} \frac{1}{f'(y^*)} - \frac{1}{\sigma_{N_L}^2} f'(y^*) + \frac{1}{\sigma_{N_L}^2} \frac{1}{g'(\ell^*)} + \frac{\ell^* - f(y^*)}{\sigma_{N_L}^2} \frac{f''(y^*)}{f'(y^*)}$$

and therefore (take the derivative with respect to ℓ^*) $\frac{f''(y^*)}{f'(y^*)} \equiv a_1$ which means $f'(y^*) = a_2 \exp(a_1 y^*)$ with $a_1, a_2 \neq 0$ because f is nonlinear. But then $\frac{1}{g'(\ell^*)} + a_1 \ell^* \equiv a_3$ is constant and using $a_1 f(y^*) = f'(y^*) + a_4$ for some a_4 we have

$$\frac{\partial^2 \log p_{Y^*}(y^*)}{(\partial y^*)^2} - \frac{2}{\sigma_{N_L}^2} f'(y^*)^2 + \frac{a_3 - a_4}{\sigma_{N_L}^2} f'(y^*) = 0 \,,$$

which implies $\log p_{Y^*}(y^*) \to \infty$ for either $y^* \to \infty$ or $y^* \to -\infty$. Obviously, this cannot be the case. This proves the corollary. ■

A.14 Proof of Lemma 32

Proof For (a) suppose that $\mathcal{G}$ has an additional edge from X_i to X_j compared to $\mathcal{G}_0$. We can then change the corresponding structural equation $X_j = f_j(\mathbf{PA}_j^{\mathcal{G}_0}) + N_j$ into $X_j = \tilde{f}_j(\mathbf{PA}_j^{\mathcal{G}_0}, X_i) + N_j$ where $\tilde{f}_j$ equals f_j in the first $\#\mathbf{PA}_j^{\mathcal{G}_0}$ components and $\tilde{f}_j$ is constant in the last component.

We now prove statement (b). Let $\mathcal{G} \leq \mathcal{G}_0$ such that $\mathcal{L}(\mathbf{X})$ is Markov with respect to $\mathcal{G}$. Suppose $i \in \mathcal{G}$ with $\mathbf{PA}_i^{\mathcal{G}} \subsetneq \mathbf{PA}_i^{\mathcal{G}_0}$. Denote $X_B = \mathbf{PA}_i^{\mathcal{G}_0} \setminus \mathbf{PA}_i^{\mathcal{G}}$. Since $\mathbf{PA}_i^{\mathcal{G}_0} \subseteq \mathbf{ND}_i^{\mathcal{G}_0} \subseteq \mathbf{ND}_i^{\mathcal{G}}$, we have from the Markov property that $X_i \perp\!\!\!\perp X_B \,|\, \mathbf{PA}_i^{\mathcal{G}}$. Analogously to the proof of Proposition 17 this implies that the (continuous) function f_i in the corresponding structural equation $X_i = f_i(\mathbf{PA}_i^{\mathcal{G}_0}) + N_i$ must be constant in X_B. We can therefore define the corresponding structural equation in $\mathcal{G}$ to be $X_i = f_i(\mathbf{PA}_i^{\mathcal{G}}, x_B) + N_i$ for some arbitrary x_B. Structural equations for variables with identical parent sets do not need to be changed. Now suppose $\mathcal{G}_1 \leq \mathcal{G}_0$ and $\mathcal{G}_2 \leq \mathcal{G}_0$. Then there is an additive noise model with graph $\mathcal{G}_{12} \leq \mathcal{G}_0$ that leads to $\mathcal{L}(\mathbf{X})$, where $\mathcal{G}_{12}$ has precisely the edges that appear in both $\mathcal{G}_1$ and $\mathcal{G}_2$. This follows by noting that the intersection property implies that $X_i \perp\!\!\!\perp (\mathbf{PA}_i^{\mathcal{G}_0} \setminus \mathbf{PA}_i^{\mathcal{G}_{12}}) \,|\, \mathbf{PA}_i^{\mathcal{G}_{12}}$, and hence f_i is constant in $(\mathbf{PA}_i^{\mathcal{G}_0} \setminus \mathbf{PA}_i^{\mathcal{G}_{12}})$. (This step is not necessarily true for densities that are not strictly positive.) The partial ordering $\leq$ defined by the subgraph property therefore has a unique least element $\mathcal{G}_0^{min}$, which satisfies causal minimality by Proposition 17. ■

A.15 Proof of Theorem 34

Proof For the correct graph, we know that N_i is independent of all ancestor variables X_j since the latter can be expressed in terms of noise variables without N_i. The correct sink nodes therefore lead to independence in step 7 of Algorithm 1. We will now show that "wrong sinks", that is nodes who are not sinks in the correct graph $\mathcal{G}_0$ do not lead to independent residuals in the first iteration of Phase 1. It follows by induction that this is true for any later iteration, too. Suppose that node Y is not a sink in $\mathcal{G}_0$ but leads to independent residuals (step 7). Since Y is not a sink in $\mathcal{G}_0$, Y has children in $\mathcal{G}_0$. Call Z the $\mathcal{G}_0$-youngest child, that is there is no directed path from Z to any other child of Y. Disregard all descendants of Z and denote the remaining set of variables $\mathbf{S} := \mathbf{X} \setminus \{Y, Z, \mathbf{DE}_Z^{\mathcal{G}_0}\}$. It therefore follows that

$$\mathbf{DE}_Z^{\mathcal{G}_0} \perp\!\!\!\perp Y \,|\, \mathbf{S} \cup \{Z\}\,. \tag{22}$$

Because Y leads to independent residuals we can think of a graph $\mathcal{G}$ in which all variables are parents of Y. From Equation (22) it follows that $Y = g_Y(\mathbf{S}, Z) + \tilde{N}_Y$ with $\tilde{N}_Y \perp\!\!\!\perp (\mathbf{S}, Z)$. We then proceed similarly as in the proof of Theorem 28 and find from $\mathcal{G}_0$ that

$$Z \,|_{\mathbf{S}=\mathbf{s}} = f_Z(s_{\mathbf{PA}_Z^{\mathcal{G}_0}}, Y \,|_{\mathbf{S}=\mathbf{s}}) + N_Z\,.$$

From $\mathcal{G}$ we conclude that

$$Y \,|_{\mathbf{S}=\mathbf{s}} = g_Y(\mathbf{s}, Z \,|_{\mathbf{S}=\mathbf{s}}) + \tilde{N}_Y\,.$$

Again, this contradicts Theorem 20. The correctness of Phase 2 follows from causal minimality and Lemma 38. ■

References

S. Acid and L. M. de Campos. Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research*, 18:445–490, 2003.

W. P. Bergsma. *Testing Conditional Independence for Continuous Random Variables*, 2004. EURANDOM-report 2004-049.

K. A. Bollen. *Structural Equations with Latent Variables*. John Wiley & Sons, 1989.

P. Bühlmann, J. Peters, and J. Ernest. CAM: Causal additive models, high-dimensional order search and penalized regression. *ArXiv e-prints (1207.5136)*, 2013.

D. M. Chickering. A transformational characterization of equivalent Bayesian network structures. In *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 1995.

D. M. Chickering. Learning Bayesian networks is NP-complete. In *Learning from Data: Artificial Intelligence and Statistics V*. Springer-Verlag, 1996.

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

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

G. Darmois. Analyse générale des liaisons stochastiques. *Revue de l'Institut International de Statistique*, 21:2–8, 1953.

A. P. Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society. Series B*, 41(1):1–31, 1979.

Deutscher Wetterdienst. Climate data. `http://www.dwd.de/`, 2008.

M. J. Druzdzel and H. van Leijen. Causal reversibility in Bayesian networks. *Journal of Experimental and Theoretical Artificial Intelligence*, 13(1):45–62, 2001.

F. Eberhardt and R. Scheines. Interventions and causal inference. *Philosophy of Science*, 74(5):981–995, 2007.

N. Friedman and D. Koller. Being Bayesian about Bayesian network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50:95–125, 2003.

K. Fukumizu, A. Gretton, X. Sun, and B. Schölkopf. Kernel measures of conditional dependence. In *Advances in Neural Information Processing Systems 20 (NIPS)*, 2008.

D. Geiger and D. Heckerman. Learning Gaussian networks. In *Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 1994.

A. Gretton, R. Herbrich, A. Smola, O. Bousquet, and B. Schölkopf. Kernel methods for measuring independence. *Journal of Machine Learning Research*, 6:2075–2129, 2005.

A. Gretton, K. Fukumizu, C. H. Teo, L. Song, B. Schölkopf, and A. Smola. A kernel statistical test of independence. In *Advances in Neural Information Processing Systems 20 (NIPS)*, 2008.

D. M. A. Haughton. On the choice of a model to fit data from an exponential family. *The Annals of Statistics*, 16:342–355, 1988.

D. Heckerman. A Bayesian approach to causal discovery. Technical report, Microsoft Research (MSR-TR-97-05), 1997.

D. Heckerman and D. Geiger. Likelihoods and parameter priors for Bayesian networks. Technical report, Microsoft Research (MSR-TR-95-54), 1995.

P. Hoyer, S. Shimizu, A. J. Kerminen, and M. Palviainen. Estimation of causal effects using linear non-Gaussian causal models with hidden variables. *International Journal of Approximate Reasoning*, 49:362–378, 2008.

P. Hoyer, D. Janzing, J. M. Mooij, J. Peters, and B. Schölkopf. Nonlinear causal discovery with additive noise models. In *Advances in Neural Information Processing Systems 21 (NIPS)*, 2009.

A. Hyvärinen and S. M. Smith. Pairwise likelihood ratios for estimation of non-Gaussian structural equation models. *Journal of Machine Learning Research*, 14:111–152, 2013.

D. Janzing and B. Schölkopf. Causal inference using the algorithmic Markov condition. *IEEE Transactions on Information Theory*, 56:5168–5194, 2010.

D. Janzing and B. Steudel. Justifying additive-noise-model based causal discovery via algorithmic information theory. *Open Systems and Information Dynamics*, 17:189–212, 2010.

D. Janzing, J. Peters, J. M. Mooij, and B. Schölkopf. Identifying confounders using additive noise models. In *Proceedings of the 25th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 2009.

M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research*, 8:613–636, 2007.

Y. Kano and S. Shimizu. Causal inference using nonnormality. In *Proceedings of the International Symposium on Science of Modeling, the 30th Anniversary of the Information Criterion*, Tokyo, Japan, 2003.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

S. Lauritzen. *Graphical Models*. Oxford University Press, New York, 1996.

C. Meek. *Graphical Models: Selecting Causal and Statistical Models*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1997.

J. M. Mooij and D. Janzing. Distinguishing between cause and effect. *Journal of Machine Learning Research W&CP*, 6:147–156, 2010.

J. M. Mooij, D. Janzing, J. Peters, and B. Schölkopf. Regression by dependence minimization and its application to causal inference. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2009.

J. M. Mooij, D. Janzing, T. Heskes, and B. Schölkopf. On causal discovery with cyclic additive noise models. In *Advances in Neural Information Processing Systems 24 (NIPS)*, 2011.

J. M. Mooij, D. Janzing, and B. Schölkopf. From ordinary differential equations to structural causal models: the deterministic case. In *Proceedings of the 29th Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 2013.

C. Nowzohour and P. Bühlmann. Score-based causal learning in additive noise models. *ArXiv e-prints (1311.6359)*, 2013.

OEIS Foundation Inc. The on-line encyclopedia of integer sequences. `http://oeis.org/A003024`, 2011.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition, 2009.

J. Peters. Asymmetries of time series under inverting their direction. Diploma Thesis, University of Heidelberg, 2008. `http://stat.ethz.ch/people/jopeters`.

J. Peters. *Restricted Structural Equation Models for Causal Inference*. PhD thesis, ETH Zurich and MPI for Intelligent Systems, 2012. `http://dx.doi.org/10.3929/ethz-a-007597940`.

J. Peters. On the intersection property of conditional independence and its application to causal discovery. *ArXiv e-prints (1403.0408)*, 2014.

J. Peters and P. Bühlmann. Structural intervention distance (SID) for evaluating causal graphs. *ArXiv e-prints (1306.1043)*, 2013.

J. Peters and P. Bühlmann. Identifiability of Gaussian structural equation models with equal error variances. *Biometrika*, 101:219–228, 2014.

J. Peters, D. Janzing, and B. Schölkopf. Causal inference on discrete data using additive noise models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33: 2436–2450, 2011a.

J. Peters, J. M. Mooij, D. Janzing, and B. Schölkopf. Identifiability of causal graphs using functional models. In *Proceedings of the 27th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 2011b.

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.

C. E. Rasmussen and H. Nickisch. Gaussian processes for machine learning (GPML) toolbox. *Journal of Machine Learning Research*, 11:3011–3015, 2010.

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

B. Schölkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. M. Mooij. On causal and anticausal learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012.

S. Shimizu, P. Hoyer, A. Hyvärinen, and A. J. Kerminen. A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7:2003–2030, 2006.

S. Shimizu, T. Inazumi, Y. Sogawa, A. Hyvärinen, Y. Kawahara, T. Washio, P. 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.

R. Silva and Z. Ghahramani. The hidden life of latent variables: Bayesian learning with mixed graph models. *Journal of Machine Learning Research*, 10:1187–1238, 2009.

V.P. Skitovič. Linear forms in independent random variables and the normal distribution law (in Russian). *Izvestiia AN SSSR, Ser. Matem.*, 18:185–200, 1954.

V.P. Skitovič. Linear combinations of independent random variables and the normal distribution law. *Select. Transl. Math. Stat. Probab.*, 2:211–228, 1962.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, 2nd edition, 2000.

Y. Tamada, S. Imoto, H. Araki, M. Nagasaki, C. G. Print, S. D. Charnock-Jones, and S. Miyano. Estimating genome-wide gene networks using nonparametric Bayesian network models on massively parallel computers. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(3):683–697, 2011a.

Y. Tamada, S. Imoto, and S. Miyano. Parallel algorithm for learning optimal Bayesian network structure. *Journal of Machine Learning Research*, 12:2437–2459, 2011b.

M. Teyssier and D. Koller. Ordering-based search: a simple and effective algorithm for learning Bayesian networks. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

C. Uhler, G. Raskutti, P. Bühlmann, and B. Yu. Geometry of the faithfulness assumption in causal inference. *Annals of Statistics*, 41(2):436–463, 2013.

T. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Proceedings of the 6th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 1991.

S. Wright. Correlation and causation. *Journal of Agricultural Research*, 20:557–585, 1921.

J. Zhang and P. Spirtes. Strong faithfulness and uniform consistency in causal inference. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 2003.

J. Zhang and P. Spirtes. Detection of unfaithfulness and robust causal inference. *Minds and Machines*, 18:239–271, 2008.

K. Zhang and A. Hyvärinen. On the identifiability of the post-nonlinear causal model. In *Proceedings of the 25th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 2009.

K. Zhang, J. Peters, D. Janzing, and B. Schölkopf. Kernel-based conditional independence test and application in causal discovery. In *Proceedings of the 27th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 2011.

Journal of Machine Learning Research 15 (2014) 2055-2060 Submitted 8/13; Revised 2/14; Published 6/14

PyStruct - Learning Structured Prediction in Python

Andreas C. Müller AMUELLER@AIS.UNI-BONN.DE
Sven Behnke BEHNKE@CS.UNI-BONN.DE
Institute of Computer Science, Department VI
University of Bonn
Bonn, Germany

Editor: Mark Reid

Abstract

Structured prediction methods have become a central tool for many machine learning applications. While more and more algorithms are developed, only very few implementations are available.

PyStruct aims at providing a general purpose implementation of standard structured prediction methods, both for practitioners and as a baseline for researchers. It is written in Python and adapts paradigms and types from the scientific Python community for seamless integration with other projects.

Keywords: structured prediction, structural support vector machines, conditional random fields, Python

1. Introduction

In recent years there has been a wealth of research in methods for learning structured prediction, as well as in their application in areas such as natural language processing and computer vision. Unfortunately only few implementations are publicly available—many applications are based on the non-free implementation of Joachims et al. (2009).

PyStruct aims at providing a high-quality implementation with an easy-to-use interface, in the high-level Python language. This allows practitioners to efficiently test a range of models, as well as allowing researchers to compare to baseline methods much more easily than this is possible with current implementations. PyStruct is BSD-licensed, allowing modification and redistribution of the code, as well as use in commercial applications. By embracing paradigms established in the scientific Python community and reusing the interface of the widely-used scikit-learn library (Pedregosa et al., 2011), PyStruct can be used in existing projects, replacing standard classifiers. The online documentation and examples help new users understand the somewhat abstract ideas behind structured prediction.

2. Structured Prediction and Casting it into Software

Structured prediction can be defined as making a prediction $f(x)$ by maximizing a compatibility function between an input x and the possible labels y (Nowozin and Lampert, 2011). Most current approaches use linear functions, leading to:

$$f(x) = \arg\max_{y \in \mathcal{Y}} \ \theta^T \Psi(x, y). \tag{1}$$

Here, y is a *structured* label, Ψ is a joint feature function of x and y, and θ are parameters of the model. *Structured* means that y is more complicated than a single output class, for example a label for each word in a sentence or a label for each pixel in an image. Learning structured prediction means learning the parameters θ from training data.

Using the above formulation, learning can be broken down into three sub-problems:

1. Optimizing the objective with respect to θ.
2. Encoding the structure of the problem in a joint feature function Ψ.
3. Solving the maximization problem in Equation 1.

The later two problems are usually tightly coupled, as the maximization in Equation 1 is usually only feasible by exploiting the structure of Ψ, while the first is treated as independent. In fact, when 3. can not be done exactly, learning θ strongly depends on the quality of the approximation. However, treating approximate inference and learning as a joint optimization problem is currently out of the scope of the package, and we implement a more modular setup. PyStruct takes an object-oriented approach to decouple the task-dependent implementation of 2. and 3. from the general algorithms used to solve 1.

Estimating θ is done in `learner` classes, which currently support cutting plane algorithms for structural support vector machines (SSVMs Joachims et al. (2009)), subgradient methods for SSVMs Ratliff et al. (2007), Block-coordinate Frank-Wolfe (BCFW) (Lacoste-Julien et al., 2012), the structured perceptron and latent variable SSVMs (Yu and Joachims, 2009). The cutting plane implementation uses the cvxopt package (Andersen et al., 2012) for quadratic optimization. Encoding the structure of the problem is done using `model` classes, which compute Ψ and encode the structure of the problem. The structure of Ψ determines the hardness of the maximization in Equation (1) and is a crucial factor in learning. PyStruct implements models (corresponding to particular forms of Ψ) for many common cases, such as multi-class and multi-label classification, conditional random fields with constant or data-dependent pairwise potentials, and several latent variable models. The maximization for finding y in Equation 1 is carried out using external libraries, such as OpenGM (Kappes et al., 2013), LibDAI (Mooij, 2010) and others. This allows the user to choose from a wide range of optimization algorithms, including (loopy) belief propagation, graph-cuts, QPBO, dual subgradient, MPBP, TRWs, LP and many other algorithms. For problems where exact inference is infeasible, PyStruct allows the use of linear programming relaxations, and provides modified loss and feature functions to work with the continuous labels. This approach, which was outlined in Finley and Joachims (2008) allows for principled learning when exact inference is intractable. When using approximate integral solvers, learning may finish prematurely and results in this case depend on the inference scheme and learning algorithm used.

Table 1 lists algorithms and models that are implemented in PyStruct and compares them to other public structured prediction libraries: Dlib (King, 2009), SVM$^{\text{struct}}$ (Joachims et al., 2009) and CRFsuite (Okazaki, 2007). We also give the programming language and the project license.

Package	Language	License	Algorithms				Models			
			CP	SG	BCFW	LV	ML	Chain	Graph	LDCRF
PyStruct	Python	BSD[1]	✓[1]	✓	✓	✓	×	✓	✓	✓
SVM^{struct}	C++	non-free	✓	×	×	✓	×	×	×	×
Dlib	C++	boost	✓	×	×	×	×	✓	✓	×
CRFsuite	C++	BSD	×	×	×	×	✓	✓	×	×

Table 1: Comparison of structured prediction software packages. CP stands for cutting plane optimization of SSVMs, SG for online subgradient optimization of SSVMs, LV for latent variable SSVMs, ML for maximum likelihood learning, Chain for chain-structured models with pairwise interactions, Graph for arbitrary graphs with pairwise interactions, and LDCRF for latent dynamic CRF (Morency et al., 2007). [1]PyStruct itself is BSD licensed, but uses the GPL-licensed package cvxopt for cutting-plane learning.

3. Usage Example: Semantic Image Segmentation

Conditional random fields are an important tool for semantic image segmentation. We demonstrate how to learn an n-slack support vector machine (Tsochantaridis et al., 2006) on a superpixel-based CRF on the popular Pascal data set. We use unary potentials generated using TextonBoost from Krähenbühl and Koltun (2012). The superpixels are generated using SLIC (Achanta et al., 2012).[1] Each sample (corresponding on one entry of the list `X`) is represented as a tuple consisting of input features and a graph representation.

```
1 model = crfs.EdgeFeatureGraphCRF(
2     class_weight=inverse_frequency, symmetric_edge_features=[0, 1],
3     antisymmetric_edge_features=[2], inference_method='qpbo')
4
5 ssvm = learners.NSlackSSVM(model, C=0.01, n_jobs=-1)
6 ssvm.fit(X, Y)
```

Listing 1: Example of defining and learning a CRF model.

The source code is shown in Listing 1. Lines 1-3 declare a model using parametric edge potentials for arbitrary graphs. Here `class_weight` re-weights the hamming loss according to inverse class frequencies. The parametric pairwise interactions have three features: a constant feature, color similarity, and relative vertical position. The first two are declared to be symmetric with respect to the direction of an edge, the last is antisymmetric. The inference method used is QPBO-fusion moves. Line 5 creates a `learner` object that will learn the parameters for the given model using the n-slack cutting plane method, and line 6 performs the actual learning. Using this simple setup, we achieve an accuracy of 30.3 on the validation set following the protocol of Krähenbühl and Koltun (2012), who report 30.2 using a more complex approach. Training the structured model takes approximately 30 minutes using a single i7 core.

1. The preprocessed data can be downloaded at `http://www.ais.uni-bonn.de/download/datasets.html`.

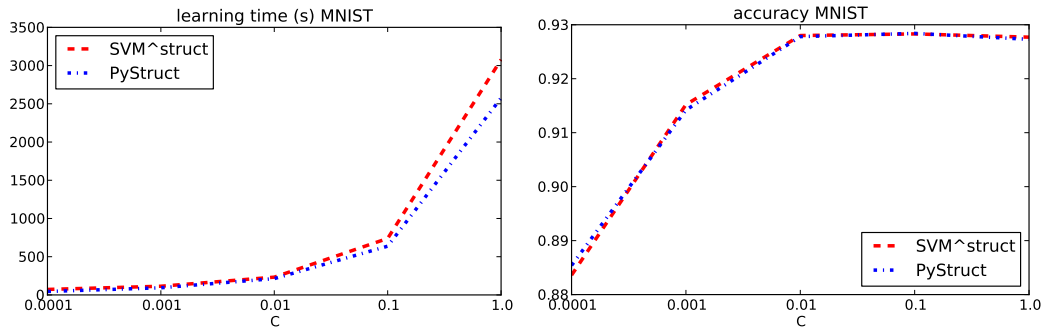


Figure 1: Runtime comparison of PyStruct and SVM$^{\text{struct}}$ for multi-class classification.

4. Experiments

While PyStruct focuses on usability and covers a wide range of applications, it is also important that the implemented learning algorithms run in acceptable time. In this section, we compare our implementation of the 1-slack cutting plane algorithm (Joachims et al., 2009) with the implementation in SVM$^{\text{struct}}$. We compare performance of the Crammer-Singer multi-class SVM with respect to learning time and accuracy on the MNIST data set of handwritten digits. While multi-class classification is not very interesting from a structured prediction point of view, this problem is well-suited to benchmark the cutting plane solvers with respect to accuracy and speed.

Results are shown in Figure 1. We report learning times and accuracy for varying regularization parameter C. The MNIST data set has 60 000 training examples, 784 features and 10 classes.[2] The figure indicates that PyStruct has competitive performance, while using a high-level interface in a dynamic programming language.

5. Conclusion

This paper introduced PyStruct, a modular structured learning and prediction library in Python. PyStruct is geared towards ease of use, while providing efficient implementations. PyStruct integrates itself into the scientific Python eco-system, making it easy to use with existing libraries and applications. Currently, PyStruct focuses on max-margin and perceptron-based approaches. In the future, we plan to integrate other paradigms, such as sampling-based learning (Wick et al., 2011), surrogate objectives (for example pseudo-likelihood), and approaches that allow for a better integration of inference and learning (Meshi et al., 2010).

Acknowledgments

The authors would like to thank Vlad Niculae and Forest Gregg for their contributions to PyStruct and André Martins for his help in integrating the AD^3 solver with PyStruct. This work was partially funded by the B-IT research school.

2. Details about the experiment and code for the experiments can be found on the project website.

References

Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *PAMI*, 2012.

Martin S. Andersen, Joachin Dahl, and Lieven Vandenberghe. CVXOPT: A Python package for convex optimization, version 1.1.5. Available at `http://cvxopt.org/`, 2012.

Thomas Finley and Thorsten Joachims. Training structural SVMs when exact inference is intractable. In *ICML*, 2008.

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural SVMs. *JMLR*, 77(1), 2009.

Jörg H Kappes, Bjoern Andres, Fred A Hamprecht, Christoph Schnörr, Sebastian Nowozin, Dhruv Batra, Sungwoong Kim, Bernhard X Kausler, Jan Lellmann, Nikos Komodakis, et al. A comparative study of modern inference techniques for discrete energy minimization problems. In *CVPR*, 2013.

Davis E. King. Dlib-ml: A machine learning toolkit. *JMLR*, 10, 2009.

Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. In *NIPS*, 2012.

Simon Lacoste-Julien, Mark Schmidt, and Francis Bach. A simpler approach to obtaining an O(1/t) convergence rate for projected stochastic subgradient descent. *arXiv preprint arXiv:1212.2002*, 2012.

Ofer Meshi, David Sontag, Tommi Jaakkola, and Amir Globerson. Learning efficiently with approximate inference via dual losses. In *ICML*, 2010.

Joris M. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *JMLR*, 2010.

L-P Morency, Ariadna Quattoni, and Trevor Darrell. Latent-dynamic discriminative models for continuous gesture recognition. In *CVPR*, 2007.

Sebastian Nowozin and Christoph H. Lampert. *Structured Learning and Prediction in Computer Vision*. Now Publishers Inc., 2011.

Naoaki Okazaki. CRFsuite: A fast implementation of conditional random fields (CRFs), 2007. URL `http://www.chokkan.org/software/crfsuite/`.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *JMLR*, 2011.

Nathan Ratliff, J. Andrew (Drew) Bagnell, and Martin Zinkevich. (Online) subgradient methods for structured prediction. In *AISTATS*, 2007.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, Yasemin Altun, and Yoram Singer. Large margin methods for structured and interdependent output variables. *JMLR*, 6(2), 2006.

Michael Wick, Khashayar Rohanimanesh, Kedar Bellare, Aron Culotta, and Andrew McCallum. Samplerank: Training factor graphs with atomic gradients. In *ICML*, 2011.

Chun-Nam John Yu and Thorsten Joachims. Learning structural SVMs with latent variables. In *ICML*, 2009.

Journal of Machine Learning Research 15 (2014) 2061-2086 Submitted 8/13; Revised 2/14; Published 6/14

The Student-t Mixture as a Natural Image Patch Prior with Application to Image Compression

Aäron van den Oord AARON.VANDENOORD@UGENT.BE
Benjamin Schrauwen BENJAMIN.SCHRAUWEN@UGENT.BE
Department of Electronics and Information Systems
Ghent University
Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium

Editor: Ruslan Salakhutdinov

Abstract

Recent results have shown that Gaussian mixture models (GMMs) are remarkably good at density modeling of natural image patches, especially given their simplicity. In terms of log likelihood on real-valued data they are comparable with the best performing techniques published, easily outperforming more advanced ones, such as deep belief networks. They can be applied to various image processing tasks, such as image denoising, deblurring and inpainting, where they improve on other generic prior methods, such as sparse coding and field of experts. Based on this we propose the use of another, even richer mixture model based image prior: the Student-t mixture model (STM). We demonstrate that it convincingly surpasses GMMs in terms of log likelihood, achieving performance competitive with the state of the art in image patch modeling. We apply both the GMM and STM to the task of lossy and lossless image compression, and propose efficient coding schemes that can easily be extended to other unsupervised machine learning models. Finally, we show that the suggested techniques outperform JPEG, with results comparable to or better than JPEG 2000.

Keywords: image compression, mixture models, GMM, density modeling, unsupervised learning

1. Introduction

Recently, there has been a growing interest in generative models for unsupervised learning. Especially latent variable models such as sparse coding, energy-based learning and deep learning techniques have received a lot of attention (Wright et al., 2010; Bengio, 2009). The research in this domain was for some time largely stimulated by the success of the models for discriminative feature extraction and unsupervised pre-training (Erhan et al., 2010). Although some of these techniques were advertised as better generative models, no experimental results could support these claims (Theis et al., 2011). Furthermore recent work (Theis et al., 2011; Tang et al., 2013) showed that many of these models, such as restricted Boltzmann machines and deep belief networks are outperformed by more basic models such as the Gaussian Mixture model (GMM) in terms of log likelihood on real-valued data.

Although arguably not as useful for the extraction of discriminative features, for the use of unsupervised pre-training, Gaussian mixture models have been shown to be very

successful in various image processing tasks, such as denoising, deblurring and inpainting (Zoran and Weiss, 2011; Yu et al., 2012). Good density models are essential for these tasks, and the log likelihood measure of these models has shown to be a good proxy for their performance. Apart from being simple and efficient, GMMs are easily interpretable methods which allow us to learn more about the nature of images (Zoran and Weiss, 2012).

In this paper we suggest the use of a similar, simple model for modeling natural image patches: the Student-t mixture model (STM). The STM uses multivariate Student-t distributed components instead of normally distributed components. We will show that a Student-t distribution, although having only one additional variable (the number of degrees of freedom), is able to model stronger dependencies than solely the linear covariance of the normal distribution, resulting in a large increase in log likelihood. Although a GMM is a universal approximator for continuous densities (Titterington et al., 1985), we will see that the gap in performance between the STM and GMM remains substantial, as the number of components increases.

Apart from comparing these methods with other published techniques for natural image modeling in terms of log likelihood, we will also apply them to image compression by proposing efficient coding schemes based on these models. Like other traditional image processing applications, it is a challenging task to improve upon the well-established existing techniques. Especially in data compression, which is one of the older, more advanced branches of computer science, research has been going on for more than 30 years. Most modern image compression techniques are therefore largely the result of designing data-transformation techniques, such as as the discrete cosine transform (DCT) and the discrete wavelet transform (DWT), and combining them with advanced engineered entropy coding schemes (Wallace, 1991; Skodras et al., 2001).

We will demonstrate that simple unsupervised machine learning techniques such as the GMM and STM are able to perform quite well on image compression, compared with conventional techniques such as JPEG and JPEG 2000. Because we want to measure the density-modeling capabilities of these models, the amount of domain-specific knowledge induced in the proposed coding schemes is kept to a minimum. This also makes it relevant from a machine learning perspective as we can more easily apply the same ideas to other types of data such as audio, video, medical data, or more specific kinds of images, such as satellite, 3D and medical images.

In Section 2 we review some work on compression, in which related techniques were used. In Section 3 we give the necessary background for this paper on the GMM and STM and the expectation-maximization (EM) steps for training them. We will also elaborate on their differences and the more theoretical aspects of their ability to model the distribution of natural image patches. In Section 4 we present the steps for encoding/decoding images with the use of these mixture models for both lossy and lossless compression. The results and their discussion follow in Section 5. We conclude in Section 6.

2. Related Work

In this section we will review related work on image compression and density modeling.

2.1 Image Compression

The coding schemes (see section 4) we use to compare the GMM and STM, can be related with other published techniques in image compression, in the way they are designed. Although little research has been done on the subject we briefly review work based on vector quantization, sparse coding and subspace clustering. The lossy coding scheme we describe in this paper is based on a preliminary version that appeared in our previous work (van den Oord et al., 2013).

In vector quantization (VQ) literature, GMMs have been proposed for the modeling of low-dimensional speech signal parameters (Hedelin and Skoglund, 2000). In this setting, the GMMs' probability density function is suggested to be used to fit a large codebook of VQ centroids on (e.g., with a technique similar to k-means), instead of on the original data set. They were introduced to help against overfitting, which is a common problem with the design of vector quantizers when the training set is relatively small compared to the size of the codebook. The same idea has also been suggested for image compression (Aiyer et al., 2005). In contrast to these approaches we will apply a (learned) data transformation in combination with simple *scalar* uniform quantization, which reduces the complexity considerably given the relatively high dimensionality of image patches. This idea called transform coding (Goyal, 2001) is widely applied in most common image compression schemes, which use designed data-transforms such as the DCT and DWT.

By some authors (Hong et al., 2005) image compression has been suggested based on a subspace clustering model. The main contribution was a piecewise linear transformation for compression, which was also extended to a multiscale method. This is by some means similar to our lossy compression scheme as we also apply a piecewise linear transform, but based on the GMM/STM instead of a subspace clustering technique. They did not suggest quantization or entropy coding steps, and therefore only evaluated their approach in terms of energy compaction instead of rate-distortion.

Image compression based on sparse coding has been proposed (Horev et al., 2012) for images in general (Bryt and Elad, 2008; Zepeda et al., 2011) and for a specific class of facial images. Aside from being another unsupervised learning technique, sparse coding has been related with GMM in another way: Some authors (Yu et al., 2012; Zoran and Weiss, 2011) have suggested the interpretation of a GMM as a structured sparse coding model. This idea is based on the observation that data can often be represented well by one of the N Gaussian mixture components, thus when combining all the eigenvectors of their covariance matrices as an overcomplete dictionary, the sparsity is $\frac{1}{N}$. The main results in Horev et al. (2012) show that sparse coding outperforms JPEG, but it does not reach JPEG 2000 performance for a general class of images.

2.2 Models of Image Patches

Sparse coding approaches (Olshausen and Field, 1997) have also been successfully applied as an image prior on various image reconstruction tasks, such as denoising and demosaicing (Elad and Aharon, 2006; Mairal et al., 2009). These models have recently been shown to be outperformed by the GMM in both image denoising (Zoran and Weiss, 2011) as density modeling (Zoran and Weiss, 2012).

The Fields of Experts (FoE) framework is another approach for learning priors that can be used for image processing applications (Roth and Black, 2005; Weiss and Freeman, 2007). In a FoE, the linear filters of a Markov random field (MRF) are trained to maximize the log-likelihood of whole images in the training set. This optimization is done approximately with contrastive divergence, as computing the log likelihood itself is intractable. The potential functions that are used in the MRF are represented by a product of experts (PoE) (Hinton, 2002). The FoE is commonly used for image restoration tasks such as denoising and inpainting, but was also recently outperformed by GMMs with the expected patch log likelihood framework (EPLL) (Zoran and Weiss, 2012).

Recently similar models to the GMM have been proposed for image modeling, such as the Deep mixture of Factor analyzers (Deep MFA) (Tang et al., 2012). This technique is a deep generalization of the Mixture of Factor Analyzers model, which is similar to the GMM. The deep MFA has a tree structure in which every node is a factor analyzer, which inherits the low-dimensional latent factors from its parent.

Another model related to the GMM and STM is the Mixture of Gaussian scale mixtures (MoGSM) (Theis et al., 2011, 2012). Instead of a Gaussian, every mixture component is a Gaussian scale mixture distribution. The MoGSM has been used for learning multi-scale image representations, by modeling each level conditioned on the higher levels.

RNADE, a new deep density estimation technique for real valued data has a very different structure (Uria et al., 2013b,a). RNADE is an extension of the NADE technique for real-valued data, where the likelihood function is factored into a product of conditional likelihood functions. Each conditional distribution is fitted with a neural mixture density network, where one variable is estimated, given the other ones. Recently a new training method has allowed a factorial number of RNADE's to be trained at once within one model. It is currently one of the few deep learning methods with good density estimation results on real-valued data and is the current state of the art on image patch modeling.

3. Mixture Models as Image Patch Priors

Mixture models are among the most widely accepted methods for clustering and probability density estimation. Especially GMMs are well known and have widespread applications in different domains. However depending on the data used, other mixture models might be more suitable.

In this work we will denote the mixture component distribution as f_k and the mixture distribution as

$$f(x) = \sum_{k=1}^{K} \pi_k f_k(x),$$

where $\pi_k, k = 1 \dots K$ are the mixing weights. The two component distributions we study here are the multivariate normal distribution:

$$f_k(x) = \mathcal{N}(x|\mu_k, \Sigma_k) = (2\pi)^{-\frac{p}{2}} |\Sigma_k|^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)},$$

and the multivariate Student-t distribution, see Equation 1. In these equations, p is the dimensionality of x. We will train the GMM with the EM-algorithm: an iterative algorithm

for finding the maximum likelihood estimate of the parameters. For completeness we summarize the expectation and maximization steps for training a GMM with EM.
E-step:

$$\gamma_{nk} = \frac{\pi_k \mathcal{N}\left(x_n | \mu_k, \Sigma_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(x_n | \mu_j, \Sigma_j\right)}.$$

M-step:

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} \gamma_{nk}, \qquad \mu_k = \frac{\sum_{n=1}^{N} \gamma_{nk} x_n}{\sum_{n=1}^{N} \gamma_{nk}},$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} \gamma_{nk} \left(x_n - \mu_k\right) \left(x_n - \mu_k\right)^T}{\sum_{n=1}^{N} \gamma_{nk}}.$$

One of the important reasons GMMs excel at modeling image patches is that the distribution of image patches has a multimodal landscape. A unimodal distribution such as the multivariate normal distribution is not able to capture this. When using a mixture however, each component can represent a different aspect or texture of the whole distribution. We can observe this by looking at the individual mixture components of a trained GMM model, see Figure 1.

Next to modeling different textures, the GMM also captures differences in contrast. It has been shown (Zoran and Weiss, 2012) that multiple components in the GMM describe a similar structure in the image, but each with a different level of contrast. The STM, however, can model different ratios of contrast within a single mixture component.

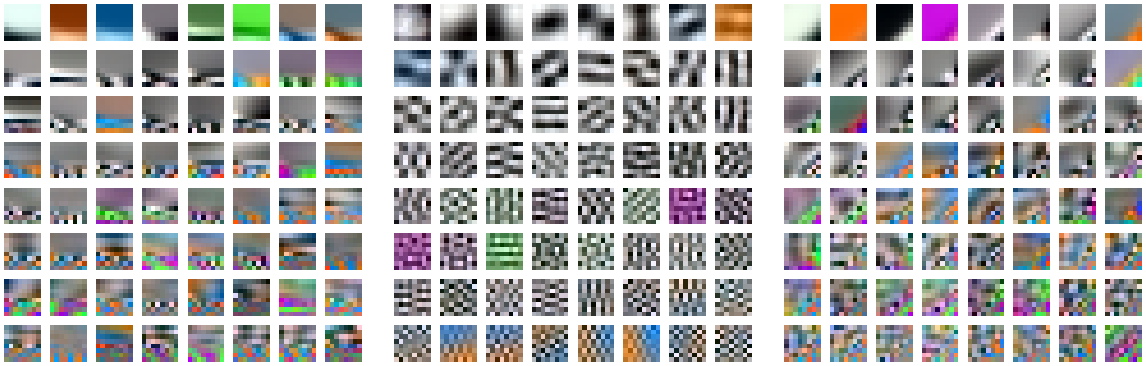
A multivariate Student-t distribution has the following density function (Kotz and Nadarajah, 2004):

$$\mathcal{T}\left(x | \nu, \mu, \Sigma\right) = \frac{\Gamma\left(\frac{\nu+p}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right) \nu^{\frac{p}{2}} \pi^{\frac{p}{2}} \left|\Sigma\right|^{1/2}} \left[1 + \frac{1}{\nu}\left(x-\mu\right)^T \Sigma^{-1} \left(x-\mu\right)\right]^{-\frac{\nu+p}{2}}. \tag{1}$$

ν is an additional parameter which represents the number of degrees of freedom. Note that for $\nu \to \infty$ the Student-t distribution converges to the normal distribution. It is interesting to see how this distribution is constructed:
If Y is a multivariate normal random vector with mean 0 and covariance Σ, and if νT is a chi-squared random variable with degrees of freedom ν, independent of Y, and

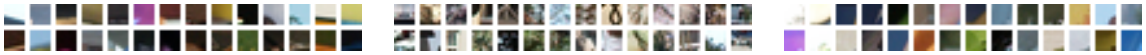
$$X = \frac{Y}{\sqrt{T}} + \mu,$$

then X has a multivariate Student-t distribution with degrees of freedom ν, mean μ, and covariance matrix Σ. This also means $X|T = \tau$ is normally distributed with mean μ and covariance $\frac{\Sigma}{\tau}$. In the setting of modeling image patches, T can be interpreted to model the

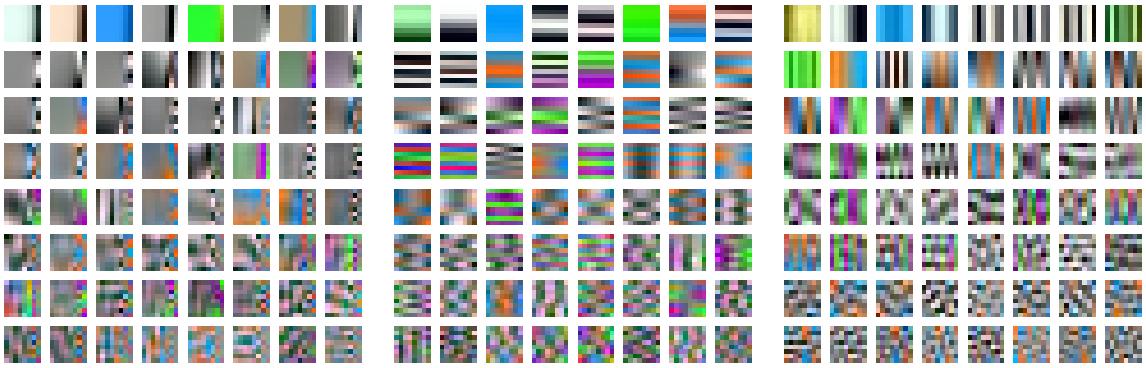


The first 64 eigenvectors of the component's covariance matrix.

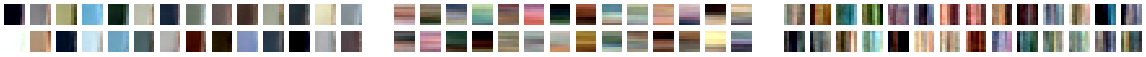
Patches generated by sampling from the component distribution.



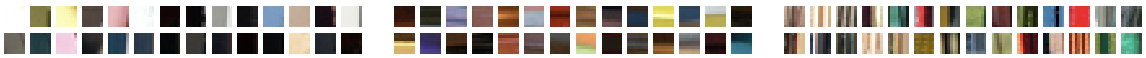
Examples from the train set that are best represented by this component.



The first 64 eigenvectors of the component's covariance matrix.



Patches generated by sampling from the component distribution.

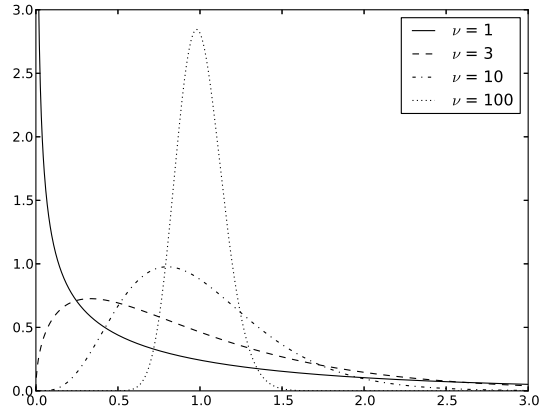


Examples from the train set that are best represented by this component.

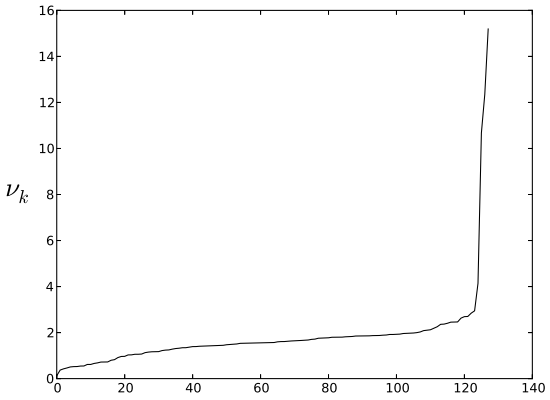
Figure 1: Six mixture components of the GMM are visualized here (the STM gives similar texture patterns). We first show the eigenvectors of the covariance matrix of each component, which show the structure of the image patches that the mixture component learns. These eigenvectors are sorted by their respective eigenvalues from large to small (left to right and top to bottom). Only the first 64 of 192 are shown. Next we show some samples that are generated by each component, and some examples from the train set that are best represented with this component (clustered with the GMM). Note that every component has specialized in a different aspect or texture, and that the samples generated by the component distributions are very similar to the real image patches. This figure is best viewed in color on the electronic version.

variety of contrast, for a given texture. The distribution of T is visualized in Figure 2(a), for different values of ν. If ν is small for a given component (texture), this means that the texture appears in natural images in a wide range of contrast. For $\nu \to \infty$ we get a Gaussian distribution and its contrast is more constrained. To obtain the same capacity with a GMM, one would need multiple components having scaled versions of the same covariance matrix.

In Figure 2(b) the value of ν is visualized for different components of a trained STM. This value differs substantially for each component, ranging from almost zero to 15. This means some component-distributions are very long-tailed (with small ν) and some are more normally distributed (higher ν). This means that some texture patterns appear in a wider range of contrast than others. However, in our experiments we saw that the STM does not learn significantly different structures compared to the GMM. The texture patterns learned by the STM were also very similar to those shown in Figure 1. This means the STM is better at generalizing to image patches with different levels of contrast, but might not be better at generalizing to different unseen texture patterns.



(a) The distribution of T in Equation 2, for different values of ν. As ν increases the distribution becomes more peaked and converges to a Dirac delta at 1.



(b) The value of ν_k for each component k of a trained STM with 128 components, sorted from low to high.

Figure 2: The distribution of T for different ν and the value of ν for different mixture components.

Given the fact that a GMM is universal approximator for continuous densities, the question that remains is if a STM still has the advantage over the GMM when the number of components increases. To this end we have trained a GMM and STM on a set of image patches for different numbers of mixture components and computed their log likelihood scores on a validation set, see Figure 3(a). Notice that the performance of a single Student-t is much better than that of a single Gaussian, and close to that of a GMM with K=4. This is in agreement with previously reported findings (Zoran and Weiss, 2012), which suggest that a GMM with a small number of components mainly learns contrast. Next we see that as N increases the gap in performance between the STM and GMM remains substantial. The most plausible explanation for this behavior is that the GMM needs more mixture components than the STM to have the same contrast modeling capabilities. However,

with more mixture components the risk of overfitting also increases. If one would tie the parameters of some of these components together, so that they have scaled versions of the same covariance matrix, the risk of overfitting would decrease. This is exploited in the mixture of Gaussian scale mixtures (MoGSM) (Theis et al., 2012).

The idea of explicitly sharing covariance parameters between mixture components has also been applied to mixtures of factor analyzers, with the deep MFA model (Tang et al., 2012). They proposed a hierarchical structure in which the mixture components partially inherit the covariance structure of their parent in the hierarchy.

The Student-t has previously been used for modeling image patches in the PoE framework (Welling et al., 2002), where each expert models a differently linearly filtered version of the input with a univariate Student-t distribution.

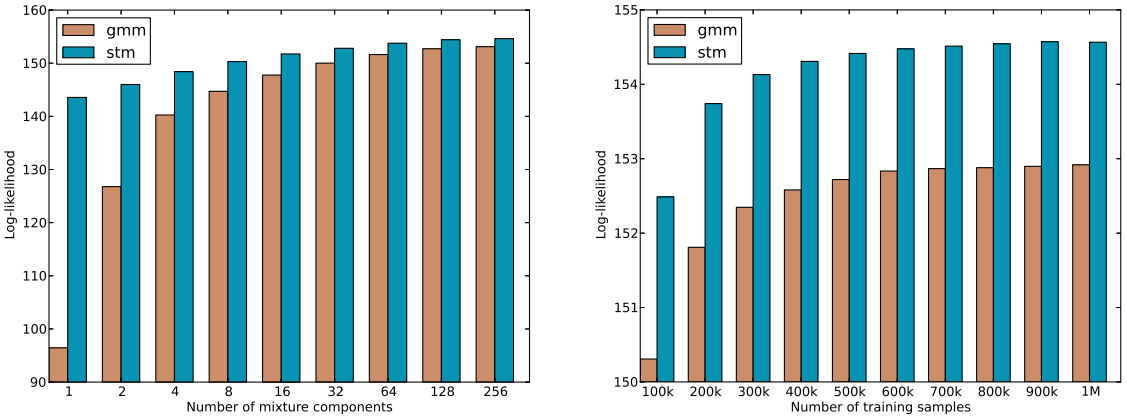


Figure 3: The average patch log likelihood for the Gaussian mixture model (GMM) and Student-t mixture model (STM) in function of its number of mixture components (a) and number of training samples (b). The models were trained on 8x8 normalized gray scale image patches, extracted from the Berkeley data set (see Section 5.1.1).

We also train the STM with the EM algorithm (Peel and McLachlan, 2000; Dempster et al., 1977):
E-step:

$$\gamma_{nk} = \frac{\pi_k \mathcal{T}\left(x_n | \nu_k, \mu_k, \Sigma_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{T}\left(x_n | \nu_j, \mu_j, \Sigma_j\right)}, \quad w_{nk} = \frac{\nu_k + p}{\nu_k + (x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k)}.$$

M-step:

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} \gamma_{nk}, \qquad \mu_k = \frac{\sum_{n=1}^{N} \gamma_{nk} w_{nk} x_n}{\sum_{n=1}^{N} \gamma_{nk} w_{nk}}.$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} \gamma_{nk} w_{nk} \left(x_n - \mu_k\right) \left(x_n - \mu_k\right)^T}{\sum_{n=1}^{N} \gamma_{nk}}.$$

For the degrees of freedom, there is no closed form update rule. Instead ν_k gets updated as the solution of:

$$\begin{aligned} &- \psi\left(\frac{\nu_k}{2}\right) + \log\left(\frac{\nu_k}{2}\right) + 1 + \frac{1}{\alpha_k} \sum_{n=1}^{N} \gamma_{nk} \left(\log\left(w_{nk}\right) - w_{nk}\right) \\ &+ \psi\left(\frac{\tilde{\nu_k} + p}{2}\right) - \log\left(\frac{\tilde{\nu_k} + p}{2}\right) = 0, \end{aligned}$$

where $\tilde{\nu_k}$ is the value of the current ν_k, $\alpha_k = \sum_{n=1}^{N} \gamma_{nk}$ and ψ is the digamma function. This scalar non-linear equation can be solved quickly with a root finding algorithm, such as Brent's method (Brent, 1973).

Note that the expectation and maximization steps are quite similar to those of the GMM. In our experiments, it did not take substantially longer to train a STM than a GMM. Typically 100 iterations were enough to train the STM or GMM, even though the log likelihood does keep improving a little bit after that (even after 500 iterations). For a big mixture model of 256 components, trained on 500.000 samples of 8x8 gray scale patches, this took about 20 hours on a standard desktop computer with four cores. For the STM, it took 21 hours. On this scale, the CPU time is linear in both the number of training samples and the number of components. Training on image patches proved to be quite stable: no components needed to be reinitialized during training.

The code for training a Student-t mixture is included in the supplementary material of this paper.

4. Compression with Mixture Models

Both the lossy and lossless algorithms we propose are patch/block based. This means they will encode each patch of an image separately. During training we randomly sample a large set of image patches from the training images. These are used to fit the GMM and STM models. Once training is finished, these density models can be used to encode the test images. Each test image is viewed as a grid of non-overlapping patches. The encoder loops over all patches, which are extracted, flattened and encoded one by one.

To speed up the algorithms, each patch will be encoded using the distribution and parameters of only one of the mixture components. We choose the mixture component which represents the given patch with the highest likelihood:

$$\beta = \arg\max_k f_k\left(x_n\right).$$

This will only slightly reduce the performance, because the "overlap" between the individual mixture components is relatively small. We can easily validate this with a simple intermediate experiment. In Table 1 we have computed the log likelihood for a trained GMM and STM

	GMM	STM
Log likelihood	152.86	154.51
Highest mixture component log likelihood	152.66	154.15

Table 1: Average patch log likelihood compared with the average highest component patch log likelihood: How well can a sample be represented by using a single mixture component? (See text)

on a validation set. We have also computed the average log likelihood when only one of the mixture components is used for each example: $\frac{1}{N}\sum_{n=1}^{N}\log\left(\max_k\left(\pi_k f_k\left(x_n\right)\right)\right)$. Note that this is strictly lower than the actual average log likelihood: $\frac{1}{N}\sum_{n=1}^{N}\log\left(\sum_{k=1}^{K}\pi_k f_k\left(x_n\right)\right)$. But as can be seen from Table 1, the difference is small.

4.1 Arithmetic Coding

Most commonplace image compression schemes follow three main steps: transformation, quantization and entropy coding (Goyal, 2001). Transformation decorrelates the data, quantization maps the values of the decorrelated continuous variables onto discrete values from a relatively small set of symbols (such as integers) and entropy coding encodes these discrete quantized values into a bit sequence. In this paper, transformation and quantization will only be used for lossy compression and not for lossless compression. However, in both cases we employ arithmetic coding (AC) for the entropy coding step.

Entropy coding is a family of algorithms that take as input a sequence of discrete values, and give as output the encoded binary sequence. Based on the statistical properties of the input, the goal is to minimize the expected length of the bit sequence (e.g., by assigning more bits to a rare symbol and less bits to a common symbol). The theoretical limit of the encoding scheme is bounded by the entropy of the input signal, which explains the name entropy coding. Arithmetic coding is a form of entropy coding, which requires a list of probabilities $\alpha_i, i = 1 \ldots N$ that describe the discrete distribution $P\left(s_j\right) = \alpha_j$ of a symbol s_j occurring in an input sequence. Based on these probabilities, the algorithm will on average spend fewer bits on common symbols, than on rare ones. However, with AC it is also possible to use different probabilities for each time step t in the sequence: $P\left(s_j^{(t)}\right) = \alpha_{jt}$, and even adapt them during the encoding/decoding based on the values of the previously encoded symbols. This is also called adaptive arithmetic coding.

4.2 Lossless Compression

In lossless compression, the image should be preserved perfectly so that after decompression the output image is identical to the input image. Because we have a probabilistic model for an image patch, the most natural way to approach this task is to use lossless predictive coding (Pearlman and Said, 2011). The idea is to predict the value (integer) of each sample within an image patch, using the values of its neighboring samples that are already encoded, based on the correlations between them. In this case, the prediction will actually consist of

a discrete probability distribution over the possible values of the current sample, which can directly be used to perform arithmetic coding.

To carry out arithmetic coding on a patch x_i, one needs to compute a list of probabilities (probability table) for each of its elements $x_{i,j}$: $P(x_{i,j} = l)$ for $l = 0 \dots L$. More specifically, because arithmetic coding can adapt the probability tables to the information of the previous symbols $x_{i,1} \dots x_{i,j-1}$ it is possible to encode every symbol conditionally with respect to the ones already encoded: $P(x_{i,j} = l | x_{i,1} \dots x_{i,j-1})$. As the image patches are modeled by continuous probability densities, this can computed as follows:

$$P(x_{i,j} = l | x_{i,1} \dots x_{i,j-1}) = \int_{l-\frac{1}{2}}^{l+\frac{1}{2}} f\left(x_{i,j} | x_{i,1} \dots x_{i,j-1}\right) \, \mathrm{d}x_{i,j}. \tag{2}$$

This scheme for performing lossless image compression can be used in combination with any density model, provided that we can compute Equation 2. This way arithmetic coding can be applied to the image using the statistics of the trained model. Algorithm 1 gives a summary for lossless compression with a mixture model.

As already mentioned, when using an mixture model, it is more efficient to use a single component for the encoding of a patch than the whole mixture. For both normally and Student-t distributed variables, the expressions for Equation 2 can be derived from their conditional distributions.
For the normal distribution this becomes:

$$\int_{l-\frac{1}{2}}^{l+\frac{1}{2}} \mathcal{N}\left(x_{i,j} | x_{i,1} \dots x_{i,j-1}\right) \, \mathrm{d}x_{i,j} = F_n\left(l + \frac{1}{2} | \tilde{\mu_j}, \tilde{\sigma_j}^2\right) - F_n\left(l - \frac{1}{2} | \tilde{\mu_j}, \tilde{\sigma_j}^2\right),$$

with F_n the cumulative distribution function (CDF) of the univariate normal distribution, and where

$$\tilde{\mu_j} = \mu_j + \Sigma_{j,1:j-1} \Sigma^{-1}_{1:j-1,1:j-1} \left(x_{1:j-1} - \mu_{1:j-1}\right),$$
$$\tilde{\sigma_j}^2 = \Sigma_{j,j} - \Sigma_{j,1:j-1} \Sigma^{-1}_{1:j-1,1:j-1} \Sigma_{1:j-1,j}.$$

For the multivariate Student-t distribution the equations are similar:

$$\int_{l-\frac{1}{2}}^{l+\frac{1}{2}} \mathcal{T}\left(x_{i,j} | x_{i,1} \dots x_{i,j-1}\right) \, \mathrm{d}x_{i,j} = F_t\left(l + \frac{1}{2} | \tilde{\nu_j}, \tilde{\mu_j}, \tilde{s_j}^2\right) - F_t\left(l - \frac{1}{2} | \tilde{\nu_j}, \tilde{\mu_j}, \tilde{s_j}^2\right),$$

with F_n the CDF of the non-standardized univariate Student-t distribution (which has a location and scale parameter), and where

$$\tilde{\nu_j} = \nu_j + j - 1,$$
$$\tilde{\mu_j} = \mu_j + \Sigma_{j,1:j-1} \Sigma^{-1}_{1:j-1,1:j-1} \left(x_{1:j-1} - \mu_{1:j-1}\right),$$
$$\tilde{s_j}^2 = \left(\frac{\nu + x^T_{1:j-1} \Sigma^{-1}_{1:j-1,1:j-1} x_{1:j-1}}{\nu + j - 1}\right) \left(\Sigma_{j,j} - \Sigma_{j,1:j-1} \Sigma^{-1}_{1:j-1,1:j-1} \Sigma_{1:j-1,j}\right).$$

When using a form of entropy coding, such as arithmetic coding, the theoretical optimal code length for a symbol i is dependent on the probability P_i of it occurring: $-\log\left(P_i\right)$. Therefore, the lower bound on the expected rate (bits per symbol) is: $-\frac{1}{N} \sum_{i=1}^{N} P_i \log\left(P_i\right)$. Because P_i is calculated by a density model (Equation 2), the log likelihood score of this model is a good indication for how well it performs on lossless compression.

Algorithm 1: Lossless image compression with a mixture model. [AC] stands for arithmetic coding.

Encoder:

for *each patch* x_i *in image* **do**
  $\beta = \arg\max_k f_k(x_i)$
  [AC] Encode symbol β with probability table π (mixing weights)
  for *each* $x_{ij},\ j = 1 \dots p$ **do**
    Use Equation 2 to compute: $\alpha_{i,j,l} = P_\beta(x_{i,j} = l | z_{i,1} \dots x_{i,j-1})$
    [AC] Encode symbol x_{ij} with probability table $\alpha_{i,j}$
  end
end

Decoder:

while *not at end of bit stream* **do**
  [AC] Decode symbol β with probability table π (mixing weights)
  initialize x_i
  for $j = 1 \dots p$ **do**
    Use Equation 2 to compute: $\alpha_{i,j,l} = P_\beta(x_{i,j} = l | z_{i,1} \dots x_{i,j-1})$
    [AC] Decode symbol x_{ij} with probability table $\alpha_{i,j}$
  end
end
Reconstruct image from patches $x_i,\ i = 1 \dots N$.

4.3 Lossy Compression

For lossy compression, the image reconstruction after decompression does not have to be identical to the original, but should match it very closely. The strength of compression should be as high as possible, given a certain tolerable amount of distortion. This freedom evidently allows stronger compression than with lossless algorithms.

Lossy image compression algorithms typically use quantization to reduce the amount of information that needs to be entropy coded. Quantization decreases the number of states of the data variables to a smaller discrete set. As mentioned above we will use simple scalar quantization as the number of variables in a patch is relatively high and vector quantization would simply be impractical. Instead of VQ, we will combine scalar quantization with a data transform step, as is done in most image compression schemes.

The main reason of a data transform step in compression schemes is to decorrelate the input, so that the different coefficients can be handled more independently afterwards. Especially when using scalar quantization it is important to use a form of transformation first, as this reduces the amount of redundancy in the data that has to be encoded. Moreover, if one would quantize the image in the original pixel domain, the reconstruction artifacts would be very obtrusive. Because a Gaussian or Student-t mixture component already models covariance, decorrelation is fairly straightforward. The transform step is as follows:

$$y_i = W^T (x_i - \mu), \tag{3}$$

where W is the eigenvector matrix of the covariance matrix Σ of the Gaussian/Student-t mixture component: $WJW^T = \Sigma$. J is the diagonal eigenvalue matrix of Σ. Subsequently, the transformed values are quantized with a uniform quantizer:

$$z_i = \text{round}\left(\frac{y_i}{\lambda}\right). \tag{4}$$

The strength of the quantization only depends on λ. When it is high, the quality of the encoded image will be low, but the compression ratio will be high.

Once quantization is done, arithmetic coding is carried out in a similar fashion as with lossless compression: we have to be able to compute Equation 2. Because the data is transformed (Equation 3), the mean of y becomes 0: $\mu_y = 0$ and the covariance matrix reduces to: $\Sigma_y = J$. The equations for calculating the conditional probabilities from before can now be simplified.
For the normal distribution:

$$\begin{aligned} P(z_{i,j} = l | z_{i,1}, \ldots, z_{i,j-1}) &= \int_{\lambda\left(l-\frac{1}{2}\right)}^{\lambda\left(l+\frac{1}{2}\right)} \mathcal{N}(y_{i,j} | \tilde{y}_{i,1} \ldots \tilde{y}_{i,j-1}) \, \mathrm{d}y_{i,j} \\ &= F_n\left(\lambda\left(l+\frac{1}{2}\right) | 0, J_j\right) - F_n\left(\lambda\left(l-\frac{1}{2}\right) | 0, J_j\right), \end{aligned} \tag{5}$$

with F_n the cumulative distribution function (CDF) of the univariate normal distribution, and $\tilde{y}_{i,*}$ is the reconstruction of $y_{i,*}$ (as we will discuss later).

For the Student-t distribution:

$$P\left(z_{i,j}=l|z_{i,1},\ldots,z_{i,j-1}\right) = \int_{\lambda\left(l-\frac{1}{2}\right)}^{\lambda\left(l+\frac{1}{2}\right)} \mathcal{T}\left(y_{i,j}|\tilde{y}_{i,1}\ldots\tilde{y}_{i,j-1}\right)\,\mathrm{d}y_{i,j} \tag{6}$$
$$= F_t\left(\lambda\left(l+\frac{1}{2}\right)|\tilde{\nu_j},0,\tilde{s_j}^2\right) - F_t\left(\lambda\left(l-\frac{1}{2}\right)|\tilde{\nu_j},0,\tilde{s_j}^2\right),$$

with F_t the CDF of the non-standardized univariate Student-t distribution (which has a location and scale parameter), and where

$$\tilde{\nu_j} = \nu_j + j - 1,$$
$$\tilde{s_j} = \left(\frac{\nu_j + \sum_{m=1}^{j-1}\frac{\tilde{y}_{i,m}^2}{J_m}}{\nu_j + j - 1}\right) J_j.$$

Because of the two additional steps (Transformation and Quantization) during compression, the decoder has to dequantize and subsequently detransform the data after arithmetic coding.
Dequantization:

$$\tilde{y_i} = \lambda z_i. \tag{7}$$

Inverse transform:

$$\tilde{x_i} = W\tilde{y_i} + \mu. \tag{8}$$

4.3.1 Uniform Threshold Quantization

It is important to note that Equation 7 might not be the best choice for reconstruction. It is indeed possible to increase the quality of dequantization by using prior knowledge of the scalar input distribution. This concept is called uniform threshold quantization (Pearlman and Said, 2011). Figure 4 shows the difference with regular uniform quantization.

Depending on the assumed distribution of the source it is possible to minimize the expected distortion: $(\tilde{y}_{i,j} - y_{i,j})^2$ (other measures of distortion can also be used). This comes down to solving the following optimization problem:

$$\tilde{y}_{i,j} = \arg\min_y \int_{\lambda\left(z_{i,j}-\frac{1}{2}\right)}^{\lambda\left(z_{i,j}+\frac{1}{2}\right)} \|y-x\|^2 f(x)\,\mathrm{d}x,$$

which can be simplified to:

$$\tilde{y}_{i,j} = \frac{\int_{\lambda\left(z_{i,j}-\frac{1}{2}\right)}^{\lambda\left(z_{i,j}+\frac{1}{2}\right)} x f(x)\,\mathrm{d}x}{\int_{\lambda\left(z_{i,j}-\frac{1}{2}\right)}^{\lambda\left(z_{i,j}+\frac{1}{2}\right)} f(x)\,\mathrm{d}x}.$$

This is actually nothing more than the centroid in that region (see Figure 4). Because we are using a probabilistic method, this improved reconstruction almost comes for free: The

Algorithm 2: Lossy image compression with a mixture model. [AC] stands for arithmetic coding.

Encoder:

for *each patch* x_i *in image* **do**
  $\beta = \arg\max_k f_k(x_i)$
  [AC] Encode symbol β with probability table π (mixing weights)
  Transform x_i with Equation 3 using the β-th component
  Quantize x_i with Equation 4
  for *each* $x_{ij},\ j = 1 \dots p$ **do**
    Use Equation 5 or 6 to compute: $\alpha_{i,j,l} = P_\beta(x_{i,j} = l | x_{i,1} \dots x_{i,j-1})$
    [AC] Encode symbol x_{ij} with probability table $\alpha_{i,j}$
  end
end

Decoder:

while *not at end of bitstream* **do**
  [AC] Decode symbol β with probability table π (mixing weights)
  initialize x_i
  for $j = 1 \dots p$ **do**
    Use Equation 5 or 6 to compute: $\alpha_{i,j,l} = P_\beta(x_{i,j} = l | x_{i,1} \dots x_{i,j-1})$
    [AC] Decode symbol x_{ij} with probability table $\alpha_{i,j}$
  end
  Dequantize x_i with Equation 7 or 9
  Inverse transform x_i with Equation 8
end
Reconstruct image from patches $x_i,\ i = 1 \dots N$.

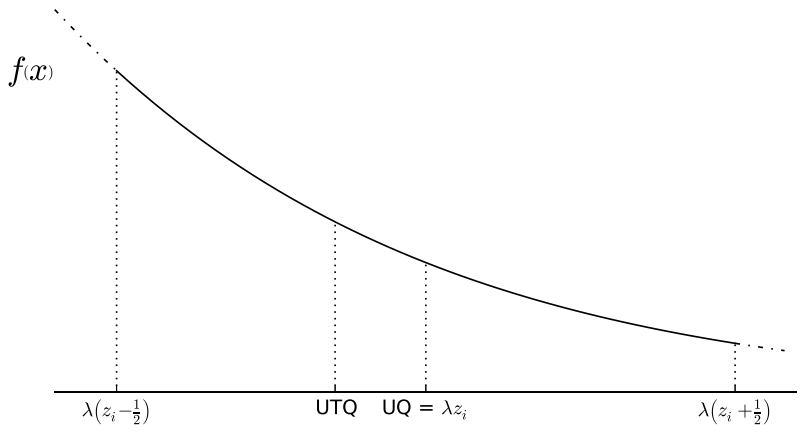


Figure 4: Uniform quantization versus uniform threshold quantization. During dequantization, UQ will reconstruct the input with the centers of the quantization intervals. UTQ uses the centroids instead.

compression scheme remains the same, only the decompression is improved. For a normally distributed variable the reconstruction is

$$\tilde{y}_{i,j} = \frac{\sqrt{\frac{J_j}{2\pi}}\left[\exp\left(-\frac{\lambda^2\left(z_{i,j}-\frac{1}{2}\right)^2}{2J_j}\right) - \exp\left(-\frac{\lambda^2\left(z_{i,j}+\frac{1}{2}\right)^2}{2J_j}\right)\right]}{F_n\left(\lambda\left(z_{i,j}+\frac{1}{2}\right)|0,J_j\right) - F_n\left(\lambda\left(z_{i,j}-\frac{1}{2}\right)|0,J_j\right)},$$

and for a Student-t distributed variable it is

$$\tilde{y}_{i,j} = \frac{\frac{\Gamma\left(\frac{\tilde{\nu}_j+1}{2}\right)}{\sqrt{\pi}\Gamma\left(\frac{\tilde{\nu}_j}{2}\right)}\frac{\tilde{s}_j^{\tilde{\nu}_j}\tilde{\nu}_j^{\frac{\tilde{\nu}_j}{2}}}{\tilde{\nu}_j-1}\left[\left(\tilde{\nu}_j\tilde{s}_j^2+\lambda^2\left(z_{i,j}-\frac{1}{2}\right)^2\right)^{\frac{1-\tilde{\nu}_j}{2}} - \left(\tilde{\nu}_j\tilde{s}_j^2+\lambda^2\left(z_{i,j}+\frac{1}{2}\right)^2\right)^{\frac{1-\tilde{\nu}_j}{2}}\right]}{F_t\left(\lambda\left(z_{i,j}+\frac{1}{2}\right)|\tilde{\nu}_j,0,\tilde{s_j}^2\right) - F_t\left(\lambda\left(z_{i,j}-\frac{1}{2}\right)|\tilde{\nu}_j,0,\tilde{s_j}^2\right)}.$$

5. Results and Discussion

In this section we will discuss the experimental results of the STM on density modelling and image compression tasks.

5.1 Data Sets and Methods

We will first introduce the data sets that we used for our experiments and also discuss the image compression standards (JPEG, JPEG 2000) which will be used to compare the compression results with.

5.1.1 Berkeley Segmentation Data Set

The Berkeley Segmentation Data set (Martin et al., 2001) consists of 200 training and 100 test JPEG-encoded images, originally intended to be used as a segmentation benchmark.

Some samples can be seen on Figure 5. This data set has been used by several authors to measure the unsupervised learning performance of their model on image patches (Zoran and Weiss, 2011; Tang et al., 2013; Uria et al., 2013a). We adopt the use of this data set for measuring density modeling performance, but not for image compression, as these images were already encoded with JPEG and will already contain some quantization noise.

5.1.2 UCID Data Set

Although images are abundant on the world wide web, large data sets containing losslessly encoded images are rather hard to find. In image processing most authors have grown to rely on a particular set of standard images, such as Lena, Baboon, Peppers, etc. [1] to measure their algorithms' performance. Although each of these images have specific features that make them interesting to test a new method on, results on this small set likely will not generalize to a wide range of images. Furthermore, because there is no clear distinction between a training and test set for these images, there is a high risk of overfitting (even when engineering a compression scheme). Finally most of these images are relatively old and noisy, so they are hardly representative for images of modern photography.

On of the few publicly available data sets is UCID (Schaefer and Stich, 2003) (Uncompressed Colour Image Data set). The UCID database consists of 1338 TIFF images on a variety of topics including natural scenes and man-made objects, both indoors and outdoors. The camera settings were all set to automatic as this resembles what the average user would do. All the images have sizes 512x384 or 384x512. The images are in true color (24-bit RGB, each color channel having 256 possible values per pixel). Some sample UCID images can be seen on Figure 6.

As the images are not in random order, we have included every 10th image (10, 20, 30, ..., 1330) of the data set in our test set, and the others were used for training. This results in 1205 images for training and 133 images for testing. We randomly sample a large set of image patches (two million) for training the mixture models. We then encode every test image with a number of different quantization strengths (only for lossy compression), and measure their compression performance and the distortion of their reconstruction.

5.1.3 JPEG and JPEG 2000

For comparison we added two image compression standards as benchmark: JPEG (Wallace, 1991) and JPEG 2000 (Skodras et al., 2001).

JPEG is a patch based compression standard which uses the DCT as its transform, with quantization and entropy coding optimized for this transform. For the JPEG standard, we employed the widely used libjpeg implementation (`ijg.org`). Optimization of the JPEG entropy encoding parameters was enabled for better performance. The quality parameter was swept from 0 (worst) to 100 (best) in steps of 5.

JPEG 2000 is a wavelet-based compression standard and because of its multiresolution decomposition structure, it is able to exploit wider spatial correlations than JPEG and our method (which are patch based). For JPEG 2000, the kakadu implementation was

1. Most of these standard images can be found here: `http://sipi.usc.edu/database/database.php?volume=misc`

used (`kakadusoftware.com`). To make a fair comparison, command line parameters were enabled to optimize the PSNR instead of perceptual error measures.

For both methods we did not take the meta information of the header into account when measuring the performance of compression.

Figure 5: Sample images from the Berkeley Segmentation Data set.

Figure 6: Sample images from the UCID data set (Uncompressed Colour Image Data set).

5.2 Average Patch Log Likelihood Comparison

Two million 8x8 patches were extracted from the training images, and 50,000 were extracted from the testing images (Berkeley data set). For every sample the mean was subtracted (DC component). Because the test patches we extracted could differ from those used by other authors, we report the mean and standard deviation across 10 randomly sampled sets of 50,000 patches. The results are listed in Table 2. The GMM and STM had 200 mixture components. As expected our GMM has a comparable result to that reported in literature. The proposed method STM significantly outperforms other methods.

We also compare our result with the recently proposed RNADE model (Uria et al., 2013b,a). Because the authors preprocess the gray scale patches differently, the results are not comparable to the ones reported in Table 2. Before subtracting the sample mean, small uniform noise (between 0 and 1) is added to the pixel values (between 0 and 255), which are then normalized by dividing by 256. Afterwards, they remove the last pixel, so that the number of variables of each datapoint equals 63. For this task we used four million patches during training and evaluated on one million patches from the test set. The results are shown in Table 3. The STM outperforms the deep RNADE model of 6 layers, but is on its turn outperformed by the ensemble of RNADE models (EoRNADE).

5.3 Lossless Compression

Because JPEG does not natively support lossless compression we excluded this benchmark for this test. For our methods we used patch size 8x8 and 128 mixture components. The results are listed in Table 4. As explained above (see Section 4.2), there is a connection

Indp. Pixel	ICA	GRBM	DBN	MFA
78.3	135.7	137.8	144.4	166.5
Deep MFA	MTA	GMM	**GMM**	**STM**
169.3	158.2	167.2	**166.97 ± 0.36**	**172.13 ± 0.42**

Table 2: Average log-likelihood (higher is better). Own results are marked in bold. The results of other methods are taken from Zoran and Weiss (2011); Tang et al. (2013). ICA: independent component analysis, GRBM: Gaussian restricted Boltzmann machine, DBN: Deep belief network, MFA: mixture of factor analyzers, MTA: Mixture of Tensor analyzers.

RNADE:	
1hl, 2hl, 3hl	143.2, 149,2, 152.0
4hl, 5hl, 6hl	153.6, 154.7, 155.2
EoRNADE (6hl)	157.0
GMM	**153.7**
STM	**155.3**

Table 3: Average log-likelihood comparison with RNADE (Uria et al., 2013a) in function of the number of hidden layers (hl). Our results are marked in bold. *These results are obtained from differently processed patches than those in Table 2 (see text).* EoRNADE stands for an ensemble of RNADE's.

between average log likelihood and the expected lossless compression strength. The STM also outperforms the GMM on this task, and both methods outperform JPEG 2000.

5.4 Lossy Compression

We will first analyze the influence of the patch size on the lossy compression performance. The results are visualized in Figure 7. All mixture models were trained for 500 iterations and consist of 128 components. The reconstruction quality of an image is measured in peak signal-to-noise ratio: $\text{PSNR} = 10 \log_{10} \frac{R^2}{\text{MSE}}$, with R being the largest possible pixel value (255 in this case) and MSE being the average mean squared error.

Bigger patch sizes show better results for low bit rates and vice versa. This can be explained by the fact that when using larger patch sizes, covariance between more pixels

JPEG 2000	**GMM**	**STM**
12.40	**12.07**	**11.83**

Table 4: Lossless compression rate (in bits per pixel - lower is better). Naive encoding would result in 24 bits per pixel (true color images).

can be modeled simultaneously. This way the transform has the ability to decorrelate better, which is important for low bit rates. For higher bit rates, we approach a near-lossless region, where the log likelihood performance of the model is crucial. When modeling smaller patch sizes, the algorithm is less prone to overfit, resulting in better performance. We can see that these high-rate effects are most apparent for the GMM. The STM, which is more robust to overfitting, is able to model larger patch sizes.

Because the 8x8 patch size has a good performance in general for both methods, our final experiments are computed with this setting. Note that JPEG also uses 8x8 patches for its compression scheme. For different compression strengths we have computed the average PSNR of the reconstructed images. For some images, JPEG or JPEG 2000 was unable to encode them at a given rate (1, 2 and 10 images for 3, 4 and 5 bpp respectively), so these images where not taken into account at those rates.

The final results are shown on Figure 8. For all compression rates, JPEG is outperformed by the other methods. The proposed compression schemes are competitive with JPEG 2000, and relatively to JPEG they score quite similar. In all experiments uniform threshold quantization improves on standard uniform quantization. The GMM is always outperformed by the STM, and the difference increases for larger bit rates. JPEG 2000 slightly exceeds the performance of the GMM in all experiments, but is in turn surpassed by the STM, with the exception of the lowest bit rate. At low bit rates, correlations on a more global scale become more important, which is why the multiresolution wavelet transform of JPEG 2000 achieves a better performance than our patch-based approach in this setting. Extending our approach to a multiscale technique might therefore be a promising direction of future research.

In Figure 9 we have visualized some reconstructed images after compression with JPEG, JPEG 2000 and the proposed method (GMM and STM), for varying levels of compression strength: 1, 2.5 and 5 bits per pixel (bpp). This Figure is best viewed on the electronic version by zooming in on the different images. Because JPEG and the proposed method are block based, they have blocking artifacts that JPEG 2000 does not. The latter has more blurring artifacts. The proposed method seems to have the strongest visual artifacts in low-frequency regions, but performs well in high-frequency regions such as trees and leaves. This can be attributed to the fact that the compression method does not take into account the properties of human visual perception and therefore quantizes both high as low frequency regions equally strongly. One could improve the visual results by adding prior knowledge about the perceptual system, using a deblocking filter (or using an image reconstruction algorithm based on GMM/STMs with the expected patch log likelihood (EPLL) framework), extending the model so that it works with overlapping blocks (with the MDCT transform for example) or by making it multi-scale. However, these extensions are outside the scope of this work.

6. Conclusion and Future Work

The presented work consists of two main contributions: the introduction and analysis of the Student-t mixture as an image patch modeling technique, and the proposal of lossless and lossy image compression techniques based on mixture models.

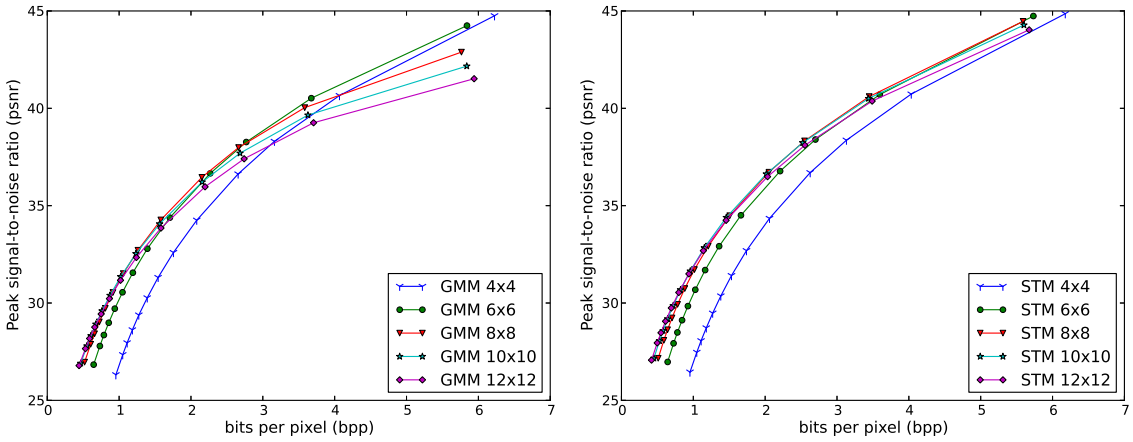


Figure 7: Average patch Quality (PSNR) - Rate (bpp) curves for different patch-sizes (GMM left, STM right). This Figure is best viewed in color.

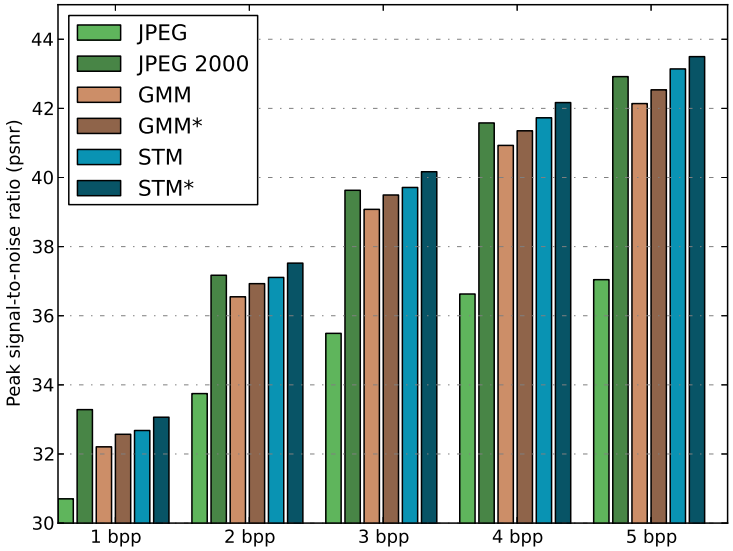


Figure 8: Results for lossy compression of colored images. Average quality (PSNR) in function of average rate (bits per pixel). Methods marked with a asterisk (*) use uniform *threshold* quantization, and thus have a better reconstruction error.

In the first part we have proposed the STM as an image patch prior. This method significantly outperformed the GMM for density modeling of image patches, with results competitive to the state-of-the-art on this task. This performance could largely be attributed to the fact that a Student-t mixture is able to model contrast in addition to linear dependencies within a single mixture component. For future work it would also be very interesting to see how it matches up with other methods on other types of data. Another possibility would be to study the Student-t mixture model for the use of image reconstruction applications (denoising, deblurring, inpainting), as was recently proposed with GMMs (Zoran and Weiss, 2011).

In the second part both the GMM and STM have been examined in this paper for the task of image compression. Lossless and lossy coding schemes were presented, which could easily be adapted for other unsupervised learning techniques. For lossy compression, experimental results demonstrated that the proposed techniques consistently outperform JPEG, with results similar to those of JPEG 2000. With the exception of the lowest bit rate, the STM has the advantage over JPEG 2000 in terms of rate-distortion. In lossless compression both the GMM and STM outperform JPEG 2000, which is mainly due to the fact that this task is even more connected with density estimation. In future work, even more advanced techniques will be considered. Moving beyond the 8x8 patch size, with for example multiscale techniques, is an especially promising direction.

One of the most important conclusions we can draw here is that relatively simple machine learning techniques can perform quite well on the task of image compression. We saw that their performance could largely be attributed to their density modeling capabilities. It would therefore be interesting to apply machine learning to compression of different types of data, such as audio, video, EEG, etc. and more specific types of data such as facial or satellite images. We also propose for compression to be used more in machine learning as a benchmark to compare models. Given the recent progress in unsupervised machine learning we expect that even better results will follow.

References

Anuradha Aiyer, Kyungsuk Pyun, Ying-zong Huang, Deirdre B OBrien, and Robert M Gray. Lloyd clustering of Gauss mixture models for image compression and classification. *Signal Processing: Image Communication*, 20(5):459–485, 2005.

Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009.

Richard P Brent. *Algorithms for Minimization without Derivatives*. Courier Dover Publications, 1973.

Ori Bryt and Michael Elad. Compression of facial images using the k-svd algorithm. *Journal of Visual Communication and Image Representation*, 19(4):270–282, 2008.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

Original Image

JPEG

JPEG 2000

GMM

STM

Figure 9: Reconstructions after compression by JPEG, JPEG 2000 or the proposed method with a GMM of STM. The rates were 1 bpp (left), 2.5 bpp (middle), 5 bpp (right). This figure is best viewed on the electronic version by zooming in on the images.

Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *Transactions on Image Processing*, 15(12):3736–3745, 2006.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

Vivek K Goyal. Theoretical foundations of transform coding. *Signal Processing Magazine*, 18(5):9–21, 2001.

Per Hedelin and Jan Skoglund. Vector quantization based on Gaussian mixture models. *Transactions on Speech and Audio Processing*, 8(4):385–401, 2000.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

Wei Hong, John Wright, Kun Huang, and Yi Ma. A multiscale hybrid linear model for lossy image representation. In *International Conference on Computer Vision*, volume 1, pages 764–771. IEEE, 2005.

Inbal Horev, Ori Bryt, and Ron Rubinstein. Adaptive image compression using sparse dictionaries. In *International Conference on Systems, Signals and Image Processing*, pages 592–595. IEEE, 2012.

Samuel Kotz and Saralees Nadarajah. *Multivariate t-Distributions and their Applications*. Cambridge University Press, 2004.

Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Non-local sparse models for image restoration. In *International Conference on Computer Vision*, pages 2272–2279. IEEE, 2009.

David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference on Computer Vision*, volume 2, pages 416–423, July 2001.

Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997.

William A Pearlman and Amir Said. *Digital Signal Compression: Principles and Practice*. Cambridge University Press, 2011.

David Peel and Geoffrey J McLachlan. Robust mixture modelling using the t distribution. *Statistics and Computing*, 10(4):339–348, 2000.

Stefan Roth and Michael J Black. Fields of experts: A framework for learning image priors. In *Computer Vision and Pattern Recognition, 2005.*, volume 2, pages 860–867. IEEE, 2005.

Gerald Schaefer and Michal Stich. UCID: an uncompressed color image database. In *Electronic Imaging 2004*, pages 472–480. International Society for Optics and Photonics, 2003.

Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The JPEG 2000 still image compression standard. *Signal Processing Magazine*, 18(5):36–58, 2001.

Yichuan Tang, Ruslan Salakhutdinov, and Geoffrey Hinton. Deep mixtures of factor analysers. In *International Conference on Machine Learning*, pages 505–512, 2012.

Yichuan Tang, Ruslan Salakhutdinov, and Geoffrey Hinton. Tensor analyzers. In *International Conference on Machine Learning*, 2013.

Lucas Theis, Sebastian Gerwinn, Fabian Sinz, and Matthias Bethge. In all likelihood, deep belief is not enough. *The Journal of Machine Learning Research*, 12:3071–3096, 2011.

Lucas Theis, Reshad Hosseini, and Matthias Bethge. Mixtures of conditional Gaussian scale mixtures applied to multiscale image representations. *PLoS ONE*, 7(7), Jul 2012. doi: 10.1371/journal.pone.0039857.

D Michael Titterington, Adrian FM Smith, Udi E Makov, et al. *Statistical Analysis of Finite Mixture Distributions*, volume 7. Wiley New York, 1985.

Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. *arXiv preprint arXiv:1310.1757*, 2013a.

Benigno Uria, Iain Murray, and Hugo Larochelle. Nade: The real-valued neural autoregressive density-estimator. *arXiv preprint arXiv:1306.0186*, 2013b.

Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Learning a piecewise linear transform coding scheme for images. In *2012 International Conference on Graphic and Image Processing*, pages 876844–876844. International Society for Optics and Photonics, 2013.

Gregory K Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.

Yair Weiss and William T Freeman. What makes a good model of natural images? In *Computer Vision and Pattern Recognition, 2007.*, pages 1–8. IEEE, 2007.

Max Welling, Simon Osindero, and Geoffrey E Hinton. Learning sparse topographic representations with products of Student-t distributions. In *Advances in Neural Information Processing Systems*, pages 1359–1366, 2002.

John Wright, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas S Huang, and Shuicheng Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044, 2010.

Guoshen Yu, Guillermo Sapiro, and Stéphane Mallat. Solving inverse problems with piecewise linear estimators: from Gaussian mixture models to structured sparsity. *Transactions on Image Processing*, 21(5):2481–2499, 2012.

Joaquin Zepeda, Christine Guillemot, and Ewa Kijak. Image compression using sparse representations and the iteration-tuned and aligned dictionary. *Journal of Selected Topics in Signal Processing*, 5(5):1061–1073, 2011.

Daniel Zoran and Yair Weiss. From learning models of natural image patches to whole image restoration. In *International Conference on Computer Vision*, pages 479–486. IEEE, 2011.

Daniel Zoran and Yair Weiss. Natural images, Gaussian mixtures and dead leaves. In *Advances in Neural Information Processing Systems*, volume 25, pages 1745–1753, 2012.

Journal of Machine Learning Research 15 (2014) 2087-2112 Submitted 10/12; Revised 1/14; Published 6/14

Parallel MCMC with Generalized Elliptical Slice Sampling

Robert Nishihara RKN@EECS.BERKELEY.EDU
Department of Electrical Engineering and Computer Science
University of California
Berkeley, CA 94720, USA

Iain Murray I.MURRAY@ED.AC.UK
School of Informatics
University of Edinburgh
Edinburgh EH8 9AB, UK

Ryan P. Adams RPA@SEAS.HARVARD.EDU
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138, USA

Editor: David Blei

Abstract

Probabilistic models are conceptually powerful tools for finding structure in data, but their practical effectiveness is often limited by our ability to perform inference in them. Exact inference is frequently intractable, so approximate inference is often performed using Markov chain Monte Carlo (MCMC). To achieve the best possible results from MCMC, we want to efficiently simulate many steps of a rapidly mixing Markov chain which leaves the target distribution invariant. Of particular interest in this regard is how to take advantage of multi-core computing to speed up MCMC-based inference, both to improve mixing and to distribute the computational load. In this paper, we present a parallelizable Markov chain Monte Carlo algorithm for efficiently sampling from continuous probability distributions that can take advantage of hundreds of cores. This method shares information between parallel Markov chains to build a scale-location mixture of Gaussians approximation to the density function of the target distribution. We combine this approximation with a recently developed method known as elliptical slice sampling to create a Markov chain with no step-size parameters that can mix rapidly without requiring gradient or curvature computations.

Keywords: Markov chain Monte Carlo, parallelism, slice sampling, elliptical slice sampling, approximate inference

1. Introduction

Probabilistic models are fundamental tools for machine learning, providing a coherent framework for finding structure in data. In the Bayesian formulation, learning is performed by computing a representation of the posterior distribution implied by the data. Unobserved quantities of interest can then be estimated as expectations of various functions under this posterior distribution.

These expectations typically correspond to high-dimensional integrals and sums, which are usually intractable for rich models. There is therefore significant interest in efficient methods for approximate inference that can rapidly estimate these expectations. In this paper, we examine Markov chain Monte Carlo (MCMC) methods for approximate inference, which estimate these quantities by simulating a Markov chain with the posterior as its equilibrium distribution. MCMC is often seen as a principled "gold standard" for inference, because (under mild conditions) its answers will be correct in the limit of the simulation. However, in practice, MCMC often converges slowly and requires expert tuning. In this paper, we propose a new method to address these issues for continuous parameter spaces. We generalize the method of *elliptical slice sampling* (Murray et al., 2010) to build a new efficient method that: 1) mixes well in the presence of strong dependence, 2) does not require hand tuning, and 3) can take advantage of multiple computational cores operating in parallel. We discuss each of these in more detail below.

Many posterior distributions arising from real-world data have strong dependencies between variables. These dependencies can arise from correlations induced by the likelihood function, redundancy in the parameterization, or directly from the prior. One of the primary challenges for efficient Markov chain Monte Carlo is making large moves in directions that reflect the dependence structure. For example, if we imagine a long, thin region of high density, it is necessary to explore the length in order to reach equilibrium; however, random-walk methods such as Metropolis–Hastings (MH) with spherical proposals can only diffuse as fast as the narrowest direction allows (Neal, 1995). More efficient methods such as Hamiltonian Monte Carlo (Duane et al., 1987; Neal, 2011; Girolami and Calderhead, 2011) avoid random walk behavior by introducing auxiliary "momentum" variables. Hamiltonian methods require differentiable density functions and gradient computations.

In this work, we are able to make efficient long-range moves—even in the presence of dependence—by building an approximation to the target density that can be exploited by elliptical slice sampling. This approximation enables the algorithm to consider the general shape of the distribution without requiring gradient or curvature information. In other words, it encodes and allows us to make use of global information about the distribution as opposed to the local information used by Hamiltonian Monte Carlo. We construct the algorithm such that it is valid regardless of the quality of the approximation, preserving the guarantees of approximate inference by MCMC.

One of the limitations of MCMC in practice is that it is often difficult for non-experts to apply. This difficulty stems from the fact that it can be challenging to tune Markov transition operators so that they mix well. For example, in Metropolis–Hastings, one must come up with appropriate proposal distributions. In Hamiltonian Monte Carlo, one must choose the number of steps and the step size in the simulation of the dynamics. For probabilistic machine learning methods to be widely applicable, it is necessary to develop black-box methods for approximate inference that do not require extensive hand tuning. Some recent attempts have been made in the area of adaptive MCMC (Roberts and Rosenthal, 2006; Haario et al., 2005), but these are only theoretically understood for a relatively narrow class of transition operators (for example, not Hamiltonian Monte Carlo). Here we propose a method based on slice sampling (Neal, 2003), which uses a local search to find an acceptable point, and avoid potential issues with convergence under adaptation.

In all aspects of machine learning, a significant challenge is exploiting a computational landscape that is evolving toward parallelism over single-core speed. When considering parallel approaches to MCMC, we can readily identify two ends of a spectrum of possible solutions. At one extreme, we could run a large number of independent Markov chains in parallel (Rosenthal, 2000; Bradford and Thomas, 1996). This will have the benefit of providing more samples and increasing the accuracy of the end result, however it will do nothing to speed up the convergence or the mixing of each individual chain. The parallel algorithm will run up against the same limitations faced by the non-parallel version. At another extreme, we could develop a single-chain MCMC algorithm which parallelizes the individual Markov transitions in a problem-specific way. For instance, if the likelihood is expensive and consists of many factors, the factors can potentially be computed in parallel. See Suchard et al. (2010); Tarlow et al. (2012) for examples. Alternatively, some Markov chain transition operators can make use of multiple parallel proposals to increase their efficiency, such as multiple-try Metropolis–Hastings (Liu et al., 2000).

We propose an intermediate algorithm to make effective use of parallelism. By sharing information between the chains, our method is able to mix faster than the naïve approach of running independent chains. However, we do not require fine-grained control over parallel execution, as would be necessary for the single-chain method. Nevertheless, if such local parallelism is possible, our sampler can take advantage of it. Our general objective is a black-box approach that mixes well with multiple cores but does not require the user to build in parallelism at a low level.

The structure of the paper is as follows. In Section 2, we review slice sampling (Neal, 2003) and elliptical slice sampling (Murray et al., 2010). In Section 3, we show how an elliptical approximation to the target distribution enables us to generalize elliptical slice sampling to continuous distributions. In Section 4, we describe a natural way to use parallelism to dynamically construct the desired approximation. In Section 5, we discuss related work. In Section 6, we evaluate our new approach against other comparable methods on several typical modeling problems.

2. Background

Throughout this paper, we will use $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ to denote the density function of a Gaussian with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ evaluated at a point $\mathbf{x} \in \mathbb{R}^D$. We will use $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ to refer to the distribution itself. Analogous notation will be used for other distributions. Throughout, we shall assume that we wish to draw samples from a probability distribution over $\mathbb{R}^D$ whose density function is π. We sometimes refer to the distribution itself as π.

The objective of Markov chain Monte Carlo is to formulate transition operators that can be easily simulated, that leave π invariant, and that are ergodic. Classical examples of MCMC algorithms are Metropolis–Hastings (Metropolis et al., 1953; Hastings, 1970) and Gibbs Sampling (Geman and Geman, 1984). For general overviews of MCMC, see Tierney (1994); Andrieu et al. (2003); Brooks et al. (2011). Simulating such a transition operator will, in the limit, produce samples from π, and these can be used to compute expectations under π. Typically, we only have access to an unnormalized version of π. However, none of the algorithms that we describe require access to the normalization constant, and so we will abuse notation somewhat and refer to the unnormalized density as π.

2.1 Slice Sampling

Slice sampling (Neal, 2003) is a Markov chain Monte Carlo algorithm with an adaptive step size. It is an auxiliary-variable method, which relies on the observation that sampling π is equivalent to sampling the uniform distribution over the set $S = \{(\mathbf{x}, y) : 0 \leq y \leq \pi(\mathbf{x})\}$ and marginalizing out the y coordinate (which in this case is accomplished simply by disregarding the y coordinate). Slice sampling accomplishes this by alternately updating $\mathbf{x}$ and y so as to leave invariant the distributions $p(\mathbf{x} \,|\, y)$ and $p(y \,|\, \mathbf{x})$ respectively. The key insight of slice sampling is that sampling from these conditionals (which correspond to uniform "slices" under the density function) is potentially much easier than sampling directly from π.

Updating y according to $p(y \,|\, \mathbf{x})$ is trivial. The new value of y is drawn uniformly from the interval $(0, \pi(\mathbf{x}))$. There are different ways of updating $\mathbf{x}$. The objective is to draw uniformly from among the "slice" $\{\mathbf{x} : \pi(\mathbf{x}) \geq y\}$. Typically, this is done by defining a transition operator that leaves the uniform distribution on the slice invariant. Neal (2003) describes such a transition operator: first, choose a direction in which to search, then place an interval around the current state, expand it as necessary, and shrink it until an acceptable point is found. Several procedures have been proposed for the expansion and contraction phases.

Less clear is how to choose an efficient direction in which to search. There are two approaches that are widely used. First, one could choose a direction uniformly at random from all possible directions (MacKay, 2003). Second, one could choose a direction uniformly at random from the D coordinate directions. We consider both of these implementations later, and we refer to them as *random-direction slice sampling* (RDSS) and *coordinate-wise slice sampling* (CWSS), respectively. In principle, any distribution over directions can be used as long as detailed balance is satisfied, but it is unclear what form this distribution should take. The choice of direction has a significant impact on how rapidly mixing occurs. In the remainder of the paper, we describe how slice sampling can be modified so that candidate points are chosen to reflect the structure of the target distribution.

2.2 Elliptical Slice Sampling

Elliptical slice sampling (Murray et al., 2010) is an MCMC algorithm designed to sample from posteriors over latent variables of the form

$$\pi(\mathbf{x}) \propto L(\mathbf{x})\,\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{1}$$

where L is a likelihood function, and $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a multivariate Gaussian prior. Such models, often called *latent Gaussian models*, arise frequently from Gaussian processes and Gaussian Markov random fields. Elliptical slice sampling takes advantage of the structure induced by the Gaussian prior to mix rapidly even when the covariance induces strong dependence. The method is easier to apply than most MCMC algorithms because it has no free tuning parameters.

Elliptical slice sampling takes advantage of a convenient invariance property of the multivariate Gaussian. Namely, if $\mathbf{x}$ and $\boldsymbol{\nu}$ are independent draws from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then the linear combination

$$\mathbf{x}' = (\mathbf{x} - \boldsymbol{\mu}) \cos\theta + (\boldsymbol{\nu} - \boldsymbol{\mu}) \sin\theta + \boldsymbol{\mu} \tag{2}$$

is also (marginally) distributed according to $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for any $\theta \in [0, 2\pi]$. Note that $\mathbf{x}'$ is nevertheless correlated with $\mathbf{x}$ and $\boldsymbol{\nu}$. This correlation has been previously used to make perturbative Metropolis–Hastings proposals in latent Gaussian models (Neal, 1998; Adams et al., 2009), but elliptical slice sampling uses it as the basis for a rejection-free method.

The elliptical slice sampling transition operator considers the locus of points defined by varying θ in Equation (2). This locus is an ellipse which passes through the current state $\mathbf{x}$ as well as through the auxiliary variable $\boldsymbol{\nu}$. Given a random ellipse induced by $\boldsymbol{\nu}$, we can slice sample $\theta \in [0, 2\pi]$ to choose the next point based purely on the likelihood term. The advantage of this procedure is that the ellipses will necessarily reflect the dependence induced by strong Gaussian priors and that the user does not have to choose a step size.

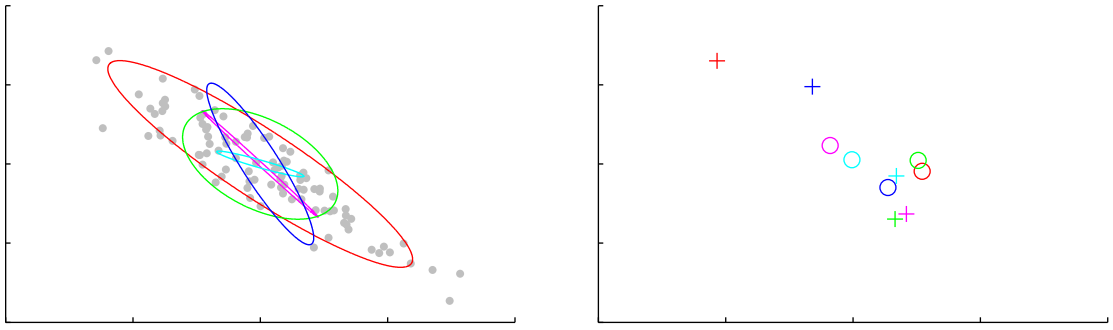
More specifically, elliptical slice sampling updates the current state $\mathbf{x}$ as follows. First, the auxiliary variable $\boldsymbol{\nu} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is sampled to define an ellipse via Equation (2), and the value $u \sim \text{Uniform}[0, 1]$ is sampled to define a likelihood threshold. Then, a sequence of angles $\{\theta_k\}$ are chosen according to a slice-sampling procedure described in Algorithm 1. These angles specify a corresponding sequence of proposal points $\{\mathbf{x}'_k\}$. We update the current state $\mathbf{x}$ by setting it equal to the first proposal point $\mathbf{x}'_k$ satisfying the slice-sampling condition $L(\mathbf{x}'_k) > uL(\mathbf{x})$. The proof of the validity of this algorithm is given in Murray et al. (2010). Intuitively, the pair $(\mathbf{x}, \boldsymbol{\nu})$ is updated to a pair $(\mathbf{x}', \boldsymbol{\nu}')$ with the same joint prior probability, and so slice sampling only needs to compare likelihood ratios. The new point $\mathbf{x}'$ is given by Equation (2), while $\boldsymbol{\nu}' = (\boldsymbol{\nu} - \boldsymbol{\mu}) \cos\theta - (\mathbf{x} - \boldsymbol{\mu}) \sin\theta + \boldsymbol{\mu}$ is never used and need not be computed.

Figure 1 depicts random ellipses produced by elliptical slice sampling superimposed on background points from some target distribution. This diagram illustrates the idea that the ellipses produced by elliptical slice sampling reflect the structure of the distribution. The full algorithm is shown in Algorithm 1.

Algorithm 1 Elliptical Slice Sampling Update

Input: Current state $\mathbf{x}$, Gaussian parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, log-likelihood function $\log L$
Output: New state $\mathbf{x}'$, with stationary distribution proportional to $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})L(\mathbf{x})$

1: $\boldsymbol{\nu} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ ▷ Choose ellipse
2: $u \sim \text{Uniform}[0, 1]$
3: $\log y \leftarrow \log L(\mathbf{x}) + \log u$ ▷ Set log-likelihood threshold
4: $\theta \sim \text{Uniform}[0, 2\pi]$ ▷ Draw an initial proposal
5: $[\theta_{\min}, \theta_{\max}] \leftarrow [\theta - 2\pi, \theta]$ ▷ Define a bracket
6: $\mathbf{x}' \leftarrow (\mathbf{x} - \boldsymbol{\mu}) \cos\theta + (\boldsymbol{\nu} - \boldsymbol{\mu}) \sin\theta + \boldsymbol{\mu}$
7: **if** $\log L(\mathbf{x}') > \log y$ **then**
8: **return** $\mathbf{x}'$ ▷ Accept
9: **else** ▷ Shrink the bracket and try a new point
10: **if** $\theta < 0$ **then**
11: $\theta_{\min} \leftarrow \theta$
12: **else**
13: $\theta_{\max} \leftarrow \theta$
14: $\theta \sim \text{Uniform}[\theta_{\min}, \theta_{\max}]$
15: **goto** 6



(a) Ellipses from ESS

(b) Corresponding values of $\mathbf{x}$ and $\boldsymbol{\nu}$

Figure 1: Background points are drawn independently from a probability distribution, and five ellipses are created by elliptical slice sampling. The distribution in question is a two-dimensional multivariate Gaussian. In this example, the same distribution is used as the prior for elliptical slice sampling. **(a)** Shows the ellipses created by elliptical slice sampling. **(b)** Shows the values of $\mathbf{x}$ (depicted as $\circ$) and $\boldsymbol{\nu}$ (depicted as $+$) corresponding to each elliptical slice sampling update. The values of $\mathbf{x}$ and $\boldsymbol{\nu}$ with a given color correspond to the ellipse of the same color in **(a)**.

3. Generalized Elliptical Slice Sampling

In this section, we generalize elliptical slice sampling to handle arbitrary continuous distributions. We refer to this algorithm as *generalized elliptical slice sampling* (GESS). In this section, our target distribution will be a continuous distribution over $\mathbb{R}^D$ with density π. In practice, π need not be normalized.

Our objective is to reframe our target distribution so that it can be efficiently sampled with elliptical slice sampling. One possible approach is to put π in the form of Equation (1) by choosing some approximation $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ to π and writing

$$\pi(\mathbf{x}) = R(\mathbf{x})\,\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

where

$$R(\mathbf{x}) = \frac{\pi(\mathbf{x})}{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})}$$

is the residual error of our approximation to the target density. Note that $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is an approximation rather than a prior and that R is not a likelihood function, but since the equation has the correct form, this representation enables us to use elliptical slice sampling.

Applying elliptical slice sampling in this manner will produce a correct algorithm, but it may mix slowly in practice. Difficulties arise when the target distribution has much heavier tails than does the approximation. In such a situation, $R(\mathbf{x})$ will increase rapidly as $\mathbf{x}$ moves away from the mean of the approximation. To illustrate this phenomenon, we use this approach with different approximations to draw samples from a Gaussian in one dimension with zero mean and unit variance. Trace plots are shown in Figure 2. The subplot corresponding to variance 0.01 illustrates the problem. Since R explodes as $|\mathbf{x}|$ gets

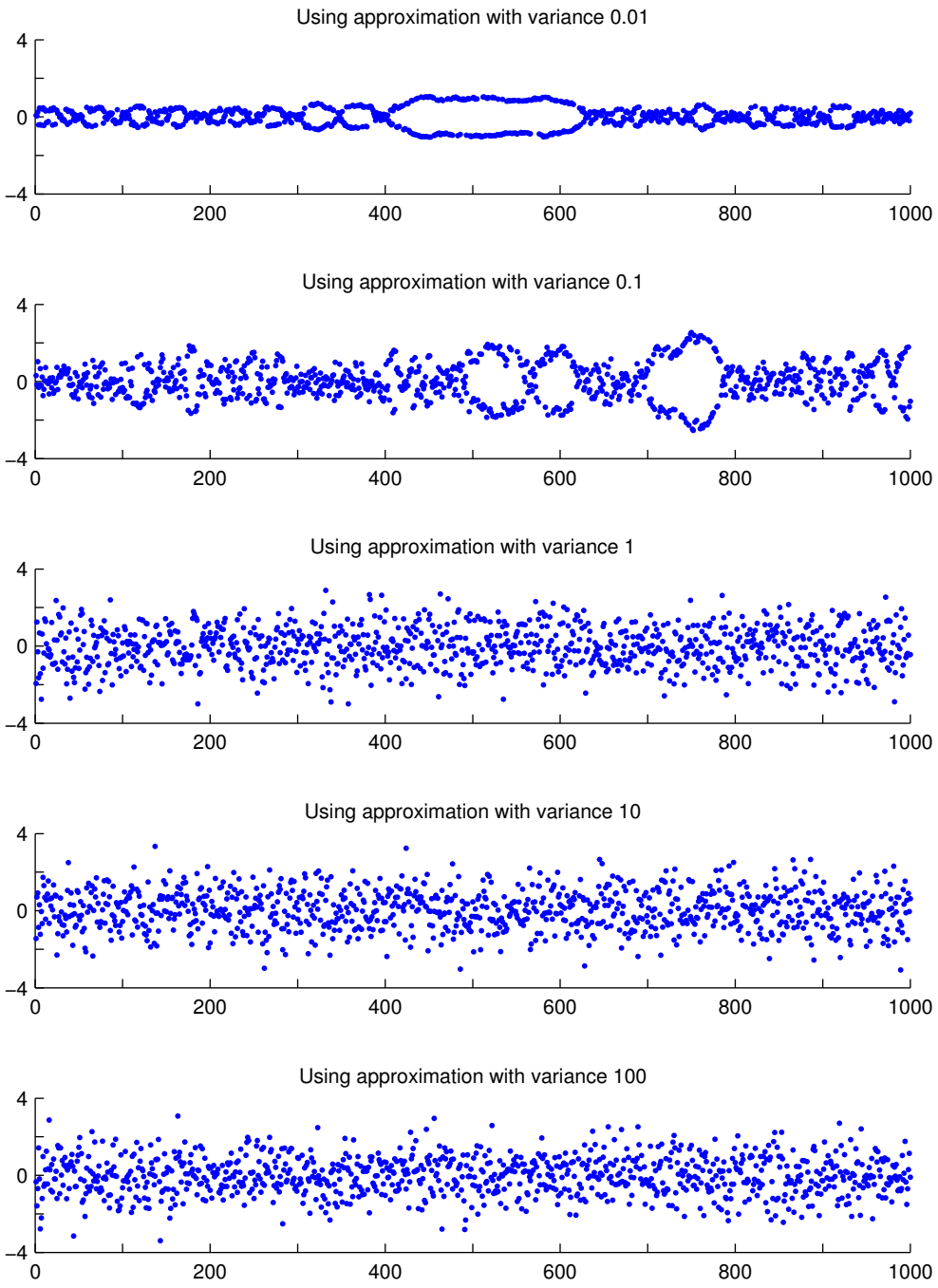


Figure 2: Samples are drawn from a Gaussian with zero mean and unit variance using elliptical slice sampling with various Gaussian approximations. These trace plots show how sampling behavior depends on how heavy the tails of the approximation are relative to how heavy the tails of the target distribution are. We plot one of every ten samples.

large, the Markov chain is unlikely to move back toward the origin. On the other hand, the size of the ellipse is limited by a draw from the Gaussian approximation, which has low variance in this case, so the Markov chain is also unlikely to move away from the origin. The result is that the Markov chain sometimes gets stuck. In the subplot corresponding to variance 0.01, this occurs between iterations 400 and 630.

In order to resolve this pathology and extend elliptical slice sampling to continuous distributions, we broaden the class of allowed approximations. To begin with, we express the density of the target distribution in the more general form

$$\pi(\mathbf{x}) \propto R(\mathbf{x}) \int \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}(s), \boldsymbol{\Sigma}(s))\, \phi(\mathrm{d}s), \tag{3}$$

where the integral represents a scale-location mixture of Gaussians (which serves as an approximation to π), and where ϕ is a measure over the auxiliary parameter s. As before, R is the residual error of the approximation. Here, ϕ can be chosen in a problem-specific way, and any residual error between π and the approximation will be compensated for by R. Equation (3) is quite flexible. Below, we will choose the measure ϕ so as to make the approximation a multivariate t distribution, but there are many other possibilities. For instance, taking ϕ to be a combination of point masses will make the approximation a discrete mixture of Gaussians.

Through Equation (3), we can view $\pi(\mathbf{x})$ as the marginal density of an augmented joint distribution over $\mathbf{x}$ and s. Using λ to denote the density of ϕ with respect to the base measure over s (this is fully general because we have control over the choice of base measure), we can write

$$p(\mathbf{x}, s) = R(\mathbf{x})\, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}(s), \boldsymbol{\Sigma}(s))\, \lambda(s).$$

Therefore, to sample π, it suffices to sample $\mathbf{x}$ and s jointly and then to marginalize out the s coordinate (by simply dropping the s coordinate). We update these components alternately so as to leave invariant the distributions

$$p(\mathbf{x} \,|\, s) \propto R(\mathbf{x})\, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}(s), \boldsymbol{\Sigma}(s)) \tag{4}$$

and

$$p(s \,|\, \mathbf{x}) \propto \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}(s), \boldsymbol{\Sigma}(s))\, \lambda(s). \tag{5}$$

Equation (4) has the correct form for elliptical slice sampling and can be updated according to Algorithm 1. Equation (5) can be updated using any valid Markov transition operator.

We now focus on a particular case in which the update corresponding to Equation (5) is easy to simulate and in which we can make the tails as heavy as we desire, so as to control the behavior of R. A simple and convenient choice is for the scale-location mixture to yield a multivariate t distribution with degrees-of-freedom parameter ν:

$$\mathcal{T}_\nu(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \int_0^\infty \mathrm{IG}(s; \tfrac{\nu}{2}, \tfrac{\nu}{2})\, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, s\boldsymbol{\Sigma})\, \mathrm{d}s,$$

where λ becomes the density function of an inverse-gamma distribution:

$$\mathrm{IG}(s; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} s^{-\alpha-1} e^{-\beta/s}.$$

Here s is a positive real-valued scale parameter. Now, in the update $p(s \mid \mathbf{x})$, we observe that the inverse-gamma distribution acts as a conjugate prior (whose "prior" parameters are $\alpha = \frac{\nu}{2}$ and $\beta = \frac{\nu}{2}$), so

$$p(s \mid \mathbf{x}) = \mathrm{IG}(s; \alpha', \beta')$$

with parameters

$$\alpha' = \frac{D+\nu}{2} \quad \text{and} \tag{6}$$

$$\beta' = \frac{1}{2}(\nu + (\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})). \tag{7}$$

We can draw independent samples from this distribution (Devroye, 1986).

Combining these update steps, we define the transition operator $S(\mathbf{x} \to \mathbf{x}'; \nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ to be the one which draws $s \sim \mathrm{IG}(s; \alpha', \beta')$, with α' and β' as described in Equations (6) and (7), and then uses elliptical slice sampling to update $\mathbf{x}$ so as to leave invariant the distribution defined in Equation (4), where $\boldsymbol{\mu}(s) = \boldsymbol{\mu}$ and $\boldsymbol{\Sigma}(s) = s\boldsymbol{\Sigma}$. From the above discussion, it follows that the stationary distribution of $S(\mathbf{x} \to \mathbf{x}'; \nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is π. Figure 3 illustrates this transition operator.

Algorithm 2 Generalized Elliptical Slice Sampling Update

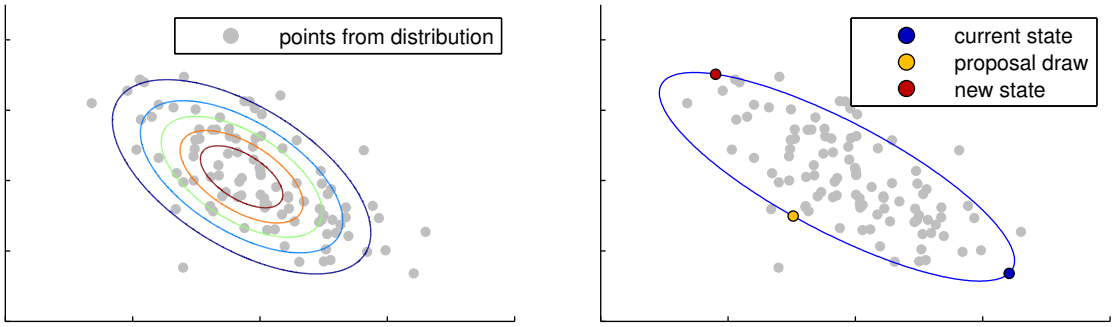
Input: Current state $\mathbf{x}$, multivariate t parameters $\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma}$, dimension D, a routine ESS that performs an elliptical slice sampling update

Output: New state $\mathbf{x}'$

1: $\alpha' \leftarrow \frac{D+\nu}{2}$
2: $\beta' \leftarrow \frac{1}{2}(\nu + (\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}))$
3: $s \sim \mathrm{IG}(\alpha', \beta')$
4: $\log L \leftarrow \log \pi - \log \mathcal{T}$ ▷ $\mathcal{T}$ is the density of a multivariate t with parameters $\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma}$
5: $\mathbf{x}' \leftarrow \mathrm{ESS}(\mathbf{x}, \boldsymbol{\mu}, s\boldsymbol{\Sigma}, \log L)$

4. Building the Approximation with Parallelism

Up to this point, we have not described how to choose the multivariate t parameters ν, $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$. These choices can be made in many ways. For instance, we may choose the maximum likelihood parameters given samples collected during a burn in period, we may build a Laplace approximation to the mode of the distribution, or we may use variational approaches. Note that this algorithm is valid regardless of the particular choice we make here. In this section, we discuss a convenient way to use parallelism to dynamically choose these parameters without requiring tuning runs or exploratory analysis of the distribution. This method creates a large number of parallel chains, each producing samples from π, and it divides them into two groups. The need for two groups of Markov chains is not immediately obvious, so we motivate our approach by first discussing two simpler algorithms that fail in different ways.



(a) Multivariate t contours

(b) Example GESS update

Figure 3: The gray points were drawn independently from a two-dimensional Gaussian to show the mode and shape of the corresponding density function. **(a)** Shows the contours of a multivariate t approximation to this distribution. **(b)** Shows a sample update step using the transition operator $S(\mathbf{x} \to \mathbf{x}'; \nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$. The blue point represents the current state. The yellow point defines an ellipse and is drawn from the Gaussian distribution corresponding to the scale s drawn from the appropriate inverse-gamma distribution. The red point corresponds to the new state and is sampled from the given ellipse.

4.1 Naïve Approaches

We begin with a collection of K parallel chains. Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$ denote the current states of the chains. We observe that $\mathcal{X}$ may contain a lot of information about the shape of the target distribution. We would like to define a transition operator $Q(\mathcal{X} \to \mathcal{X}')$ that uses this information to intelligently choose the multivariate t parameters ν, $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$ and then uses these parameters to update each $\mathbf{x}_k$ via generalized elliptical slice sampling. Additionally, we would like Q to have two properties. First, each $\mathbf{x}_k$ should have the marginal stationary distribution π. Second, we should be able to parallelize the update of $\mathcal{X}$ over K cores.

Here we describe two simple approaches for parallelizing generalized elliptical slice sampling, each of which lacks one of the desired properties. The first approach begins with K parallel Markov chains, and it requires a procedure for choosing the multivariate t parameters given $\mathcal{X}$ (for example, maximum likelihood estimation). In this setup, Q uses this procedure to determine the multivariate t parameters $\nu_{\mathcal{X}}$, $\boldsymbol{\mu}_{\mathcal{X}}$, $\boldsymbol{\Sigma}_{\mathcal{X}}$ from $\mathcal{X}$ and then applies $S(\mathbf{x} \to \mathbf{x}'; \nu_{\mathcal{X}}, \boldsymbol{\mu}_{\mathcal{X}}, \boldsymbol{\Sigma}_{\mathcal{X}})$ to each $\mathbf{x}_k$ individually. These updates can be performed in parallel, but the variables $\mathbf{x}_k$ no longer have the correct marginal distributions because of the coupling between the chains introduced by the approximation (this update violates detailed balance).

A second approach creates a valid MCMC method by including the multivariate t parameters in a joint distribution

$$p(\mathcal{X}, \nu, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = p(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma} \mid \mathcal{X}) \left[\prod_{k=1}^{K} \pi(\mathbf{x}_k) \right]. \tag{8}$$

Note that in Equation (8), each $\mathbf{x}_k$ has marginal distribution π. We can sample this joint distribution by alternately updating the variables and the multivariate t parameters so as to leave invariant the conditional distributions $p(\mathcal{X} \mid \nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $p(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma} \mid \mathcal{X})$. Ideally, we would like to update the collection $\mathcal{X}$ by updating each $\mathbf{x}_k$ in parallel. However, we cannot easily parallelize the update in this formulation because of the factor of $p(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma} \mid \mathcal{X})$, which nontrivially couples the chains.

4.2 The Two-Group Approach

Our proposed method creates a transition operator Q that satisfies both of the desired properties. That is, each $\mathbf{x}_k$ has marginal distribution π, and the update can be efficiently parallelized. This method circumvents the problems of the previous approaches by maintaining two groups of Markov chains and using each group to choose multivariate t parameters to update the other group. Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{K_1}\}$ and $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_{K_2}\}$ denote the states of the Markov chains in these two groups (in practice, we set $K_1 = K_2 = K$, where K is the number of available cores). The stationary distribution of the collection is

$$\Pi(\mathcal{X}, \mathcal{Y}) = \Pi_1(\mathcal{X})\Pi_2(\mathcal{Y}) = \left[\prod_{k=1}^{K_1} \pi(\mathbf{x}_k) \right] \left[\prod_{k=1}^{K_2} \pi(\mathbf{y}_k) \right].$$

By simulating a Markov chain which leaves this product distribution invariant, this method generates samples from the target distribution. Our Markov chain is based on a transition operator, Q, defined in two parts. The first part of the transition operator, Q_1, uses $\mathcal{Y}$ to determine parameters $\nu_{\mathcal{Y}}$, $\boldsymbol{\mu}_{\mathcal{Y}}$, and $\boldsymbol{\Sigma}_{\mathcal{Y}}$. It then uses these parameters to update $\mathcal{X}$. The second part of the transition operator, Q_2, uses $\mathcal{X}$ to determine parameters $\nu_{\mathcal{X}}$, $\boldsymbol{\mu}_{\mathcal{X}}$, and $\boldsymbol{\Sigma}_{\mathcal{X}}$. It then uses these parameters to update $\mathcal{Y}$. The transition operator Q is the composition of Q_1 and Q_2. The idea of maintaining a group of Markov chains and updating the states of some Markov chains based on the states of other Markov chains has been discussed in the literature before. For example, see Zhang and Sutton (2011); Gilks et al. (1994).

In order to make these descriptions more precise, we define Q_1 as follows. First, we specify a procedure for choosing the multivariate t parameters given the population $\mathcal{Y}$. We use an extension of the expectation-maximization algorithm (Liu and Rubin, 1995) to choose the maximum-likelihood multivariate t parameters given the data $\mathcal{Y}$. The details of this algorithm are described in Algorithm 4 in the Appendix. More concretely, we choose

$$\nu_{\mathcal{Y}}, \boldsymbol{\mu}_{\mathcal{Y}}, \boldsymbol{\Sigma}_{\mathcal{Y}} = \arg\max_{\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma}} \prod_{k=1}^{K_2} \mathcal{T}_{\nu}(\mathbf{y}_k \,;\, \boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

After choosing $\nu_{\mathcal{Y}}$, $\boldsymbol{\mu}_{\mathcal{Y}}$, and $\boldsymbol{\Sigma}_{\mathcal{Y}}$ in this manner, we update $\mathcal{X}$ by applying the transition operator $S(\mathbf{x} \to \mathbf{x}'; \nu_{\mathcal{Y}}, \boldsymbol{\mu}_{\mathcal{Y}}, \boldsymbol{\Sigma}_{\mathcal{Y}})$ to each $\mathbf{x}_k \in \mathcal{X}$ in parallel. The operator Q_2 is defined analogously.

Algorithm 3 Building the Approximation Using Parallelism

Input: Two groups of states $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{K_1}\}$ and $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_{K_2}\}$, a subroutine FIT-MVT which takes data and returns the maximum-likelihood t parameters, a subroutine GESS which performs a generalized elliptical slice sampling update

Output: Updated groups $\mathcal{X}'$ and $\mathcal{Y}'$

$\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma} \leftarrow \text{FIT-MVT}(\mathcal{Y})$
for all $\mathbf{x}_k \in \mathcal{X}$ **do**
    $\mathbf{x}'_k = \text{GESS}(\mathbf{x}_k, \nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$
$\mathcal{X}' \leftarrow \{\mathbf{x}'_1, \ldots, \mathbf{x}'_{K_1}\}$
$\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma} \leftarrow \text{FIT-MVT}(\mathcal{X}')$
for all $\mathbf{y}_k \in \mathcal{Y}$ **do**
    $\mathbf{y}'_k = \text{GESS}(\mathbf{y}_k, \nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$
$\mathcal{Y}' \leftarrow \{\mathbf{y}'_1, \ldots, \mathbf{y}'_{K_2}\}$

In the case where the number of chains in the collection $\mathcal{Y}$ is less than or close to the dimension of the distribution, the particular algorithm that we use to choose the parameters (Liu and Rubin, 1995) may not converge quickly (or at all). Suppose we are in the setting where $K < 2D$. In this situation, we can perform a regularized estimate of the parameters. We describe this procedure below. The choice $K < 2D$ probably overestimates the regime in which the algorithm for fitting the parameters performs poorly. The particular algorithm that we use appears to work well as long as $K \geq D$.

Let $\bar{\mathbf{y}}$ be the mean of $\mathcal{Y}$, and let $\{\mathbf{v}_1, \ldots, \mathbf{v}_J\}$ be the first J principal components of the set $\{\mathbf{y}_1 - \bar{\mathbf{y}}, \ldots, \mathbf{y}_K - \bar{\mathbf{y}}\}$, where $J = \lfloor \frac{K}{2} \rfloor$, and let $V = \text{span}(\mathbf{v}_1, \ldots, \mathbf{v}_J)$. Let $\mathbf{A}$ be the $D \times J$ matrix defined by $\mathbf{A}\mathbf{e}_j = \mathbf{v}_j$, where $\mathbf{e}_j$ is the jth standard basis vector in $\mathbb{R}^J$. This map identifies $\mathbb{R}^J$ with V.

Let the set $\hat{\mathcal{Y}}$ consist of the projections of the elements of $\mathcal{Y}$ onto $\mathbb{R}^J$ by $\hat{\mathbf{y}}_k = \mathbf{A}^\mathsf{T}\mathbf{y}_k$. Using the algorithm from Liu and Rubin (1995), fit the multivariate t parameters $\nu_{\hat{\mathcal{Y}}}$, $\boldsymbol{\mu}_{\hat{\mathcal{Y}}}$ and, $\boldsymbol{\Sigma}_{\hat{\mathcal{Y}}}$ to $\hat{\mathcal{Y}}$. At this point, we have constructed a J-dimensional multivariate t distribution, but we would like a D-dimensional one. We construct the desired distribution by rotating back to the original space. Concretely, we can set

$$\begin{aligned} \nu_{\mathcal{Y}} &= \nu_{\hat{\mathcal{Y}}} \\ \boldsymbol{\mu}_{\mathcal{Y}} &= \mathbf{A}\,\boldsymbol{\mu}_{\hat{\mathcal{Y}}} + \bar{\mathbf{y}} \\ \boldsymbol{\Sigma}_{\mathcal{Y}} &= \mathbf{A}\,\boldsymbol{\Sigma}_{\hat{\mathcal{Y}}}\,\mathbf{A}^\mathsf{T} + \epsilon\,\mathbf{I}_D, \end{aligned}$$

where $\mathbf{I}_D$ is the $D \times D$ identity matrix and ϵ is the median entry on the diagonal of $\boldsymbol{\Sigma}_{\hat{\mathcal{Y}}}$. We add a scaled identity matrix to the covariance parameter to avoid producing a degenerate distribution. The choice of ϵ is based on intuition about typical values of the variance of π in the directions orthogonal to V.

We emphasize that the nature of the procedure for fitting a multivariate t distribution to some points is not important to our algorithm. One could devise more sophisticated approaches drawing on ideas from the literature on high-dimensional covariance estimation, see Ravikumar et al. (2011) for instance, but we merely choose a simple idea that

seems to work in practice. Since our default choice (if there are at least $2D$ chains, then choose the maximum-likelihood parameters, otherwise project to a lower dimension, choose the maximum-likelihood parameters, and then pad the diagonal of the covariance parameter) works well, the fact that one could design a more sophisticated procedure does not compromise the tuning-free nature of our algorithm.

4.3 Correctness

To establish the correctness of our algorithm, we treat the collection of chains as a single aggregate Markov chain, and we show that this aggregate Markov chain with transition operator Q correctly samples from the product distribution Π.

We wish to show that Q_1 and Q_2 preserve the invariant distributions Π_1 and Π_2 respectively. As the two cases are identical, we consider only the first. We have

$$\begin{aligned}
\int \Pi_1(\mathcal{X})\, Q_1(\mathcal{X} \to \mathcal{X}')\, \mathrm{d}\mathcal{X} &= \int \Pi_1(\mathcal{X})\, Q_1(\mathcal{X} \to \mathcal{X}' \,|\, \nu_\mathcal{Y}, \boldsymbol{\mu}_\mathcal{Y}, \boldsymbol{\Sigma}_\mathcal{Y})\, \mathrm{d}\mathcal{X} \\
&= \prod_{k=1}^{K_1} \left[\int \pi(\mathbf{x}_k)\, S(\mathbf{x}_k \to \mathbf{x}'_k; \nu_\mathcal{Y}, \boldsymbol{\mu}_\mathcal{Y}, \boldsymbol{\Sigma}_\mathcal{Y})\, \mathrm{d}\mathbf{x}_k \right] \\
&= \Pi_1(\mathcal{X}').
\end{aligned}$$

The last equality uses the fact that π is the stationary distribution of $S(\mathbf{x} \to \mathbf{x}'; \nu_\mathcal{Y}, \boldsymbol{\mu}_\mathcal{Y}, \boldsymbol{\Sigma}_\mathcal{Y})$, so we see that Q leaves the desired product distribution invariant.

Within a single chain, elliptical slice sampling has a nonzero probability of transitioning to any region that has nonzero probability under the posterior, as described by Murray et al. (2010). The transition operator Q updates the chains in a given group independently of one another. Therefore Q has a nonzero probability of transitioning to any region that has nonzero probability under the product distribution. It follows that the transition operator is both irreducible and aperiodic. These conditions together ensure that this Markov transition operator has a unique invariant distribution, namely Π, and that the distribution over the state of the Markov chain created from this transition operator will converge to this invariant distribution (Roberts and Rosenthal, 2004). It follows that, in the limit, samples derived from the repeated application of Q will be drawn from the desired distribution.

4.4 Cost and Complexity

There is a cost to the construction of the multivariate t approximation. Although the user has some flexibility in the choice of t parameters, we fit them with the iterative algorithm described by Liu and Rubin (1995) and in Algorithm 4 of the Appendix. Let D be the dimension of the distribution and let K be the number of parallel chains. Then the complexity of each iteration is $O(D^3K)$, which comes from the fact that we invert a $D \times D$ matrix for each of the K chains. Empirically, Algorithm 4 appears to converge in a small number of iterations when the number of parallel Markov chains in each group exceeds the dimension of the distribution. As described in the next section, this cost can be amortized by reusing the same approximation for multiple updates. On the challenging distributions that most interest us, the cost of constructing the approximation (when amortized in this manner), will be negligible compared to the cost of evaluating the density function.

An additional concern is the overhead from sharing information between chains. The chains must communicate in order to build a multivariate t approximation, and so the updates must be synchronized. Since elliptical slice sampling requires a variable amount of time, updating the different chains will take different amounts of time, and the faster chains may end up waiting for the slower ones. We can mitigate this cost by performing multiple updates between such periods of information sharing. In this manner, we can perform as much computation as we want between synchronizations without compromising the validity of the algorithm. As we increase the number of updates performed between synchronizations, the fraction of time spent waiting will diminish.

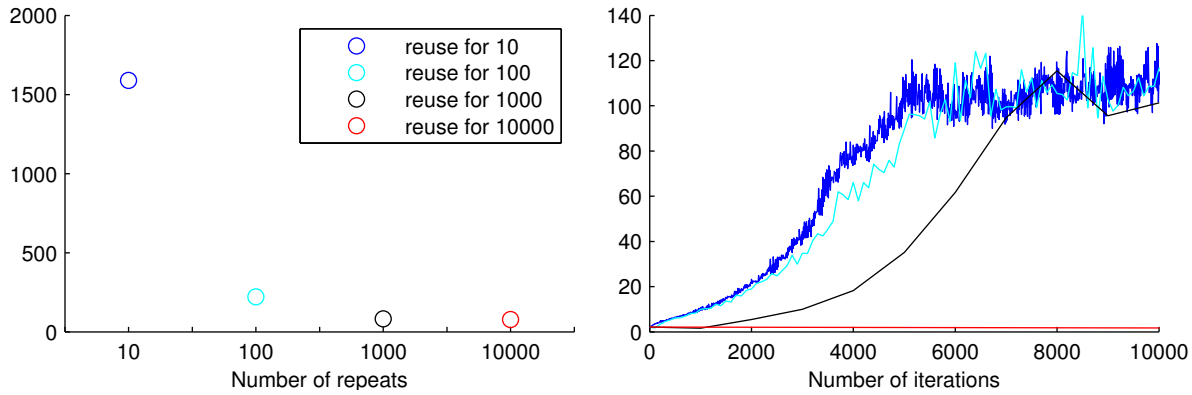
The time measured in our experiments is wall-clock time, which includes the overhead from constructing the approximation and from synchronizing the chains.

4.5 Reusing the Approximation

Here we explain that reusing the same approximation is valid. To illustrate this point, let the transition operators Q_1 and Q_2 be defined as before. In our description of the algorithm, we defined the transition operator Q as the composition $Q = Q_2 Q_1$. However, both Q_1 and Q_2 preserve the desired product distribution, so we may use any transition operator of the form $Q = Q_2^{r_2} Q_1^{r_1}$, where this notation indicates that we first apply Q_1 for r_1 rounds and then we apply Q_2 for r_2 rounds. As long as $r_2, r_1 \geq 1$, the composite transition operator is ergodic. When we apply Q_1 multiple times in a row, the states $\mathcal{Y}$ do not change, so if Q_1 computes $\nu_{\mathcal{Y}}$, $\boldsymbol{\mu}_{\mathcal{Y}}$, and $\boldsymbol{\Sigma}_{\mathcal{Y}}$ deterministically from $\mathcal{Y}$, then we need only compute these values once. Reusing the approximation works even if Q_1 samples $\nu_{\mathcal{Y}}$, $\boldsymbol{\mu}_{\mathcal{Y}}$, and $\boldsymbol{\Sigma}_{\mathcal{Y}}$ from some distribution. In this case, we can model the randomness by introducing a separate variable $r_{\mathcal{Y}}$ in the Markov chain, and letting Q_1 compute $\nu_{\mathcal{Y}}$, $\boldsymbol{\mu}_{\mathcal{Y}}$, and $\boldsymbol{\Sigma}_{\mathcal{Y}}$ deterministically from $\mathcal{Y}$ and $r_{\mathcal{Y}}$.

Our algorithm maintains two collections of Markov chains, one of which will always be idle. Therefore, each collection can take advantage of all available cores. Given K cores, it makes sense to use two collections of K Markov chains. In general, it seems to be a good idea to sample equally from both collections so that the chains in both collections burn in.

To motivate reusing the approximation, we demonstrate the effect of reusing the approximation for different numbers of iterations on a Gaussian distribution in 100 dimensions (the same one that we use in Section 6.2). For each value of i from 1 to 4, we sample this distribution for 10^4 iterations and we reuse each approximation for 10^i iterations. We show plots of the running time of GESS and the convergence of the approximation for different values of i. Figure 4 shows how the amount of time required by GESS changes as we vary i, and how the covariance matrix parameter of the fitted multivariate t approximation changes over time for the different values of i. We summarize the covariance matrix parameter by its trace $\mathrm{tr}(\boldsymbol{\Sigma})$. The figure shows that increasing the number of iterations for which we reuse the approximation can dramatically reduce the amount of time required by GESS. It also shows that if we rebuild the approximation frequently, the approximation will settle on a reasonable approximation in fewer iterations. However, there is little difference between rebuilding the approximation every 10 iterations versus every 100 iterations (in terms of the number of iterations required), while there is a dramatic difference in the time required.



(a) Durations of GESS (seconds)

(b) Plots of $\mathrm{tr}(\mathbf{\Sigma})$

Figure 4: We used GESS to sample a multivariate Gaussian distribution in 100 dimensions for 10^4 iterations. We repeated this procedure four times, reusing the approximation for 10^1, 10^2, 10^3, and 10^4 iterations. **(a)** Shows the durations (in seconds) of the sampling procedures as we varied the number of iterations for which we reused the approximation. **(b)** Shows how $\mathrm{tr}(\mathbf{\Sigma})$ changes over time in the four different settings.

5. Related Work

Our work uses updates on a product distribution in the style of Adaptive Direction Sampling (Gilks et al., 1994), which has inspired a large literature of related methods. The closest research to our work makes use of slice-sampling based updates of product distributions along straight-line directions chosen by sampling pairs of points (MacKay, 2003; Ter Braak, 2006). The work on elliptical slice sampling suggests that in high dimensions larger steps can be taken along curved trajectories, given an appropriate Gaussian fit. Using closed ellipses also removes the need to set an initial step size or to build a bracket.

The recent affine invariant ensemble sampler (Goodman and Weare, 2010) also uses Gaussian fits to a population, in that case to make Metropolis proposals. Our work differs by using a scale-mixture of Gaussians and elliptical slice sampling to perform updates on a variety of scales with self-adjusting step-sizes. Rather than updating each member of the population in sequence, our approach splits the population into two groups and allows the members of each group to be updated in parallel.

Population MCMC with parallel tempering (Friel and Pettitt, 2008) is another parallel sampling approach that involves sampling from a product distribution. It uses separate chains to sample a sequence of distributions interpolating between the target distribution and a simpler distribution. The different chains regularly swap states to encourage mixing. In this setting, samples are generated only from a single chain, and all of the others are auxiliary. However, some tasks such as computing model evidence can make use of samples from all of the chains (Friel and Pettitt, 2008).

Recent work on Hamiltonian Monte Carlo has attempted to reduce the tuning burden (Hoffman and Gelman, 2014). A user friendly tool that combines this work with a software

stack supporting automatic differentiation is under development (Stan Development Team, 2012). We feel that this alternative line of work demonstrates the interest in more practical MCMC algorithms applicable to a variety of continuous-valued parameter spaces and is very promising. Our complementary approach introduces simpler algorithms with fewer technical software requirements. In addition, our two-population approach to parallelization could be applied with whichever methods become dominant in the future.

6. Experiments

In this section, we compare Algorithm 3 with other parallel MCMC algorithms by measuring how quickly the Markov chains mix on a number of different distributions. Second, we compare how the performance of Algorithm 3 scales with the dimension of the target distribution, the number of cores used, and the number of chains used per core.

These experiments were run on an EC2 cluster with 5 nodes, each with two eight-core Intel Xeon E5-2670 CPUs. We implement all algorithms in Python, using the IPython environment (Pérez and Granger, 2007) for parallelism.

6.1 Comparing Mixing

We empirically compare the mixing of the parallel MCMC samplers on seven distributions. We quantify their mixing by comparing the effective number of samples produced by each method. This quantity can be approximated as the product of the number of chains with the effective number of samples from the product distribution. We estimate the effective number of samples from the product distribution by computing the effective number of samples from its sequence of log likelihoods. We compute effective sample size using R-CODA (Plummer et al., 2006), and we compare the results using two metrics: effective samples per second and effective samples per density function evaluation (in the case of Hamiltonian Monte Carlo, we count gradient evaluations as density function evaluations).

In each experiment, we run each algorithm with 100 parallel chains. Unless otherwise noted, we burn in for 10^4 iterations and sample for 10^5 iterations. We run five trials for each experiment to estimate variability.

Figure 5 shows the average effective number of samples, with error bars, according to the two different metrics. Bars are omitted where the sequence of aggregate log likelihoods did not converge according to the Geweke convergence diagnostic (Geweke, 1992). We diagnose this using the tool from R-CODA (Plummer et al., 2006).

6.1.1 Samplers Considered

We compare generalized elliptical slice sampling (GESS) with parallel versions of several different sampling algorithms.

First, we consider random-direction slice sampling (RDSS) (MacKay, 2003) and coordinate-wise slice sampling (CWSS) (Neal, 2003). These are variants of slice sampling which differ in their choice of direction (a random direction versus a random axis-aligned direction) in which to sample. RDSS is rotation invariant like GESS, but CWSS is not.

In addition, we compare to a simple Metropolis–Hastings (MH) (Metropolis et al., 1953) algorithm whose proposal distribution is a spherical Gaussian centered on the current state.

A tuning period is used to adjust the MH step size so that the acceptance ratio is as close as possible to the value 0.234, which is optimal in some settings (Roberts and Rosenthal, 1998). This tuning is done independently for each chain. We also compare to an adaptive MCMC (AMH) algorithm following the approach in Roberts and Rosenthal (2006) in which the covariance of a Metropolis–Hastings proposal is adapted to the history of the "Markov" chain.

We also compare to the No-U-Turn sampler (Hoffman and Gelman, 2014), which is an implementation of Hamiltonian Monte Carlo (HMC) combined with procedures to automatically tune the step size parameter and the number of steps parameter. Due to the large number of function evaluations per sample required by HMC, we run HMC for a factor of 10 or 100 fewer iterations in order to make the algorithms roughly comparable in terms of wall-clock time. Though we include the comparisons, we do not view HMC as a perfectly comparable algorithm due to its requirement that the density function of the target distribution be differentiable. Though the target distribution is often differentiable in principle, there are many practical situations in which the gradient is difficult to access, either by manual computation or by automatic differentiation, possibly because evaluating the density function requires running a complicated black-box subroutine. For instance, in computer vision problems, evaluating the likelihood function may require rendering an image or running graph cuts. See Tarlow and Adams (2012) or Lang and Hogg (2012) for examples.

We compare to parallel tempering (PT) (Friel and Pettitt, 2008), using each Markov chain to sample the distribution at a different temperature (if the target distribution has density $\pi(\mathbf{x})$, then the distribution "at temperature t" has density proportional to $\pi(\mathbf{x})^{1/t}$) and swapping states between the Markov chains at regular intervals. Samples from the target distribution are produced by only one of the chains. Using PT requires the practitioner to pick a temperature schedule, and doing so often requires a significant amount of experimentation (Neal, 2001). We follow the practice of Friel and Pettitt (2008) and use a geometric temperature schedule. As with HMC, we do not view PT as entirely comparable in the absence of an automatic and principled way to choose the temperatures of the different Markov chains. One of the main goals of GESS is to provide a black-box MCMC algorithm that imposes as few restrictions on the target distribution as possible and that requires no expertise or experimentation on the part of the user.

6.1.2 Distributions

In this section, we describe the different distributions that we use to compare the mixing of our samplers.

Funnel: A ten-dimensional funnel-shaped distribution described in Neal (2003). The first coordinate is distributed normally with mean zero and variance nine. Conditioned on the first coordinate v, the remaining coordinates are independent identically-distributed normal random variables with mean zero and variance e^v. In this experiment, we initialize each Markov chain from a spherical multivariate Gaussian centered on the origin.

Gaussian Mixture: An eight-component mixture of Gaussians in eight dimensions. Each component is a spherical Gaussian with unit variance. The components are distributed

uniformly at random within a hypercube of edge length four. In this experiment, we initialize each Markov chain from a spherical multivariate Gaussian centered on the origin.

Breast Cancer: The posterior density of a linear logistic regression model for a binary classification problem with thirty explanatory variables (thirty-one dimensions) using the Breast Cancer Wisconsin data set (Street et al., 1993). The data set consists of 569 data points. We scale the data so that each coordinate has unit variance, and we place zero-mean Gaussian priors with variance 100 on each of the regression coefficients. In this experiment, we initialize each Markov chain from a spherical multivariate Gaussian centered on the origin.

German Credit: The posterior density of a linear logistic regression model for a binary classification problem with twenty-four explanatory variables (twenty-five dimensions) from the UCI repository (Frank and Asuncion, 2010). The data set consists of 1000 data points. We scale the data so that each coordinate has unit variance, and we place zero-mean Gaussian priors with variance 100 on each of the regression coefficients. In this experiment, we initialize each Markov chain from a spherical multivariate Gaussian centered on the origin.

Stochastic Volatility: The posterior density of a simple stochastic volatility model fit to synthetic data in fifty-one dimensions. This distribution is a smaller version of a distribution described in Hoffman and Gelman (2014). In this experiment, we burn-in for 10^5 iterations and sample for 10^5 iterations. We initialize each Markov chain from a spherical multivariate Gaussian centered on the origin and we take the absolute value of the first parameter, which is constrained to be positive.

Ionosphere: The posterior density on covariance hyperparameters for Gaussian process regression applied to the Ionosphere data set (Sigillito et al., 1989). We use a squared exponential kernel with thirty-four length-scale hyperparameters and 100 data points. We place exponential priors with rate 0.1 on the length-scale hyperparameters. In this experiment, we burn-in for 10^4 iterations and sample for 10^4 iterations. We initialize each Markov chain from a spherical multivariate Gaussian centered on the vector $(1, \ldots, 1)^\mathsf{T}$.

SNP Covariates: The posterior density of the parameters of a generative model for gene expression levels simulated in thirty-eight dimensions using actual genomic sequences from 480 individuals for covariate data (Engelhardt and Adams, 2014). In this experiment, we burn-in for 2000 iterations and sample for 10^4 iterations. We initialize each Markov chain from a spherical multivariate Gaussian centered on the origin and we take the absolute value of the first three parameters, which are constrained to be positive.

6.1.3 Mixing Results

The results of the mixing experiments are shown in Figure 5. For the most part, GESS sampled more effectively than the other algorithms according to both metrics. The poor performance of PT can be attributed to the fact that PT only produces samples from one of its chains, unlike the other algorithms, which produce samples from 100 chains. HMC also performed well, although it failed to converge on the SNP Covariates distribution. The density function of this particular distribution is only piecewise continuous, with the discontinuities arising from thresholding in the model. In this case, the gradient and curvature largely reflect the prior, whereas the likelihood mostly manifests itself in the discontinuities of the distribution.

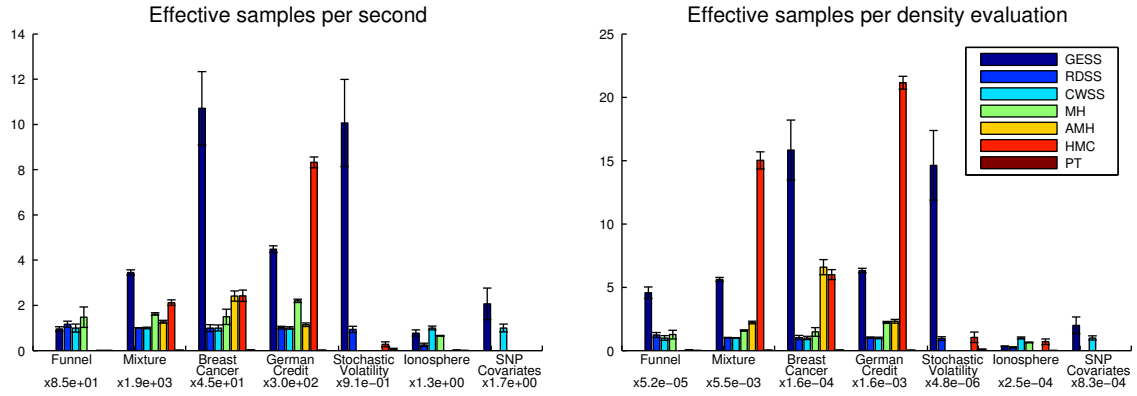


Figure 5: The results of experimental comparisons of seven parallel MCMC methods on seven distributions. Each figure shows seven groups of bars, (one for each distribution) and the vertical axis shows the effective number of samples per unit cost. Error bars are included. Bars are omitted where the method failed to converge according to the Geweke diagnostic (Geweke, 1992). The costs are *per second* (left) and *per density function evaluation* (right). Mean and standard error for five runs are shown. Each group of bars has been rescaled for readability: the number beneath each group gives the effective samples corresponding to CWSS, which always has height one.

One reason for the rapid mixing of GESS is that GESS performs well even on highly-skewed distributions. RDSS, CWSS, MH, and PT propose steps in uninformed directions, the vast majority of which lead away from the region of high density. As a result, these algorithms take very small steps, causing successive states to be highly correlated. In the case of GESS, the multivariate t approximation builds information about the global shape of the distribution (including skew) into the transition operator. As a consequence, the Markov chain can take long steps along the length of the distribution, allowing the Markov chain to mix much more rapidly. Skewed distributions can arise as a result of the user not knowing the relative length scales of the parameters or as a result of redundancy in the parameterization. Therefore, the ability to perform well on such distributions is frequently relevant.

These results show that a multivariate t approximation to the target distribution provides enough information to greatly speed up the mixing of the sampler and that this information can be used to improve the convergence of the sampler. These improvements occur on top of the performance gain from using parallelism.

6.2 Scaling the Number of Cores

We wish to explore the performance of GESS as a function of the dimension D of the target distribution, the number C of cores available, and the number K of parallel chains. In this

$D = 50$	$K = C$	$K = 2C$	$K = 3C$	$K = 4C$	$K = 5C$
$C = 20$	$10^{-0.5} \pm 10^{-0.4}$	$10^{-1.2} \pm 10^{-1.5}$	$10^{-1.5} \pm 10^{-1.8}$	$10^{-1.7} \pm 10^{-1.8}$	$10^{-1.6} \pm 10^{-1.6}$
$C = 40$	$10^{-0.8} \pm 10^{-0.9}$	$10^{-2.6} \pm 10^{-2.6}$	$10^{-1.9} \pm 10^{-1.9}$	$10^{-1.8} \pm 10^{-1.8}$	$10^{-2.4} \pm 10^{-2.6}$
$C = 60$	$10^{-1.6} \pm 10^{-1.5}$	$10^{-1.6} \pm 10^{-1.7}$	$10^{-2.1} \pm 10^{-2.2}$	$10^{-2.1} \pm 10^{-2.2}$	$10^{-2.2} \pm 10^{-2.4}$
$C = 80$	$10^{-1.3} \pm 10^{-1.1}$	$10^{-2.4} \pm 10^{-2.8}$	$10^{-2.4} \pm 10^{-2.4}$	$10^{-2.1} \pm 10^{-2.4}$	$10^{-2.3} \pm 10^{-2.5}$
$C = 100$	$10^{-1.6} \pm 10^{-1.7}$	$10^{-1.7} \pm 10^{-1.7}$	$10^{-2.0} \pm 10^{-2.0}$	$10^{-2.2} \pm 10^{-2.4}$	$10^{-2.5} \pm 10^{-2.3}$

$D = 100$	$K = C$	$K = 2C$	$K = 3C$	$K = 4C$	$K = 5C$
$C = 20$	$10^{+0.3} \pm 10^{+0.2}$	$10^{-1.3} \pm 10^{-2.2}$	$10^{-1.7} \pm 10^{-2.1}$	$10^{-1.9} \pm 10^{-2.2}$	$10^{-2.4} \pm 10^{-3.5}$
$C = 40$	$10^{-1.1} \pm 10^{-1.1}$	$10^{-1.9} \pm 10^{-2.1}$	$10^{-2.5} \pm 10^{-3.2}$	$10^{-2.5} \pm 10^{-2.6}$	$10^{-2.7} \pm 10^{-3.0}$
$C = 60$	$10^{-1.7} \pm 10^{-2.0}$	$10^{-2.5} \pm 10^{-2.8}$	$10^{-2.8} \pm 10^{-3.0}$	$10^{-2.9} \pm 10^{-3.4}$	$10^{-2.9} \pm 10^{-3.1}$
$C = 80$	$10^{-2.1} \pm 10^{-2.7}$	$10^{-2.7} \pm 10^{-2.8}$	$10^{-2.7} \pm 10^{-3.0}$	$10^{-2.9} \pm 10^{-3.2}$	$10^{-3.1} \pm 10^{-4.0}$
$C = 100$	$10^{-2.4} \pm 10^{-2.6}$	$10^{-2.8} \pm 10^{-3.3}$	$10^{-3.0} \pm 10^{-3.5}$	$10^{-3.0} \pm 10^{-3.6}$	$10^{-2.9} \pm 10^{-3.0}$

$D = 150$	$K = C$	$K = 2C$	$K = 3C$	$K = 4C$	$K = 5C$
$C = 20$	$10^{+2.3} \pm 10^{+1.4}$	$10^{+2.3} \pm 10^{+1.7}$	$10^{+1.4} \pm 10^{+1.0}$	$10^{+0.5} \pm 10^{+0.2}$	$10^{-0.7} \pm 10^{-1.0}$
$C = 40$	$10^{+2.1} \pm 10^{+1.6}$	$10^{-0.1} \pm 10^{-0.0}$	$10^{-1.1} \pm 10^{-1.2}$	$10^{-1.4} \pm 10^{-1.4}$	$10^{-1.8} \pm 10^{-1.7}$
$C = 60$	$10^{+1.3} \pm 10^{+0.7}$	$10^{-1.2} \pm 10^{-1.2}$	$10^{-1.6} \pm 10^{-1.5}$	$10^{-1.9} \pm 10^{-2.0}$	$10^{-1.7} \pm 10^{-1.6}$
$C = 80$	$10^{-0.0} \pm 10^{-0.0}$	$10^{-1.7} \pm 10^{-1.8}$	$10^{-2.2} \pm 10^{-2.3}$	$10^{-1.9} \pm 10^{-2.0}$	$10^{-2.1} \pm 10^{-2.6}$
$C = 100$	$10^{-0.7} \pm 10^{-1.0}$	$10^{-1.8} \pm 10^{-2.1}$	$10^{-1.9} \pm 10^{-2.1}$	$10^{-2.0} \pm 10^{-2.1}$	$10^{-2.3} \pm 10^{-2.3}$

$D = 200$	$K = C$	$K = 2C$	$K = 3C$	$K = 4C$	$K = 5C$
$C = 20$	$10^{+2.8} \pm 10^{+2.5}$	$10^{+3.0} \pm 10^{+2.4}$	$10^{+3.1} \pm 10^{+2.1}$	$10^{+3.1} \pm 10^{+1.9}$	$10^{+3.0} \pm 10^{+1.5}$
$C = 40$	$10^{+3.1} \pm 10^{+1.6}$	$10^{+3.1} \pm 10^{+1.7}$	$10^{+2.7} \pm 10^{+1.6}$	$10^{+1.1} \pm 10^{+0.6}$	$10^{-1.4} \pm 10^{-1.6}$
$C = 60$	$10^{+3.1} \pm 10^{+1.6}$	$10^{+2.6} \pm 10^{+1.8}$	$10^{-0.6} \pm 10^{-0.8}$	$10^{-1.7} \pm 10^{-2.0}$	$10^{-2.0} \pm 10^{-2.8}$
$C = 80$	$10^{+3.1} \pm 10^{+1.7}$	$10^{+0.7} \pm 10^{+0.1}$	$10^{-1.7} \pm 10^{-2.3}$	$10^{-1.9} \pm 10^{-1.9}$	$10^{-2.1} \pm 10^{-2.5}$
$C = 100$	$10^{+3.0} \pm 10^{+2.1}$	$10^{-1.4} \pm 10^{-1.6}$	$10^{-2.3} \pm 10^{-2.8}$	$10^{-2.0} \pm 10^{-2.6}$	$10^{-2.3} \pm 10^{-2.9}$

$D = 250$	$K = C$	$K = 2C$	$K = 3C$	$K = 4C$	$K = 5C$
$C = 20$	$10^{+3.5} \pm 10^{+2.0}$	$10^{+3.5} \pm 10^{+1.5}$	$10^{+3.5} \pm 10^{+1.7}$	$10^{+3.5} \pm 10^{+1.4}$	$10^{+3.5} \pm 10^{+1.6}$
$C = 40$	$10^{+3.5} \pm 10^{+2.3}$	$10^{+3.5} \pm 10^{+1.3}$	$10^{+3.5} \pm 10^{+1.6}$	$10^{+3.5} \pm 10^{+2.1}$	$10^{+3.6} \pm 10^{+1.8}$
$C = 60$	$10^{+3.5} \pm 10^{+2.0}$	$10^{+3.5} \pm 10^{+1.6}$	$10^{+3.6} \pm 10^{+2.1}$	$10^{+3.6} \pm 10^{+2.4}$	$10^{+2.3} \pm 10^{+1.9}$
$C = 80$	$10^{+3.5} \pm 10^{+1.6}$	$10^{+3.5} \pm 10^{+1.9}$	$10^{+3.5} \pm 10^{+2.2}$	$10^{+1.1} \pm 10^{+0.8}$	$10^{-0.8} \pm 10^{-0.9}$
$C = 100$	$10^{+3.5} \pm 10^{+1.8}$	$10^{+3.6} \pm 10^{+2.0}$	$10^{+2.2} \pm 10^{+1.7}$	$10^{+0.3} \pm 10^{+0.2}$	$10^{-0.1} \pm 10^{-0.2}$

Figure 6: For each choice of D, C, and K, we run GESS, estimate $\boldsymbol{\sigma}$, and report the squared error averaged over 5 trials along with error bars. Smaller numbers are better. Average errors less than 1 are shown in blue.

experiment, we consider all 125 triples (D, C, K) such that

$$\begin{aligned} D &\in \{50, 100, 150, 200, 250\} \\ C &\in \{20, 40, 60, 80, 100\} \\ K &\in \{C, 2C, 3C, 4C, 5C\}. \end{aligned}$$

It makes sense to let K be an integer multiple of C so that each core will be tasked with updating the same number of chains (the experiments in Section 6.1 set K equal to C).

For each triple (D, C, K), we sample a D-dimensional multivariate Gaussian distribution centered on the origin whose precision matrix was generated from a Wishart distribution with identity scale matrix and D degrees of freedom. The distributions used in this experiment were modeled off of one of the distributions considered in Hoffman and Gelman (2014). We initialize GESS from a broad spherical Gaussian distribution centered on the origin, and we run GESS for 500 seconds. The first half of the resulting samples are discarded, and the second half of the resulting samples are used to estimate the vector $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_D)$, where σ_d is the marginal standard deviation of the dth coordinate. For each triple (D, C, K), we run five trials. Figure 6 shows the resulting average squared error in the empirical estimate of $\boldsymbol{\sigma}$ after 500 seconds. Error bars are included as well.

When aggregating samples from K independent Markov chains, we would expect the squared error of our estimator to decrease at the rate $1/K$. However, in the setting of GESS, additional chains not only provide additional samples, but may enable the construction of a more accurate approximation to the target distribution thereby enabling the other chains to sample more effectively. In some situations, the presence of additional chains can even enable the sampler to converge in situations where it otherwise would not.

We can see this effect in Figure 6. Singling out the column corresponding to $D = 200$ and $K = 3C$, we notice that using either 20 or 40 cores, GESS fails to estimate $\boldsymbol{\sigma}$, indeed the Markov chain fails to burn in during the allotted time (the average squared errors are $10^{3.1}$ and $10^{2.7}$ respectively). However, once we increase the number of cores to 60, 80, and 100, GESS provides an accurate estimate of $\boldsymbol{\sigma}$ (the average squared errors are $10^{-0.6}$, $10^{-1.7}$, and $10^{-2.3}$ respectively). In this case, increasing the number of cores enabled our estimator to converge. This property contrasts sharply with many other approaches to parallel sampling. If a single Markov chain running MH will not converge, then one-hundred chains running MH will not converge either.

7. Discussion

In this paper, we generalized elliptical slice sampling to handle arbitrary continuous distributions using a scale mixture of Gaussians to approximate the target distribution. We then showed that parallelism can be used to dynamically choose the parameters of the scale mixture of Gaussians in a way that encodes information about the shape of the target distribution in the transition operator. The result is Markov chain Monte Carlo algorithm with a number of desirable properties. In particular, it mixes well in the presence of strong dependence, it does not require hand tuning, and it can be parallelized over hundreds of cores.

We compared our algorithm to several other parallel MCMC algorithms in a variety of settings. We found that generalized elliptical slice sampling (GESS) mixed more rapidly

than the other algorithms on a variety of distributions, and we found evidence that the performance of GESS can scale superlinearly in the number of available cores.

One possible area of future work is reducing the overhead from the information sharing. In Section 4.5 we remarked that the synchronization requirement leads to faster chains waiting for slower chains. There are a number of factors which contribute to the difference in speed from chain to chain. Most obviously, some chains may be running on faster machines. More subtly, the slice sampling procedure performs a variable number of function evaluations per update, and the average number of required updates may be a function of location. For instance, Markov chains whose current states lie in narrow portions of the distribution may require more function evaluations per update. In each situation, the chains with the rapid updates end up waiting for the chains with the slower updates, leaving some processors idle. We imagine that a cleverly-engineered system would be able to account for the potentially different update speeds, perhaps by sending the chains in the narrower parts of the distribution to the faster machines or by allowing the slower chains to spawn multiple threads. Properly done, the performance gain in wall-clock time due to using GESS should approach the gain as measured by function evaluations.

In addition to using parallelism to distribute the computational load of MCMC, we saw that our algorithm was able to use information from the parallel chains to speed up mixing. One area of future work is extending the algorithm to take advantage of a greater number of cores. The magnitude of this performance gain depends on the accuracy of our multivariate t approximation, which will increase, to a point, as the number of available cores grows. However, there is a limit to how well a multivariate t distribution can approximate an arbitrary distribution. We chose to use the multivariate t distribution because it has the flexibility to capture the general allocation of probability mass of a given distribution, but it is too coarse to capture more complex features such as the locations of multiple modes. After some point, the approximation will not significantly improve. A more general approach would be to use a scale-location mixture of Gaussians, which could accurately approximate a much larger class of distributions. The idea of approximating the target distribution with a mixture of Gaussians has been explored by Ji and Schmidler (2010) in the context of adaptive Metropolis–Hastings. We leave it to future work to explore this more general setting.

Acknowledgments

We thank Barbara Engelhardt for kindly sharing the simulated genetics data. We thank Eddie Kohler, Michael Gelbart, and Oren Rippel for valuable discussions. This work was funded by DARPA Young Faculty Award N66001-12-1-4219 and an Amazon AWS in Research grant.

Appendix A

In Algorithm 4, we detail the algorithm for estimating the maximum likelihood multivariate t parameters ν, $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ from Liu and Rubin (1995).

Algorithm 4 Computing the maximum likelihood multivariate t parameters

Input: I points $\mathbf{x}_i$ (each D dimensional)
Output: Maximum likelihood multivariate t parameters ν, $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$

1: $t \leftarrow 0$
2: Initialize $\nu^{(0)}$, $\boldsymbol{\mu}^{(0)}$, and $\boldsymbol{\Sigma}^{(0)}$
3: **while** $|\nu^{(t)} - \nu^{(t-1)}| < \epsilon$ **do**
4: Compute the distances from each point $\mathbf{x}_i$ to $\boldsymbol{\mu}^{(t)}$ with respect to $\boldsymbol{\Sigma}^{(t)}$

$$\delta_i^{(t)} = \left(\mathbf{x}_i - \boldsymbol{\mu}^{(t)}\right)^{\mathsf{T}} \left(\boldsymbol{\Sigma}^{(t)}\right)^{-1} \left(\mathbf{x}_i - \boldsymbol{\mu}^{(t)}\right)$$

5: Set

$$w_i^{(t+1)} = \frac{\nu^{(t)} + D}{\nu^{(t)} + \delta_i^{(t)}}$$

6: Update the mean and covariance parameters via

$$\begin{aligned} \boldsymbol{\mu}^{(t+1)} &= \frac{\sum_{i=1}^{I} w_i^{(t+1)} \mathbf{x}_i}{\sum_{i=1}^{I} w_i^{(t+1)}} \\ \boldsymbol{\Sigma}^{(t+1)} &= \frac{1}{I} \sum_{i=1}^{I} w_i^{(t+1)} \left(\mathbf{x}_i - \boldsymbol{\mu}^{(t)}\right) \left(\mathbf{x}_i - \boldsymbol{\mu}^{(t)}\right)^{\mathsf{T}} \end{aligned}$$

7: Using the updated mean and covariance parameters, recompute the distance

$$\delta_i^{(t+1)} = \left(\mathbf{x}_i - \boldsymbol{\mu}^{(t+1)}\right)^{\mathsf{T}} \left(\boldsymbol{\Sigma}^{(t+1)}\right)^{-1} \left(\mathbf{x}_i - \boldsymbol{\mu}^{(t+1)}\right)$$

8: Let ψ be the digamma function, and let

$$w_i = \frac{\nu + D}{\nu + \delta_i^{(t+1)}}$$

9: Set $\nu^{(t+1)}$ by solving for ν in the equation

$$-\psi\left(\frac{\nu}{2}\right) + \log\left(\frac{\nu}{2}\right) + \frac{1}{I} \sum_{i=1}^{I} \left(\log\left(w_i\right) - w_i\right) + \psi\left(\frac{\nu + D}{2}\right) - \log\left(\frac{\nu + D}{2}\right) = -1$$

10: $t \leftarrow t + 1$
11: **return** $\nu^{(t)}$, $\boldsymbol{\mu}^{(t)}$, and $\boldsymbol{\Sigma}^{(t)}$

References

Ryan P. Adams, Iain Murray, and David MacKay. The Gaussian process density sampler. In *Advances in Neural Information Processing Systems 21*, 2009.

Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. Introduction to MCMC for machine learning. *Machine Learning*, 50(1):5–43, 2003.

Russell Bradford and Alun Thomas. Markov chain Monte Carlo methods for family trees using a parallel processor. *Statistics and Computing*, 6:67–75, 1996.

Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng, editors. *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, 2011.

Luc Devroye. *Non-Uniform Random Variate Generation*. Springer, 1986.

Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195:216–222, 1987.

Barbara E. Engelhardt and Ryan P. Adams. Bayesian structured sparsity from Gaussian fields. 2014. In Preparation.

Andrew Frank and Arthur Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

Nial Friel and Anthony N. Pettitt. Marginal likelihood estimation via power posteriors. *Journal of the Royal Statistical Society B*, 70(3):589–607, 2008.

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

John Geweke. Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. *Bayesian Statistics*, 4:169–193, 1992.

Walter R. Gilks, Gareth O. Roberts, and Edward I. George. Adaptive direction sampling. *The Statistician*, 43(1):179–189, 1994.

Mark Girolami and Ben Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo. *Journal of the Royal Statistical Society B*, 73:1–37, 2011.

Jonathan Goodman and Jonathan Weare. Ensemble samplers with affine invariance. *Communications in Applied Mathematics and Computational Science*, 5(1):65–80, 2010.

Heikki Haario, Eero Saksman, and Johanna Tamminen. Componentwise adaptation for high dimensional MCMC. *Computational Statistics*, 20:265–273, 2005.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 1:97–109, 1970.

Matthew D. Hoffman and Andrew Gelman. The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15: 1351–1381, 2014.

Chunlin Ji and Scott C. Schmidler. Adaptive Markov chain Monte Carlo for Bayesian variable selection. *Journal of Computational and Graphical Statistics*, 2010. To Appear.

Dustin Lang and David W. Hogg. Searching for comets on the World Wide Web: The orbit of 17P/Holmes from the behavior of photographers. *The Astronomical Journal*, 144(2), 2012.

Chuanhai Liu and Donald B. Rubin. ML estimation of the t distribution using EM and its extensions, ECM and ECME. *Statistica Sinica*, 5:19–39, 1995.

Jun S. Liu, Faming Liang, and Wing Hung Wong. The multiple-try method and local optimization in Metropolis sampling. *Journal of the American Statistical Association*, 95:121–134, 2000.

David MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

Iain Murray, Ryan P. Adams, and David MacKay. Elliptical slice sampling. *Journal of Machine Learning Research: W&CP*, 9:541–548, 2010.

Radford M. Neal. Suppressing random walks in Markov chain Monte Carlo using ordered overrelaxation. Technical Report 9508, Department of Statistics and Department of Computer Science, University of Toronto, 1995.

Radford M. Neal. Regression and classification using Gaussian process priors. *Bayesian Statistics*, 6:475–501, 1998.

Radford M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

Radford M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705–767, 2003.

Radford M. Neal. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*, pages 113–162. Chapman & Hall/CRC, 2011.

Fernando Pérez and Brian E. Granger. IPython: a System for Interactive Scientific Computing. *Computing in Science and Engineering*, 9(3):21–29, May 2007. URL `http://ipython.org`.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. CODA: Convergence diagnosis and output analysis for MCMC. *R News*, 6(1):7–11, 2006. URL `http://CRAN.R-project.org/doc/Rnews/`.

Pradeep Ravikumar, Martin J. Wainwright, Garvesh Raskutti, and Bin Yu. High-dimensional covariance estimation by minimizing ℓ_1-penalized log-determinant divergence. *Electronic Journal of Statistics*, 5:935–980, 2011.

Gareth O. Roberts and Jeffrey S. Rosenthal. Markov chain Monte Carlo: Some practical implications of theoretical results. *Canadian Journal of Statistics*, 26:5–31, 1998.

Gareth O. Roberts and Jeffrey S. Rosenthal. General state space Markov chains and MCMC algorithms. *Probability Surveys*, 1:20–71, 2004.

Gareth O. Roberts and Jeffrey S. Rosenthal. Examples of adaptive MCMC. Technical report, Department of Statistics, University of Toronto, 2006.

Jeffrey S. Rosenthal. Parallel computing and Monte Carlo algorithms. *Far East Journal of Theoretical Statistics*, 4:207–236, 2000.

Vince G. Sigillito, Simon P. Wing, Larrie V. Hutton, and K. B. Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10:262–266, 1989.

Stan Development Team. Stan: A C++ library for probability and sampling, version 1.0, 2012. URL `http://mc-stan.org/`.

William Nick Street, William H. Wolberg, and Olvi L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology*, volume 1905, pages 861–870, 1993.

Marc A. Suchard, Quanli Wang, Cliburn Chan, Jacob Frelinger, Andrew Cron, and Mike West. Understanding GPU programming for statistical computation: Studies in massively parallel massive mixtures. *Journal of Computational and Graphical Statistics*, 19:419–438, 2010.

Daniel Tarlow and Ryan P. Adams. Revisiting uncertainty in graph cut solutions. In *The 25th IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

Daniel Tarlow, Ryan P. Adams, and Richard Zemel. Randomized optimum models for structured prediction. In *The 15th International Conference on Artificial Intelligence and Statistics*, 2012.

Cajo J. F. Ter Braak. A Markov chain Monte Carlo version of the genetic algorithm differential evolution: easy Bayesian computing for real parameter spaces. *Statistical Computing*, 16:239–249, 2006.

Luke Tierney. Markov chains for exploring posterior distributions. *Annals of Statistics*, 22 (4):1701–1728, 1994.

Yichuan Zhang and Charles Sutton. Quasi-Newton methods for Markov chain Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 2393–2401, 2011.

Journal of Machine Learning Research 15 (2014) 2113-2144 Submitted 4/13; Revised 12/13; Published 6/14

Classifier Cascades and Trees for Minimizing Feature Evaluation Cost

Zhixiang (Eddie) Xu XUZX@CSE.WUSTL.EDU
Matt J. Kusner MKUSNER@WUSTL.EDU
Kilian Q. Weinberger KILIAN@WUSTL.EDU
Department of Computer Science
Washington University
1 Brookings Drive
St. Louis, MO 63130, USA

Minmin Chen M.CHEN@CRITEO.COM
Olivier Chapelle OLIVIER@CHAPELLE.CC
Criteo
411 High Street
Palo Alto, CA 94301, USA

Editor: Balazs Kegl

Abstract

Machine learning algorithms have successfully entered industry through many real-world applications (*e.g.* , search engines and product recommendations). In these applications, the test-time CPU cost must be budgeted and accounted for. In this paper, we examine two main components of the test-time CPU cost, *classifier evaluation cost* and *feature extraction cost*, and show how to balance these costs with the classifier accuracy. Since the computation required for feature extraction dominates the test-time cost of a classifier in these settings, we develop two algorithms to efficiently balance the performance with the test-time cost. Our first contribution describes how to construct and optimize a tree of classifiers, through which test inputs traverse along individual paths. Each path extracts different features and is optimized for a specific sub-partition of the input space. Our second contribution is a natural reduction of the tree of classifiers into a cascade. The cascade is particularly useful for class-imbalanced data sets as the majority of instances can be early-exited out of the cascade when the algorithm is sufficiently confident in its prediction. Because both approaches only compute features for inputs that benefit from them the most, we find our trained classifiers lead to high accuracies at a small fraction of the computational cost.

Keywords: budgeted learning, resource efficient machine learning, feature cost sensitive learning, web-search ranking, tree of classifiers

1. Introduction

In real-world machine learning applications, such as email-spam (Weinberger et al., 2009), adult content filtering (Fleck et al., 1996), and web-search engines (Zheng et al., 2008; Chapelle et al., 2011), managing the CPU cost at test-time becomes increasingly important. In applications of such large scale, computation must be budgeted and accounted for.

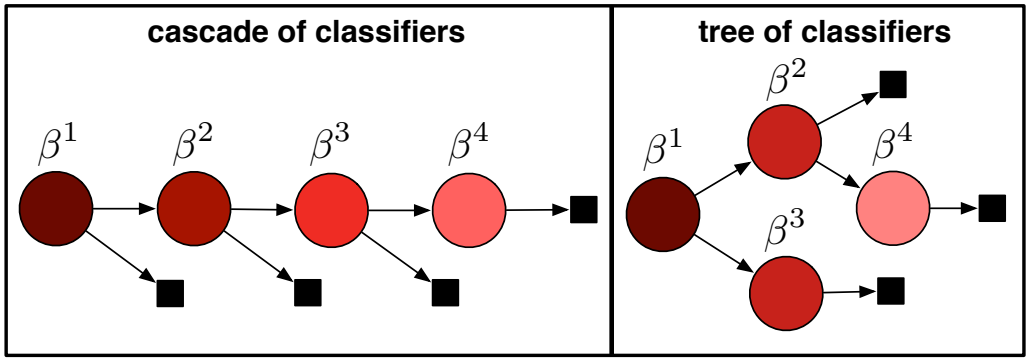


Figure 1: An illustration of two different techniques for learning under a test-time budget. Circular nodes represent classifiers (with parameters $\boldsymbol{\beta}$) and black squares predictions. The color of a classifier node indicates the number of inputs passing through it (darker means more). *Left:* CSCC, a classifier cascade that optimizes the average cost by rejecting easier inputs early. *Right:* CSTC, a tree that trains expert leaf classifiers specialized on subsets of the input space.

Two main components contribute to the *test-time cost*. The time required to evaluate a classifier and the time to extract features used by that classifier. Since the features are often heterogeneous, extraction time for different features is highly variable. Imagine introducing a new feature to a product recommendation system that requires 1 second to extract per recommendation. If a web-service provides 100 million recommendations a day (which is not uncommon), it would require 1200 extra CPU days to extract just this feature. While this additional feature may increase the accuracy of the recommendation system, the cost of computing it for *every recommendation* is prohibitive. This introduces the problem of balancing the test-time cost and the classifier accuracy. Addressing this trade-off in a principled manner is crucial for the applicability of machine learning.

A key observation for minimizing test-time cost is that not all inputs require the same amount of computation to obtain a confident prediction. One celebrated example is face detection in images, where the majority of all image regions do not contain faces and can often be easily rejected based on the response of a few simple Haar features (Viola and Jones, 2004). A variety of algorithms utilize this insight by constructing a cascade of classifiers (Viola and Jones, 2004; Lefakis and Fleuret, 2010; Saberian and Vasconcelos, 2010; Pujara et al., 2011; Chen et al., 2012; Trapeznikov et al., 2013b). Each stage in the cascade can reject an input or pass it on to a subsequent stage. These algorithms significantly reduce the test-time complexity, particularly when the data is *class-imbalanced*, and *few features* are needed to classify instances into a certain class, as in face detection.

Another observation is that it is not only possible that many inputs can be classified correctly using a small subset of all features, but also, such subsets are likely to *vary across inputs*. Particularly for the case in which data is *not* class-imbalanced it may be possible to further lower the test-time cost by extracting fewer, more specialized features per input than the features that would be extracted using a cascade of classifiers. In this paper, we provide a detailed analysis of a new algorithm, *Cost-Sensitive Tree of Classifiers (CSTC)* (Xu et al.,

2013a) that is derived based on this observation. CSTC minimizes an approximation of the exact expected test-time cost required to predict an instance. An illustration of a CSTC tree is shown in the right plot of Figure 1. Because the input space is partitioned by the tree, different features are only extracted where they are most beneficial, and therefore, the average test-time cost is reduced. Unlike prior approaches, which reduce the total cost for every input (Efron et al., 2004) or which combine feature cost with mutual information to select features (Dredze et al., 2007), a CSTC tree incorporates *input-dependent feature selection* into training and dynamically allocates higher feature budgets for infrequently traveled tree-paths.

CSTC incorporates two novelties: 1. it relaxes the expected per-instance test-time cost into a well-behaved optimization; and 2. it is a generalization of cascades to trees. Full trees, however, are not always necessary. In data scenarios with highly skewed class imbalance, cascades might be a better model by rejecting many instances using a small number of features. We therefore apply the same test-time cost derivation to a stage-wise classifier for cascades. The resulting algorithm, *Cost-Sensitive Cascade of Classifiers (CSCC)*, is shown in the left plot of Figure 1. This algorithm supersedes an approach previously proposed, *Cronus* (Chen et al., 2012), which is not derived through a formal relaxation of the test-time cost, but performs a clever weighting scheme. We compare and contrast Cronus with CSCC.

Two earlier short papers already introduce CSTC (Xu et al., 2013a) and Cronus (Chen et al., 2012) algorithms, however the present manuscript provides significantly more detailed analysis, experimental results and insightful discussion—and it introduces CSCC, which combines insights from all prior work. The paper is organized as follows. Section 2 introduces and defines the test-time cost learning setting. Section 3 presents the tree of classifiers approach, CSTC. In Section 4 we lay out CSCC and relate it to prior work, Cronus, (Chen et al., 2012). Section 5 introduces non-linear extensions to CSTC and CSCC. Section 6 presents the experimental results on several data sets and discusses the performance differences. Section 7 reviews the prior and related contributions that inspires our work. We conclude in Section 8 by summarizing our contributions and proposing a few future directions.

2. Test-Time Cost

There are several key aspects towards learning under test-time cost budgets that need to be considered: 1. feature extraction cost is relevant and varies significantly across features; 2. features are extracted *on-demand* rather than prior to evaluation; 3. different features can be extracted for different inputs; 4. the test cost is evaluated in average rather than in absolute (/worst-case) terms (*i.e.* , several cheap classifications can free up budget for an expensive classification). In this section we focus on learning a single cost-sensitive classifier. We will combine these classifiers to form our tree and cascade algorithms in later sections. We first introduce notation and our general setup, and then provide details on how we address these specific aspects.

Let the data consist of inputs $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathcal{R}^d$ with corresponding class labels $\{y_1, \ldots, y_n\} \subseteq \mathcal{Y}$, where $\mathcal{Y} = \mathcal{R}$ in the case of regression ($\mathcal{Y}$ could also be a finite set of categorical labels). We summarize all notation in Table 1.

$\mathbf{x}_i$	Input instance i
y_i	Input label i
$\mathcal{H}$	Set of all weak learner t
H	Linear classifier on input
$\boldsymbol{\beta}$	Parameters of linear classifier H
ℓ	Non-negative loss function over input
ρ	Coefficient for regularization
λ	Accuracy/cost trade-off parameter
c_α	Feature extraction cost of feature α
v^k	Classifier node k
θ^k	Splitting threshold of node v^k
p_i^k	Traversal probability to node v^k of input $\mathbf{x}_i$
π^l	Set of classifier node along the path from root to v^l

Table 1: Notation used throughout this manuscript.

2.1 Cost-Sensitive Loss Minimization

We learn a classifier $H : \mathcal{R}^d \to \mathcal{Y}$, parameterized by $\boldsymbol{\beta}$, to minimize a continuous, non-negative loss function ℓ over $\mathcal{D}$,

$$\frac{1}{n} \min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \ell(H(\mathbf{x}_i; \boldsymbol{\beta}), y_i).$$

We assume that H is a linear classifier, $H(\mathbf{x}; \boldsymbol{\beta}) = \boldsymbol{\beta}^\top \mathbf{x}$. To avoid overfitting, we deploy a standard l_1 regularization term, $|\boldsymbol{\beta}|$ to control model complexity. This regularization term has the known side-effect to keep $\boldsymbol{\beta}$ sparse (Tibshirani, 1996), which requires us to only evaluate a subset of all features. In addition, to balance the test-time cost incurred by the classifier, we also incorporate the cost term $c(\boldsymbol{\beta})$ described in the following section. The combined test-time cost-sensitive optimization becomes

$$\min_{\boldsymbol{\beta}} \underbrace{\sum_i \ell(\mathbf{x}_i^\top \boldsymbol{\beta}, y_i) + \rho|\boldsymbol{\beta}|}_{\text{regularized loss}} \; + \; \lambda \underbrace{c(\boldsymbol{\beta})}_{\text{test-cost}}, \tag{1}$$

where λ is the accuracy/cost trade-off parameter, and ρ controls the strength of the regularization.

2.2 Test-Time Cost

The test-time cost of H is regulated by the features extracted for that classifier. Different from traditional settings, where all features are computed prior to the application of H, we assume that features are computed *on demand* the first time they are used.

We denote the extraction cost for feature α as c_α. The cost $c_\alpha \geq 0$ is suffered at most once, only for the initial extraction, as feature values can be cached for future use. For a classifier H, parameterized by $\boldsymbol{\beta}$, we can record the features used:

$$\|\beta_\alpha\|_0 = \begin{cases} 1 & \text{if feature } \alpha \text{ is used in } H \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Here, $\|\cdot\|_0$ denotes the l_0 norm with $\|a\|_0 = 1$ if $a \neq 0$ and $\|a\|_0 = 0$ otherwise. With this notation, we can formulate the total test-time cost required to evaluate a test input $\mathbf{x}$ with classifier H (and parameters $\boldsymbol{\beta}$) as

$$c(\boldsymbol{\beta}) = \sum_{\alpha=1}^{d} c_\alpha \|\beta_\alpha\|_0. \tag{3}$$

The equation (3) can be in any units of cost. For example in medical applications, the feature extraction cost may be in units of "patient agony" or in "examination cost". The current formulation (1) with cost term (3) still extracts the same features for all inputs and is NP-hard to optimize (Korte and Vygen, 2012, Chapter 15). We will address these issues in the following sections.

3. Cost-Sensitive Tree of Classifiers

We introduce an algorithm that is inspired by the observation that many inputs could be classified correctly based on only a small subset of all features, and this subset may vary across inputs. Our algorithm employs a tree structure to extract particular features for particular inputs, and we refer to it as the *Cost-Sensitive Tree of Classifiers* (*CSTC*). We begin by introducing foundational concepts regarding the CSTC tree and derive a global cost term that extends (3) to trees of classifiers and then we relax the resulting loss function into a well-behaved optimization problem.

3.1 CSTC Nodes

The fundamental building block of the CSTC tree is a CSTC node—a linear classifier as described in Section 2.1. Our classifier design is based on the assumption that instances with similar labels tend to have similar features. Thus, we design our tree algorithm to partition the input space based on classifier predictions. Intermediate classifiers determine the path of instances through the tree and leaf classifiers become experts for a small subset of the input space.

Correspondingly, there are two different nodes in a CSTC tree (depicted in Figure 2): *classifier nodes* (white circles) and *terminal elements* (black squares). Each *classifier node* v^k is associated with a weight vector $\boldsymbol{\beta}^k$ and a threshold θ^k. These classifier nodes branch inputs by their threshold θ^k, sending inputs to their upper child if $\mathbf{x}_i^\top \boldsymbol{\beta}^k > \theta^k$, and to their lower child otherwise. *Terminal elements* are "dummy" structures and are *not* real classifiers. They return the predictions of their direct parent classifier nodes—essentially functioning as a placeholder for an exit out of the tree. The tree structure may be a full balanced binary tree of some depth (e.g., Figure 2), or can be pruned based on a validation set. For simplicity, we assume at this point that nodes with terminal element children must be leaf nodes (as depicted in the figure)—an assumption that we will relax later on.

During test-time, inputs traverse through the tree, starting from the root node v^0. The root node produces predictions $\mathbf{x}_i^\top \boldsymbol{\beta}^0$ and sends the input $\mathbf{x}_i$ along one of two different paths, depending on whether $\mathbf{x}_i^\top \boldsymbol{\beta}^0 > \theta^0$. By repeatedly branching the test inputs, classifier nodes sitting deeper in the tree only handle a small subset of all inputs and become specialized towards that subset of the input space.

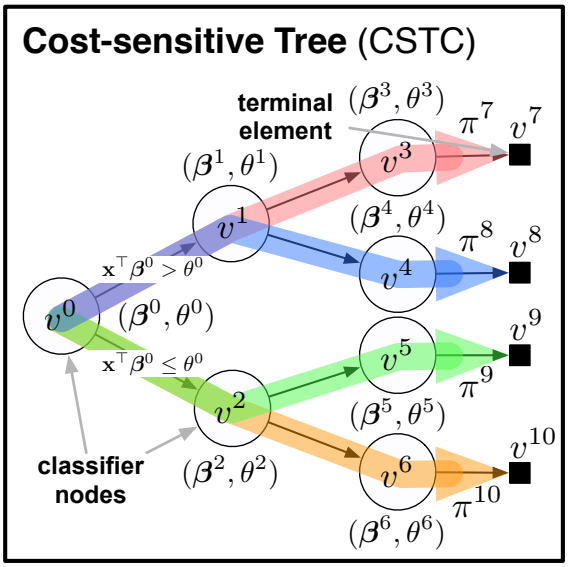


Figure 2: A schematic layout of a CSTC tree. Each node v^k is associated with a weight vector $\boldsymbol{\beta}^k$ for prediction and a threshold θ^k to send instances to different parts of the tree. We solve for $\boldsymbol{\beta}^k$ and θ^k that best balance the accuracy/cost trade-off for the whole tree. All paths of a CSTC tree are shown in color.

3.2 CSTC Loss

In this section, we discuss the loss and test-time cost of a CSTC tree. We then derive a single global loss function over all nodes in the CSTC tree.

3.2.1 Soft Tree Traversal

As we described before, inputs are partitioned at each node during test-time, and we use a hard threshold to achieve this partitioning. However, modeling a CSTC tree with hard thresholds leads to a combinatorial optimization problem that is NP-hard (Korte and Vygen, 2012, Chapter 15). As a remedy, during training, we *softly* partition the inputs and assign *traversal probabilities* $p(v^k|\mathbf{x}_i)$ to denote the likelihood of input $\mathbf{x}_i$ traversing through node v^k. Every input $\mathbf{x}_i$ traverses through the root, so we define $p(v^0|\mathbf{x}_i) = 1$ for all i. We define a "sigmoidal" soft belief that an input $\mathbf{x}_i$ will transition from classifier node v^k with threshold θ^k to its *upper* child v^u as

$$p(v^u|\mathbf{x}_i, v^k) = \frac{1}{1 + \exp(-(\mathbf{x}_i^\top \boldsymbol{\beta}^k - \theta^k))}. \tag{4}$$

Let v^k be a node with upper child v^u and lower child v^l. We can express the probabilities of reaching nodes v^u and v^l recursively as $p(v^u|\mathbf{x}_i) = p(v^u|\mathbf{x}_i, v^k)p(v^k|\mathbf{x}_i)$ and $p(v^l|\mathbf{x}_i) = \big[1 - p(v^u|\mathbf{x}_i, v^k)\big]p(v^k|\mathbf{x}_i)$ respectively. Note that it follows immediately, that if $\mathcal{V}^d$ contains

all nodes at tree-depth d, we have

$$\sum_{v \in \mathcal{V}^d} p(v|\mathbf{x}) = 1. \tag{5}$$

In the following paragraphs we incorporate this probabilistic framework into the loss and cost terms of (1) to obtain the corresponding *expected* tree loss and tree cost.

3.2.2 Expected Tree Loss

To obtain the *expected tree loss*, we sum over all nodes $\mathcal{V}$ in a CSTC tree and all inputs and weight the loss $\ell(\cdot)$ of input $\mathbf{x}_i$ at each node v^k by the probability that the input reaches v^k, $p_i^k = p(v^k|\mathbf{x}_i)$,

$$\frac{1}{n} \sum_{i=1}^{n} \sum_{v^k \in \mathcal{V}} p_i^k \ell(\mathbf{x}_i^\top \boldsymbol{\beta}^k, y_i). \tag{6}$$

This has two effects: 1. the local loss for each node focuses more on likely inputs; 2. the global objective attributes more weight to classifiers that serve many inputs. Technically, the prediction of the CSTC tree is made entirely by the terminal nodes (*i.e.* , the leaves), and an obvious suggestion may be to only minimize their classification losses and leave the interior nodes as "gates" without any predictive abilities. However, such a setup creates local minima that send all inputs to the terminal node with the lowest initial error rate. These local minima are hard to escape from and therefore we found it to be important to minimize the loss for *all* nodes. Effectively, this forces a structure onto the tree that similarly labeled inputs leave through similar leaves and achieves robustness by assigning high loss to such pathological solutions.

3.2.3 Expected Tree Costs

The cost of a test input is the cumulative cost across all classifiers along its path through the CSTC tree. Figure 2 illustrates an example of a CSTC tree with all paths highlighted in color. Every test input must follow along exactly one of the paths from the root to a terminal element. Let $\mathcal{L}$ denote the set of all terminal elements (*e.g.*, in Figure 2 we have $\mathcal{L} = \{v^7, v^8, v^9, v^{10}\}$), and for any $v^l \in \mathcal{L}$ let π^l denote the set of all *classifier nodes* along the unique path from the root v^0 before terminal element v^l (*e.g.*, $\pi^9 = \{v^0, v^2, v^5\}$).

For an input $\mathbf{x}$, exiting through terminal node v^l, a feature α needs to be extracted if and only if at least one classifier along the path π^l uses this feature. We extend the indicator function defined in (2) accordingly:

$$\left\| \sum_{v^j \in \pi^l} |\beta_\alpha^j| \right\|_0 = \begin{cases} 1 & \text{if feature } \alpha \text{ is used along path to terminal node } v^l \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

We can extend the cost term in (3) to capture the traversal cost from root to node v^l as

$$c^l = \sum_\alpha c_\alpha \left\| \sum_{v^j \in \pi^l} |\beta_\alpha^j| \right\|_0. \tag{8}$$

Given an input $\mathbf{x}_i$, the expected cost is then $E[c^l|\mathbf{x}_i] = \sum_{l\in\mathcal{L}} p(v^l|\mathbf{x}_i)c^l$. To approximate the data distribution, we sample uniformly at random from our training set, *i.e.* , we set $p(\mathbf{x}_i) \approx \frac{1}{n}$, and obtain the unconditional expected cost

$$E[cost] = \sum_{i=1}^{n} p(\mathbf{x}_i) \sum_{l\in\mathcal{L}} p(v^l|\mathbf{x}_i)c^l \approx \sum_{l\in\mathcal{L}} c^l \underbrace{\sum_{i=1}^{n} p(v^l|\mathbf{x}_i)\frac{1}{n}}_{:=p^l} = \sum_{l\in\mathcal{L}} c^l p^l. \tag{9}$$

Here, p^l denotes the probability that a randomly picked training input exits the CSTC tree through terminal node v^l. We can combine (8), (9) with (6) and obtain the objective function,

$$\sum_{v^k\in\mathcal{V}} \underbrace{\left(\frac{1}{n}\sum_{i=1}^{n} p_i^k \ell_i^k + \rho|\boldsymbol{\beta}^k|\right)}_{\text{regularized loss}} + \lambda \sum_{v^l\in\mathcal{L}} p^l \underbrace{\left[\sum_{\alpha} c_\alpha \left\| \sum_{v^j\in\pi^l} |\beta_\alpha^j| \right\|_0\right]}_{\text{test-time cost}}, \tag{10}$$

where we use the abbreviations $p_i^k = p(v^k|\mathbf{x}_i)$ and $\ell_i^k = \ell(\mathbf{x}_i^\top \boldsymbol{\beta}^k, y_i)$.

3.3 Test-Cost Relaxation

The cost penalties in (10) are exact but difficult to optimize due to the discontinuity and non-differentiability of the l_0 norm. As a solution, throughout this paper we use the mixed-norm relaxation of the l_0 norm over sums,

$$\sum_j \left\| \sum_i |A_{ij}| \right\|_0 \rightarrow \sum_j \sqrt{\sum_i (A_{ij})^2}, \tag{11}$$

described by Kowalski (2009). Note that for a vector, this relaxes the l_0 norm to the l_1 norm, *i.e.* , $\sum_j \|a_j\|_0 \rightarrow \sum_j \sqrt{(a_j)^2} = \sum_j |a_j|$, recovering the commonly used approximation to encourage sparsity. For matrices $\mathbf{A}$, the mixed norm applies the l_1 norm over rows and the l_2 norm over columns, thus encouraging a whole row to be all-zero or non-sparse. In our case this has the natural interpretation to encourage re-use of features that are already extracted along a path. Using the relaxation in (11) on the l_0 norm in (10) gives the final optimization problem:

$$\min_{\boldsymbol{\beta}^0,\theta^0,\ldots,\boldsymbol{\beta}^{|V|},\theta^{|V|}} \sum_{v^k\in\mathcal{V}} \underbrace{\left(\frac{1}{n}\sum_{i=1}^{n} p_i^k \ell_i^k + \rho|\boldsymbol{\beta}^k|\right)}_{\text{regularized loss}} + \lambda \sum_{v^l\in\mathcal{L}} p^l \underbrace{\left[\sum_{\alpha} c_\alpha \sqrt{\sum_{v^j\in\pi^l} (\beta_\alpha^j)^2}\right]}_{\text{test-time cost penalty}} \tag{12}$$

We can illustrate the fact that the mixed-norm encourages re-use of features with a simple example. If two classifiers $v^k \neq v^{k'}$ along a path π^l use *different* features with identical weight, *i.e.* , $\beta_t^k = \epsilon = \beta_s^{k'}$ and $t \neq s$, the test-time cost penalty along π^l is $\sqrt{\epsilon^2} + \sqrt{\epsilon^2} = 2\epsilon$. However, if the two classifiers *re-use* the same feature, *i.e.* , $t = s$, the test-time cost penalty reduces to $\sqrt{\epsilon^2 + \epsilon^2} = \sqrt{2}\epsilon$.

3.4 Optimization

There are many techniques to minimize the objective in (12). We use block coordinate descent, optimizing with respect to the parameters of a single classifier node v^k at a time, keeping all other parameters fixed. We perform a level order tree traversal, optimizing each node in order: $v^1, v^2, \ldots, v^{|V|}$. To minimize (12) (up to a local minimum) with respect to parameters $\boldsymbol{\beta}^k, \theta^k$ we use the lemma below to overcome the non-differentiability of the square-root term (and l_1 norm, which we can rewrite as $|a| = \sqrt{a^2}$) resulting from the l_0-relaxation (11).

Lemma 1. *Given a positive function* $g(x)$, the following holds:

$$\sqrt{g(x)} = \inf_{z>0} \frac{1}{2}\left[\frac{g(x)}{z} + z\right]. \tag{13}$$

It is straight-forward to see that $z = \sqrt{g(x)}$ minimizes the function on the right hand side and satisfies the equality, which leads to the proof of the lemma.

For each square-root or l_1 term we 1) introduce an auxiliary variable (*i.e.*, z above), 2) substitute in (13), and 3) alternate between minimizing the objective in (12) with respect to $\boldsymbol{\beta}^k, \theta^k$ and solving for the auxiliary variables. The former minimization is performed with conjugate gradient descent and the latter can be computed efficiently in closed form. This pattern of block-coordinate descent followed by a closed form minimization is repeated until convergence. Note that the objective is guaranteed to converge to a fixed point because each iteration decreases the objective function, which is bounded below by zero. In the following subsection, we detail the block coordinate descent optimization technique. Lemma 1 is only defined for strictly positive functions $g(x)$. As we are performing function minimization, we can reach cases where $g(x) = 0$ and Lemma 1 is ill defined. Thus, as a practical work-around, we clamp values to zero once they are below a small threshold (10^{-4}).

3.4.1 Optimization Details

For reproducibility, we describe the optimization in more detail. Readers not interested in the exact procedure may skip to Section 3.5. As terminal nodes are only placeholders and do not have their own parameters, we only focus on classifier nodes, which are depicted as round circles in Figure 2.

Leaf Nodes. The optimization of leaf nodes (*e.g.* , v^3, v^4, v^5, v^6 in Fig. 2) is simpler because there are no downstream dependencies. Let v^k be such a classifier node with only a single "dummy" terminal node $v^{k'}$. During optimization of (12), we fix all other parameters $\boldsymbol{\beta}^j, \theta^j$ of other nodes v^j and the respective terms become constants. Therefore, we remove all other paths, and only minimize over the path $\pi^{k'}$ from the root to terminal node $v^{k'}$. Even along the path $\pi^{k'}$ most terms become constant and the only non-constant parameter is $\boldsymbol{\beta}^k$ (the branching parameter θ^k can be set to $-\infty$ because v^k has only one child). We color non-constant terms in the remaining function in blue below,

$$\sum_i p_i^k \ell(\phi(\mathbf{x}_i)^\top \boldsymbol{\beta}^k, y_i) + \rho|\boldsymbol{\beta}^k| + \lambda\, p^{k'}\left[\sum_\alpha c_\alpha \sqrt{(\beta_\alpha^k)^2 + \sum_{v^j \in \pi^{k'} \setminus v^k} (\beta_\alpha^j)^2}\right], \tag{14}$$

where $\mathcal{S}\backslash b$ contain all of the elements in $\mathcal{S}$ except b. After identifying the non-constant terms, we can apply Lemma 1, making (14) differentiable with respect to β_α^k. Let us define auxiliary variables γ_α and η_α for $1 \leq \alpha \leq d$ for the l_1-regularization term and the test-time cost term. Further, let us collect the constants in the test-time cost term $c_{\text{test-time}} = \sum_{v^j \in \pi^{k'} \setminus v^k} (\beta_\alpha^j)^2$. Applying Lemma 1 results in the following substitutions:

$$\sum_\alpha \rho |\beta_\alpha^k| = \sum_\alpha \rho \sqrt{(\beta_\alpha^k)^2} \longrightarrow \sum_\alpha \rho \frac{1}{2} \left(\frac{(\beta_\alpha^k)^2}{\gamma_\alpha} + \gamma_\alpha \right),$$
$$\sum_\alpha c_\alpha \sqrt{(\beta_\alpha^k)^2 + c_{\text{test-time}}} \longrightarrow \sum_\alpha c_\alpha \frac{1}{2} \left(\frac{(\beta_\alpha^k)^2 + c_{\text{test-time}}}{\eta_\alpha} + \eta_\alpha \right). \tag{15}$$

As a result, we obtain a differentiable objective function after making the above substitutions. We can solve $\boldsymbol{\beta}^k$ by alternately minimizing the obtained differentiable function w.r.t. $\boldsymbol{\beta}^k$ with $\gamma_\alpha, \eta_\alpha$ fixed, and minimizing $\gamma_\alpha, \eta_\alpha$ with $\boldsymbol{\beta}^k$ fixed (*i.e.*, minimizing η_α is equivalent to setting $\eta_\alpha = \sqrt{(\beta_\alpha^k)^2 + c_{\text{test-time}}}$). Recall that θ^k does not require optimization as v^k does not further branch inputs.

It is straight-forward to show (Boyd and Vandenberghe, 2004, page 72), that the right hand side of Lemma 1 is jointly convex in x and z, so as long as $g(x)$ is a quadratic function of x. Thus, if $\ell(\mathbf{x}_i^\top \boldsymbol{\beta}^k, y_i)$ is the squared loss, the substituted objective function is jointly convex in $\boldsymbol{\beta}^k$ and in $\gamma_\alpha, \eta_\alpha$ and therefore we can obtain a globally-optimal solution. Moreover, we can solve $\boldsymbol{\beta}^k$ in closed form. Let us define three design matrices

$$\mathbf{X}_{i\alpha} = [\mathbf{x}_i]_\alpha, \quad \boldsymbol{\Omega}_{ii} = p_i^k, \quad \boldsymbol{\Gamma}_{\alpha\alpha} = \frac{\rho}{\gamma_\alpha} + \lambda \left(\frac{p^k c_\alpha}{\eta_\alpha} \right),$$

where $\boldsymbol{\Omega}$ and $\boldsymbol{\Gamma}$ are both diagonal and $[\mathbf{x}_i]_\alpha$ is the α feature of instance $\mathbf{x}_i$. The closed-form solution for $\boldsymbol{\beta}^k$ is as follows,

$$\boldsymbol{\beta}^k = (\mathbf{X}^\top \boldsymbol{\Omega} \mathbf{X} + \boldsymbol{\Gamma})^{-1} \mathbf{X}^\top \boldsymbol{\Omega} \mathbf{y}. \tag{16}$$

Intermediate Nodes. We further generalize this approach to all classifier nodes. As before, we optimize one node at a time, fixing the parameters of all other nodes. However, optimizing the parameters $\boldsymbol{\beta}^k, \theta^k$ of an *internal* node v^k, which has two children affects the parameters of descendant nodes. This affects the optimization of the regularized classifier loss and the test-time cost separately. We state how these terms in the global objective (12) are affected, and then show how to minimize it.

Let $\mathcal{S}$ be the set containing all descendant nodes of v^k. Changes to the parameters $\boldsymbol{\beta}^k, \theta^k$ will affect the traversal probabilities p_i^j for all $v^j \in \mathcal{S}$ and therefore enter the downstream loss functions. We first state the regularized loss part of (12) and once again color non-constant parameters in blue,

$$\frac{1}{n} \sum_i p_i^k \ell(\mathbf{x}_i^\top \boldsymbol{\beta}^k, y_i) + \frac{1}{n} \sum_{v^j \in \mathcal{S}} \sum_i p_i^j \ell(\mathbf{x}_i^\top \boldsymbol{\beta}^j, y_i) + \rho |\boldsymbol{\beta}^k|. \tag{17}$$

For the cost terms in (12), recall that the cost of each path π^l is weighted by the probability p^l of traversing that path. Changes to $\boldsymbol{\beta}^k, \theta^k$ affect the probability of any path

Algorithm 1 CSTC global optimization

Input: data $\{\mathbf{x}_i, y_i\} \in \mathcal{R}^d \times \mathcal{R}$, initialized CSTC tree
repeat
  for $k = 1$ **to** $N = \#$ CSTC nodes **do**
    repeat
      Solve for γ, η (fix $\boldsymbol{\beta}^k, \theta^k$) using left hand side of (15)
      Solve for $\boldsymbol{\beta}^k, \theta^k$ (fix γ, η) with conjugate gradient descent, or in closed-form
    until objective changes less than ε
  end for
until objective changes less than ϵ

that passes through v^k and its corresponding probability p^l. Let $\mathcal{P}$ be the terminal elements associated with paths passing through v^k. We state the cost function with non-constant parameters in blue,

$$\sum_{v^l \in \mathcal{P}} p^l \underbrace{\left[\sum_{\alpha} c_\alpha \sqrt{\left(\sum_{v^j \in \pi^l \setminus v^k} (\beta_\alpha^j)^2 + (\beta_\alpha^k)^2 \right)} \right]}_{\text{test-time cost}} \tag{18}$$

Adding (17) and (18), with the latter weighted by λ, gives the internal node loss. To make the combined objective function differentiable we apply Lemma 1 to the l_1-regularization, and test-time cost terms and introduce auxiliary variables $\gamma_\alpha, \eta_\alpha$ as in (15). Similar to the leaf node case, we solve $\boldsymbol{\beta}^k, \theta^k$ by alternately minimizing the new objective w.r.t. $\boldsymbol{\beta}^k, \theta^k$ with $\gamma_\alpha, \eta_\alpha$ fixed, and minimizing $\gamma_\alpha, \eta_\alpha$ with fixed $\boldsymbol{\beta}^k, \theta^k$. Unlike leaf nodes, optimizing the objective function w.r.t. $\boldsymbol{\beta}^k, \theta^k$ cannot be expressed in closed form even with squared loss. Therefore, we optimize it with conjugate gradient descent. Algorithm 1 describes how the entire CSTC tree is optimized.

3.4.2 Node Initialization

The minimization of (12) is non-convex and is therefore initialization dependent. However, minimizing (12) with respect to the parameters of leaf classifiers *is convex*. We therefore initialize the tree top-to-bottom, starting at v^0, and optimizing over $\boldsymbol{\beta}^k$ by minimizing (12) while considering all descendant nodes of v^k as "cut-off" (thus pretending node v^k is a leaf). This initialization is also very fast in the case of a quadratic loss, as it can be solved for in closed form.

3.5 Fine-Tuning

The original test-time cost term in (3) sums over the cost of all features that are extracted during test-time. The relaxation in (11) makes the exact l_0 cost differentiable and is still well suited to *select which* features to extract. However, the mixed-norm does also impact the performance of the classifiers, because (different from the l_0 norm) larger weights in $\boldsymbol{\beta}$ incur larger cost penalties. We therefore introduce a post-processing step to correct the classifiers from this unwanted regularization effect. We re-optimize the loss of all *leaf* classifiers (*i.e.*

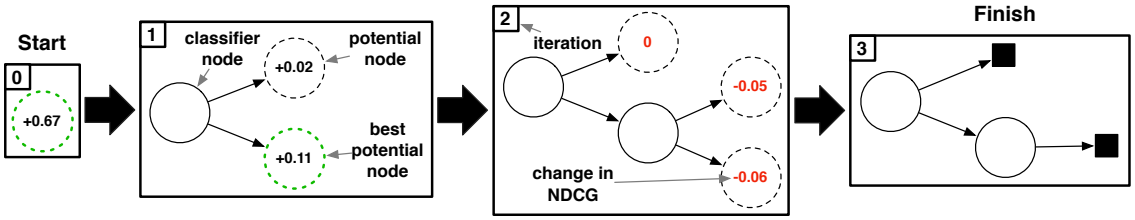


Figure 3: A schematic layout of the greedy tree building algorithm. Each iteration we add the best performing potential node (dashed, above) to the tree. Each potential node is annotated by the improvement in validation-NDCG, obtained with its inclusion (number inside the circle). In this example, after two iterations no more nodes improve the NDCG and the algorithm terminates, converting all remaining potential nodes into terminal elements (black boxes).

, classifiers that make final predictions), while clamping all features with zero-weight to strictly remain zero.

$$\min_{\bar{\boldsymbol{\beta}}^k} \sum_i p_i^k \ell(\mathbf{x}_i^\top \bar{\boldsymbol{\beta}}^k, y_i) + \rho|\bar{\boldsymbol{\beta}}^k|$$
$$\text{subject to: } \bar{\beta}_t^k = 0 \text{ if } \beta_t^k = 0.$$

Here, we do not include the cost-term, because the decision regarding which features to use is already made. The final CSTC tree uses these re-optimized weight vectors $\bar{\boldsymbol{\beta}}^k$ for all leaf classifier nodes v^k.

3.6 Determining the Tree Structure

As the CSTC tree does not need to be balanced, its structure is an implicit parameter of the algorithm. We learn and fix the tree structure prior to the optimization and fine-tuning steps in Sections 3.4 and 3.5. We discuss two approaches to determine the structure of the tree in the absence of prior knowledge, the first prunes a balanced tree bottom-up, the second adds nodes top-down, only when necessary. In practice, both techniques produce similar results and we settled on using the pruning technique for all of our experiments.

3.6.1 Tree Pruning

We build a full (balanced) CSTC tree of depth d and initialize all nodes. To obtain a more compact model and to avoid over-fitting, the CSTC tree can be pruned with the help of a validation set. We compute the validation error of the initialized CSTC tree at each node. Starting with the leaf nodes, we then prune away nodes that, upon removal, do not decrease the validation performance (in the case of ranking data, we even can use validation NDCG (Järvelin and Kekäläinen, 2002) as our pruning criterion). After pruning, the tree structure is fixed and all nodes are optimized with the procedure described in Section 3.4.1.

3.6.2 Greedy Tree Building

In contrast to the bottom-up pruning, we can also use a top-down approach to construct the tree structure. Figure 3 illustrates our greedy heuristic for CSTC tree construction. In each iteration, we add the child node that improves the validation criteria (*e.g.*, NDCG) the most on the validation set.

More formally, we distinguish between CSTC *classifier nodes* and *potential nodes*. Potential nodes (dotted circles in Figure 3) can turn into classifier nodes or terminal elements. Each potential node is initially a trained classifier and annotated with the NDCG value that the CSTC tree would reach on validation with its inclusion. At iteration 0 we learn a single CSTC node by minimizing (12) for the root node v^0, and make it a *potential node*. At iteration $i > 0$ we pick the potential node whose inclusion improves the validation NDCG the most (depicted as the dotted green circle) and add it to the tree. Then we create two new potential nodes as its children, and initialize their classifiers by minimizing (12) with all other weight-vectors and thresholds fixed. The splitting threshold θ^k is set to move 50% of the validation inputs to the upper child (the thresholds will be re-optimized subsequently). This procedure continues until no more potential nodes improve the validation NDCG, and we convert all remaining potential nodes into terminal elements.

4. Cost-Sensitive Cascade of Classifiers

Many real world applications have data distributions with high class imbalance. One example is face detection, where the vast majority of all image patches does not contain faces; another example is web-search ranking, where almost all web-pages are irrelevant to a given query. Often, a few features may suffice to detect that an image does not contain a face or that a web-page is irrelevant. Further, in applications such as web-search ranking, the accuracy of bottom ranked instances is irrelevant as long as they are not retrieved at the top (and therefore are not displayed to the end user).

In these settings, the entire focus of the algorithm should be on the most confident positive samples. Sub-trees that lead to only negative predictions, can be pruned effectively as there is no value in providing fine-grained differentiation between negative samples. This further reduces the average feature cost, as negative inputs traverse through shorter paths and require fewer features to be extracted. Previous work obtains these unbalanced trees by explicitly learning cascade structured classifiers (Viola and Jones, 2004; Dundar and Bi, 2007; Lefakis and Fleuret, 2010; Saberian and Vasconcelos, 2010; Chen et al., 2012; Trapeznikov et al., 2013b; Trapeznikov and Saligrama, 2013a). CSTC can incorporate cascades naturally as a special case, in which the tree of classifiers has only a single node per level of depth. However, further modifications can be made to accommodate the specifics of these settings. We introduce two changes to the learning algorithm:

- Inputs of different classes are re-weighted to account for the severe class imbalance.
- Every classifier node v^k has a terminal element as child and is weighted by the probability of *exiting* rather than the probability of traversing through node v^k.

We refer to the modified algorithm as *Cost-Sensitive Cascade of Classifiers (CSCC)*. An example cascade is illustrated in Figure 4. A CSCC with K-stages is defined by a set of

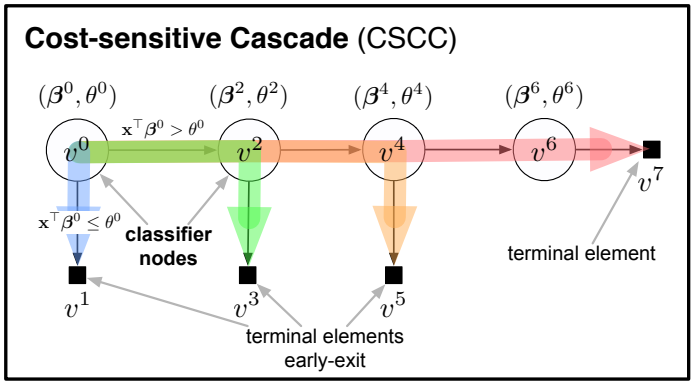


Figure 4: Schematic layout of our classifier cascade with four classifier nodes. All paths are colored in different colors.

weight vectors $\boldsymbol{\beta}^k$ and thresholds θ^k, $\mathcal{C} = \{(\boldsymbol{\beta}^1, \theta^1), (\boldsymbol{\beta}^2, \theta^2), \cdots, (\boldsymbol{\beta}^K, -)\}$. An input is early-exited from the cascade at node v^k if $\mathbf{x}^\top \boldsymbol{\beta}^k < \theta^k$ and is sent to its terminal element v^{k+1}. Otherwise, the input is sent to the next classifier node. At the final node v^K a prediction is made for all remaining inputs via $\mathbf{x}^\top \boldsymbol{\beta}^K$.

In CSTC, most classifier nodes are internal and branch inputs. As such, the predictions need to be similarly accurate for all inputs to ensure that they are passed on to the correct part of the tree. In CSCC, each classifier node early-exits a fraction of its inputs, providing their *final* prediction. As mistakes of such exiting inputs are irreversible, the classifier needs to ensure particularly low error rates for this fraction of inputs. All other inputs are passed down the chain to later nodes. This key insight inspires us to modify the loss function of CSCC from the original CSTC formulation in (6). Instead of weighting the contribution of classifier loss $\ell(\mathbf{x}_i^\top \boldsymbol{\beta}^k, y_i)$ by p_i^k, the probability of input $\mathbf{x}_i$ *traversing* through node v^k, we weight it with p_i^{k+1}, the probability of *exiting* through terminal node v^{k+1}. As a second modification, we introduce an optional class-weight $w_{y_i} > 0$ which absorbs some of the impact of the class imbalance. The resulting loss becomes:

$$\frac{1}{n} \sum_{i=1}^{n} \sum_{v^k \in \mathcal{V}} w_{y_i} p_i^{k+1} \ell(\mathbf{x}_i^\top \boldsymbol{\beta}^k, y_i).$$

The cost term is unchanged and the combined cost-sensitive loss function of CSCC becomes

$$\underbrace{\sum_{v^k \in \mathcal{V}} \frac{1}{n} \left(\sum_{i=1}^{n} w_{y_i} p_i^{k+1} \ell_i^k \right) + \rho |\boldsymbol{\beta}^k|}_{\text{regularized loss}} + \underbrace{\lambda \sum_{v^l \in L} p^l \left[\sum_{\alpha=1}^{d} c_\alpha \sqrt{\sum_{v^j \in \pi^l} (\beta_\alpha^j)^2} \right]}_{\text{feature cost penalty}}. \tag{19}$$

We optimize (19) using the same block coordinate descent optimization described in Section 3.4. Similar as before, we initialize the cascade from left to right, while assuming the currently initialized node is the last node.

4.1 Cronus

CSCC supersedes previous work on cost sensitive learning of cascades by the same authors, Chen et al. (2012). The previous algorithm, named Cronus, shares the same loss terms as CSCC, however the feature and evaluation cost of each node is weighted by the expected number of inputs, p^k, *within* the mixed-norm (highlighted in color):

$$\underbrace{\sum_{v^k \in \mathcal{V}} \frac{1}{n} \left(\sum_{i=1}^{n} w_{y_i} p_i^{k+1} \ell_i^k \right) + \rho |\boldsymbol{\beta}^k|}_{\text{regularized loss}} + \underbrace{\lambda \sum_{\alpha=1}^{d} c_\alpha \sqrt{\sum_{v^k \in \mathcal{V}} (p^k \beta_\alpha^k)^2}}_{\text{feature cost penalty}} .$$

In contrast, CSCC in (19) sums over the weighted cost of all exit paths. The two formulations are similar, but CSCC may be considered more principled as it is derived from the exact expected cost of the cascade. As we show in Section 6, this does translate into better empirical accuracy/cost trade-offs.

5. Extension to Non-Linear Classifiers

Although CSTC's decision boundary may be non-linear, each individual node classifier is linear. For many problems this may be too restrictive and insufficient to divide the input space effectively. In order to allow non-linear decision boundaries we map the input into a more expressive feature space with the "boosting trick" (Friedman, 2001; Chapelle et al., 2011), prior to our optimization. In particular, we first train gradient boosted regression trees with a squared loss penalty for T iterations and obtain a classifier $H'(\mathbf{x}_i) = \sum_{t=1}^{T} h_t(\mathbf{x}_i)$, where each function $h_t(\cdot)$ is a limited-depth CART tree (Breiman, 1984). We then define the mapping $\phi(\mathbf{x}_i) = [h_1(\mathbf{x}_i), \ldots, h_T(\mathbf{x}_i)]^\top$ and apply it to all inputs. The boosting trick is particularly well suited for our feature cost sensitive setting, as each CART tree only uses a small number of features. Nevertheless, this pre-processing step does affect the loss function in two ways: 1. the feature extraction now happens within the CART trees; and 2. the evaluation time of the CART trees needs to be taken into account.

5.1 Feature Cost After the Boosting Trick

After the transformation $\mathbf{x}_i \to \phi(\mathbf{x}_i)$, each input is $T-$dimensional and consequently, we have the weight vectors $\boldsymbol{\beta} \in \mathcal{R}^T$. To incorporate the feature extraction cost into our loss, we define an auxiliary matrix $\mathbf{F} \in \{0,1\}^{d \times T}$ with $F_{\alpha t} = 1$ if and only if the CART tree h_t uses feature f_α. With this notation, we can incorporate the CART-trees into the original feature extraction cost term for a weight vector $\boldsymbol{\beta}$, as stated in (3). The new formulation and its relaxed version, following the mixed-norm relaxation as stated in (11), are then:

$$\sum_{\alpha=1}^{d} c_\alpha \left\| \sum_{t=1}^{T} |F_{\alpha t} \beta_t| \right\|_0 \longrightarrow \sum_{\alpha=1}^{d} c_\alpha \sqrt{\sum_{t=1}^{T} (F_{\alpha t} \beta_t)^2}.$$

The non-negative sum inside the l_0 norm is non-zero if and only if feature α is used by at least one tree with non-zero weight, *i.e.* , $|\beta_t| > 0$. Similar to a single classifier, we can also

adapt the feature extraction cost of the path through a CSTC tree, originally defined in (8), which becomes:

$$\sum_{\alpha=1}^{d} c_\alpha \left\| \sum_{v^j \in \pi^l} \sum_{t=1}^{T} |F_{\alpha t} \beta_t^j| \right\|_0 \longrightarrow \sum_{\alpha=1}^{d} c_\alpha \sqrt{\sum_{v^j \in \pi^l} \sum_{t=1}^{T} (F_{\alpha t} \beta_t^j)^2}. \tag{20}$$

5.2 CART Evaluation Cost

The evaluation of a CART tree may be non-trivial or comparable to the cost of feature extraction and its cost must be accounted for. We define a constant $e_t \geq 0$, which captures the cost of the evaluation of the t^{th} CART tree. We can express this evaluation cost for a single classifier with weight vector $\boldsymbol{\beta}$ in terms of the l_0 norm and again apply the mixed norm relaxation (11). The exact (left term) and relaxed evaluation cost penalty (right term) can be stated as follows:

$$\sum_{t=1}^{T} e_t \|\beta_t\|_0 \longrightarrow \sum_{t=1}^{T} e_t |\beta_t|$$

The left term incurs a cost of e_t for each tree h_t if and only if it is assigned a non-zero weight by the classifier, *i.e.* , $\beta_t \neq 0$. Similar to feature values, we assume that CART tree evaluations can be cached and only incur a cost once (the first time they are computed). With this assumption, we can express the exact and relaxed CART evaluation cost along a path π^l in a CSTC tree as

$$\sum_{t=1}^{T} e_t \left\| \sum_{v^j \in \pi^l} |\beta_t^j| \right\|_0 \longrightarrow \sum_{t=1}^{T} e_t \sqrt{\sum_{v^j \in \pi^l} (\beta_t^j)^2}. \tag{21}$$

It is worth pointing out, that (21) is analogous to the feature extraction cost with linear classifiers (8) and its relaxation, as stated in (12).

5.3 CSTC and CSCC with Non-Linear Classifiers

We can integrate the two CART tree aware cost terms (20) and (21) into the optimization problem in (12). The final objective of the CSTC tree after the "boosting trick" becomes then

$$\sum_{v^k \in \mathcal{V}} \underbrace{\left(\frac{1}{n} \sum_{i=1}^{n} p_i^k \ell_i^k + \rho |\boldsymbol{\beta}^k| \right)}_{\text{regularized loss}} + \lambda \sum_{v^l \in \mathcal{L}} p^l \Bigg[\underbrace{\sum_t e_t \sqrt{\sum_{v^j \in \pi^l} (\beta_t^j)^2}}_{\text{CART evaluation cost penalty}} + \underbrace{\sum_{\alpha=1}^{d} c_\alpha \sqrt{\sum_{v^j \in \pi^l} \sum_{t=1}^{T} (F_{\alpha t} \beta_t^j)^2}}_{\text{feature cost penalty}} \Bigg]. \tag{22}$$

The objective in (22) can be optimized with the same block coordinate descent algorithm, as described in Section 3.4. Similarly, the CSCC loss function with non-linear classifiers becomes

$$\underbrace{\sum_{v^k \in \mathcal{V}} \frac{1}{n} \left(\sum_{i=1}^{n} w_{y_i} p_i^{k+1} \ell_i^k \right) + \rho |\boldsymbol{\beta}^k|}_{\text{regularized loss}} + \lambda \sum_{v^l \in L} p^l \Bigg[\underbrace{\sum_t e_t \sqrt{\sum_{v^j \in \pi^l} (\beta_t^j)^2}}_{\text{evaluation cost}} + \underbrace{\sum_{\alpha=1}^{d} c_\alpha \sqrt{\sum_{v^j \in \pi^l} \sum_{t=1}^{T} (F_{\alpha t} \beta_t^j)^2}}_{\text{feature cost}} \Bigg].$$

In the same way, Cronus may be adapted for non-linear classification (see: Chen et al., 2012). To avoid over-fitting, we use validation set to perform early-stopping during optimizing objective function 22.

6. Results

In this section, we evaluate CSTC on a synthetic cost-sensitive learning task and compare it with competing algorithms on two large-scale, real world benchmark problems. Additionally, we discuss the differences between our models for several learning settings. We provide further insight by analyzing the features extracted on a these data sets and looking at how CSTC tree partitions the input space. We judge the effect of the cost-sensitive regularization by looking at how removing terms and varying parameters affects CSTC on real world data sets. We also present detailed results of CSTC on a cost-sensitive version of the MNIST data set, demonstrating that it extracts intelligent *per-instance* features. We end by proposing a criterion that is designed to judge if CSTC will perform well on a data set.

6.1 Synthetic Data

We construct a synthetic regression data set consisting of points sampled from the four quadrants of the X, Z-plane, where $X = Z \in [-1, 1]$. The features belong to two categories: cheap features: $sign(x), sign(z)$ with cost $c = 1$, which can be used to identify the quadrant of an input; and expensive features: $z_{++}, z_{+-}, z_{-+}, z_{--}$ with cost $c = 10$, which equal the exact label of an input if it is from the corresponding quadrant (or a random number otherwise). Since in this synthetic data set we do not transform the feature space, we have $\phi(\mathbf{x}) = \mathbf{x}$, and $\mathbf{F}$ (the weak learner feature-usage variable) is the 6×6 identity matrix. By design, a perfect classifier can use the two cheap features to identify the sub-region of an instance and then extract the correct expensive feature to make a perfect prediction. The minimum feature cost of such a perfect classifier is exactly $c = 12$ per instance. We construct the data set to be a regression problem, with labels sampled from Gaussian distributions with quadrant-specific means $\mu_{++}, \mu_{-+}, \mu_{+-}, \mu_{--}$ and variance 1. The individual values for the label means are picked to satisfy the CSTC assumption, *i.e.* , that the prediction of similar labels requires similar features. In particular, as can be seen in Figure 5 (top left), label means from quadrants with negative $z-$coordinates (μ_{+-}, μ_{--}) are higher than those with positive $z-$coordinates (μ_{++}, μ_{-+}).

Figure 5 shows the raw data (top left) and a CSTC tree trained on this data with its predictions of test inputs made by each node. The semi-transparent gray hyperplane shows the values of thresholds, θ, and vertical gray lines show the difference between predicted label and true label, for each instance. In general, in every path along the tree, the first two classifiers split on the two cheap features and identify the correct sub-region of the input. The leaf classifiers extract a single expensive feature to predict the labels. As such, the mean squared error of the training and testing data both approach 0 (and the gray lines vanish) at optimal cost $c = 12$.

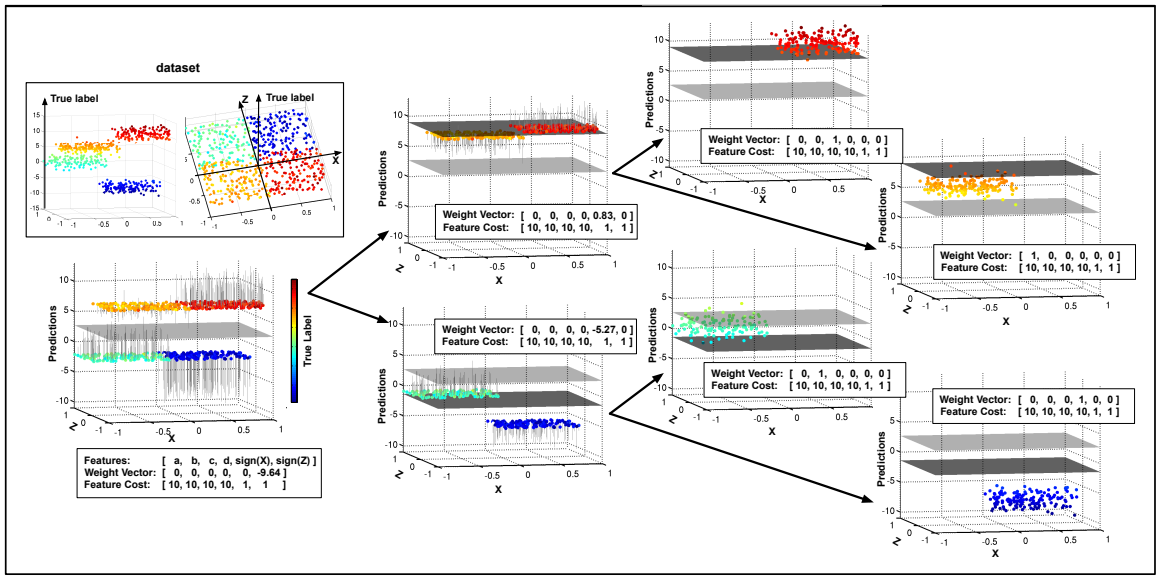


Figure 5: CSTC on synthetic data. The box at left describes the data set. The rest of the figure shows the trained CSTC tree. At each node we show a plot of the predictions made by that classifier and the feature weight vector. The tree obtains a perfect (0%) test-error at the optimal cost of 12 units.

6.2 Yahoo! Learning to Rank

To evaluate the performance of CSTC on real-world tasks, we test it on the Yahoo! Learning to Rank Challenge (LTR) data set. The set contains 19,944 queries and 473,134 documents. Each query-document pair $\mathbf{x}_i$ consists of 519 features. An extraction cost, which takes on a value in the set $\{1, 5, 20, 50, 100, 150, 200\}$, is associated with each feature.[1] The unit of these values turns out to be approximately the number of weak learner evaluations $h_t(\cdot)$ that can be performed while the feature is being extracted. The label $y_i \in \{4, 3, 2, 1, 0\}$ denotes the relevancy of a document to its corresponding query, with 4 indicating a perfect match. We measure the performance using normalized discounted cumulative gain at the 5^{th} position (NDCG@5) (Järvelin and Kekäläinen, 2002), a preferred ranking metric when multiple levels of relevance are available. Let π be an ordering of all inputs associated with a particular query ($\pi(r)$ is the index of the r^{th} ranked document and $y_{\pi(r)}$ is its relevance label), then the NDCG of π at position P is defined as

$$NDCG@P(\pi) = \frac{DCG@P(\pi)}{DCG@P(\pi^*)} \text{ with } DCG@P(\pi) = \sum_{r=1}^{P} \frac{2^{y_{\pi(r)}} - 1}{\log_2(r+1)},$$

where π^* is an optimal ranking (*i.e.* , documents are sorted in decreasing order of relevance). To introduce non-linearity, we transform the input features into a non-linear feature space $\mathbf{x} \to \phi(\mathbf{x})$ with the boosting trick (see Section 5) with $T = 3000$ iterations of gradient boosting and CART trees of maximum depth 4. Unless otherwise stated, we determine the CSTC depth by validation performance (with a maximum depth of 10).

1. The extraction costs were provided by a Yahoo! employee.

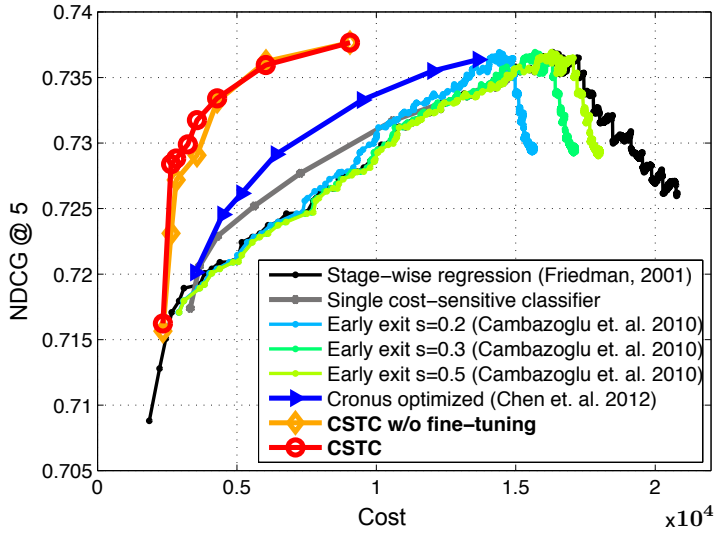


Figure 6: The test ranking accuracy (NDCG@5) and cost of various cost-sensitive classifiers. CSTC maintains its high retrieval accuracy significantly longer as the cost-budget is reduced.

Figure 6 shows a comparison of CSTC with several recent algorithms for test-time budgeted learning. We show NDCG versus cost (in units of weak learner evaluations). We obtain the curves of CSTC by varying the accuracy/cost trade-off parameter λ (and perform early stopping based on the validation data, for fine-tuning). For CSTC we evaluate eight settings, $\lambda = \{\frac{1}{3}, \frac{1}{2}, 1, 2, 3, 4, 5, 6\}$. In the case of *stage-wise regression*, which is not cost-sensitive, the curve is simply a function of boosting iterations. We include CSTC with and without fine-tuning. The comparison shows that there is a small but consistent benefit to fine-tuning the weights as described in Section 3.5.

For competing algorithms, we include *Early exit* (Cambazoglu et al., 2010) which improves upon stage-wise regression by short-circuiting the evaluation of unpromising documents at test-time, reducing the overall test-time cost. The authors propose several criteria for rejecting inputs early and we use the best-performing method "early exits using proximity threshold", where at the i^{th} early-exit, we remove all test-inputs that have a score that is at least $\frac{300-i}{299}s$ lower than the fifth best input, and s determines the power of the early-exit. The *single cost-sensitive classifier* is a trivial CSTC tree consisting of only the root node *i.e.* , a cost-sensitive classifier without the tree structure. We also include Cronus, which is described in Section 4. We set the maximum number of Cronus nodes to 10, and set all other parameters (*e.g.*, keep ratio, discount, early-stopping) based on a validation set. As shown in the graph, both Cronus and CSTC improve the cost/accuracy trade-off curve over all other algorithms. The power of Early exit is limited in this case as the test-time cost is dominated by feature extraction, rather than the evaluation cost. Compared with Cronus, CSTC has the ability to identify features that are most beneficial to different groups of inputs. It is this ability, which allows CSTC to maintain the high NDCG significantly longer

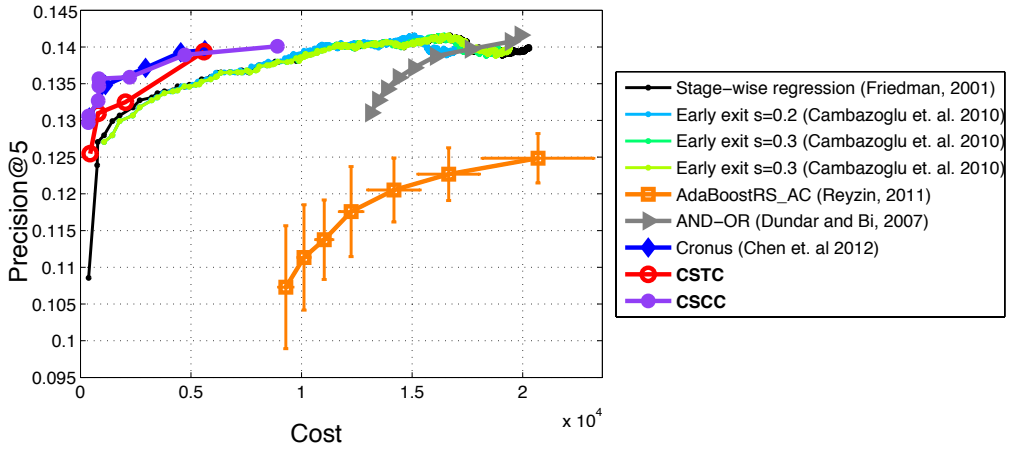


Figure 7: The test ranking accuracy (Precision@5) and cost of various budgeted cascade classifiers on the Skew-LTR data set with high class imbalance. CSCC outperforms similar techniques, requiring less cost to achieve the same performance.

as the cost-budget is reduced. It is interesting to observe that the single cost-sensitive classifier outperforms stage-wise regression (due to the cost sensitive regularization) but obtains much worse cost/accuracy trade offs than the full CSTC tree. This demonstrates that the tree structure is indeed an important part of the high cost effectiveness of CSTC.

6.3 Yahoo! Learning to Rank: Skewed, Binary

To evaluate the performance of our cascade approach CSCC, we construct a highly class-skewed binary data set using the Yahoo! LTR data set. We define inputs having labels $y_i \geq 3$ as 'relevant' and label the rest as 'irrelevant', binarizing the data in this way. We also replicate each negative, irrelevant example 10 times to simulate the scenario where only a few documents are highly relevant, out of many candidate documents. After these modifications, the inputs have one of two labels $\{-1, 1\}$, and the ratio of $+1$ to -1 is $1/100$. We call this data set LTR-Skewed. This simulates an important setting, as in many time-sensitive real life applications the class distributions are often very skewed.

For the binary case, we use the ranking metric Precision@5 (the fraction of top 5 documents retrieved that are relevant to a query). It best reflects the capability of a classifier to precisely retrieve a small number of relevant instances within a large set of irrelevant documents. Figure 7 compares CSCC and Cronus with several recent algorithms for binary budgeted learning. We show Precision@5 versus cost (in units of weak learner evaluations). Similar to CSTC, we obtain the curves of CSCC by varying the accuracy/cost trade-off parameter λ. For CSCC we evaluate eight settings, $\lambda = \{\frac{1}{3}, \frac{1}{2}, 1, 2, 3, 4, 5, 6\}$.

For competing algorithms, in addition to *Early exit* (Cambazoglu et al., 2010) described above, we also include *AND-OR* proposed by Dundar and Bi (2007), which is designed specifically for binary budgeted learning. They formulate a global optimization of a cas-

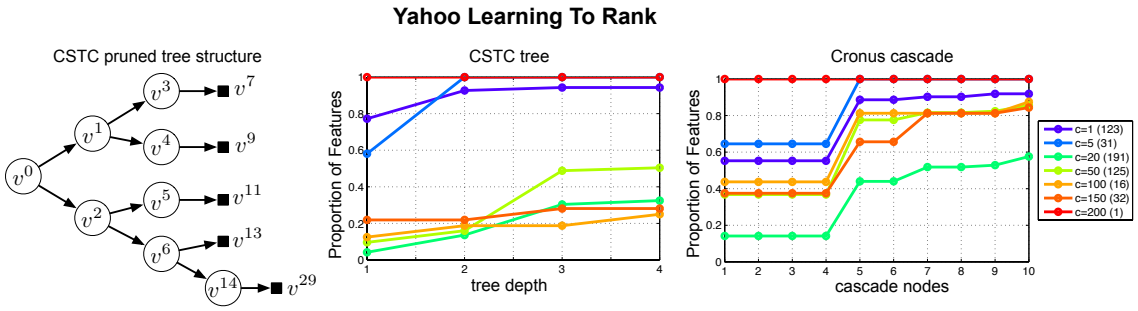


Figure 8: *Left:* The pruned CSTC tree, trained on the Yahoo! LTR data set. The ratio of features, grouped by cost, are shown for CSTC (*center*) and Cronus (*right*). The number of features in each cost group is indicated in parentheses in the legend. More expensive features ($c \geq 20$) are gradually extracted deeper in the structure of each algorithm.

cade of classifiers and employ an *AND-OR* scheme with the loss function that treats negative inputs and positive inputs separately. This setup is based on the insight that positive inputs are carried all the way through the cascade (*i.e.* , each classifier must classify them as positive), whereas negative inputs can be rejected at any time (*i.e.* , it is sufficient if a single classifier classifies them as negative). The loss for positive inputs is the maximum loss across all stages, which corresponds to the AND operation, and encourages all classifiers to make correct predictions. For negative inputs the loss is the minimum loss of all classifiers, which corresponds to the OR operation, and which enforces that at least one classifier makes a correct prediction. Different from our approach, their algorithm requires pre-assigning features to each node. We therefore use five nodes in total, assigning features of cost $\leq 5, \leq 20, \leq 50, \leq 150, \leq 200$. The curve is generated by varying a loss/cost trade-off parameter (similar to λ). Finally, we also compare with the cost sensitive version of AdaboostRS (Reyzin, 2011). This algorithm resamples decision trees, learned with AdaBoost (Freund et al., 1999), inversely proportional to a tree's feature cost. As this algorithm involves random sampling, we averaged over 10 runs and show the standard deviations in both precision and cost.

As shown in the graph, AdaBoostRS obtains lower precision than other algorithms. This may be due to the known sensitivity of AdaBoost towards noisy data, (Melville et al., 2004). AND-OR also under-performs. It requires pre-assigning features prior to training, which makes it impossible to obtain high precision at a low cost. On the other hand, Cronus, CSCC, and CSTC have the ability to cherry pick good but expensive features at an early node, which in turn can reduce the overall cost while improving performance over other algorithms. We take a closer look at this effect in the following section. Cronus and CSCC in general outperform CSTC because they can exit a large portion of the data set early on. As mentioned before, CSCC outperforms Cronus a little bit, which we attribute to the more principled optimization.

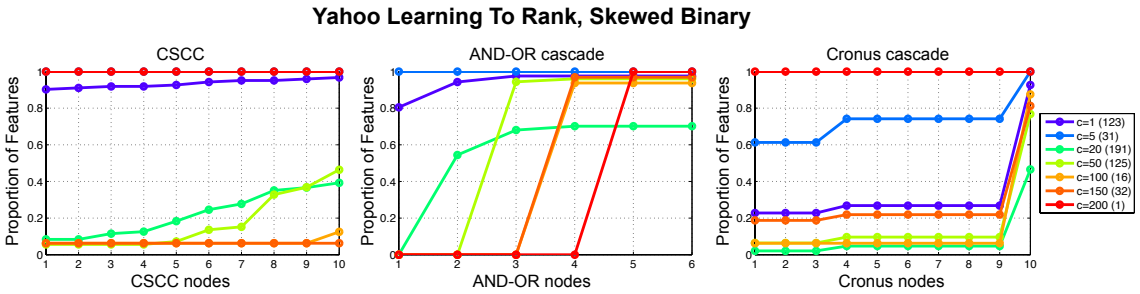


Figure 9: The ratio of features, grouped by cost, that are extracted at different depths of CSCC (*left*), AND-OR (*center*) and Cronus (*right*). The number of features in each cost group is indicated in parentheses in the legend.

6.4 Feature Extraction

Based on the LTR and LTR-Skewed data sets, we investigate the features extracted by various algorithms in each scenario. We fist show the features retrieved in the regular balanced class data set (LTR). Figure 8 (*left*) shows the pruned CSTC tree learned on the LTR data set. The plot in the center demonstrates the fraction of features, with a particular cost, extracted at different depths of the CSTC tree. The rightmost plot shows the features extracted at different nodes of Cronus. We observe a general trend that for both CSTC and Cronus, as depth increases, more features are being used. However, cheap features ($c \leq 5$) are all extracted early-on, whereas expensive features ($c \geq 20$) are extracted by classifiers sitting deeper in the tree. Here, individual classifiers only cope with a small subset of inputs and the expensive features are used to classify these subsets more precisely. The only feature that has cost 200 is extracted at all depths—which seems essential to obtain high NDCG (Chen et al., 2012). Although Cronus has larger depth than CSTC (10 vs 4), most nodes in Cronus are basically dummy nodes (as can be seen by the flat parts of the feature usage curve). For these nodes all weights are zeros, and the threshold is a very small negative number, allowing all inputs to pass through.

In the second scenario, where the class-labels are binarized and are highly skewed (LTR-Skewed), we compare the features extracted by CSCC, Cronus and AND-OR. For a fair comparison, we set the trade-off parameter λ for each algorithm to achieve similar precision 0.135 ± 0.001. We also set the maximum number of nodes of CSCC and Cronus to 10. Figure 9 (*left*) shows the fraction of features, with a particular cost, extracted at different nodes of the CSCC. The center plot illustrates the features used by AND-OR, and the right plot shows the features extracted at different nodes of Cronus. Note that while the features are pre-assigned in the AND-OR algorithm, it still has the ability to only use some of the assigned features at each node. In general, all algorithms use more features as the depth increases. However, compared to AND-OR, both Cronus and CSCC can cherry pick some good but expensive features early-on to achieve high accuracy at a low cost. Some of the expensive features (*e.g.*, $c = 100, 150$) are extracted from the very first node in CSCC and Cronus, whereas in AND-OR, they are only available at the fourth node. This ability is

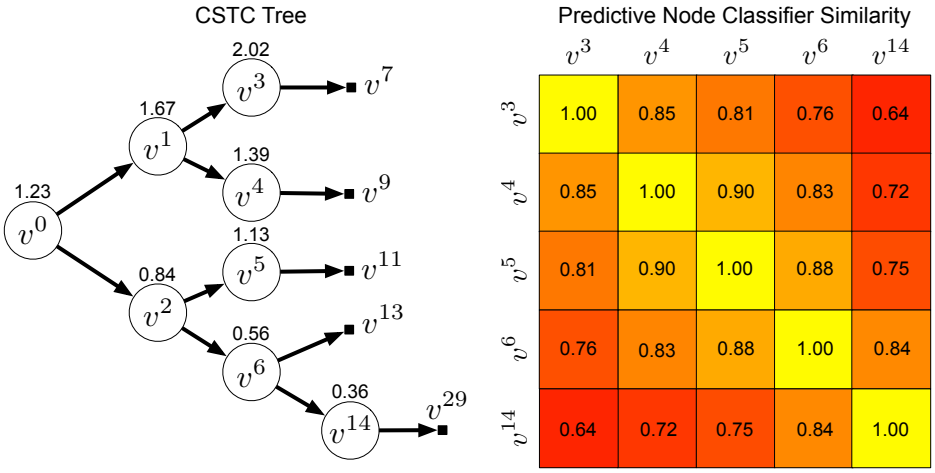


Figure 10: (*Left*) The pruned CSTC-tree generated from the Yahoo! Learning to Rank data set. (*Right*) Jaccard similarity coefficient between classifiers within the learned CSTC tree.

one of the reasons that CSCC and Cronus achieve better performance over existing cascade algorithms.

6.5 Input Space Partition

CSTC has the ability to split the input space and learn more specialized classifiers sitting deeper in the tree. Figure 10 (*left*) shows a pruned CSTC tree ($\lambda = 4$) for the LTR data set. The number above each node indicates the average label of the testing inputs passing through that node. We can observe that different branches aim at different parts of the input domain. In general, the upper branches focus on correctly classifying higher-ranked documents, while the lower branches target low-rank documents. Figure 10 (*right*) shows the Jaccard matrix of the leaf classifiers ($v^3, v^4, v^5, v^6, v^{14}$) from this CSTC tree. The number in field i, j indicates the fraction of shared features between v^i and v^j. The matrix shows a clear trend that the Jaccard coefficients decrease monotonically away from the diagonal. This indicates that classifiers share fewer features in common if their average labels are further apart—the most different classifiers v^3 and v^{14} have only 64% of their features in common—and validates that classifiers in the CSTC tree extract different features in different regions of the tree.

6.6 CSTC Sensitivity

Recall the CSTC objective function with non-linear classifiers,

$$\sum_{v^k \in \mathcal{V}} \underbrace{\left(\frac{1}{n} \sum_{i=1}^{n} p_i^k \ell_i^k + \rho |\boldsymbol{\beta}^k| \right)}_{\text{regularized loss}} + \lambda \sum_{v^l \in \mathcal{L}} p^l \left[\underbrace{\sum_t e_t \sqrt{\sum_{v^j \in \pi^l} (\beta_t^j)^2}}_{\text{CART evaluation cost penalty}} + \underbrace{\sum_{\alpha=1}^{d} c_\alpha \sqrt{\sum_{v^j \in \pi^l} \sum_{t=1}^{T} (F_{\alpha t} \beta_t^j)^2}}_{\text{feature cost penalty}} \right].$$

In order to judge the effect of different terms in the cost-sensitive regularization we experiment with removing the CART evaluation cost penalty (or simply 'evaluation cost') and/or the feature cost penalty. Figure 11 (*Left*) shows the performance of CSTC and the less cost-sensitive variants, after removing one or both of the penalty terms, on the Yahoo! LTR data set. As we suspected, the feature cost term seems to contribute most to the performance of CSTC. Indeed, only taking into account evaluation cost severely impairs the model. Without considering cost, CSTC seems to overfit even though the remaining l_1-regularization prevents the model from extracting all possible features.

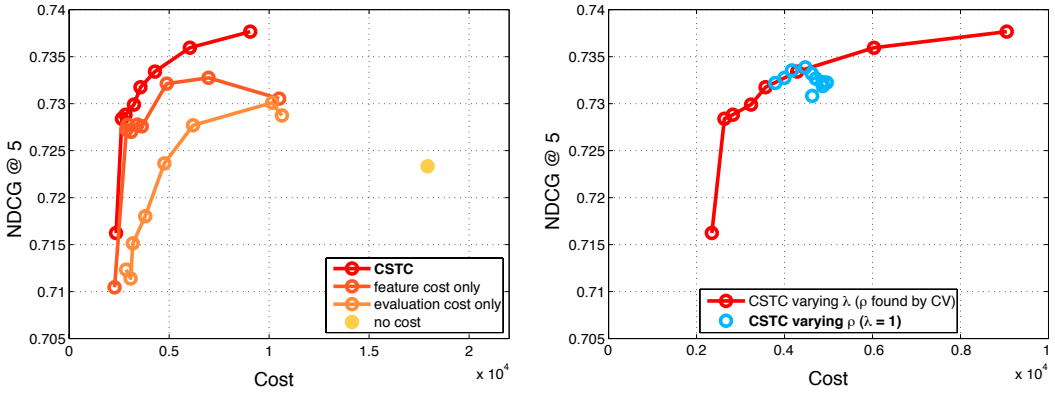


Figure 11: Two plots showing the sensitivity of CSTC to different cost regularization and hyperparameters. *Left:* The test ranking accuracy (NDCG@5) and cost of different CSTC cost-sensitive variants. *Right:* The performance of CSTC for different values of hyperparameter $\rho \in [0.5, 0.6, 0.7, 0.8, 0.9, 1, 2, 3, 4, 5]$.

We are also interested in judging the effect of the l_1-regularization hyperparameter ρ on the performance of CSTC. Figure 11 (*Right*) shows for $\lambda = 1$ different settings of $\rho \in [0.5, 0.6, 0.7, 0.8, 0.9, 1, 2, 3, 4, 5]$ and the resulting change in cost and NDCG. The result shows that varying ρ does follow the CSTC NDCG/cost trade-off for a little bit, however ultimately leads to a reduction in accuracy. This supports our hypothesis that the cost term is crucial to obtain low cost classifiers.

6.7 Cost-Sensitive MNIST

We created a cost-sensitive binary MNIST data set by first extracting all images of digits 3 and 8. We resized them to four different resolutions: 4×4, 8×8, 16×16, and 28×28 (the

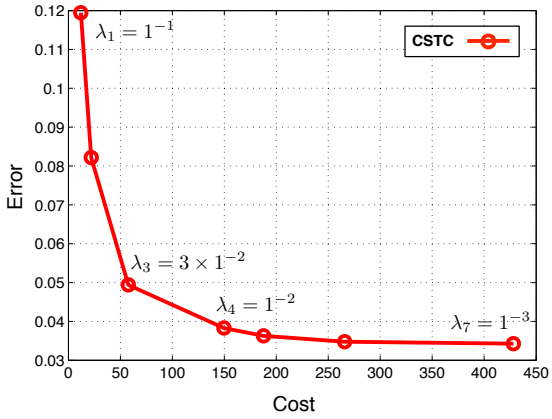


Figure 12: The test error vs. cost trade-off for CSTC on the cost-sensitive MNIST data set

original size), and concatenated all features, resulting in $d = 1120$ features. We assigned each feature a cost $c = 1$. To obtain a baseline error rate for the data set we trained a support vector machine (SVM) with a radial basis function (RBF) kernel. To select hyperparameters C (SVM cost) and γ (kernel width) we used 100 rounds of Bayesian optimization on a validation set (we found $C = 753.1768$, $\gamma = 0.0198$). An RBF-SVM trained with these hyperparameters achieves a test error of 0.005 for cost $c = 1120$. Figure 12 shows error versus cost for different values of the trade-off parameter λ. We note that CSTC smoothly trades off feature cost for error, quickly reducing error initially at small increases in cost.

Figure 13 shows the features extracted and trees built for different values of λ for the cost-sensitive MNIST data set. For each value, we show the paths of one randomly selected 3-instance (lower paths) and one 8-instance (upper paths). For each node in each path we show the features extracted at each of the four resolutions in boxes (red indicates a feature was not extracted). In general, as λ is decreased, more features are extracted. Additionally, for a single λ, nodes along a path tend to use the same features. Finally, even when the algorithm is restricted to use very little cost (*i.e.*, the λ_1 tree) it is still able to find features that distinguish the classes in the data set, sending the 3 and 8-instances along different paths in the tree.

6.8 CSTC Criterion

CSTC implicitly assumes that similarly-labeled inputs can be classified using similar features by sending instances with different predictions to different classifiers. Since not all data sets have such a property, we propose a simple test which indicates if a data set satisfies this assumption. We train binary classifiers with l_1-regularization for neighboring pairs of labels. As an example, the LTR data set contains five labels $\{0, 1, 2, 3, 4\}$, so we train four binary classifiers, (0 *vs.* 1, 1 *vs.* 2, etc.). We then compute the Jaccard coefficient matrix $\mathbf{J}$ between the sparse (because of the l_1-regularizer) feature vectors $\boldsymbol{\beta}$ of all classifiers, where an element $\mathbf{J}_{ij}$ indicates the percentage of overlapping features selected by classifiers i and j. Figure 14 shows this Jaccard matrix on the LTR data set. The figure shows a clear trend that the

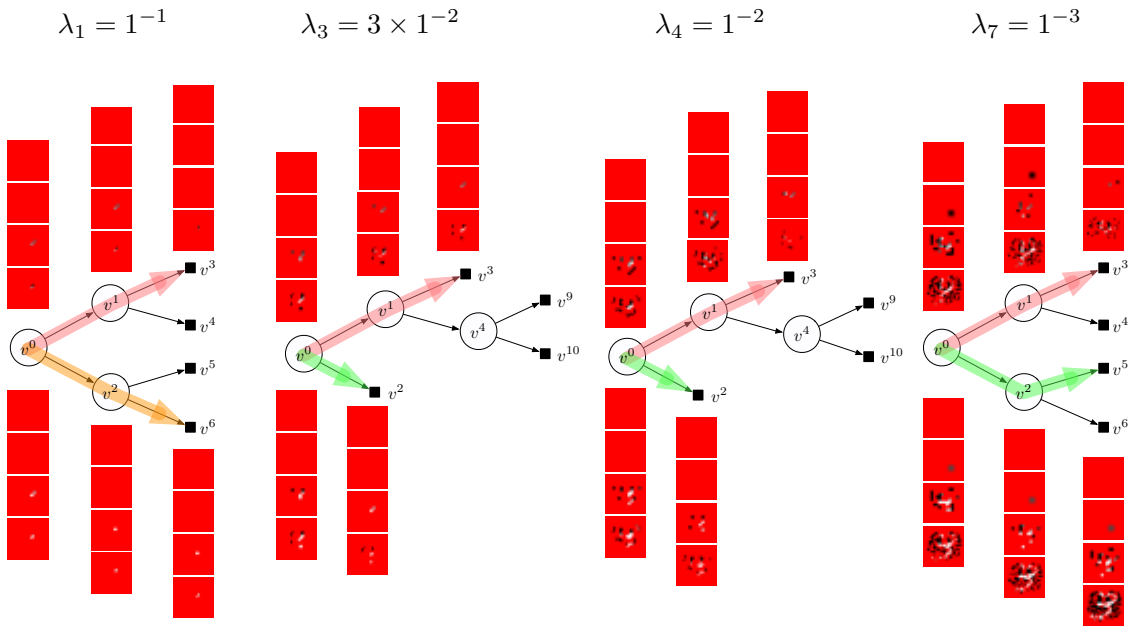


Figure 13: Trees generated for different λ on the cost-sensitive MNIST data set. For each λ we show the path of one 3-instance (orange or green) and one 8-instance (red). For each node these instances visit, we show the features extracted above (for the 8-instance) and below (for the 3-instance) the trees in boxes for different resolutions (red indicates a feature was not extracted).

Jaccard coefficients decrease monotonically away from the diagonal. This indicates that classifiers share fewer features in common if the average label of their training data sets are further apart—a good indication that on this data set CSTC will perform well.

7. Related Work

In the following we review different methods for budgeting the computational cost during test-time starting with simply reducing the feature space via l_1-regularization up to recent work in budgeted learning.

7.1 l_1 Norm Regularization

A related approach to control test-time cost is *feature selection* with l_1 norm regularization (Efron et al., 2004),

$$\min_{\boldsymbol{\beta}} \ell(H(\mathbf{x}; \boldsymbol{\beta}), \mathbf{y}) + \lambda |\boldsymbol{\beta}|,$$

where $\lambda \geq 0$ controls the magnitude of regularization. The l_1-norm regularization results in a sparse feature set (Schölkopf and Smola, 2001), and can significantly reduce the feature cost during test-time (as unused features are never computed). Individual feature cost can be incorporated with feature specific regularization trade-offs, λ_α. The downside of this

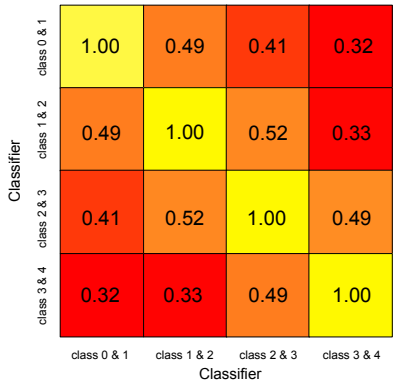


Figure 14: Jaccard similarity coefficient between binary classifiers learned on the LTR data set. The binary classifiers are trained with l_1-regularization for neighboring pairs of labels. As an example, the LTR data set contains five labels $\{0, 1, 2, 3, 4\}$, four binary classifiers are trained, (0 *vs.* 1, 1 *vs.* 2, etc.). There is a clear trend that the Jaccard coefficients decrease monotonically away from the diagonal. This indicates that classifiers share fewer features in common if the average label of their training data sets are further apart—an indication that CSTC will perform well.

approach is that it extracts a feature for *all* inputs or *none*, which makes it uncompetitive with more flexible cascade or tree models.

7.2 Feature Selection

Another approach, extending l_1-regularization, is to select features using some external criterion that naturally limits the number of features used. Cesa-Bianchi et al. (2011) construct a linear classifier given a budget by selecting one instance at a time, then using the current parameters to select a useful feature. This process is repeated until the budget is met. Globerson and Roweis (2006) formulate feature selection as an adversarial game and use minimax to develop a worst-case strategy, assuming feature removal at test-time. These approaches, however, are unaware of the test-time cost in (3), and fail to pick the optimal feature set that best trades-off loss and cost. Dredze et al. (2007) gets closer to directly balancing this trade-off by combining the cost to select a feature with the mutual information of that feature to build a decision tree that reduces the feature extraction cost. This work, though, does not directly minimize the total test-time cost vs. accuracy trade-off of the classifier. Most recently, Xu et al. (2013b) proposed to learn a new feature representation entirely using selected features.

7.3 Linear Cascades

Grubb and Bagnell (2012) and Xu et al. (2012) focus on training a classifier that explicitly trades-off the test-time cost with the loss. Grubb and Bagnell (2012) introduce *SpeedBoost*, a generalization of functional gradient descent for *anytime predictions* (Zilberstein, 1996),

which incorporates the prediction cost during training. The resulting algorithms obtain good approximations at very low cost and refine their predictions if more resources are available. Both algorithms learn classifier cascades that schedule the computational budget and can terminate prematurely if easy inputs reach confident predictions early, to save overall CPU budget for more difficult inputs. This schedule is identical for all inputs, whereas CSTC decides to send inputs along different paths within the tree of classifiers to potentially extract fundamentally different features.

Based on the earlier observation that not all inputs require the same amount of computation to obtain a confident prediction, there is much previous work that addresses this by building classifier cascades (mostly for binary classification) (Viola and Jones, 2004; Dundar and Bi, 2007; Lefakis and Fleuret, 2010; Saberian and Vasconcelos, 2010; Pujara et al., 2011; Chen et al., 2012; Reyzin, 2011; Trapeznikov et al., 2013b). They chain several classifiers into a sequence of stages. Each classifier can either early-exit inputs (predicting them), or pass them on to the next stage. This decision is made based on the prediction of each instance. Different from CSCC, these algorithms typically do not take into account feature cost and implement more ad hoc rules to trade-off accuracy and cost.

7.4 Dynamic Feature Selection During Test-Time

For learning tasks with *balanced classes* and *specialized features*, the linear cascade model is less well-suited. Because all inputs follow exactly the same linear path, it cannot capture the scenario in which different subsets of inputs require different expert features. Chai et al. (2004) introduce the value of unextracted features, where the value of a feature is the increase (gain) in expected classification accuracy minus the cost of including that feature. During test-time, each iteration, their algorithm picks the feature that has the highest value and retrains a classifier with the new feature. The algorithm stops when there is no increase in expected classification rate, or all features are included. Because they employ a naive Bayes classifier, retraining incurs very little cost. Similarly, Gao and Koller (2011) use locally weighted regression during test-time to predict the information gain of unknown features.

Most recently, Karayev et al. (2012) use reinforcement learning during test-time to dynamically select object detectors for a particular image. He et al. (2013) use imitation learning to select instance-specific features for graph-based dependency parsing. Our approach shares the same idea that different inputs require different features. However, instead of learning the best feature for each input during test-time, which introduces an additional cost, we learn and fix a tree structure in training. Each branch of the tree only handles a subset of the input space and, as such, the classifiers in a given branch become specialized for those inputs. Moreover, because we learn a fixed tree structure, it has a test-time complexity that is constant with respect to the training set size.

7.5 Budgeted Tree-Structured Classifiers

Concurrently, there has been work (Deng et al., 2011) to speed up the training and evaluation of tree-structured classifiers (specifically label trees: Bengio et al., 2010), by avoiding many binary one-vs-all classifier evaluations. In many real world data sets the test-time cost is largely composed of feature extraction time and so our aim is different from their work.

Another model (Beygelzimer et al., 2009) learns a tree of classifiers online for estimating the conditional probability of an input label. Their aim is also different from ours as they only consider reducing the training time necessary for the estimation problem. Goetschalckx and Driessens also introduces parsimonious linear model tree to control test-time cost. Possibly most similar to our work is Busa-Fekete et al. (2012), who apply a Markov decision process to learn a directed acyclic graph. At each step, they select features for different instances. Although similar in motivation, their algorithmic framework is very different and can be regarded complementary to ours.

It is worth mentioning that, although Hierarchical Mixture of Experts (HME) (Jordan and Jacobs, 1994) also builds tree-structured classifiers, it does not consider reducing the test-time cost and thus results in fundamentally different models. In contrast, we train each classifier with the test-time cost in mind and each test input only traverses a *single* path from the root down to a terminal element, accumulating path-specific costs. In HME, all test inputs traverse all paths and all leaf-classifiers contribute to the final prediction, incurring the same cost for all test inputs.

8. Conclusions

In this paper, we systematically investigate the trade off between test-time CPU cost and accuracy in real-world applications. We formulate this trade off mathematically for a tree of classifiers and relax it into a well-behaved optimization problem. Our algorithm, Cost-Sensitive Tree of Classifiers (CSTC), partitions the input space into sub-regions and identifies the most cost-effective features for each one of these regions—allowing it to match the high accuracy of the state-of-the-art at a small fraction of the cost. This cost function can be minimized while learning the parameters of all classifiers in the tree jointly.

As the use of machine learning algorithms becomes more and more wide spread, addressing the CPU test-time cost becomes a problem of ever increasing importance. CSTC is one solution but there are still many unanswered questions. Future work will investigate learning theoretical guarantees for test-time budgeted learning, worst-case scenarios (in contrast to average cost), other learning frameworks (*e.g.* , SVM classifiers: Cortes and Vapnik, 1995) and the incorporation of hardware architectures constraints (*e.g.* , clusters, GPUs and shared memory machines). We consider the principled approach of CSTC an important step towards the ultimate goal of fully integrating test-time budgets into machine learning.

Acknowledgements

KQW, ZX, and MK are supported by NIH grant U01 1U01NS073457-01 and NSF grants 1149882 and 1137211.

References

S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. *Advances in Neural Information Processing Systems*, 23:163–171, 2010.

A. Beygelzimer, J. Langford, Y. Lifshits, G. Sorkin, and A. Strehl. Conditional probability tree estimation analysis and algorithms. In *Conference on Uncertainty in Artificial Intelligence*, pages 51–58, 2009.

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

L. Breiman. *Classification and Regression Trees*. Chapman & Hall/CRC, 1984.

R. Busa-Fekete, D. Benbouzid, B. Kégl, et al. Fast classification using sparse decision DAGs. In *International Conference on Machine Learning*, 2012.

B.B. Cambazoglu, H. Zaragoza, O. Chapelle, J. Chen, C. Liao, Z. Zheng, and J. Degenhardt. Early exit optimizations for additive machine learned ranking systems. In *Web Search and Data Mining*, pages 411–420, 2010.

N. Cesa-Bianchi, S. Shalev-Shwartz, and O. Shamir. Efficient learning with partially observed attributes. *The Journal of Machine Learning Research*, 12:2857–2878, 2011.

X. Chai, L. Deng, Q. Yang, and C.X. Ling. Test-cost sensitive naive Bayes classification. In *International Conference on Data Mining*, pages 51–58. IEEE, 2004.

O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng. Boosted multi-task learning. *Machine Learning*, 85(1):149–173, 2011.

M. Chen, Z. Xu, K. Q. Weinberger, and O. Chapelle. Classifier cascade for minimizing feature evaluation cost. In *International Conference on Artificial Intelligence and Statistics*, 2012.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

J. Deng, S. Satheesh, A.C. Berg, and L. Fei-Fei. Fast and balanced: Efficient label tree learning for large scale object recognition. In *Advances in Neural Information Processing Systems*, 2011.

M. Dredze, R. Gevaryahu, and A. Elias-Bachrach. Learning fast classifiers for image spam. In *Conference on Email and Anti-Spam*, 2007.

M.M. Dundar and J. Bi. Joint optimization of cascaded classifiers for computer aided detection. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

M. Fleck, D. Forsyth, and C. Bregler. Finding naked people. *European Conference on Computer Vision*, pages 593–602, 1996.

Y. Freund, R. Schapire, and N. Abe. A short introduction to boosting. *Japanese Society for Artificial Intelligence*, 14(771-780):1612, 1999.

J.H. Friedman. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, pages 1189–1232, 2001.

T. Gao and D. Koller. Active classification based on value of classifier. In *Advances in Neural Information Processing Systems*, pages 1062–1070. 2011.

A. Globerson and S. Roweis. Nightmare at test time: robust learning by feature deletion. In *International Conference on Machine Learning*, pages 353–360. ACM, 2006.

R. Goetschalckx and K. Driessens. Parsimonious linear model trees.

A. Grubb and J. A. Bagnell. Speedboost: Anytime prediction with uniform near-optimality. In *International Conference on Artificial Intelligence and Statistics*, 2012.

H. He, H. Daumé III, and J. Eisner. Dynamic feature selection for dependency parsing. In *Conference on Empirical Methods in Natural Language Processing*, 2013.

K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *Transactions on Information Systems*, 20(4):422–446, 2002.

M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

S. Karayev, T. Baumgartner, M. Fritz, and T. Darrell. Timely object recognition. In *Advances in Neural Information Processing Systems*, pages 899–907, 2012.

B. Korte and J. Vygen. *Combinatorial Optimization*, volume 21. Springer, 2012.

M. Kowalski. Sparse regression using mixed norms. *Applied and Computational Harmonic Analysis*, 27(3):303–324, 2009.

L. Lefakis and F. Fleuret. Joint cascade optimization using a product of boosted classifiers. In *Advances in Neural Information Processing Systems*, pages 1315–1323. 2010.

P. Melville, N. Shah, L. Mihalkova, and R.J. Mooney. Experiments on ensembles with missing and noisy data. In *Multiple Classifier Systems*, pages 293–302. Springer, 2004.

J. Pujara, H. Daumé III, and L. Getoor. Using classifier cascades for scalable e-mail classification. In *Conference on Email and Anti-Spam*, 2011.

L. Reyzin. Boosting on a budget: Sampling for feature-efficient prediction. In *International Conference on Machine Learning*, pages 529–536, 2011.

M. Saberian and N. Vasconcelos. Boosting classifier cascades. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, pages 2047–2055. 2010.

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

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

K. Trapeznikov and V. Saligrama. Supervised sequential classification under budget constraints. In *International Conference on Artificial Intelligence and Statistics*, pages 581–589, 2013a.

K. Trapeznikov, V. Saligrama, and D. Castañón. Multi-stage classifier design. *Machine Learning*, 92(2-3):479–502, 2013b.

P. Viola and M.J. Jones. Robust real-time face detection. *International Journal on Computer Vision*, 57(2):137–154, 2004.

K.Q. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *International Conference on Machine Learning*, pages 1113–1120, 2009.

Z. Xu, K. Weinberger, and O. Chapelle. The greedy miser: Learning under test-time budgets. In *International Conference on Machine Learning*, pages 1175–1182, 2012.

Z. Xu, M.J. Kusner, M. Chen, and K.Q. Weinberger. Cost-sensitive tree of classifiers. In *International Conference on Machine Learning*, pages 133–141. JMLR Workshop and Conference Proceedings, 2013a.

Z. Xu, M.J. Kusner, G. Huang, and K.Q. Weinberger. Anytime representation learning. In *International Conference on Machine Learning*, pages 1076–1084, 2013b.

Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In *Advances in Neural Information Processing Systems*, pages 1697–1704. Cambridge, MA, 2008.

S. Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73, 1996.

Journal of Machine Learning Research 15 (2014) 2145-2184 Submitted 1/14; Revised 4/14; Published 6/14

Particle Gibbs with Ancestor Sampling

Fredrik Lindsten FREDRIK.LINDSTEN@ENG.CAM.AC.UK
Department of Engineering
University of Cambridge
Cambridge, CB2 1PZ, UK, and

Division of Automatic Control
Linköping University
Linköping, 581 83, Sweden

Michael I. Jordan JORDAN@CS.BERKELEY.EDU
Computer Science Division and Department of Statistics
University of California
Berkeley, CA 94720, USA

Thomas B. Schön THOMAS.SCHON@IT.UU.SE
Department of Information Technology
Uppsala University
Uppsala, 751 05, Sweden

Editor: Yee Whye Teh

Abstract

Particle Markov chain Monte Carlo (PMCMC) is a systematic way of combining the two main tools used for Monte Carlo statistical inference: sequential Monte Carlo (SMC) and Markov chain Monte Carlo (MCMC). We present a new PMCMC algorithm that we refer to as *particle Gibbs with ancestor sampling* (PGAS). PGAS provides the data analyst with an off-the-shelf class of Markov kernels that can be used to simulate, for instance, the typically high-dimensional and highly autocorrelated state trajectory in a state-space model. The *ancestor sampling* procedure enables fast mixing of the PGAS kernel even when using seemingly few particles in the underlying SMC sampler. This is important as it can significantly reduce the computational burden that is typically associated with using SMC. PGAS is conceptually similar to the existing *PG with backward simulation* (PGBS) procedure. Instead of using separate forward and backward sweeps as in PGBS, however, we achieve the same effect in a single forward sweep. This makes PGAS well suited for addressing inference problems not only in state-space models, but also in models with more complex dependencies, such as non-Markovian, Bayesian nonparametric, and general probabilistic graphical models.

Keywords: particle Markov chain Monte Carlo, sequential Monte Carlo, Bayesian inference, non-Markovian models, state-space models

1. Introduction

Monte Carlo methods are one of the standard tools for inference in statistical models as they, among other things, provide a systematic approach to the problem of computing Bayesian posterior probabilities. Sequential Monte Carlo (SMC, see, e.g., Doucet and Jo-

hansen, 2011; Del Moral et al., 2006) and Markov chain Monte Carlo (MCMC, see, e.g., Robert and Casella, 2004; Liu, 2001) methods in particular have found application to a wide range of data analysis problems involving complex, high-dimensional models. These include state-space models (SSMs) which are used in the context of time series and dynamical systems modeling in a wide range of scientific fields. The strong assumptions of linearity and Gaussianity that were originally invoked for SSMs have indeed been weakened by decades of research on SMC and MCMC. These methods have not, however, led to a substantial weakening of a further strong assumption, that of Markovianity. It remains a major challenge to develop efficient inference algorithms for models containing a latent stochastic process which, in contrast with the state process in an SSM, is non-Markovian. Such non-Markovian latent variable models arise in various settings, either from direct modeling or via a transformation or marginalization of an SSM. We discuss this further in Section 6; see also Lindsten and Schön (2013, Section 4).

In this paper we present a new tool in the family of Monte Carlo methods which is particularly useful for inference in SSMs and, importantly, in non-Markovian latent variable models. However, the proposed method is by no means limited to these model classes. We work within the framework of particle MCMC (PMCMC, Andrieu et al., 2010) which is a systematic way of combining SMC and MCMC, exploiting the strengths of both techniques. More specifically, PMCMC samplers make use of SMC to construct efficient, high-dimensional MCMC kernels. These kernels can then be used as off-the-shelf components in MCMC algorithms and other inference strategies relying on Markov kernels. PMCMC has in a relatively short period of time found many applications in areas such as hydrology (Vrugt et al., 2013), finance (Pitt et al., 2012), systems biology (Golightly and Wilkinson, 2011), and epidemiology (Rasmussen et al., 2011), to mention a few.

Our method builds on the particle Gibbs (PG) sampler proposed by Andrieu et al. (2010). In PG, the aforementioned Markov kernel is constructed by running an SMC sampler in which one particle trajectory is set deterministically to a reference trajectory that is specified *a priori*. After a complete run of the SMC algorithm, a new trajectory is obtained by selecting one of the particle trajectories with probabilities given by their importance weights. The effect of the reference trajectory is that the resulting Markov kernel leaves its target distribution invariant, regardless of the number of particles used in the underlying SMC algorithm.

However, PG suffers from a serious drawback, which is that the mixing of the Markov kernel can be very poor when there is path degeneracy in the underlying SMC sampler (Lindsten and Schön, 2013; Chopin and Singh, 2014). Unfortunately, path degeneracy is inevitable for high-dimensional problems, which significantly reduces the applicability of PG. This problem has been addressed in the generic setting of SSMs by adding a backward simulation step to the PG sampler, yielding a method denoted as *PG with backward simulation* (PGBS, Whiteley, 2010; Whiteley et al., 2010; Lindsten and Schön, 2012). It has been found that this considerably improves mixing, making the method much more robust to a small number of particles as well as growth in the size of the data (Lindsten and Schön, 2013; Chopin and Singh, 2014; Whiteley et al., 2010; Lindsten and Schön, 2012).

Unfortunately, however, the application of backward simulation is problematic for models with more intricate dependencies than in SSMs, such as non-Markovian latent variable models. The reason is that we need to consider complete trajectories of the latent process

during the backward simulation pass (see Section 6.2 for details). The method proposed in this paper, which we refer to as *particle Gibbs with ancestor sampling* (PGAS), is geared toward this issue. PGAS alleviates the problem with path degeneracy by modifying the original PG kernel with a so-called ancestor sampling (AS) step, thereby achieving the same effect as backward sampling, but without an explicit backward pass.

After giving some background on SMC in Section 2, the PGAS Markov kernel is constructed and analyzed theoretically in Sections 3 and 4, respectively. This extends the preliminary work that we have previously published (Lindsten et al., 2012) with a more straightforward construction, a more complete proof of invariance, and a new uniform ergodicity result. We then show specifically how PGAS can be used for inference and learning of SSMs and of non-Markovian latent variable models in Sections 5 and 6, respectively. As part of our development, we also propose a truncation strategy specifically for non-Markovian models. This is a generic method that is also applicable to PGBS, but, as we show in a simulation study in Section 7, the effect of the truncation error is much less severe for PGAS than for PGBS. Indeed, we obtain up to an order-of-magnitude increase in accuracy in using PGAS when compared to PGBS in this study. We also evaluate PGAS on a stochastic volatility SSM and on an epidemiological model. Finally, in Section 8 we conclude and point out possible directions for future work.

2. Sequential Monte Carlo

Let $\gamma_{\theta,t}(x_{1:t})$, for $t = 1, \ldots, T$, be a sequence of unnormalized densities[1] on the measurable space $(\mathsf{X}^t, \mathcal{X}^t)$, parameterized by $\theta \in \Theta$. Let $\bar{\gamma}_{\theta,t}(x_{1:t})$ be the corresponding normalized probability densities:

$$\bar{\gamma}_{\theta,t}(x_{1:t}) = \frac{\gamma_{\theta,t}(x_{1:t})}{Z_{\theta,t}},$$

where $Z_{\theta,t} = \int \gamma_{\theta,t}(x_{1:t})\, dx_{1:t}$ and where it is assumed that $Z_{\theta,t} > 0, \forall \theta \in \Theta$. For instance, in the (important) special case of an SSM we have $\bar{\gamma}_{\theta,t}(x_{1:t}) = p_\theta(x_{1:t} \mid y_{1:t})$, $\gamma_{\theta,t}(x_{1:t}) = p_\theta(x_{1:t}, y_{1:t})$, and $Z_{\theta,t} = p_\theta(y_{1:t})$. We discuss this special case in more detail in Section 5.

To make inference about the latent variables $x_{1:T}$, as well as to enable learning of the model parameter θ, a useful approach is to construct a Monte Carlo algorithm to draw samples from $\bar{\gamma}_{\theta,T}(x_{1:T})$. The sequential nature of the problem suggests the use of SMC methods; in particular, particle filters (PFs); see, e.g., Doucet and Johansen (2011); Del Moral et al. (2006); Pitt and Shephard (1999).

We start by reviewing a standard SMC sampler, which will be used to construct the PGAS algorithm in the consecutive section. We will refer to the index variable t as time, but in general it might not have any temporal meaning. Let $\{x^i_{1:t-1}, w^i_{t-1}\}_{i=1}^N$ be a weighted particle system targeting $\bar{\gamma}_{\theta,t-1}(x_{1:t-1})$. That is, the weighted particles define an empirical point-mass approximation of the target distribution given by

$$\widehat{\gamma}^N_{\theta,t-1}(dx_{1:t-1}) = \sum_{i=1}^N \frac{w^i_{t-1}}{\sum_l w^l_{t-1}} \delta_{x^i_{1:t-1}}(dx_{1:t-1}).$$

1. The dominating measure is denoted simply as $dx_{1:t}$.

This particle system is propagated to time t by sampling $\{a_t^i, x_t^i\}_{i=1}^N$ independently, conditionally on the particles generated up to time $t-1$, from a proposal kernel,

$$M_{\theta,t}(a_t, x_t) = \frac{w_{t-1}^{a_t}}{\sum_l w_{t-1}^l} r_{\theta,t}(x_t \mid x_{1:t-1}^{a_t}). \tag{1}$$

Note that $M_{\theta,t}$ depends on the complete particle system up to time $t-1$, $\{x_{1:t-1}^i, w_{t-1}^i\}_{i=1}^N$, but for notational convenience we shall not make that dependence explicit. Here, a_t^i is the index of the ancestor particle of x_t^i. In this formulation, the resampling step is implicit and corresponds to sampling these *ancestor indices*. When we write $x_{1:t}^i$ we refer to the ancestral path of particle x_t^i. That is, the particle trajectory is defined recursively as

$$x_{1:t}^i = (x_{1:t-1}^{a_t^i}, x_t^i).$$

Once we have generated N ancestor indices and particles from the proposal kernel (1), the particles are weighted according to $w_t^i = W_{\theta,t}(x_{1:t}^i)$ where the weight function is given by

$$W_{\theta,t}(x_{1:t}) = \frac{\gamma_{\theta,t}(x_{1:t})}{\gamma_{\theta,t-1}(x_{1:t-1}) r_{\theta,t}(x_t \mid x_{1:t-1})}, \tag{2}$$

for $t \geq 2$. The procedure is initialized by sampling from a proposal density $x_1^i \sim r_{\theta,1}(x_1)$ and assigning importance weights $w_1^i = W_{\theta,1}(x_1^i)$ with $W_{\theta,1}(x_1) = \gamma_{\theta,1}(x_1)/r_{\theta,1}(x_1)$. The SMC sampler is summarized in Algorithm 1.

Algorithm 1 Sequential Monte Carlo (each step is for $i = 1, \ldots, N$)

1: Draw $x_1^i \sim r_{\theta,1}(x_1)$.
2: Set $w_1^i = W_{\theta,1}(x_1^i)$.
3: **for** $t = 2$ **to** T **do**
4: Draw $\{a_t^i, x_t^i\} \sim M_{\theta,t}(a_t, x_t)$.
5: Set $x_{1:t}^i = (x_{1:t-1}^{a_t^i}, x_t^i)$.
6: Set $w_t^i = W_{\theta,t}(x_{1:t}^i)$.
7: **end for**

It is interesting to note that the joint law of all the random variables generated by Algorithm 1 can be written down explicitly. Let

$$\mathbf{x}_t = \{x_t^1, \ldots, x_t^N\} \qquad \text{and} \qquad \mathbf{a}_t = \{a_t^1, \ldots, a_t^N\},$$

refer to all the particles and ancestor indices, respectively, generated at time t of the algorithm. It follows that the SMC sampler generates a collection of random variables $\{\mathbf{x}_{1:T}, \mathbf{a}_{2:T}\} \in \mathsf{X}^{NT} \times \{1, \ldots, N\}^{N(T-1)}$. Furthermore, $\{a_t^i, x_t^i\}_{i=1}^N$ are drawn independently (conditionally on the particle system generated up to time $t-1$) from the proposal kernel $M_{\theta,t}$, and similarly at time $t = 1$. Hence, the joint probability density function (with respect to a natural product of dx and counting measure) of these variables is given by

$$\psi_\theta(\mathbf{x}_{1:T}, \mathbf{a}_{2:T}) \triangleq \prod_{i=1}^N r_{\theta,1}(x_1^i) \prod_{t=2}^T \prod_{i=1}^N M_{\theta,t}(a_t^i, x_t^i).$$

3. The PGAS Kernel

We now turn to the construction of PGAS, a class of Markov kernels on the space of trajectories $(\mathsf{X}^T, \mathcal{X}^T)$. We will provide an algorithm for generating samples from these Markov kernels, which are thus defined implicitly by the algorithm.

3.1 Particle Gibbs

Before stating the PGAS algorithm, we review the main ideas of the PG algorithm of Andrieu et al. (2010) and we then turn to our proposed modification of this algorithm via the introduction of an *ancestor sampling* step.

PG is based on an SMC sampler, akin to a standard PF, but with the difference that one particle trajectory is specified *a priori*. This path, denoted as $x'_{1:T} = (x'_1, \ldots, x'_T)$, serves as a reference trajectory. Informally, it can be thought of as guiding the simulated particles to a relevant region of the state space. After a complete pass of the SMC algorithm, a trajectory $x^\star_{1:T}$ is sampled from among the particle trajectories. That is, we draw $x^\star_{1:T}$ with $\mathbb{P}(x^\star_{1:T} = x^i_{1:T}) \propto w^i_T$. This procedure thus maps $x'_{1:T}$ to a probability distribution on $\mathcal{X}^T$, implicitly defining a Markov kernel on $(\mathsf{X}^T, \mathcal{X}^T)$.

In a standard PF, the samples $\{a^i_t, x^i_t\}$ are drawn independently from the proposal kernel (1) for $i = 1, \ldots, N$. When sampling from the PG kernel, however, we condition on the event that the reference trajectory $x'_{1:T}$ is retained throughout the sampling procedure. To accomplish this, we sample according to (1) only for $i = 1, \ldots, N-1$. The Nth particle and its ancestor index are then set deterministically as $x^N_t = x'_t$ and $a^N_t = N$, respectively. This implies that after a complete pass of the algorithm, the Nth particle path coincides with the reference trajectory, i.e., $x^N_{1:T} = x'_{1:T}$.

The fact that $x'_{1:T}$ is used as a reference trajectory in the SMC sampler implies an invariance property of the PG kernel which is of key relevance. More precisely, as shown by Andrieu et al. (2010, Theorem 5), for any number of particles $N \geq 1$ and for any $\theta \in \Theta$, the PG kernel leaves the exact target distribution $\bar{\gamma}_{\theta,T}$ invariant. We return to this invariance property below, when it is shown to hold also for the proposed PGAS kernel.

3.2 Ancestor Sampling

As noted above, the PG algorithm keeps the reference trajectory $x'_{1:T}$ intact throughout the sampling procedure. While this results in a Markov kernel which leaves $\bar{\gamma}_{\theta,T}$ invariant, it has been recognized that the mixing properties of this kernel can be very poor due to path degeneracy (Lindsten and Schön, 2013; Chopin and Singh, 2014).

To address this fundamental problem we now turn to our new procedure, PGAS. The idea is to sample a new value for the index variable a^N_t in an *ancestor sampling* step. While this is a small modification of the algorithm, the improvement in mixing can be quite considerable; see Section 3.3 and the numerical evaluation in Section 7. The AS step is implemented as follows.

At time $t \geq 2$, we consider the part of the reference trajectory $x'_{t:T}$ ranging from the current time t to the final time point T. The task is to artificially assign a history to this partial path. This is done by connecting $x'_{t:T}$ to one of the particles $\{x^i_{1:t-1}\}_{i=1}^N$. Recall that the ancestry of a particle is encoded via the corresponding ancestor index. Hence, we can

connect the partial reference path to one of the particles $\{x^i_{1:t-1}\}_{i=1}^N$ by assigning a value to the variable $a^N_t \in \{1, \ldots, N\}$. To do this, first we compute the weights

$$\widetilde{w}^i_{t-1|T} \triangleq w^i_{t-1} \frac{\gamma_{\theta,T}((x^i_{1:t-1}, x'_{t:T}))}{\gamma_{\theta,t-1}(x^i_{1:t-1})} \tag{3}$$

for $i = 1, \ldots, N$. Here, $(x^i_{1:t-1}, x'_{t:T})$ refers to the point in X^T formed by concatenating the two partial trajectories. Then, we sample a^N_t with $\mathbb{P}(a^N_t = i) \propto \widetilde{w}^i_{t-1|T}$. The expression above can be understood as an application of Bayes' theorem, where the importance weight w^i_{t-1} is the prior probability of the particle $x^i_{1:t-1}$ and the ratio between the target densities in (3) can be seen as the likelihood that $x'_{t:T}$ originated from $x^i_{1:t-1}$. A formal argument for why (3) provides the correct AS distribution, in order to retain the invariance properties of the kernel, is detailed in the proof of Theorem 1 in Section 4.

The sampling procedure outlined above is summarized in Algorithm 2 and the class of PGAS kernels is formally defined below. Note that the only difference between PG and PGAS is on line 8 of Algorithm 2 (where, for PG, we would simply set $a^N_t = N$). However, as we shall see, the effect of this small modification on the mixing of the kernel is quite significant.

Definition 1 (PGAS kernels)**.** *For any $N \geq 1$ and any $\theta \in \Theta$, Algorithm 2 maps $x'_{1:T}$ stochastically into $x^\star_{1:T}$, thus implicitly defining a Markov kernel P^N_θ on $(\mathsf{X}^T, \mathcal{X}^T)$. The class of Markov kernels $\{P^N_\theta : \theta \in \Theta\}$, indexed by $N \geq 1$, is referred to as the PGAS class of kernels.*

Algorithm 2 PGAS Markov kernel

Input: Reference trajectory $x'_{1:T} \in \mathsf{X}^T$ and parameter $\theta \in \Theta$.
Output: Sample $x^\star_{1:T} \sim P^N_\theta(x'_{1:T}, \cdot)$ from the PGAS Markov kernel.

1: Draw $x^i_1 \sim r_{\theta,1}(x_1)$ for $i = 1, \ldots, N-1$.
2: Set $x^N_1 = x'_1$.
3: Set $w^i_1 = W_{\theta,1}(x^i_1)$ for $i = 1, \ldots, N$.
4: **for** $t = 2$ **to** T **do**
5: Draw $\{a^i_t, x^i_t\} \sim M_{\theta,t}(a_t, x_t)$ for $i = 1, \ldots, N-1$.
6: Set $x^N_t = x'_t$.
7: Compute $\{\widetilde{w}^i_{t-1|T}\}_{i=1}^N$ according to (3).
8: Draw a^N_t with $\mathbb{P}(a^N_t = i) \propto \widetilde{w}^i_{t-1|T}$.
9: Set $x^i_{1:t} = (x^{a^i_t}_{1:t-1}, x^i_t)$ for $i = 1, \ldots, N$.
10: Set $w^i_t = W_{\theta,t}(x^i_{1:t})$ for $i = 1, \ldots, N$.
11: **end for**
12: Draw k with $\mathbb{P}(k = i) \propto w^i_T$.
13: **return** $x^\star_{1:T} = x^k_{1:T}$.

3.3 The Effect of Path Degeneracy on PG and on PGAS

We have argued that AS can considerably improve the mixing of PG. To illustrate this effect and to provide an explanation of its cause, we consider a simple numerical example. Further

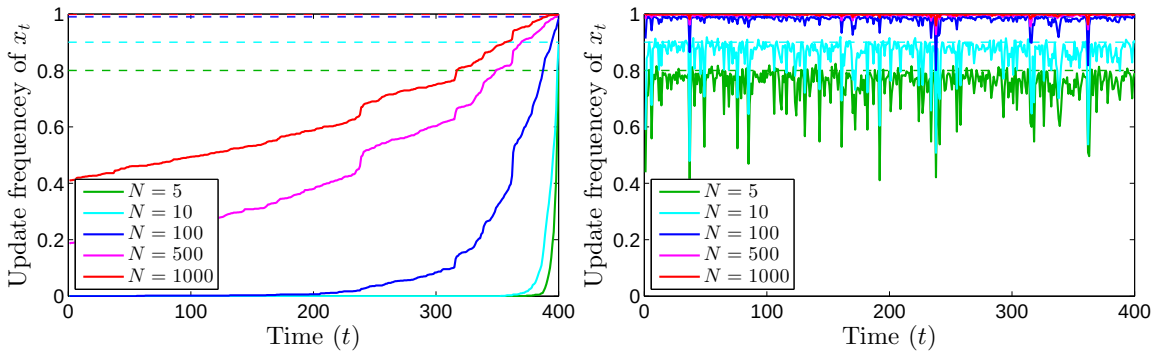


Figure 1: Update rates for x_t versus $t \in \{1, \ldots, 400\}$ for PG (left) and for PGAS (right). The dashed lines correspond to the ideal rates $(N-1)/N$. (This figure is best viewed in color.)

empirical evaluation of PGAS is provided in Section 7. Consider the one-dimensional linear Gaussian state-space (LGSS) model,

$$\begin{aligned} x_{t+1} &= ax_t + v_t, & v_t &\sim \mathcal{N}(0, \sigma_v^2), \\ y_t &= x_t + e_t & e_t &\sim \mathcal{N}(0, \sigma_e^2), \end{aligned}$$

where the state process $\{x_t\}_{t\geq 1}$ is latent and observations are made only via the measurement process $\{y_t\}_{t\geq 1}$. For simplicity, the parameters $\theta = (a, \sigma_v, \sigma_e) = (0.9, 0.32, 1)$ are assumed to be known. A batch of $T = 400$ observations are simulated from the system. Given these, we seek the joint smoothing density $p(x_{1:T} \mid y_{1:T})$. To generate samples from this density we employ both PG and PGAS with varying number of particles ranging from $N = 5$ to $N = 1\,000$. We simulate sample paths of length $1\,000$ for each algorithm. To compare the mixing, we look at the update rate of x_t versus t, which is defined as the proportion of iterations where x_t changes value. The results are reported in Figure 1, which reveals that AS significantly increases the probability of updating x_t for t far from T.

The poor update rates for PG is a manifestation of the well-known path degeneracy problem of SMC samplers (see, e.g., Doucet and Johansen 2011). Consider the process of sampling from the PG kernel for a fixed reference trajectory $x'_{1:T}$. A particle system generated by the PG algorithm (corresponding to Algorithm 2, but with line 8 replaced with $a_t^N = N$) is shown in Figure 2 (left). For clarity of illustration, we have used a small number of particles and time steps, $N = 20$ and $T = 50$, respectively. By construction, the reference trajectory (shown by a thick blue line) is retained throughout the sampling procedure. As a consequence, the particle system degenerates toward this trajectory which implies that $x^\star_{1:T}$ (shown as a red line) to a large extent will be identical to $x'_{1:T}$.

What is, perhaps, more surprising is that PGAS is so much more insensitive to the degeneracy issue. To understand why this is the case, we analyze the procedure for sampling from the PGAS kernel $P_\theta^N(x'_{1:T}, \cdot)$ for the same reference trajectory $x'_{1:T}$ as above. The particle system generated by Algorithm 2 (with AS) is shown in Figure 2 (right). The thick

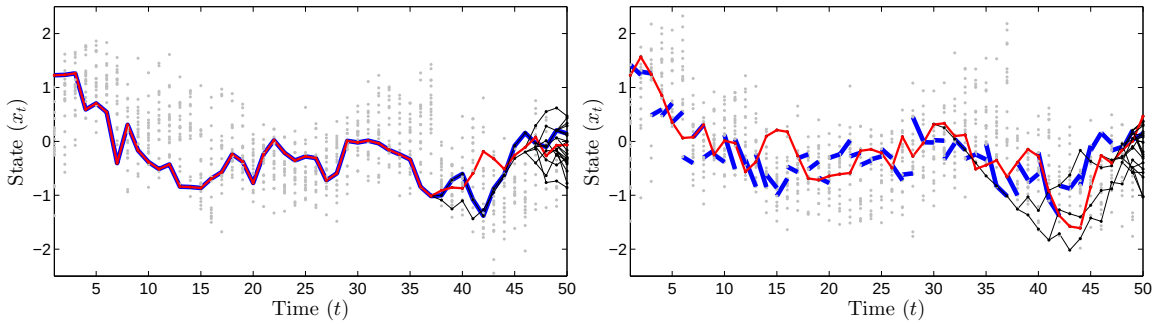


Figure 2: Particle systems generated by the PG algorithm (left) and by the PGAS algorithm (right), for the same reference trajectory $x'_{1:T}$ (shown as a thick blue line in the left panel, partly underneath the red line). The gray dots show the particle positions and the thin black lines show the ancestral dependencies of the particles. The extracted trajectory $x^\star_{1:T}$ is illustrated with a red line. In the right panel, AS has the effect of breaking the reference trajectory into pieces, causing the particle system to degenerate toward something different than $x'_{1:T}$. (This figure is best viewed in color.)

blue lines are again used to illustrate the reference particles, but now with updated ancestor indices. That is, the blue line segments are drawn between $x_{t-1}^{a_t^N}$ and x'_t for $t \geq 2$. It can be seen that the effect of AS is that, informally, the reference trajectory is broken into pieces. It is worth pointing out that the particle system still collapses; AS does not prevent path degeneracy. However, it causes the particle system to degenerate toward something different than the reference trajectory. As a consequence, $x^\star_{1:T}$ (shown as a red line in the figure) will with high probability be substantially different from $x'_{1:T}$, enabling high update rates and thereby much faster mixing.

4. Theoretical Justification

In this section we investigate the invariance and ergodicity properties of the PGAS kernel.

4.1 Stationary Distribution

We begin by stating a theorem, whose proof is provided later in this section, which shows that the invariance property of PG is not violated by the AS step.

Theorem 1. *For any $N \geq 1$ and $\theta \in \Theta$, the PGAS kernel P_θ^N leaves $\bar{\gamma}_{\theta,T}$ invariant:*

$$\bar{\gamma}_{\theta,T}(B) = \int P_\theta^N(x'_{1:T}, B)\bar{\gamma}_{\theta,T}(dx'_{1:T}), \qquad \forall B \in \mathcal{X}^T.$$

An apparent difficulty in establishing this result is that it is not possible to write down a simple, closed-form expression for P_θ^N. In fact, the PGAS kernel is given by

$$P_\theta^N(x'_{1:T}, B) = \mathbb{E}_{\theta, x'_{1:T}}\left[\mathbb{1}_B(x_{1:T}^k)\right], \tag{4}$$

where $\mathbb{1}_B$ is the indicator function for the set $B \in \mathcal{X}^T$ and where $\mathbb{E}_{\theta,x'_{1:T}}$ denotes expectation with respect to all the random variables generated by Algorithm 2, i.e., all the particles $\mathbf{x}_{1:T}$ and ancestor indices $\mathbf{a}_{2:T}$, as well as the index k. Computing this expectation is not possible in general. Instead of working directly with (4), however, we can adopt the strategy employed by Andrieu et al. (2010). That is, we treat all the random variables generated by Algorithm 2, $\{\mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k\}$, as auxiliary variables, thus avoiding an intractable integration. In the following, it is convenient to view x_t^N as a random variable with distribution $\delta_{x'_t}$.

Recall that the particle trajectory $x_{1:T}^k$ is the ancestral path of the particle x_T^k. That is, we can write

$$x_{1:T}^k = x_{1:T}^{b_{1:T}} \triangleq (x_1^{b_1}, \ldots, x_T^{b_T}),$$

where the indices $b_{1:T}$ are given recursively by the ancestor indices: $b_T = k$ and $b_t = a_{t+1}^{b_{t+1}}$. Let $\Omega \triangleq \mathsf{X}^{NT} \times \{1, \ldots, N\}^{N(T-1)+1}$ be the space of all random variables generated by Algorithm 2. Following Andrieu et al. (2010), we then define a probability density function $\phi_\theta : \Omega \mapsto \mathbb{R}$ as follows:

$$\begin{aligned}\phi_\theta(\mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k) &= \phi_\theta(x_{1:T}^{b_{1:T}}, b_{1:T})\phi_\theta(\mathbf{x}_{1:T}^{-b_{1:T}}, \mathbf{a}_{2:T}^{-b_{2:T}} \mid x_{1:T}^{b_{1:T}}, b_{1:T}) \\ &\triangleq \underbrace{\frac{\bar{\gamma}_{\theta,T}(x_{1:T}^{b_{1:T}})}{N^T}}_{\text{marginal}} \underbrace{\prod_{\substack{i=1 \\ i\neq b_1}}^{N} r_{\theta,1}(x_1^i) \prod_{t=2}^{T} \prod_{\substack{i=1 \\ i\neq b_t}}^{N} M_{\theta,t}(a_t^i, x_t^i)}_{\text{conditional}},\end{aligned} \tag{5}$$

where we have introduced the notation

$$\mathbf{x}_t^{-i} = \{x_t^1, \ldots, x_t^{i-1}, x_t^{i+1}, \ldots, x_t^N\}, \qquad \mathbf{x}_{1:T}^{-b_{1:T}} = \{\mathbf{x}_1^{-b_1}, \ldots, \mathbf{x}_T^{-b_T}\},$$

and similarly for the ancestor indices. By construction, ϕ_θ is nonnegative and integrates to one, i.e., ϕ_θ is indeed a probability density function on Ω. We refer to this density as the *extended target density*.

The factorization into a marginal and a conditional density is intended to reveal some of the structure inherent in the extended target density. In particular, the marginal density of the variables $\{x_{1:T}^{b_{1:T}}, b_{1:T}\}$ is defined to be equal to the original target density $\bar{\gamma}_{\theta,T}(x_{1:T}^{b_{1:T}})$, up to a factor N^{-T} corresponding to a uniform distribution over the index variables $b_{1:T}$. This has the important implication that if $\{\mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k\}$ are distributed according to ϕ_θ, then, by construction, the marginal distribution of $x_{1:T}^{b_{1:T}}$ is $\bar{\gamma}_{\theta,T}$.

By constructing an MCMC kernel with invariant distribution ϕ_θ, we will thus obtain a kernel with invariant distribution $\bar{\gamma}_{\theta,T}$ (the PGAS kernel) as a byproduct. To prove Theorem 1 we will reinterpret all the steps of the PGAS algorithm as partially collapsed Gibbs steps for ϕ_θ. The meaning of partial collapsing will be made precise in the proof

of Lemma 2 below, but basically it refers to the process of marginalizing out some of the variables of the model in the individual steps of the Gibbs sampler. This is done in such a way that it does not violate the invariance property of the Gibbs kernel, i.e., each such Gibbs step will leave the extended target distribution invariant. As a consequence, the invariance property of the PGAS kernel follows. First we show that the PGAS algorithm in fact implements the following sequence of partially collapsed Gibbs steps for ϕ_θ.

Procedure 1 (Instrumental reformulation of PGAS)**.** *Given* $x'^{,b'_{1:T}}_{1:T} \in \mathsf{X}^T$ *and* $b'_{1:T} \in \{1, \ldots, N\}^T$*:*

(i) *Draw* $\mathbf{x}_1^{-b'_1} \sim \phi_\theta(\,\cdot \mid x'^{,b'_{1:T}}_{1:T}, b'_{1:T})$ *and, for* $t = 2$ *to* T*, draw:*

$$\{\mathbf{x}_t^{-b_t}, \mathbf{a}_t^{-b_t}\} \sim \phi_\theta(\,\cdot \mid \mathbf{x}_{1:t-1}^{-b'_{1:t-1}}, \mathbf{a}_{2:t-1}, x'^{,b'_{1:T}}_{1:T}, b'_{t-1:T}),$$
$$a_t^{b_t} \sim \phi_\theta(\,\cdot \mid \mathbf{x}_{1:t-1}^{-b'_{1:t-1}}, \mathbf{a}_{2:t-1}, x'^{,b'_{1:T}}_{1:T}, b'_{t:T}),$$

(ii) *Draw* $k \sim \phi_\theta(\,\cdot \mid \mathbf{x}_{1:T}^{-b_{1:T}}, \mathbf{a}_{2:T}, x'^{,b'_{1:T}}_{1:T})$.

Lemma 1. *Algorithm 2 is equivalent to the partially collapsed Gibbs sampler of Procedure 1, conditionally on* $x'^{,b'_{1:T}}_{1:T} = x'_{1:T}$ *and* $b'_{1:T} = (N, \ldots, N)$.

Proof. From (5) we have, by construction,

$$\phi_\theta(\mathbf{x}_{1:T}^{-b_{1:T}}, \mathbf{a}_{2:T}^{-b_{2:T}} \mid x_{1:T}^{b_{1:T}}, b_{1:T}) = \prod_{\substack{i=1\\ i\neq b_1}}^{N} r_{\theta,1}(x_1^i) \prod_{t=2}^{T} \prod_{\substack{i=1\\ i\neq b_t}}^{N} M_{\theta,t}(a_t^i, x_t^i).$$

By marginalizing this expression over $\{\mathbf{x}_{t+1:T}^{-b_{t+1:T}}, \mathbf{a}_{t+1:T}^{-b_{t+1:T}}\}$ we get

$$\phi_\theta(\mathbf{x}_{1:t}^{-b_{1:t}}, \mathbf{a}_{2:t}^{-b_{2:t}} \mid x_{1:T}^{b_{1:T}}, b_{1:T}) = \prod_{\substack{i=1\\ i\neq b_1}}^{N} r_{\theta,1}(x_1^i) \prod_{s=2}^{t} \prod_{\substack{i=1\\ i\neq b_s}}^{N} M_{\theta,s}(a_s^i, x_s^i),$$

It follows that

$$\phi_\theta(\mathbf{x}_1^{-b_1} \mid x_{1:T}^{b_{1:T}}, b_{1:T}) = \prod_{\substack{i=1\\ i\neq b_1}}^{N} r_{\theta,1}(x_1^i), \tag{6a}$$

and, for $t = 2, \ldots, T$,

$$\begin{aligned}\phi_\theta(\mathbf{x}_t^{-b_t}, \mathbf{a}_t^{-b_t} \mid \mathbf{x}_{1:t-1}^{-b_{1:t-1}}, \mathbf{a}_{2:t-1}^{-b_{2:t-1}}, x_{1:T}^{b_{1:T}}, b_{1:T}) \\ = \frac{\phi_\theta(\mathbf{x}_{1:t}^{-b_{1:t}}, \mathbf{a}_{2:t}^{-b_{2:t}} \mid x_{1:T}^{b_{1:T}}, b_{1:T})}{\phi_\theta(\mathbf{x}_{1:t-1}^{-b_{1:t-1}}, \mathbf{a}_{2:t-1}^{-b_{2:t-1}} \mid x_{1:T}^{b_{1:T}}, b_{1:T})} = \prod_{\substack{i=1\\ i\neq b_t}}^{N} M_{\theta,t}(a_t^i, x_t^i).\end{aligned} \tag{6b}$$

Hence, we can sample from (6a) and (6b) by drawing $x_1^i \sim r_{\theta,1}(\cdot)$ for $i \in \{1, \ldots, N\} \setminus b_1$ and $\{a_t^i, x_t^i\} \sim M_{\theta,t}(\cdot)$ for $i \in \{1, \ldots, N\} \setminus b_t$, respectively. Consequently, with the choice

$b_t = N$ for $t = 1, \ldots, T$, the initialization at line 1 and the particle propagation at line 5 of Algorithm 2 correspond to sampling from (6a) and (6b), respectively.

Next, we consider the AS step. Recall that $a_t^{b_t}$ identifies to b_{t-1}. We can thus write

$$\begin{aligned}
\phi_\theta(a_t^{b_t} \mid \mathbf{x}_{1:t-1}, \mathbf{a}_{2:t-1}, x_{t:T}^{b_{t:T}}, b_{t:T}) &\propto \phi_\theta(\mathbf{x}_{1:t-1}, \mathbf{a}_{2:t-1}, x_{t:T}^{b_{t:T}}, b_{t-1:T}) \\
&= \phi_\theta(x_{1:T}^{b_{1:T}}, b_{1:T}) \phi_\theta(\mathbf{x}_{1:t-1}^{-b_{1:t-1}}, \mathbf{a}_{2:t-1}^{-b_{2:t-1}} \mid x_{1:T}^{b_{1:T}}, b_{1:T}) \\
&= \frac{\gamma_{\theta,T}(x_{1:T}^{b_{1:T}})}{\gamma_{\theta,t-1}(x_{1:t-1}^{b_{1:t-1}})} \frac{\gamma_{\theta,t-1}(x_{1:t-1}^{b_{1:t-1}})}{Z_{\theta,T} N^T} \prod_{\substack{i=1 \\ i \neq b_1}}^{N} r_{\theta,1}(x_1^i) \prod_{s=2}^{t-1} \prod_{\substack{i=1 \\ i \neq b_s}}^{N} M_{\theta,s}(a_s^i, x_s^i). \qquad (7)
\end{aligned}$$

To simplify this expression, note first that we can write

$$\gamma_{\theta,t-1}(x_{1:t-1}) = \gamma_{\theta,1}(x_1) \prod_{s=2}^{t-1} \frac{\gamma_{\theta,s}(x_{1:s})}{\gamma_{\theta,s-1}(x_{1:s-1})}.$$

By using the definition of the weight function (2), this expression can be expanded according to

$$\gamma_{\theta,t-1}(x_{1:t-1}) = W_{\theta,1}(x_1) r_{\theta,1}(x_1) \prod_{s=2}^{t-1} W_{\theta,s}(x_{1:s}) r_{\theta,s}(x_s \mid x_{1:s-1}).$$

Plugging the trajectory $x_{1:t-1}^{b_{1:t-1}}$ into the above expression, we get

$$\begin{aligned}
\gamma_{\theta,t-1}(x_{1:t-1}^{b_{1:t-1}}) &= w_1^{b_1} r_{\theta,1}(x_1^{b_1}) \prod_{s=2}^{t-1} w_s^{b_s} r_{\theta,s}(x_s^{b_s} \mid x_{1:s-1}^{b_{1:s-1}}) \\
&= \left(\prod_{s=1}^{t-1} \sum_{l=1}^{N} w_s^l \right) \frac{w_1^{b_1}}{\sum_l w_1^l} r_{\theta,1}(x_1^{b_1}) \prod_{s=2}^{t-1} \frac{w_s^{b_s}}{\sum_l w_s^l} r_{\theta,s}(x_s^{b_s} \mid x_{1:s-1}^{b_{1:s-1}}) \\
&= \frac{w_{t-1}^{b_{t-1}}}{\sum_l w_{t-1}^l} \left(\prod_{s=1}^{t-1} \sum_{l=1}^{N} w_s^l \right) r_{\theta,1}(x_1^{b_1}) \prod_{s=2}^{t-1} M_{\theta,s}(a_s^{b_s}, x_s^{b_s}). \qquad (8)
\end{aligned}$$

Expanding the numerator in (7) according to (8) results in

$$\begin{aligned}
&\phi_\theta(a_t^{b_t} \mid \mathbf{x}_{1:t-1}, \mathbf{a}_{2:t-1}, x_{t:T}^{b_{t:T}}, b_{t:T}) \\
&\quad \propto \frac{\gamma_{\theta,T}(x_{1:T}^{b_{1:T}})}{\gamma_{\theta,t-1}(x_{1:t-1}^{b_{1:t-1}})} \frac{w_{t-1}^{b_{t-1}}}{\sum_l w_{t-1}^l} \frac{\left(\prod_{s=1}^{t-1} \sum_l w_s^l \right)}{Z_{\theta,T} N^T} \prod_{i=1}^{N} r_{\theta,1}(x_1^i) \prod_{s=2}^{t-1} \prod_{i=1}^{N} M_{\theta,s}(a_s^i, x_s^i) \\
&\quad \propto w_{t-1}^{b_{t-1}} \frac{\gamma_{\theta,T}((x_{1:t-1}^{b_{1:t-1}}, x_{t:T}^{b_{t:T}}))}{\gamma_{\theta,t-1}(x_{1:t-1}^{b_{1:t-1}})}. \qquad (9)
\end{aligned}$$

Consequently, with $b_t = N$ and $x_{t:T}^{b_{t:T}} = x'_{t:T}$, sampling from (9) corresponds to the AS step of line 8 of Algorithm 2. Finally, analogously to (9), it follows that $\phi_\theta(k \mid \mathbf{x}_{1:T}, \mathbf{a}_{2:T}) \propto w_T^k$, which corresponds to line 12 of Algorithm 2. ■

Next, we show that Procedure 1 leaves ϕ_θ invariant. This is done by concluding that the procedure is a properly collapsed Gibbs sampler; see Dyk and Park (2008). Marginalization, or collapsing, is commonly used within Gibbs sampling to improve the mixing and/or to simplify the sampling procedure. However, it is crucial that the collapsing is carried out in the correct order to respect the dependencies between the variables of the model.

Lemma 2. *The Gibbs sampler of Procedure 1 is properly collapsed and thus leaves ϕ_θ invariant.*

Proof. Consider the following sequence of *complete* Gibbs steps:

(i) Draw $\{\mathbf{x}_1^{-b'_1}, \underline{\mathbf{x}}_{2:T}^{-b'_{2:T}}, \underline{\mathbf{a}}_{2:T}^{-b'_{2:T}}\} \sim \phi_\theta(\,\cdot \mid x_{1:T}^{\prime,b'_{1:T}}, b'_{1:T})$ and, for $t = 2$ to T, draw:

$$\{\mathbf{x}_t^{-b_t}, \mathbf{a}_t, \underline{\mathbf{x}}_{t+1:T}^{-b'_{t+1:T}}, \underline{\mathbf{a}}_{t+1:T}^{-b'_{t+1:T}}\} \sim \phi_\theta(\,\cdot \mid \mathbf{x}_{1:t-1}^{-b'_{1:t-1}}, \mathbf{a}_{2:t-1}, x_{1:T}^{\prime,b'_{1:T}}, b'_{t:T}).$$

(ii) Draw $k \sim \phi_\theta(\,\cdot \mid \mathbf{x}_{1:T}^{-b'_{1:T}}, \mathbf{a}_{2:T}, x_{1:T}^{\prime,b'_{1:T}})$.

In the above, all the samples are drawn from conditionals under the full joint density $\phi_\theta(\mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k)$. Hence, it is clear that the above procedure will leave ϕ_θ invariant. Note that some of the variables above have been marked by an underline. It can be seen that these variables are in fact never conditioned upon in any subsequent step of the procedure. That is, the underlined variables are never used. Therefore, to obtain a valid sampler it is sufficient to sample all the non-underlined variables from their respective marginals. Furthermore, from (6b) it can be seen that $\{\mathbf{x}_t^{-b_t}, \mathbf{a}_t^{-b_t}\}$ are conditionally independent of $a_t^{b_t}$, i.e., it follows that the complete Gibbs sweep above is equivalent to the partially collapsed Gibbs sweep of Procedure 1. Hence, the Gibbs sampler is properly collapsed and it will therefore leave ϕ_θ invariant. ■

Proof (Theorem 1). Let $\mathcal{L}(d\mathbf{x}_{1:T}^{-b'_{1:T}}, d\mathbf{a}_{2:T}, dk \mid x'_{1:T}, b'_{1:T})$ denote the law of the random variables generated by Procedure 1, conditionally on $x_{1:T}^{\prime,b'_{1:T}} = x'_{1:T}$ and on $b'_{1:T}$. Using Lemma 2 and recalling that $\phi_\theta(x_{1:T}^{b_{1:T}}, b_{1:T}) = N^{-T}\bar{\gamma}_{\theta,T}(x_{1:T}^{b_{1:T}})$ we have

$$\begin{aligned}\bar{\gamma}_{\theta,T}(B) = \int \mathbb{1}_B(x_{1:T}^k)\mathcal{L}(d\mathbf{x}_{1:T}^{-b'_{1:T}}, d\mathbf{a}_{2:T}, dk \mid x'_{1:T}, b'_{1:T}) \\ \times \delta_{x'_1}(dx_1^{b'_1}) \cdots \delta_{x'_T}(dx_T^{b'_T}) \frac{\bar{\gamma}_{\theta,T}(x'_{1:T})}{N^T} dx'_{1:T} db'_{1:T}, \quad \forall B \in \mathcal{X}^T. \end{aligned} \tag{10}$$

By Lemma 1 we know that Algorithm 2, which implicitly defines P_θ^N, is equivalent to Procedure 1 conditionally on $x_{1:T}^{\prime,b'_{1:T}} = x'_{1:T}$ and $b'_{1:T} = (N, \ldots, N)$. That is to say,

$$\begin{aligned}P_\theta^N(x'_{1:T}, B) = \int \mathbb{1}_B(x_{1:T}^k)\mathcal{L}(d\mathbf{x}_{1:T}^{-(N,\ldots,N)}, d\mathbf{a}_{2:T}, dk \mid x'_{1:T}, (N, \ldots, N)) \\ \times \delta_{x'_1}(dx_1^N) \cdots \delta_{x'_T}(dx_T^N),\end{aligned}$$

However, the law of $x_{1:T}^\star$ in Algorithm 2 is invariant to permutations of the particle indices. That is, it does not matter if we place the reference particles on the Nth positions, or on

some other positions, when enumerating the particles.[2] This implies that for any $b'_{1:T} \in \{1, \ldots, N\}^T$,

$$P_\theta^N(x'_{1:T}, B) = \int \mathbb{1}_B(x^k_{1:T}) \mathcal{L}(d\mathbf{x}_{1:T}^{-b'_{1:T}}, d\mathbf{a}_{2:T}, dk \mid x'_{1:T}, b'_{1:T}) \delta_{x'_1}(dx_1^{b'_1}) \cdots \delta_{x'_T}(dx_T^{b'_T}). \quad (11)$$

Plugging (11) into (10) gives the desired result,

$$\bar{\gamma}_{\theta,T}(B) = \int P_\theta^N(x'_{1:T}, B) \bar{\gamma}_{\theta,T}(x'_{1:T}) \underbrace{\left(\sum_{b'_{1:T}} \frac{1}{N^T} \right)}_{=1} dx'_{1:T}, \qquad \forall B \in \mathcal{X}^T.$$

■

4.2 Ergodicity

To show ergodicity of the PGAS kernel we need to characterize the support of the target and the proposal densities. Let,

$$\mathcal{S}_{\theta,t} = \{x_{1:t} \in \mathsf{X}^t : \bar{\gamma}_{\theta,t}(x_{1:t}) > 0\},$$
$$\mathcal{Q}_{\theta,t} = \{x_{1:t} \in \mathsf{X}^t : r_{\theta,t}(x_t \mid x_{1:t-1}) \bar{\gamma}_{\theta,t-1}(x_{1:t-1}) > 0\},$$

with obvious modifications for $t = 1$. The following is a minimal assumption.

(A1) For any $\theta \in \Theta$ and $t \in \{1, \ldots, T\}$ we have $\mathcal{S}_t^\theta \subseteq \mathcal{Q}_t^\theta$.

Assumption (A1) basically states that the support of the proposal density should cover the support of the target density. Ergodicity of PG under Assumption (A1) has been established by Andrieu et al. (2010). The same argument can be applied also to PGAS.

Theorem 2 (Andrieu et al. (2010, Theorem 5))**.** *Assume (A1). Then, for any $N \geq 2$ and $\theta \in \Theta$, P_θ^N is $\bar{\gamma}_{\theta,T}$-irreducible and aperiodic. Consequently,*

$$\lim_{n \to \infty} \|(P_\theta^N)^n(x'_{1:T}, \cdot) - \bar{\gamma}_{\theta,T}(\cdot)\|_{\mathrm{TV}} = 0, \qquad \bar{\gamma}_{\theta,T}\text{-a.a. } x'_{1:T}.$$

To strengthen the ergodicity results for the PGAS kernel, we use a boundedness condition for the importance weights, given in assumption (A2) below. Such a condition is typical also in classical importance sampling and is, basically, a stronger version of assumption (A1).

(A2) For any $\theta \in \Theta$ and $t \in \{1, \ldots, T\}$, there exists a constant $\kappa_\theta < \infty$ such that $\|W_{\theta,t}\|_\infty \leq \kappa_\theta$.

Theorem 3. *Assume (A2). Then, for any $N \geq 2$ and $\theta \in \Theta$, P_θ^N is uniformly ergodic. That is, there exist constants $R_\theta < \infty$ and $\rho_\theta \in [0, 1)$ such that*

$$\|(P_\theta^N)^n(x'_{1:T}, \cdot) - \bar{\gamma}_{\theta,T}(\cdot)\|_{\mathrm{TV}} \leq R_\theta \rho_\theta^n, \qquad \forall x'_{1:T} \in \mathsf{X}^T.$$

2. A formal proof of this statement is given for the PG sampler by Chopin and Singh (2014). The same argument can be used also for PGAS.

Proof. We show that P_θ^N satisfies a Doeblin condition,

$$P_\theta^N(x'_{1:T}, B) \geq \varepsilon_\theta \bar{\gamma}_{\theta,T}(B), \qquad \forall x'_{1:T} \in \mathsf{X}^T, \forall B \in \mathcal{X}^T, \tag{12}$$

for some constant $\varepsilon_\theta > 0$. Uniform ergodicity then follows from Tierney (1994, Proposition 2). To prove (12) we use the representation of the PGAS kernel in (4),

$$\begin{aligned} P_\theta^N(x'_{1:T}, B) &= \mathbb{E}_{\theta, x'_{1:T}}\left[\mathbb{1}_B(x^k_{1:T})\right] = \sum_{j=1}^N \mathbb{E}_{\theta, x'_{1:T}}\left[\frac{w_T^j}{\sum_l w_T^l}\mathbb{1}_B(x^j_{1:T})\right] \\ &\geq \frac{1}{N\kappa_\theta}\sum_{j=1}^{N-1}\mathbb{E}_{\theta, x'_{1:T}}\left[w_T^j \mathbb{1}_B(x^j_{1:T})\right] = \frac{N-1}{N\kappa_\theta}\mathbb{E}_{\theta, x'_{1:T}}\left[W_{\theta,T}(x^1_{1:T})\mathbb{1}_B(x^1_{1:T})\right]. \end{aligned} \tag{13}$$

Here, the inequality follows from bounding the weights in the normalization by κ_θ and by simply discarding the Nth term of the sum (which is clearly nonnegative). The last equality follows from the fact that the particle trajectories $\{x^i_{1:T}\}_{i=1}^{N-1}$ are equally distributed under Algorithm 2. Let $h_{\theta,t} : \mathsf{X}^t \mapsto \mathbb{R}_+$ and consider

$$\begin{aligned} \mathbb{E}_{\theta, x'_{1:T}}\left[h_{\theta,t}(x^1_{1:t})\right] &= \mathbb{E}_{\theta, x'_{1:T}}\left[\mathbb{E}_{\theta, x'_{1:T}}\left[h_{\theta,t}(x^1_{1:t}) \mid \mathbf{x}_{1:t-1}, \mathbf{a}_{2:t-1}\right]\right] \\ &= \mathbb{E}_{\theta, x'_{1:T}}\left[\sum_{j=1}^N \int h_{\theta,t}((x^j_{1:t-1}, x_t))\frac{w^j_{t-1}}{\sum_l w^l_{t-1}} r_{\theta,t}(x_t \mid x^j_{1:t-1})\,dx_t\right] \\ &\geq \frac{N-1}{N\kappa_\theta}\mathbb{E}_{\theta, x'_{1:T}}\left[\int h_{\theta,t}((x^1_{1:t-1}, x_t)) W_{\theta,t-1}(x^1_{1:t-1}) r_{\theta,t}(x_t \mid x^1_{1:t-1})\,dx_t\right], \end{aligned} \tag{14}$$

where the inequality follows analogously to (13). Now, let

$$\begin{aligned} h_{\theta,T}(x_{1:T}) &= W_{\theta,T}(x_{1:T})\mathbb{1}_B(x_{1:T}), \\ h_{\theta,t-1}(x_{1:t-1}) &= \int h_{\theta,t}(x_{1:t}) W_{\theta,t-1}(x_{1:t-1}) r_{\theta,t}(x_t \mid x_{1:t-1})\,dx_t, \qquad t \leq T. \end{aligned}$$

Then, by iteratively making use of (14) and changing the order of integration, we can bound (13) according to

$$\begin{aligned} \left(\frac{N-1}{N\kappa_\theta}\right)^{-T} P_\theta^N(x'_{1:T}, B) &\geq \mathbb{E}_{\theta, x'_{1:T}}\left[h_{\theta,1}(x^1_1)\right] \\ &= \int W_{\theta,1}(x_1) r_{\theta,1}(x_1) \prod_{t=2}^T \left(W_{\theta,t}(x_{1:t}) r_{\theta,t}(x_t \mid x_{1:t-1})\right) \mathbb{1}_B(x_{1:T})\,dx_{1:T} \\ &= \int \gamma_{\theta,1}(x_1) \prod_{t=2}^T \left(\frac{\gamma_{\theta,t}(x_{1:t})}{\gamma_{\theta,t-1}(x_{1:t-1})}\right) \mathbb{1}_B(x_{1:T})\,dx_{1:T} \\ &= \int \gamma_{\theta,T}(x_{1:T})\mathbb{1}_B(x_{1:T})\,dx_{1:T} = Z_{\theta,T}\bar{\gamma}_{\theta,T}(B). \end{aligned}$$

With $N \geq 2$ and since $Z_{\theta,T} > 0$ the result follows. ■

5. PGAS for State-Space Models

SSMs comprise an important special case of the model class treated above. In this section, we illustrate how PGAS can be used for inference and learning of these models.

5.1 Sampling from the Joint Smoothing Distribution with PGAS

Consider the (possibly) nonlinear/non-Gaussian SSM

$$x_{t+1} \sim f_\theta(x_{t+1} \mid x_t), \tag{15a}$$
$$y_t \sim g_\theta(y_t \mid x_t), \tag{15b}$$

and $x_1 \sim \mu_\theta(x_1)$, where $\theta \in \Theta$ is a static parameter, x_t is the latent state and y_t is the observation at time t, respectively. Given a batch of measurements $y_{1:T}$, we wish to make inferences about θ and/or about the latent states $x_{1:T}$. In the subsequent section we will provide both a Bayesian and a frequentist learning algorithm based on the PGAS kernel. However, we start by discussing how to implement the PGAS algorithm for this specific model.

For an SSM the target distribution of interest is typically the joint smoothing distribution $p_\theta(x_{1:T} \mid y_{1:T})$. Consequently, since $p_\theta(x_{1:T} \mid y_{1:T}) \propto p_\theta(x_{1:T}, y_{1:T})$, the sequence of *unnormalized* target densities is given by

$$\gamma_{\theta,t}(x_{1:t}) = p_\theta(x_{1:t}, y_{1:t}), \qquad t = 1, \ldots, T. \tag{16}$$

As we have previously discussed, the process of sampling from the PGAS kernel is similar to running a PF. The only non-standard (and nontrivial) operation is the AS step. By plugging the specific choice of unnormalized target densities (16) into the general expression for the AS weights (3), we get

$$\widetilde{w}^i_{t-1|T} = w^i_{t-1} \frac{p_\theta((x^i_{1:t-1}, x'_{t:T}), y_{1:T})}{p(x^i_{1:t-1}, y_{1:t-1})} = w^i_{t-1} p_\theta(x'_{t:T}, y_{t:T} \mid x^i_{t-1}) \propto w^i_{t-1} f_\theta(x'_t \mid x^i_{t-1}). \tag{17}$$

This expression can be understood as an application of Bayes' theorem. Recall that we want to assign an ancestor at time $t-1$ to the reference particle x'_t. The importance weight w^i_{t-1} is the prior probability of the particle x^i_{t-1} and the factor $f_\theta(x'_t \mid x^i_{t-1})$ is the likelihood of moving from x^i_{t-1} to x'_t. The product of these two factors is thus proportional to the posterior probability that x'_t originated from x^i_{t-1}, which gives us the AS probability.

Expression (17) can also be recognized as the backward sampling weights in a backward simulator; see Godsill et al. (2004); Lindsten and Schön (2013). Consequently, the AS step corresponds to a one-step backward simulation, which highlights the close relationship between PGAS and PGBS for SSMs. The latter method is conceptually similar to PGAS, but it makes use of an explicit backward simulation pass; see Whiteley (2010); Whiteley et al. (2010) or Lindsten and Schön (2013, Section 5.4). We discuss this relationship in more detail in Appendix A. In particular, we show that PGAS and PGBS are in fact probabilistically equivalent under certain conditions when applied to SSMs. Note, however, that this equivalence *does not* hold in general for models outside the class of SSMs. In particular, for the class of non-Markovian models, discussed in the subsequent section, we have found that PGAS and PGBS have quite different properties.

For concreteness we provide a restatement of the PGAS algorithm, specifically for the case of SSMs, in Algorithm 3. To highlight the similarities between PGAS and a standard PF, we have chosen to present Algorithm 3 using a notation and nomenclature that is common in the particle filtering literature, but that differs slightly from our previous notation. However, we emphasize that Algorithm 3 is completely equivalent to Algorithm 2 when the target distributions are given by (16). Note that the computational cost of the AS step is $O(N)$ per time step, i.e., of the same order as the PF. Consequently, for an SSM, the computational complexity of PGAS is the same as for PG, in total $O(NT)$.

Algorithm 3 PGAS Markov kernel for the joint smoothing distribution $p_\theta(x_{1:T} \mid y_{1:T})$

Input: Reference trajectory $x'_{1:T} \in \mathsf{X}^T$ and parameter $\theta \in \Theta$.
Output: Sample $x^\star_{1:T} \sim P^N_\theta(x'_{1:T}, \cdot)$ from the PGAS Markov kernel.

1: Draw $x^i_1 \sim r_{\theta,1}(x_1 \mid y_1)$ for $i = 1, \ldots, N-1$.
2: Set $x^N_1 = x'_1$.
3: Set $w^i_1 = g_\theta(y_1 \mid x^i_1)\mu_\theta(x^i_1)/r_{\theta,1}(x^i_1 \mid y_1)$ for $i = 1, \ldots, N$.
4: **for** $t = 2$ **to** T **do**
   / Resampling and ancestor sampling */*
5: Generate $\{\widetilde{x}^i_{1:t-1}\}_{i=1}^{N-1}$ by sampling $N-1$ times with replacement from $\{x^i_{1:t-1}\}_{i=1}^N$ with probabilities proportional to the importance weights $\{w^i_{t-1}\}_{i=1}^N$.
6: Draw J with

$$\mathbb{P}(J = i) = \frac{w^i_{t-1} f_\theta(x'_t \mid x^i_{t-1})}{\sum_{l=1} w^l_{t-1} f_\theta(x'_t \mid x^l_{t-1})}, \qquad i = 1, \ldots, N$$

   and set $\widetilde{x}^N_{1:t-1} = x^J_{1:t-1}$.

   / Particle propagation */*
7: Simulate $x^i_t \sim r_{\theta,t}(x_t \mid \widetilde{x}^i_{t-1}, y_t)$ for $i = 1, \ldots, N-1$.
8: Set $x^N_t = x'_t$.
9: Set $x^i_{1:t} = (\widetilde{x}^i_{1:t-1}, x^i_t)$ for $i = 1, \ldots, N$.

   / Weighting */*
10: Set $w^i_t = g_\theta(y_t \mid x^i_t) f_\theta(x^i_t \mid \widetilde{x}^i_{t-1})/r_{\theta,t}(x^i_t \mid \widetilde{x}^i_{t-1}, y_t)$ for $i = 1, \ldots, N$.
11: **end for**
12: Draw k with $\mathbb{P}(k = i) \propto w^i_T$.
13: **return** $x^\star_{1:T} = x^k_{1:T}$.

5.2 Learning Algorithms for State-Space Models

We now turn to the problem of learning the model parameter θ in the SSM (15), given a batch of observations $y_{1:T}$. Consider first the Bayesian setting where a prior distribution $\pi(\theta)$ is assigned to θ. We seek the parameter posterior $p(\theta \mid y_{1:T})$ or, more generally, the joint state and parameter posterior $p(\theta, x_{1:T} \mid y_{1:T})$. Gibbs sampling can be used to simulate from this distribution by sampling the state variables $\{x_t\}$ one at a time and the parameters θ from their respective conditionals. However, it has been recognized that this can result in

Algorithm 4 PGAS for Bayesian learning of SSMs

1: Set $\theta[0]$ and $x_{1:T}[0]$ arbitrarily.
2: **for** $n \geq 1$ **do**
3: Draw $x_{1:T}[n] \sim P^N_{\theta[n-1]}(x_{1:T}[n-1], \cdot)$. */* By running Algorithm 3 */*
4: Draw $\theta[n] \sim p(\theta \mid x_{1:T}[n], y_{1:T})$.
5: **end for**

poor mixing, due to the often high autocorrelation of the state sequence. The PGAS kernel offers a different approach, namely to sample the complete state trajectory $x_{1:T}$ in one block. This can considerably improve the mixing of the sampler (de Jong and Shephard, 1995). Due to the invariance and ergodicity properties of the kernel (Theorems 1– 3), the validity of the Gibbs sampler is not violated. We summarize the procedure in Algorithm 4.

PGAS is also useful for maximum-likelihood-based learning of SSMs. A popular strategy for computing the maximum likelihood estimator

$$\widehat{\theta}_{\mathrm{ML}} = \arg\max_{\theta \in \Theta} \log p_\theta(y_{1:T})$$

is to use the expectation maximization (EM) algorithm (Dempster et al., 1977; McLachlan and Krishnan, 2008). EM is an iterative method, which maximizes $\log p_\theta(y_{1:T})$ by iteratively maximizing an auxiliary quantity: $\theta[n] = \arg\max_{\theta \in \Theta} Q(\theta, \theta[n-1])$, where

$$Q(\theta, \theta[n-1]) = \int \log p_\theta(x_{1:T}, y_{1:T}) p_{\theta[n-1]}(x_{1:T} \mid y_{1:T})\, dx_{1:T}.$$

When the above integral is intractable to compute, one can use a Monte Carlo approximation or a stochastic approximation of the intermediate quantity, leading to the MCEM (Wei and Tanner, 1990) and the SAEM (Delyon et al., 1999) algorithms, respectively. When the underlying Monte Carlo simulation is computationally involved, SAEM is particularly useful since it makes efficient use of the simulated values. The SAEM approximation of the auxiliary quantity is given by

$$\widehat{Q}_n(\theta) = (1 - \alpha_n)\widehat{Q}_{n-1}(\theta) + \alpha_n \log p_\theta(x_{1:T}[n], y_{1:T}), \tag{18}$$

where α_n is the step size and, in the vanilla form of SAEM, $x_{1:T}[n]$ is drawn from the joint smoothing density $p_{\theta[n-1]}(x_{1:T} \mid y_{1:T})$. In practice, the stochastic approximation update (18) is typically made on some sufficient statistic for the complete data log-likelihood; see Delyon et al. (1999) for details. While the joint smoothing density is intractable for a general nonlinear/non-Gaussian SSM, it has been recognized that it is sufficient to sample from a uniformly ergodic Markov kernel, leaving the joint smoothing distribution invariant (Benveniste et al., 1990; Andrieu et al., 2005). A practical approach is therefore to compute the auxiliary quantity according to the stochastic approximation (18), but where $x_{1:T}[n]$ is simulated from the PGAS kernel $P^N_{\theta[n-1]}(x_{1:T}[n-1], \cdot)$. This particle SAEM algorithm, previously presented by Lindsten (2013), is summarized in Algorithm 5.

6. Beyond State-Space Models

For SSMs, the Markovianity implies a simple expression for the AS weights, depending only of the one-step transition density according to (17). For models with more intricate

Algorithm 5 PGAS for frequentist learning of SSMs

1: Set $\theta[0]$ and $x_{1:T}[0]$ arbitrarily. Set $\widehat{Q}_0(\theta) \equiv 0$.
2: **for** $n \geq 1$ **do**
3: Draw $x_{1:T}[n] \sim P^N_{\theta[n-1]}(x_{1:T}[n-1], \cdot)$. */* By running Algorithm 3 */*
4: Compute $\widehat{Q}_n(\theta)$ according to (18).
5: Compute $\theta[n] = \arg\max_{\theta \in \Theta} \widehat{Q}_n(\theta)$.
6: **if** convergence criterion is met **then**
7: **return** $\theta[n]$.
8: **end if**
9: **end for**

dependencies between the latent variables, however, this is not the case and the general expression (3) needs to be used. In this section we consider the computational aspects of the AS step, first in a very general setting and then specifically for the class of non-Markovian latent variable models.

6.1 Modifications and Mixed Strategies

The interpretation of the PGAS algorithm as a standard MCMC sampler on an extended space opens up for straightforward modifications of the algorithm while still making sure that it retains its desirable theoretical properties. In particular, for models where the computation of the AS weights in (3) is costly—that is, when evaluating the unnormalized joint target density $\gamma_{\theta,T}$ is computationally involved—it can be beneficial to modify the AS step to reduce the overall computational cost of the algorithm. Let

$$\rho(i) = \frac{\widetilde{w}^i_{t-1|T}}{\sum_{l=1}^N \widetilde{w}^l_{t-1|T}}, \qquad i = 1, \ldots, N, \tag{19}$$

denote the law of the ancestor index a^N_t, sampled at line 8 of Algorithm 2. From Lemma 1, we know that this step of the algorithm in fact corresponds to a Gibbs step for the extended target distribution (5). To retain the correct limiting distribution of the PGAS kernel, it is therefore sufficient that a^N_t is sampled from a Markov kernel leaving (19) invariant (resulting in a standard combination of MCMC kernels; see, e.g., Tierney 1994).

A simple modification is to carry out the AS step only for a fraction of the time steps. For instance, we can generate the ancestor index a^N_t according to:

$$\begin{cases} \text{With probability } 1-\eta, \text{ set } a^N_t = N, \\ \text{Otherwise, simulate } a^N_t \text{ with } \mathbb{P}(a^N_t = i) = \rho(i), \end{cases} \tag{20}$$

where $\eta \in [0, 1]$ is a user specified parameter, controlling the probability of executing the AS step. This strategy results in a mix between PG and PGAS; for $\eta = 0$ we recover the original PG algorithm and for $\eta = 1$ we obtain the basic PGAS algorithm. For complex models, this modification can be quite useful. In fact, there is no immediate gain in changing the ancestry of the reference trajectory as long as the particle trajectories have not degenerated. That is, it is sufficient to carry out the AS step "once in a while" to obtain high update

rates for the complete trajectory (cf. Figure 1 where we get high update rates for PG for the last few time steps, even when using a small number of particles). We illustrate this empirically in the simulation study in Section 7.3.

Another modification, that can be used either on its own or in conjunction with (20), is to use MH to simulate from (19). Let $q(i' \mid i)$ be an MH proposal kernel on $\{1, \ldots, N\}$. We can thus propose a move for the ancestor index a_t^N, from N to i', by simulating $i' \sim q(\cdot \mid N)$. With probability

$$1 \wedge \frac{\widetilde{w}_{t-1|T}^{i'}}{\widetilde{w}_{t-1|T}^{N}} \frac{q(N \mid i')}{q(i' \mid N)} \tag{21}$$

the sample is accepted and we set $a_t^N = i'$, otherwise we keep the ancestry $a_t^N = N$. Using this approach, we avoid computing the normalizing constant in (19), i.e., we only need to evaluate the AS weights for the proposed values. This will reduce the computational cost of the AS step by, roughly, a factor N which can be very useful whenever N is moderately large. Since the variable a_t^N is discrete-valued, it is recommended to use a *forced move* proposal in the spirit of Liu (1996). That is, q is constructed so that $q(i \mid i) = 0, \forall i$, ensuring that the current state of the chain is not proposed anew, which would be a wasteful operation.

6.2 Non-Markovian Latent Variable Models

A very useful generalization of SSMs is the class of non-Markovian latent variable models,

$$\begin{aligned} x_{t+1} &\sim f_\theta(x_{t+1} \mid x_{1:t}), \\ y_t &\sim g_\theta(y_t \mid x_{1:t}). \end{aligned}$$

Similarly to the SSM (15), this model is characterized by a latent process $x_t \in \mathsf{X}$ and an observed process $y_t \in \mathsf{Y}$. However, it does not share the conditional independence properties that are central to SSMs. Instead, both the transition density f_θ and the measurement density g_θ may depend on the entire past history of the latent process. Below we discuss the AS step of the PGAS algorithm specifically for these non-Markovian models and derive a truncation strategy for the AS weights. First, however, to motivate the present development we review some application areas in which this type of models arise.

In Bayesian nonparametrics (Hjort et al., 2010) the latent random variables of the classical Bayesian model are replaced by latent stochastic processes, which are typically non-Markovian. This includes popular models based on the Dirichlet process, e.g., Teh et al. (2006); Escobar and West (1995), and Gaussian process regression and classification models (Rasmussen and Williams, 2006). These processes are also commonly used as components in hierarchical Bayesian models, which then inherit their non-Markovianity. An example is the Gaussian process SSM (Turner and Deisenroth, 2010; Frigola et al., 2013), a flexible nonlinear dynamical systems model, for which PGAS has been successfully applied (Frigola et al., 2013).

Another typical source of non-Markovianity is by marginalization over part of the state vector, i.e., Rao-Blackwellization, (Chen and Liu, 2000; Whiteley et al., 2010; Lindsten et al., 2013) or by a change of variables in an SSM. This type of operation typically results in a loss of the Markov property, but can, however, be very useful. For instance, by expressing

an SSM in terms of its "innovations" (i.e., the driving noise of the state process), it is possible to use backward and ancestor sampling in models for which the state transition density is not available. This includes many models for which the transition is implicitly given by a simulator (Gander and Stephens, 2007; Fearnhead et al., 2008; Golightly and Wilkinson, 2008; Murray et al., 2013) or degenerate models where the transition density does not even exist (Ristic et al., 2004; Gustafsson et al., 2002). We illustrate these ideas in Section 7. See also Lindsten and Schön (2013, Section 4) for a more in-depth discussion on reformulations of SSMs as non-Markovian models.

Finally, it is worth to point out that many statistical models which are not sequential "by nature" can be conveniently viewed as non-Markovian latent variable models. This includes, among others, probabilistic graphical models such as Markov random fields; see Lindsten and Schön (2013, Section 4).

To employ PGAS, or in fact any backward-simulation-based method (see Lindsten and Schön 2013), we need to evaluate the AS weights (3) which depend on the ratio

$$\frac{\gamma_{\theta,T}(x_{1:T})}{\gamma_{\theta,t-1}(x_{1:t-1})} = \frac{p_\theta(x_{1:T}, y_{1:T})}{p_\theta(x_{1:t-1}, y_{1:t-1})} = \prod_{s=t}^{T} g_\theta(y_s \mid x_{1:s}) f_\theta(x_s \mid x_{1:s-1}). \tag{22}$$

Assuming that g_θ and f_θ can both be evaluated in constant time, the computational cost of computing the backward sampling weights (3) will thus be $O(NT)$. This implies that the overall computational complexity of the PGAS kernel will scale quadratically with T which can be prohibitive in some cases. The general strategies discussed in Section 6.1, i.e., using AS sporadically and/or using MH within PGAS, can of course be used to mitigate this issue. Nonetheless, the AS step can easily become the computational bottleneck when applying the PGAS algorithm to a non-Markovian model.

To make further progress we consider non-Markovian models in which there is a decay in the influence of the past on the present, akin to that in Markovian models but without the strong Markovian assumption. Hence, it is possible to obtain a useful approximation of the AS weights by truncating the product (22) to a smaller number of factors, say ℓ. We can thus replace (3) with the approximation

$$\begin{aligned}\widetilde{w}^{\ell,i}_{t-1|T} &\triangleq w^i_{t-1} \frac{\gamma_{\theta,t-1+\ell}((x^i_{1:t-1}, x'_{t:t-1+\ell}))}{\gamma_{\theta,t-1}(x^i_{1:t-1})} \\ &= w^i_{t-1} \prod_{s=t}^{t-1+\ell} g_\theta(y_s \mid x^i_{1:t-1}, x'_{t:s}) f_\theta(x'_s \mid x^i_{1:t-1}, x'_{t:s-1}).\end{aligned} \tag{23}$$

Let $\widehat{\rho}_\ell(k)$ be the probability distribution defined by the truncated AS weights (23), analogously to (19). The following proposition formalizes our assumption.

Proposition 1. *Let* $h_s(k) = g_\theta(y_{t-1+s} \mid x^k_{1:t-1}, x'_{t:t-1+s}) f_\theta(x'_{t-1+s} \mid x^k_{1:t-1}, x'_{t:t-1+s})$ *and assume that* $\max_{k,l} (h_s(k)/h_s(l) - 1) \leq A \exp(-cs)$*, for some constants* A *and* $c > 0$*. Then,* $D_{\mathsf{KLD}}(\rho \| \widehat{\rho}_\ell) \leq C \exp(-c\ell)$ *for some constant* C*, where* D_{KLD} *is the Kullback-Leibler (KL) divergence.*

Proof. See Appendix B. ■

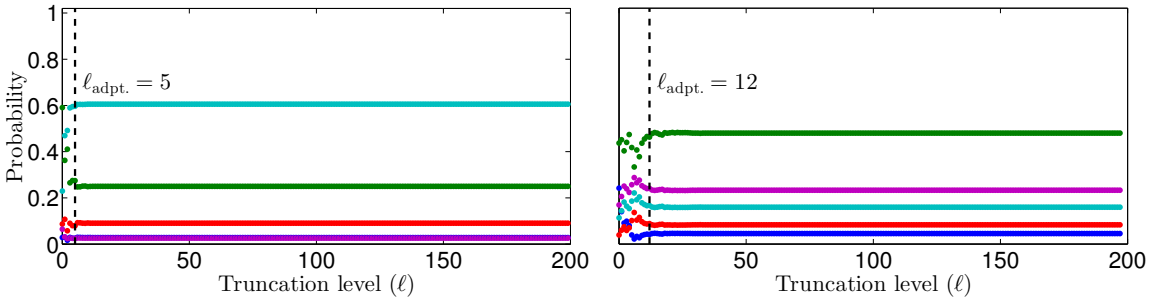


Figure 3: Probability under $\widehat{\rho}_\ell$ as a function of the truncation level ℓ for two different systems; one 5-dimensional (left) and one 20-dimensional (right). The $N = 5$ dotted lines correspond to $\widehat{\rho}_\ell(k)$ for $k \in \{1, \ldots, N\}$, respectively (N.B. two of the lines overlap in the left figure). The dashed vertical lines show the value of the truncation level $\ell_{\text{adpt.}}$, resulting from the adaption scheme with $\upsilon = 0.1$ and $\tau = 10^{-2}$. See Section 7.2 for details on the experiments.

Using the approximation given by (23), the AS weights can be computed in constant time within the PGAS framework. The resulting approximation can be quite useful; indeed, in our experiments we have seen that even $\ell = 1$ can lead to very accurate inferential results. In general, however, it will not be known *a priori* how to set the truncation level ℓ. To address this problem, we propose to use an adaptive strategy. Since the approximative weights (23) can be evaluated sequentially, the idea is to start with $\ell = 1$ and then increase ℓ until the weights have, in some sense, converged. In particular, in our experimental work, we have used the following simple approach.

Let $\varepsilon_\ell = D_{\mathsf{TV}}(\widehat{\rho}_\ell, \widehat{\rho}_{\ell-1})$ be the total variation (TV) distance between the approximative AS distributions for two consecutive truncation levels. We then compute the exponentially decaying moving average of the sequence ε_ℓ, with forgetting factor $\upsilon \in [0, 1]$, and stop when this falls below some threshold $\tau \in [0, 1]$. This adaption scheme removes the requirement to specify ℓ directly, but instead introduces the design parameters υ and τ. However, these parameters are much easier to reason about—a small value for υ gives a rapid response to changes in ε_ℓ whereas a large value gives a more conservative stopping rule, improving the accuracy of the approximation at the cost of higher computational complexity. A similar tradeoff holds for the threshold τ as well. Most importantly, we have found that the same values for υ and τ can be used for a wide range of models, with very different mixing properties.

To illustrate the effect of the adaption rule, and how the distribution $\widehat{\rho}_\ell$ typically evolves as we increase ℓ, we provide two examples in Figure 3. These examples are taken from the simulation study provided in Section 7.2. Note that the untruncated distribution ρ is given for the maximal value of ℓ, i.e., furthest to the right in the figures. By using the adaptive truncation, we can stop the evaluation of the weights at a much earlier stage, and still obtain an accurate approximation of ρ.

The approximation (23) can be used in a few different ways. First, as discussed above, we can simply replace ρ with $\widehat{\rho}_\ell$ in the PGAS algorithm, resulting in a total computational cost of $O(NT\ell)$. This is the approach that we have favored, owing to its simplicity and the fact that we have found the truncation to lead to very accurate approximations. Another approach, however, is to use $\widehat{\rho}_\ell$ as an efficient proposal distribution for the MH algorithm suggested in Section 6.1, leading to an $O(NT\ell + T^2)$ complexity. The MH accept/reject decision will then compensate for the approximation error caused by the truncation. A third approach is to use the MH algorithm, but to make use of the approximation (23) when evaluating the acceptance probability (21). By doing so, the algorithm can be implemented with $O(NT + T\ell)$ computational complexity.

7. Numerical Evaluation

In this section we illustrate the properties of PGAS in a simulation study. First, in Section 7.1 we consider a stochastic volatility SSM and investigate the improvement in mixing offered by AS when PGAS is compared with PG. We do not consider PGBS in this example since, as we show in Proposition 2 in Appendix A, PGAS and PGBS are probabilistically equivalent in this scenario. The conditions of Proposition 2 imply that the weight function in the PF is independent of the ancestor indices. When applied to non-Markovian models, however, Proposition 2 does not apply, since the weight function then will depend on the complete history of the particles. PGAS and PGBS will then have different properties as is illustrated empirically in Section 7.2 where we consider inference in degenerate SSMs reformulated as non-Markovian models. Finally, in Section 7.3 we use a similar reformulation and apply PGAS for identification of an epidemiological model for which the state transition kernel is not available.

7.1 Stochastic Volatility Model with Leverage

Stochastic volatility (SV) models are commonly used to model the variation (or volatility) of a financial asset; see, e.g., Kim et al. (1998); Shephard (2005). Let r_t be the price of an asset at time t and let $y_t = \log(r_t/r_{t-1})$ denote the so-called *log-returns*. A typical SV model is then given by the SSM:

$$\begin{aligned} x_{t+1} &= \mu(1-\varphi) + \varphi x_t + \sigma v_t, \\ y_t &= \exp(-\tfrac{1}{2}x_t)e_t, \end{aligned} \qquad \begin{pmatrix} v_t \\ e_t \end{pmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \right). \tag{24}$$

The correlation between the process noise and the observation noise allows for a leverage effect by letting the price of the asset influence the future volatility. Assuming stationarity, the distribution of the initial state is given by $\mu_\theta(x_1) = \mathcal{N}(x_1; \mu, \sigma^2/(1-\varphi^2))$. The unknown parameters of the model are $\theta = (\mu, \varphi, \sigma^2, \rho)$.

This system is used as a proof of concept, primarily to illustrate the superior mixing of PGAS when compared to PG. However, we also compare PGAS with the particle marginal MH (PMMH) algorithm by Andrieu et al. (2010), which has previously been used to calibrate SV models on the form (24) (Hallgren and Koski, 2014; Pitt et al., 2010). We analyze the Standard and Poor's (S&P) 500 data from 3/April/2006 to 31/March/2014, consisting

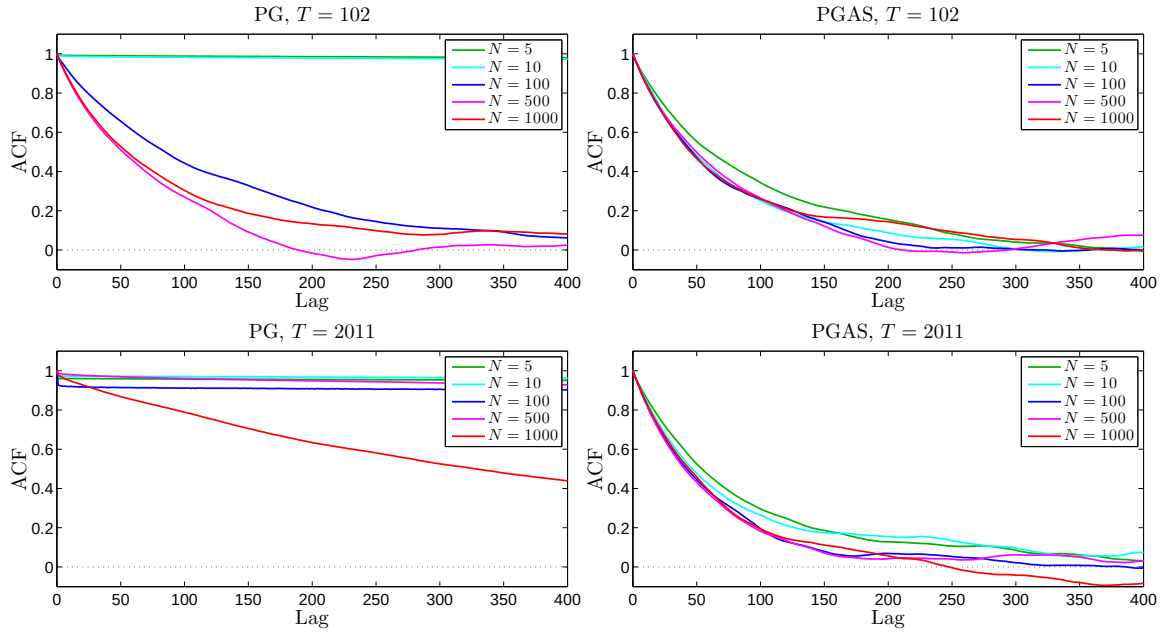


Figure 4: ACFs the parameter σ^2 for PG (left column) and for PGAS (right column) for the S&P 500 data consisting of $T = 102$ (top row) and $T = 2\,011$ (bottom row) observations, respectively. The results are reported for different number of particles N. (This figure is best viewed in color.)

of $T = 2\,011$ observations.[3] We consider the the PGAS sampler (Algorithm 4) as well as the PG and PMMH samplers by Andrieu et al. (2010), all for a range of different number of particles, $N \in \{5, 10, 100, 500, 1\,000\}$. All methods are simulated for 50 000 iterations, whereafter the first 10 000 samples are discarded as burn-in. For updating θ, PG and PGAS simulate the parameters one at a time from their respective conditionals, whereas PMMH uses a Gaussian random walk tuned according to an initial trial run. Additional details on the experiments are given in Appendix C.

To evaluate the mixing of the samplers, we compute the autocorrelation functions (ACFs) for the sequences $\theta[n] - \mathbb{E}[\theta \mid y_{1:T}]$.[4] We start by considering the simpler problem of analyzing a small subset of the data, consisting of $T = 102$ samples (1/November/2013–31/March/2014). The results for PG and PGAS for the parameter σ^2 are reported in the top row of Figure 4. Similar results hold for the other parameters as well. We see that the PG sampler requires a fairly large N to obtain good mixing and using $N \leq 10$ causes the sampler to get completely stuck. For PGAS, on the other hand, the ACF is much more robust to the choice of N. Indeed, we obtain comparable mixing rates for any number of particles $N \geq 5$. This suggests that the sampler in fact performs very closely to a fictive

3. The data was acquired from the Yahoo Finance web page `https://finance.yahoo.com/q/hp?s=%5EGSPC&a=03&b=3&c=2006&d=02&e=31&f=2014&g=d`
4. The "true" posterior mean is computed from a long (500 000 samples) run of PMMH.

	w/o sub-sampling		w sub-sampling	
	PGAS	PMMH	PGAS	PMMH
$N = 5$	111.7	$> 10^4$	24.6	3 923.3
$N = 10$	96.6	$> 10^4$	**20.7**	6 187.0
$N = 100$	71.3	4 796.1	21.1	1 146.8
$N = 500$	73.3	59.2	47.8	33.5
$N = 1\,000$	72.6	31.5	80.9	**31.5**

Table 1: Average inefficiencies for the SV model.

"ideal" Gibbs sampler, i.e., a sampler that simulates $x_{1:T}$ from the true joint smoothing distribution.

Next, we rerun the methods on the whole data set with $T = 2\,011$ observations. The results are shown in the bottom row of Figure 4. The effect can be seen even more clearly in this more challenging scenario. Again, we find PGAS to perform very closely to an ideal Gibbs sampler for any $N \geq 5$. In other words, for this model it is the mixing of the ideal Gibbs sampler, not the intrinsic particle approximation, that is the limitation of PGAS. The big difference in mixing between PG and PGAS can be understood as a manifestation of how they are affected by path degeneracy. These results are in agreement with the discussion in Section 3.3.

We now turn to a comparison between PGAS and PMMH. However, in doing so it is important to realize that these two methods, while both being instances of PMCMC, have quite different properties. In particular, just as PGAS can be thought of as an approximation of an ideal Gibbs sampler, PMMH can be viewed as an approximation of an ideal marginal MH sampler. Consequently, their respective performances depend on the properties of these ideal samplers and the preference for one method over the other heavily depends on the specific problem under study. Nevertheless, we apply both methods to the S&P 500 data (with $T = 2\,011$). To evaluate the mixing, we compute[5] the *inefficiencies*:

$$\mathsf{IF} \triangleq 1 + 2\sum_{j=1}^{\infty} \mathsf{ACF}(j)$$

for the four parameters of the model, where $\mathsf{ACF}(j)$ is the ACF at lag j. The interpretation of the inefficiency is that we need $n \times \mathsf{IF}$ draws from the Markov chain to obtain the same precision as using n i.i.d. draws from the posterior. The average inefficiencies for the four parameters for PGAS and PMMH are reported in Table 1.

In the two columns to the left, the inefficiencies for PGAS and PMMH, respectively, are given without taking the computational cost of the algorithms into account. As above we find that PGAS is quite insensitive to the number of particles N, with only a minor increase in inefficiency as we reduce N from 1 000 to 5. PMMH requires a larger number of particles to mix well—this is in agreement with previous analyzes and empirical studies (Doucet et al., 2014; Andrieu et al., 2010). Using too few particles with PMMH causes the method to get stuck, hence the very large inefficiency values. For large N, however, PMMH

5. We use the *initial monotone sequence estimator* by Geyer (1992) to estimate the inefficiencies.

outperforms PGAS. The reason for this is that the limiting behavior (as we increase N) of PMMH is that of an ideal marginal MH sampler. For the model under study, it is apparent that this marginal MH sampler has better mixing properties than the ideal Gibbs sampler.

Finally, in the rightmost two columns of Table 1 we report the average inefficiencies for the two samplers when we have matched their computational costs. We use PMMH with $N = 1\,000$ as the base algorithm. We then match the computational times (as measured by the `tic/toc` commands in Matlab) of the algorithms and modify the inefficiencies accordingly. This corresponds to sub-sampled versions of the algorithms, such that each iteration of any method take the same amount of time as one iteration of PMMH with $N = 1\,000$. In this comparison, we obtain the best overall performance for PGAS with $N = 10$.

While the computational complexities of both algorithms scale like $O(N)$, it is worth to note that the computational overhead is quite substantial (we use a vectorized implementation of the particle filters in Matlab). In fact, when reducing N from 1 000 to 5, the reduction in computational time is closer to a factor 5 than to a factor 200. Of course, the computational overhead will be less noticeable in more difficult scenarios where it is required to use $N \gg 1\,000$ for PMMH to mix well. Nevertheless, this effect needs to be taken into account when comparing algorithms with very different computational properties, such as PGAS and PMMH, and increasingly so when considering implementations on parallel computer architectures. That is, matching the algorithms "particle by particle" would be overly favorable for PGAS. We discuss this further in Section 8.

7.2 Degenerate LGSS Models

Many dynamical systems are most naturally modeled as *degenerate* in the sense that the transition kernel of the state-process does not admit any density with respect to a dominating measure. It is problematic to use (particle-filter-based) backward sampling methods for these models, owing to the fact that the backward kernel of the state process will also be degenerate. As a consequence, it is not possible to approximate the backward kernel using the forward filter particles.

To illustrate how this difficulty can be remedied by a change of variables, consider an LGSS model of the form

$$\begin{pmatrix} x_{t+1} \\ z_{t+1} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} x_t \\ z_t \end{pmatrix} + \begin{pmatrix} v_t \\ 0 \end{pmatrix}, \qquad v_t \sim \mathcal{N}(0, Q), \tag{25a}$$

$$y_t = C \begin{pmatrix} x_t \\ z_t \end{pmatrix} + e_t, \qquad e_t \sim \mathcal{N}(0, R). \tag{25b}$$

Since the Gaussian process noise enters only on the first part of the state vector (or, equivalently, the process noise covariance matrix is rank deficient) the state transition kernel is degenerate. However, for the same reason, the state component z_t is $\sigma(x_{1:t})$-measurable and we can write $z_t = z_t(x_{1:t})$. Therefore, it is possible to rephrase (25) as a non-Markovian model with latent process given by $\{x_t\}_{t\geq 1}$.

As a first illustration, we simulate $T = 200$ samples from a four-dimensional, single output system with poles[6] at -0.65, -0.12, and $0.22 \pm 0.10i$. We let $\dim(x_t) = 1$ and

6. The poles of a linear system are given by the eigenvalues of the matrix A and they encode the frequency response of the system.

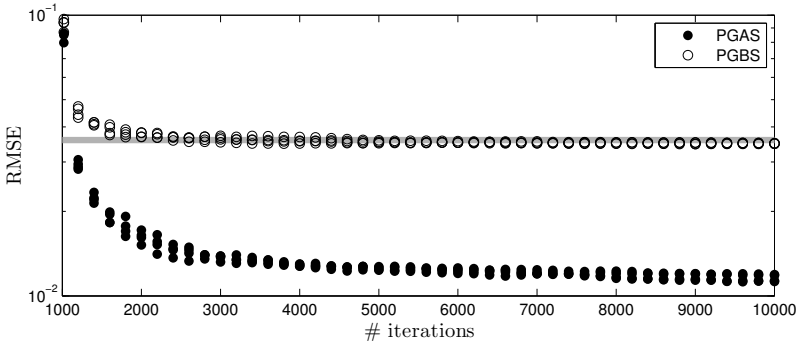


Figure 5: Running RMSEs for $x_{1:T}$ for five independent runs of PGAS (•) and PGBS (◦), respectively. The truncation level is set to $\ell = 1$. The thick gray line corresponds to a run of an untruncated FFBS particle smoother.

$Q = R = 0.1$. For simplicity, we assume that the system parameters are known and seek the joint smoothing distribution $p(x_{1:T} \mid y_{1:T})$. In the non-Markovian formulation it is possible to apply backward-simulation-based methods, such as PGAS and PGBS, as described in Section 6.2. The problem, however, is that the non-Markovianity gives rise to an $O(T^2)$ computational complexity. To obtain more practical inference algorithms we employ the weight truncation strategy (23).

First, we consider the coarse approximation $\ell = 1$. We run PGAS and PGBS, both with $N = 5$ particles for 10 000 iterations (with the first 1 000 discarded as burn-in). We then compute running means of the latent variables $x_{1:T}$ and, from these, we compute the running root-mean-squared errors (RMSEs) ϵ_n *relative to the true posterior means* (computed with a modified Bryson-Frazier smoother, Bierman, 1973). Hence, if no approximation would have been made, we would expect $\epsilon_n \to 0$, so any static error can be seen as the effect of the truncation. The results for five independent runs are shown in Figure 5. First, we note that both methods give accurate results. Still, the error for PGAS is significantly lower than for PGBS. For further comparison, we also run an *untruncated* forward filter/backward simulator (FFBS) particle smoother (Godsill et al., 2004), using $N = 10\,000$ particles and $M = 1\,000$ backward trajectories, with a computational cost of $O(NMT^2)$. The resulting RMSE value is shown as a thick gray line in Figure 5. This result suggest that PGAS can be a serious competitor to more "classical" particle smoothers, even when there are no unknown parameters of the model. Already with $\ell = 1$, PGAS outperforms FFBS in terms of accuracy and, due to the fact that AS allows us to use as few as $N = 5$ particles at each iteration, at a much lower computational cost.

To see how the samplers are affected by the choice of truncation level ℓ and by the mixing properties of the system, we consider randomly generated systems of the form (25) of different orders (i.e., with different state dimensions d). We generate 150 random systems, using the Matlab function `drss` from the Control Systems Toolbox, with model orders 2, 5 and 20 (50 systems for each model order). The number of outputs are taken as 1, 2 and 4 for the different model orders, respectively. We consider different fixed truncation levels ($\ell \in \{1, 2, 3\}$ for 2nd order systems and $\ell \in \{1, 5, 10\}$ for 5th and 20th order systems), as

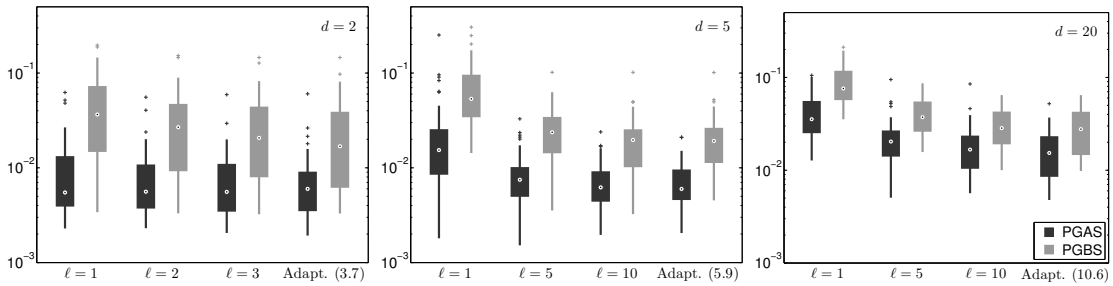


Figure 6: Box plots of the RMSE errors for PGAS (black) and PGBS (gray), for 150 random systems of different dimensions d (left, $d = 2$; middle, $d = 5$; right, $d = 20$). Different values for the truncation level ℓ are considered. The rightmost boxes correspond to an adaptive truncation and the values in parentheses are the average truncation levels over all systems and MCMC iterations (the same for both methods). The dots within the boxes show the median errors.

well as an adaptive level with $\upsilon = 0.1$ and $\tau = 10^{-2}$ (see Section 6.2). All other settings are as above.

Again, we compute the posterior means of $x_{1:T}$ (discarding 1 000 samples) and RMSE values relative to the true posterior mean. Box plots over the different systems are shown in Figure 6. Since the process noise only enters on one of the state components, the mixing tends to deteriorate as we increase the model order. Figure 3 shows how the probability distributions on $\{1, \ldots, N\}$ change as we increase the truncation level, in two representative cases for a 5th and a 20th order system, respectively. By using an adaptive level, we can obtain accurate results for systems of different dimensions, without having to change any settings between the runs.

7.3 Epidemiological Model

As a final numerical illustration, we consider identification of an epidemiological model using PGAS. Seasonal influenza epidemics each year cause millions of severe illnesses and hundreds of thousands of deaths world-wide (Ginsberg et al., 2009). Furthermore, new strains of influenza viruses can possibly cause pandemics with very severe effects on the public health. The ability to accurately predict disease activity can enable early response to such epidemics, which in turn can reduce their impact.

We consider a susceptible/infected/recovered (SIR) model with environmental noise and seasonal fluctuations (Keeling and Rohani, 2007; Rasmussen et al., 2011). The model, specified by a stochastic differential equation, is discretized according to the Euler-Maruyama method, yielding

$$S_{t+dt} = S_t + \mu\mathcal{P}dt - \mu S_t dt - (1 + Fv_t)\,\beta_t S_t \mathcal{P}^{-1} I_t dt, \tag{26a}$$

$$I_{t+dt} = I_t - (\gamma + \mu) I_t dt + (1 + Fv_t)\,\beta_t S_t \mathcal{P}^{-1} I_t dt, \tag{26b}$$

$$R_{t+dt} = R_t + \gamma I_t dt - \mu R_t dt, \tag{26c}$$

where $v_t \sim \mathcal{N}(0, 1/\sqrt{dt})$ and dt is the sampling time. Here, S_t, I_t and R_t represent the number of susceptible, infected and recovered individuals at time t (months), respectively. The total population size $\mathcal{P} = 10^6$ and the host birth/death rate $\mu = 0.0012$ are assumed known. The seasonally varying transmission rate is given by $\beta_t = R_0(\gamma+\mu)(1+\alpha\sin(2\pi t/12))$ where R_0 is the basic reproductive ratio, γ is the rate of recovery and α is the strength of seasonality.

Furthermore, we consider an observation model which is inspired by the Google Flu Trends project (Ginsberg et al., 2009). The idea is to use the frequency of influenza-related search engine queries to infer knowledge of the dynamics of the epidemic. Let Q_k be the proportion of influenza-related queries counted during a time interval $(\Delta(k-1), \Delta k]$. Following Ginsberg et al. (2009), we use a linear relationship between the log-odds of the relative query counts and the log-odds of the proportion of infected individuals,

$$y_k \triangleq \operatorname{logit}(Q_k) = \rho \operatorname{logit}(\bar{I}_k/\mathcal{P}) + e_k, \qquad e_k \sim \mathcal{N}(0, \sigma^2), \tag{27}$$

where $\bar{I}_k$ is the mean value of I_t during the time interval $(\Delta(k-1), \Delta k]$ and $\operatorname{logit}(p) = \log(p/(1-p))$. As in Ginsberg et al. (2009) we consider weekly query counts, i.e., $\Delta = 7/30$ (assuming for simplicity that we have 30 days in each month). Using this value of Δ as sampling time will, however, result in overly large discretization errors. Instead, we sample the model (26) $m = 7$ times per week: $dt = \Delta/m$.

Rasmussen et al. (2011) use the PMMH sampler (Andrieu et al., 2010) to identify a similar SIR model, though with a different observation model. A different Monte Carlo strategy, based on a particle filter with an augmented state space, for identification of an SIR model is proposed by Skvortsov and Ristic (2012). We investigate the possibility of using PGAS for joint state and parameter inference in the model (26)–(27). However, there are two difficulties in applying PGAS directly to this model. Firstly, the transition kernel of the state process, as defined between consecutive observation time points $\Delta(k-1)$ and Δk, is not available in closed form. Secondly, since the state is three-dimensional, whereas the driving noise v_t is scalar, the transition kernel is degenerate. To cope with these difficulties we (again) suggest collapsing the model to the driving noise variables. Let $V_k = \begin{pmatrix} v_{\Delta(k-1)} & v_{\Delta(k-1)+dt} & \cdots & v_{\Delta k-dt} \end{pmatrix}^T$. It follows that the model (26)–(27) can be equivalently expressed as the non-Markovian latent variable model,

$$V_k \sim \mathcal{N}(0, I_m/\sqrt{dt}), \tag{28a}$$

$$y_k \sim g_\theta(y_k \mid V_{1:k}), \tag{28b}$$

for some likelihood function g_θ; see (29). A further motivation for using this reformulation is that the latent variables V_k are *a priori* independent of the model parameters θ. This can result in a significant improvement in mixing of the Gibbs sampler, in particular when there are strong dependencies between the system state and the parameters (Golightly and Wilkinson, 2008; Papaspiliopoulos et al., 2003).

The parameters of the model are $\theta = (\gamma, R_0, \alpha, F, \rho, \sigma)$, with the true values given by $\gamma = 3$, $R_0 = 10$, $\alpha = 0.16$, $F = 0.03$, $\rho = 1.1$ and $\sigma = 0.224$. We use an improper flat prior on $\mathbb{R}^6_+$ for θ. We generate eight years of data with weekly observations. The number of infected individuals I_t over this time period is shown in Figure 7. The first half of the data batch is used for estimation of the model parameters using PGAS. It is

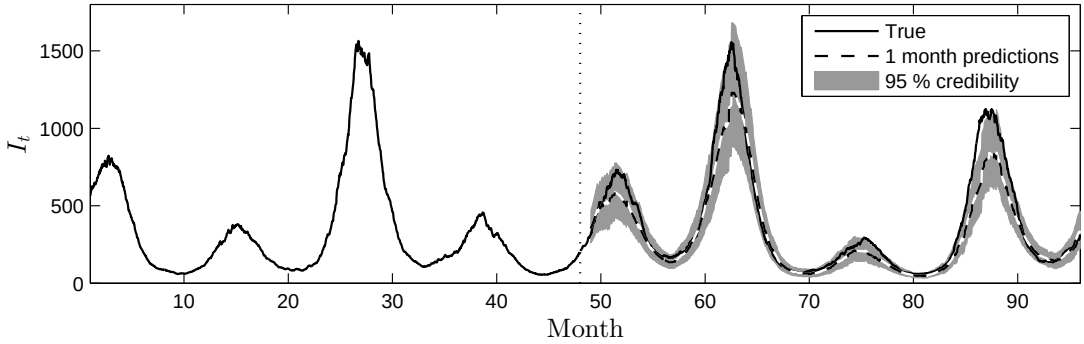


Figure 7: Disease activity (number of infected individuals I_t) over an eight year period. The first four years are used as estimation data, to find the unknown parameters of the model. For the consecutive four years, one-month-ahead predictions are computed using the estimated model.

worth pointing out that while the sampler effectively targets the collapsed model (28), it is most straightforwardly implemented using the original state variables from (26). With $x_k = (S_{\Delta k}, I_{\Delta k}, R_{\Delta k})^\mathsf{T}$ we can simulate x_{k+1} given x_k according to (26) which is used in the underlying particle filter. The innovation variables V_k need only be taken into account for the AS step. Let $V'_{1:T}$ be the reference innovation trajectory. To compute the AS weights (3) we need to evaluate the ratios,

$$\frac{p_\theta((V^i_{1:k-1}, V'_{k:T}), y_{1:T})}{p_\theta(V^i_{1:k-1}, y_{1:k-1})} \propto \prod_{\ell=k}^{T} g_\theta(y_\ell \mid V^i_{1:k-1}, V'_{k:\ell}).$$

Using (27), the observation likelihood can be written as

$$g_\theta(y_\ell \mid V^i_{1:k-1}, V'_{k:\ell}) = \mathcal{N}(y_\ell \mid \rho \operatorname{logit}(\bar{I}_\ell\{x^i_{k-1}, V'_{k:\ell}\}/\mathcal{P}), \sigma^2), \tag{29}$$

where $I_\ell\{x^i_{k-1}, V'_{k:\ell}\}$ is obtained by simulating the system (26) from time $\Delta(k-1)$ to time $\Delta\ell$, initialized at x^i_{k-1} and using the innovation sequence $V'_{k:\ell}$.

We run PGAS with $N = 10$ particles for 50 000 iterations (discarding the first 10 000). For sampling θ, we use MH steps with a Gaussian random walk proposal, tuned according to an initial trial run. The innovation variables $V_{1:T}$ are sampled from the PGAS kernel by Algorithm 2. Since the latter step is the computational bottleneck of the algorithm, we execute ten MH steps for θ, for each draw from the PGAS kernel. No truncation is used for the AS weights; instead we investigate the effect of using the strategy proposed in (20). That is, to reduce the computational cost we execute the AS step only with some probability η, otherwise we keep the current ancestry of the reference trajectory.

In Figure 8 we report the ACFs for the six parameters of the model, for η ranging from 0 to 1. As a comparison, we also provide the results for a run of the PMMH algorithm with $N = 1\,000$ particles and a random walk proposal distribution tuned according to an initial trial run. For most parameters, PMMH achieves better mixing than PGAS (however,

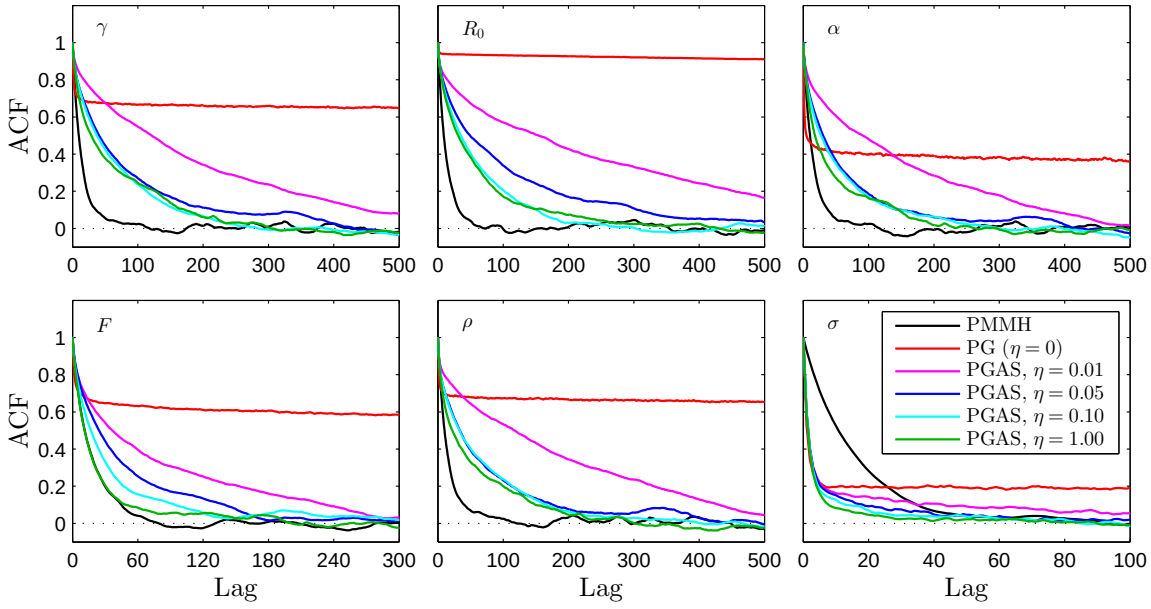


Figure 8: ACFs for PGAS with $N = 10$ and η ranging from 0 to 1. As comparison, we also show the ACF for PMMH with $N = 1\,000$. (This figure is best viewed in color.)

requiring a much larger N) which can be accredited to the fact that the ideal marginal MH sampler mixes better than the ideal Gibbs sampler.

Note that for $\eta = 0$, PGAS reduces to the standard PG algorithm. Since we use only $N = 10$ particles this sampler mixes very poorly, in agreement with our previous findings. However, interestingly, increasing the probability of ancestor sampling to as little as $\eta = 0.01$ results in a large improvement in mixing and with $\eta = 0.1$ we get results that are comparable to $\eta = 1$. This suggests that, in cases when the AS step is the computational bottleneck of the algorithm, it can indeed be a good idea to carry out this step only sporadically.

To further investigate the effect of η, we plot the update rates for the trajectory $V_{1:T}$ in Figure 9 (cf. Figure 1). As expected, the update rate deteriorates as we decrease η, but the relationship is clearly nonlinear. Specifically, for small values of η we get an average update rate which is larger than η; for instance $\eta = 0.1$ gives an average update rate of 0.31. The reason for this is that any ancestor index update will result in an update, not only for the corresponding latent variable, but also for a collection of neighboring latent variables. The number of variables that will be affected by changing one of the ancestor indices depends on how quickly the PF degenerates. Consequently, there is an inverse relationship between η and N; by increasing the number of particles we can get away with a smaller η and still obtain high update rates for the entire trajectory.

In Figure 10 we show histograms representing the estimated posterior parameter distributions, reported for PGAS with $\eta = 0.1$ and for PMMH. As can be seen, the true system parameters fall well within the credible regions. Finally, the identified model, based on PGAS with $\eta = 0.1$, is used to make one-month-ahead predictions of the disease activity for

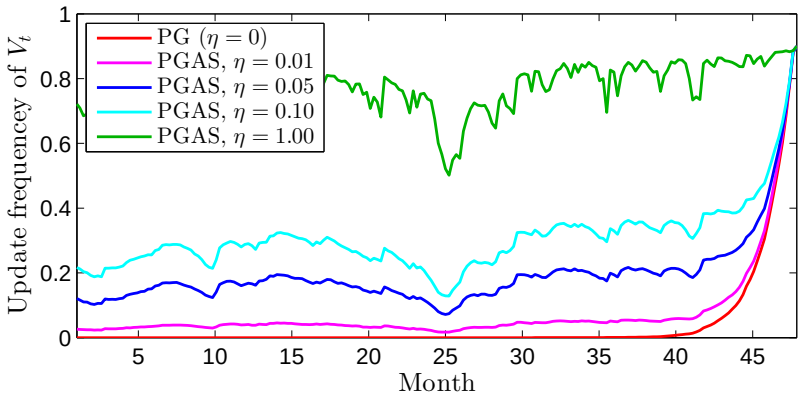


Figure 9: Update rates for $V_{1:T}$ for PGAS with $N = 10$ and with η ranging from 0 to 1. As a comparison, the PMMH sampler with $N = 1\,000$ particles attains an average acceptance probability of 0.19. (This figure is best viewed in color.)

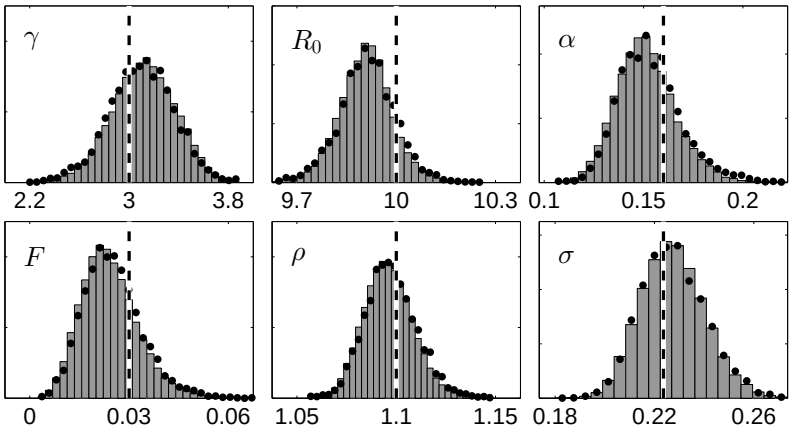


Figure 10: Posterior densities for the parameters of model (26)–(27) for PGAS; $N = 10$, $\eta = 0.1$ (gray bars) and for PMMH; $N = 1\,000$ (black dots). The true values are marked by vertical dashed lines.

the subsequent four years, as shown in Figure 7. The predictions are made by sub-sampling the Markov chain and, for each sample, running a particle filter on the validation data using 100 particles. As can be seen, we obtain an accurate prediction of the disease activity, which falls within the estimated 95 % credibility intervals, one month in advance.

8. Discussion

PGAS is a novel approach to PMCMC that provides the statistician with an off-the-shelf class of Markov kernels which can be used to simulate, for instance, the typically high-dimensional and highly autocorrelated state trajectory in a state-space model. This opens

up the possibility of using PGAS as a key component in different inference algorithms, enabling both Bayesian and frequentist parameter inference as well as state inference. However, PGAS is by no means limited to inference in state-space models. Indeed, we believe that the method can be particularly useful for models with more complex dependencies, such as non-Markovian, nonparametric, and graphical models.

The PGAS Markov kernels are built upon two main ideas. First, by conditioning the underlying SMC sampler on a reference trajectory the correct stationary distribution of the kernel is enforced. Second, *ancestor sampling* enables movement around the reference trajectory which drastically improves the mixing of the sampler. In particular, we have shown empirically that ancestor sampling makes the mixing of the PGAS kernels robust to a small number of particles as well as to large data records.

Ancestor sampling is basically a way of exploiting backward simulation ideas without needing an explicit backward pass. Compared to PGBS, a conceptually similar method that does require an explicit backward pass, PGAS has several advantages, most notably for inference in non-Markovian models. When using the proposed truncation of the backward weights, we have found PGAS to be more robust to the approximation error than PGBS, yielding up to an order-of-magnitude improvement in accuracy. An interesting topic for future work is to further investigate the effect on these samplers by errors in the backward weights, whether these errors arise from a truncation or some other approximation of the transition density function. It is also worth pointing out that for non-Markovian models PGAS is simpler to implement than PGBS as it requires less bookkeeping. It can also be more memory efficient; by using the techniques proposed by Jacob et al. (2013), it is possible to store the paths of the particle filter in PGAS with an expected memory cost bounded by $T + CN \log N$ for some constant C. This is in contrast with PGBS, which requires storage of all NT intermediate particles.

The aforementioned samplers—PG, PGAS, and PGBS—share the same interpretation of being PMCMC-versions of an ideal Gibbs sampler. A different type of PMCMC, however, is the PMMH sampler by Andrieu et al. (2010). To comprehensively compare PGAS with PMMH is nontrivial, since the two samplers have quite different properties. However, some of the most important differences are that, *(i)* in the limit $N \to \infty$, PMMH approaches a marginal sampler for θ, whereas PGAS approaches an ideal Gibbs sampler for θ and $x_{1:T}$, *(ii)* empirically, PGAS is more robust to small N/large T than PMMH, and *(iii)* PGAS defines a Markov kernel on the space of trajectories, which is not the case for PMMH, making it more suitable to use as a component in composite sampling schemes. Due to these differences—*(i)* being in favor for PMMH and *(ii)*–*(iii)* for PGAS—the preference for one sampler over the other depends heavily on the specific properties of the problem at hand.

Another important difference is that PMMH readily allows for parallelization over the particles. While this is of course possible also for PGAS, the fact that the sampler typically requires only a small number of particles limits the computational benefits of doing so. To enable PGAS to make better use of modern computational architectures, other approaches might therefore prove to be more fruitful. This includes, for instance, to couple PGAS with parallel MCMC methods (see, e.g., VanDerwerken and Schmidler 2013; Wilkinson 2005) or to use the PGAS Markov kernels together with SMC samplers (Del Moral et al., 2006)

instead of with classical MCMC. The practical usefulness of these approaches is a topic that requires further investigation.

Other directions for future work include further analysis of the ergodicity of PGAS. While the established uniform ergodicity result is encouraging, it does not provide information about how fast the mixing rate improves with the number of particles. Finding informative rates with an explicit dependence on N is an interesting, though challenging, topic for future work. It would also be interesting to further investigate empirically the convergence rate of PGAS for different settings, such as the number of particles, the amount of data, and the dimension of the latent process.

Acknowledgments

The authors would like to thank Sumeetpal S. Singh for pointing out that Metropolis-Hastings sampling can be useful for reducing the computational complexity of the AS step in the non-Markovian setting. This work was supported by: the project Probabilistic modelling of dynamical systems (Contract number: 621-2013-5524) funded by the Swedish Research Council, CADICS, a Linnaeus Center also funded by the Swedish Research Council, and the project Bayesian Tracking and Reasoning over Time (Reference: EP/K020153/1), funded by the EPSRC.

Appendix A. The Relationship between PGAS and PGBS for SSMs

As pointed out in Section 5, there is a close relationship between PGAS and PGBS, in particular when considering the special case of SSMs. PGBS is conceptually similar to PGAS, but it makes use of an explicit backward simulation pass; see Whiteley (2010); Whiteley et al. (2010) or Lindsten and Schön (2013, Section 5.4). More precisely, to generate a draw from the PGBS kernel, we first run a particle filter with reference trajectory $x'_{1:T}$ *without* AS (i.e., in Algorithm 2, we replace line 8 with $a_t^N = N$, as in the basic PG sampler). Thereafter, we extract a new trajectory by running a backward simulator. That is, we draw $j_{1:T}$ with $\mathbb{P}(j_T = i) \propto w_T^i$ and then, for $t = T-1$ to 1,

$$\mathbb{P}(j_t = i \mid j_{t+1}) \propto w_t^i f_\theta(x_{t+1}^{j_{t+1}} \mid x_t^i), \tag{30}$$

and take $x^\star_{1:T} = x^{j_{1:T}}_{1:T}$ as the output from the algorithm. In the above, the conditioning on the forward particle system $\{\mathbf{x}_{1:T}, \mathbf{a}_{2:T}\}$ is implicit.

Let the Markov kernel on $(\mathsf{X}^T, \mathcal{X}^T)$ defined by this procedure be denoted as $P^N_{\mathsf{BS},\theta}$. An interesting question to ask is whether or not the PGAS kernel P^N_θ and the PGBS kernel $P^N_{\mathsf{BS},\theta}$ are probabilistically equivalent. In the specific setting when both methods use the bootstrap proposal kernel in the internal particle filters, it turns out that this is indeed the case. We formalize this is Proposition 2 below. The analysis builds upon Olsson and Rydén (2011, Proposition 5), where the equivalence between a (standard) bootstrap PF and a backward simulator is established. Below, we adapt their argument to handle the case with conditioning on a reference trajectory and the AS step. For improved readability we provide the complete proof, though it should be noted that the main part is due to Olsson and Rydén (2011).

Proposition 2. *Assume that PGAS and PGBS both target the joint smoothing distribution for an SSM and that both methods use the bootstrap proposal kernel in the internal particle filters, i.e.,* $r_{\theta,t}(x_t \mid x_{1:t-1}) = f_\theta(x_t \mid x_{t-1})$. *Then, for any* $x'_{1:T} \in \mathsf{X}^T$ *and* $B \in \mathcal{X}^T$, $P^N_\theta(x'_{1:T}, B) = P^N_{\mathsf{BS},\theta}(x'_{1:T}, B)$.

Proof. For ease of notation, we write $\mathbb{E}$ for $\mathbb{E}_{\theta,x'_{1:T}}$. First, note that for a bootstrap proposal kernel, the weight function (2) is given by $W_{\theta,t}(x_t) = g_\theta(y_t \mid x_t)$, i.e., it depends only on the current state and not on its ancestor. As a consequence, the law of the forward particle system is independent of the ancestor variables $\{a^N_t\}_{t=2}^T$, meaning that the particle systems, excluding $\{a^N_t\}_{t=2}^T$, are equally distributed for PGAS and for PGBS.

Let $B \in \mathcal{X}^T$ be a measurable rectangle: $B = \times_{t=1}^T B_t$ with $B_t \in \mathcal{X}$ for $t = 1, \ldots, T$. Then,

$$P^N_\theta(x'_{1:T}, B) = \mathbb{E}\left[\prod_{t=1}^T \mathbb{1}_{B_t}(x_t^{b_t})\right], \qquad \text{and} \qquad P^N_{\mathsf{BS},\theta}(x'_{1:T}, B) = \mathbb{E}\left[\prod_{t=1}^T \mathbb{1}_{B_t}(x_t^{j_t})\right].$$

Since the measurable rectangles form a π-system generating $\mathcal{X}^T$, it is by the π-λ theorem sufficient to show that $\mathbb{E}[h(x_{1:T}^{b_{1:T}})] = \mathbb{E}[h(x_{1:T}^{j_{1:T}})]$ for all bounded, multiplicative functionals, $h(x_{1:T}) = \prod_{t=1}^T h_t(x_t)$. As Olsson and Rydén (2011), we establish this result by induction. Hence, for $t < T$, assume that

$$\mathbb{E}\left[\prod_{s=t+1}^T h_s(x_s^{b_s})\right] = \mathbb{E}\left[\prod_{s=t+1}^T h_s(x_s^{j_s})\right].$$

For $t = T-1$, the induction hypothesis holds since b_T and j_T are equally distributed (both are drawn from the discrete distribution induced by the weights $\{w^i_T\}_{i=1}^N$). Let

$$\begin{aligned}\Lambda_t(x_{t+1}^{j_{t+1}}, h) &\triangleq \mathbb{E}\left[h(x_t^{j_t}) \mid x_{t+1}^{j_{t+1}}\right] = \mathbb{E}\left[\mathbb{E}\left[h(x_t^{j_t}) \mid \mathbf{x}_t, x_{t+1}^{j_{t+1}}\right] \mid x_{t+1}^{j_{t+1}}\right] \\ &= \mathbb{E}\left[\sum_{i=1}^N h(x_t^i) \frac{w_t^i f_\theta(x_{t+1}^{j_{t+1}} \mid x_t^i)}{\sum_l w_t^l f_\theta(x_{t+1}^{j_{t+1}} \mid x_t^l)} \mid x_{t+1}^{j_{t+1}}\right],\end{aligned}$$

where we recall that $w_t^i = W_{\theta,t}(x_t^i)$ and where the last equality follows from (30). Consider,

$$\mathbb{E}\left[\prod_{s=t}^T h_s(x_s^{b_s})\right] = \mathbb{E}\left[\mathbb{E}\left[h_t(x_t^{b_t}) \mid x_{t+1:T}^{b_{t+1:T}}, b_{t+1:T}\right] \prod_{s=t+1}^T h_s(x_s^{b_s})\right]. \tag{31}$$

Using the Markov property of the generated particle system and the tower property of conditional expectation, we have

$$\mathbb{E}\left[h_t(x_t^{b_t}) \mid x_{t+1:T}^{b_{t+1:T}}, b_{t+1:T}\right] = \mathbb{E}\left[\mathbb{E}\left[h_t(x_t^{b_t}) \mid \mathbf{x}_t, x_{t+1}^{b_{t+1}}, b_{t+1}\right] \mid x_{t+1}^{b_{t+1}}, b_{t+1}\right]. \tag{32}$$

Recall that $b_t = a_{t+1}^{b_{t+1}}$. Consider first the case $b_{t+1} < N$. From (1), we have that $\mathbb{P}(b_t = i \mid \mathbf{x}_t) \propto w_t^i$ and $x_{t+1}^{b_{t+1}} \mid x_t^{b_t} \sim f_\theta(\cdot \mid x_t^{b_t})$. If follows from Bayes' theorem that

$\mathbb{P}(b_t = i \mid \mathbf{x}_t, x_{t+1}^{b_{t+1}}) \propto w_t^i f_\theta(x_{t+1}^{b_{t+1}} \mid x_t^{b_t})$. However, by the AS procedure (Algorithm 2, line 8), the same expression holds also for $b_{t+1} = N$. We can thus write (32) as

$$\mathbb{E}\left[h_t(x_t^{b_t}) \mid x_{t+1:T}^{b_{t+1:T}}, b_{t+1:T}\right] = \mathbb{E}\left[\sum_{i=1}^N h_t(x_t^i) \frac{w_t^i f_\theta(x_{t+1}^{b_{t+1}} \mid x_t^i)}{\sum_l w_t^l f_\theta(x_{t+1}^{b_{t+1}} \mid x_t^l)} \mid x_{t+1}^{b_{t+1}}, b_{t+1}\right] = \Lambda_t(x_{t+1}^{b_{t+1}}, h_t),$$

Hence, since the function $x_{t+1} \mapsto \Lambda_t(x_{t+1}, h_t)$ is bounded, we can use the induction hypothesis to write (31) as

$$\begin{aligned}\mathbb{E}\left[\prod_{s=t}^T h_s(x_s^{b_s})\right] &= \mathbb{E}\left[\Lambda_t(x_{t+1}^{b_{t+1}}, h_t) \prod_{s=t+1}^T h_s(x_s^{b_s})\right] = \mathbb{E}\left[\Lambda_t(x_{t+1}^{j_{t+1}}, h_t) \prod_{s=t+1}^T h_s(x_s^{j_s})\right] \\ &= \mathbb{E}\left[\mathbb{E}\left[h_t(x_t^{j_t}) \mid x_{t+1:T}^{j_{t+1:T}}, j_{t+1:T}\right] \prod_{s=t+1}^T h_s(x_s^{j_s})\right] = \mathbb{E}\left[\prod_{s=t}^T h_s(x_s^{j_s})\right].\end{aligned}$$

■

Appendix B. Proof of Proposition 1

With $M = T - t + 1$ and $w(k) = w_{t-1}^k$, the distributions of interest are given by

$$\rho(k) = \frac{w(k)\prod_{s=1}^M h_s(k)}{\sum_l w(l)\prod_{s=1}^M h_s(l)} \qquad \text{and} \qquad \widehat{\rho}_\ell(k) = \frac{w(k)\prod_{s=1}^\ell h_s(k)}{\sum_l w(l)\prod_{s=1}^\ell h_s(l)},$$

respectively. Let $\varepsilon_s \triangleq \max_{k,l}(h_s(k)/h_s(l) - 1) \leq A\exp(-cs)$ and consider

$$\begin{aligned}\left(\sum_l w(l)\prod_{s=1}^\ell h_s(l)\right)\prod_{s=\ell+1}^M h_s(k) &\leq \sum_l \left(w(l)\prod_{s=1}^\ell h_s(l)\prod_{s=\ell+1}^M h_s(l)(1+\varepsilon_s)\right) \\ &= \left(\sum_l w(l)\prod_{s=1}^M h_s(l)\right)\prod_{s=\ell+1}^M (1+\varepsilon_s).\end{aligned}$$

It follows that the KL divergence is bounded according to,

$$\begin{aligned}D_{\mathsf{KLD}}(\rho\|\widehat{\rho}_\ell) &= \sum_k \rho(k)\log\frac{\rho(k)}{\widehat{\rho}_\ell(k)} = \sum_k \rho(k)\log\left(\frac{\prod_{s=\ell+1}^M h_s(k)\left(\sum_l w(l)\prod_{s=1}^\ell h_s(l)\right)}{\sum_l w(l)\prod_{s=1}^M h_s(l)}\right) \\ &\leq \sum_k \rho(k)\sum_{s=\ell+1}^M \log(1+\varepsilon_s) \leq \sum_{s=\ell+1}^M \varepsilon_s \leq A\sum_{s=\ell+1}^M \exp(-cs) = A\frac{e^{-c(\ell+1)} - e^{-c(M+1)}}{1-e^{-c}}.\end{aligned}$$

■

Appendix C. Details on the Experiment in Section 7.1

The parameters of the SV model (24) are $\theta = (\mu, \varphi, \sigma^2, \rho)$. For μ and φ, we use the priors proposed by Kim et al. (1998) (who consider inference in an SV model without the

correlation parameter ρ), namely $\mu \sim \mathcal{N}(0, 10)$ and $\varphi = 2\varphi^\star - 1$ where $\varphi^\star$ is beta distributed; $\varphi^\star \sim \mathcal{B}(20, 1.5)$. Consequently, φ is supported on $(-1, 1)$ with a prior mean of 0.86. This choice is made to ensure stationarity and identifiability of the model. We also use the efficient rejection sampler proposed by Kim et al. (1998) to simulate φ from its posterior conditional distribution. For σ^2 and ρ, we note that the model (24) can be written as

$$\begin{aligned} x_{t+1} &= \mu(1-\varphi) + \varphi x_t + \sigma\rho y_t \exp(-\tfrac{1}{2}x_t) + \sigma\sqrt{1-\rho^2}v_t^\star, \\ y_t &= \exp(-\tfrac{1}{2}x_t)e_t, \end{aligned}$$

where $v_t^\star$ and e_t are mutually independent standard normal. To obtain an efficient updating formula for (σ^2, ρ), we assume a conjugate normal-inverse-gamma prior for the pair $(\vartheta, \varsigma^2) \triangleq (\sigma\rho, \sigma^2(1-\rho^2))$, with $\vartheta \mid \varsigma^2 \sim \mathcal{N}(0, \varsigma^2/0.05)$ and $\varsigma^2 \sim \mathcal{IG}(5/2, 0.05/2)$. We also investigated the possibility of letting σ^2 and ρ be *a priori* independent with an inverse gamma and a uniform prior, respectively, but we did not experience any notable differences in the posterior distributions.

In the experiments, all the samplers are initialized at $\theta[0] = (0, 0.975, 0.05, 0)$. For PMMH, we tune the covariance matrix of the random walk proposal distribution according to the posterior distribution obtained from an initial trial run, using PGAS with $N = 20$ for 10 000 iterations.

References

C. Andrieu, E. Moulines, and P. Priouret. Stability of stochastic approximation under verifiable conditions. *SIAM Journal on Control and Optimization*, 44(1):283–312, 2005.

C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B*, 72(3):269–342, 2010.

A. Benveniste, M. Métivier, and P. Priouret. *Adaptive Algorithms and Stochastic Approximations*. Springer-Verlag, New York, USA, 1990.

G. J. Bierman. Fixed interval smoothing with discrete measurements. *International Journal of Control*, 18(1):65–75, 1973.

R. Chen and J. S. Liu. Mixture Kalman filters. *Journal of the Royal Statistical Society: Series B*, 62(3):493–508, 2000.

N. Chopin and S. S. Singh. On particle Gibbs sampling. *Bernoulli*, 2014. Forthcoming.

P. de Jong and N. Shephard. The simulation smoother for time series models. *Biometrika*, 82(2):339–350, 1995.

P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B*, 68(3):411–436, 2006.

B. Delyon, M. Lavielle, and E. Moulines. Convergence of a stochastic approximation version of the EM algorithm. *The Annals of Statistics*, 27(1):94–128, 1999.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

A. Doucet and A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovskii, editors, *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011.

A. Doucet, M. K. Pitt, G. Deligiannidis, and R. Kohn. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. arXiv.org, arXiv:1210.1871v3, March 2014.

D. A. Van Dyk and T. Park. Partially collapsed Gibbs samplers: Theory and methods. *Journal of the American Statistical Association*, 103(482):790–796, 2008.

M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588, 1995.

P. Fearnhead, O. Papaspiliopoulos, and G. O. Roberts. Particle filters for partially observed diffusions. *Journal of the Royal Statistical Society: Series B*, 70(4):755–777, 2008.

R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen. Bayesian inference and learning in Gaussian process state-space models with particle MCMC. In *Advances in Neural Information Processing Systems (NIPS) 26*. December 2013.

M. P. S. Gander and D. A. Stephens. Stochastic volatility modelling in continuous time with general marginal distributions: Inference, prediction and model selection. *Journal of Statistical Planning and Inference*, 137(10):3068–3081, 2007.

C. J. Geyer. Practical Markov chain Monte Carlo. *Statistical Science*, 7(4):473–483, 1992.

J. Ginsberg, M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, and L. Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457:1012–1014, 2009.

S. J. Godsill, A. Doucet, and M. West. Monte Carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, 99(465):156–168, March 2004.

A. Golightly and D. J. Wilkinson. Bayesian inference for nonlinear multivariate diffusion models observed with error. *Computational Statistics & Data Analysis*, 52(3):1674–1693, 2008.

A. Golightly and D. J. Wilkinson. Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo. *Interface Focus*, 1(6):807–820, 2011.

F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing*, 50(2):425–437, 2002.

J. Hallgren and T. Koski. Decomposition sampling applied to parallelization of Metropolis-Hastings. arXiv.org, arXiv:1402.2828, February 2014.

N. L. Hjort, C. Holmes, P. Müller, and S. G. Walker, editors. *Bayesian Nonparametrics*. Cambridge University Press, 2010.

P. E. Jacob, L. M. Murray, and S. Rubenthaler. Path storage in the particle filter. *Statistics and Computing*, 2013. doi: 10.1007/s11222-013-9445-x. (available online).

M.J. Keeling and P. Rohani. *Modeling Infectious Diseases in Humans and Animals*. Princeton University Press, 2007.

S. Kim, N. Shephard, and S. Chib. Stochastic volatility: Likelihood inference and comparison with ARCH models. *The Review of Economic Studies*, 65(3):361–393, 1998.

F. Lindsten. An efficient stochastic approximation EM algorithm using conditional particle filters. In *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013.

F. Lindsten and T. B. Schön. On the use of backward simulation in the particle Gibbs sampler. In *Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, March 2012.

F. Lindsten and T. B. Schön. Backward simulation methods for Monte Carlo statistical inference. *Foundations and Trends in Machine Learning*, 6(1):1–143, 2013.

F. Lindsten, M. I. Jordan, and T. B. Schön. Ancestor sampling for particle Gibbs. In P. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS) 25*, pages 2600–2608. 2012.

F. Lindsten, P. Bunch, S. J. Godsill, and T. B. Schön. Rao-Blackwellized particle smoothers for mixed linear/nonlinear state-space models. In *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013.

J. S. Liu. Peskun's theorem and a modified discrete-state Gibbs sampler. *Biometrika*, 83 (3):681–682, 1996.

J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, 2001.

G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Whiley Series in Probability and Statistics. John Wiley & Sons, New York, USA, second edition, 2008.

L. M. Murray, E. M. Jones, and J. Parslow. On disturbance state-space models and the particle marginal Metropolis-Hastings sampler. *SIAM/ASA Journal on Uncertainty Quantification*, 1(1):494–521, 2013.

J. Olsson and T. Rydén. Rao-Blackwellization of particle Markov chain Monte Carlo methods using forward filtering backward sampling. *IEEE Transactions on Signal Processing*, 59(10):4606–4619, 2011.

O. Papaspiliopoulos, G. O. Roberts, and M. Sköld. Non-centered parameterisations for hierarchical models and data augmentation. In J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith, and M. West, editors, *Bayesian Statistics 7*, pages 307–326. Oxford University Press, 2003.

M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

M. K. Pitt, R. S. Silva, P. Giordani, and R. Kohn. Auxiliary particle filtering within adaptive Metropolis-Hastings sampling. arXiv.org, arXiv:1006.1914, June 2010.

M. K. Pitt, R. S. Silva, P. Giordani, and R. Kohn. On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171:134–151, 2012.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

D. A. Rasmussen, O. Ratmann, and K. Koelle. Inference for nonlinear epidemiological models using genealogies and time series. *PLoS Comput Biology*, 7(8), 2011.

B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman filter: particle filters for tracking applications*. Artech House, London, UK, 2004.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 2004.

N. Shephard, editor. *Stochastic Volatility: Selected Readings*. Oxford University Press, 2005.

A. Skvortsov and B. Ristic. Monitoring and prediction of an epidemic outbreak using syndromic observations. *Mathematical Biosciences*, 240(1):12–19, 2012.

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701–1728, 1994.

R. Turner and C. E. Deisenroth, M. P. Rasmussen. State-space inference and learning with Gaussian processes. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010.

D. N. VanDerwerken and S. C. Schmidler. Parallel Markov chain Monte Carlo. arXiv.org, arXiv:1312.7479, December 2013.

J. A. Vrugt, J. F. ter Braak, C. G. H. Diks, and G. Schoups. Hydrologic data assimilation using particle Markov chain Monte Carlo simulation: Theory, concepts and applications. *Advances in Water Resources*, 51:457–478, 2013.

G. C. G. Wei and M. A. Tanner. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85(411):699–704, 1990.

N. Whiteley. Discussion on Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B*, 72(3):306–307, 2010.

N. Whiteley, C. Andrieu, and A. Doucet. Efficient Bayesian inference for switching state-space models using discrete particle Markov chain Monte Carlo methods. Technical report, Bristol Statistics Research Report 10:04, 2010.

D. J. Wilkinson. Parallel Bayesian computation. In *Handbook of Parallel Computing and Statistics*. Chapman & Hall/CRC, 2005.

Journal of Machine Learning Research 15 (2014) 2185-2211 Submitted 7/13; Revised 11/13; Published 6/14

Ramp Loss Linear Programming Support Vector Machine

Xiaolin Huang HUANGXL06@MAILS.TSINGHUA.EDU.CN
Department of Electrical Engineering, ESAT-STADIUS, KU Leuven
Kasteelpark Arenberg 10, Leuven, B-3001, Belgium
Lei Shi LEISHI@FUDAN.EDU.CN
Department of Electrical Engineering, ESAT-STADIUS, KU Leuven
School of Mathematical Sciences, Fudan University, Shanghai, 200433, P.R. China
Johan A.K. Suykens JOHAN.SUYKENS@ESAT.KULEUVEN.BE
Department of Electrical Engineering, ESAT-STADIUS, KU Leuven
Kasteelpark Arenberg 10, Leuven, B-3001, Belgium

Editor: Mikhail Belkin

Abstract

The ramp loss is a robust but non-convex loss for classification. Compared with other non-convex losses, a local minimum of the ramp loss can be effectively found. The effectiveness of local search comes from the piecewise linearity of the ramp loss. Motivated by the fact that the ℓ_1-penalty is piecewise linear as well, the ℓ_1-penalty is applied for the ramp loss, resulting in a ramp loss linear programming support vector machine (ramp-LPSVM). The proposed ramp-LPSVM is a piecewise linear minimization problem and the related optimization techniques are applicable. Moreover, the ℓ_1-penalty can enhance the sparsity. In this paper, the corresponding misclassification error and convergence behavior are discussed. Generally, the ramp loss is a truncated hinge loss. Therefore ramp-LPSVM possesses some similar properties as hinge loss SVMs. A local minimization algorithm and a global search strategy are discussed. The good optimization capability of the proposed algorithms makes ramp-LPSVM perform well in numerical experiments: the result of ramp-LPSVM is more robust than that of hinge SVMs and is sparser than that of ramp-SVM, which consists of the $\|\cdot\|_{\mathcal{K}}$-penalty and the ramp loss.

Keywords: support vector machine, ramp loss, ℓ_1-regularization, generalization error analysis, global optimization

1. Introduction

In a binary classification problem, the input space is a compact subset $X \subset \mathbb{R}^n$ and the output space $Y = \{-1, 1\}$ represents two classes. Classification algorithms produce binary classifiers $\mathcal{C} : X \to Y$ induced by real-valued functions $f : X \to \mathbb{R}$ as $\mathcal{C} = \mathrm{sgn}(f)$, where the sign function is defined by $\mathrm{sgn}(f(x)) = 1$ if $f(x) \geq 0$ and $\mathrm{sgn}(f(x)) = -1$ otherwise. Since proposed by Cortes and Vapnik (1995), the support vector machine (SVM) has become a popular classification method, because of its good statistical property and generalization capability. SVM is usually based on a Mercer kernel $\mathcal{K}$ to produce non-linear classifiers. Such a kernel is a continuous, symmetric, and positive semi-definite function defined on $X \times X$. Given training data $\mathbf{z} = \{x_i, y_i\}_{i=1}^m$ with $x_i \in X, y_i \in Y$ and a loss function $L : \mathbb{R} \to \mathbb{R}^+$, in the functional analysis setting, SVM can be formulated as the following

optimization problem

$$\min_{f\in\mathcal{H}_\mathcal{K},b\in\mathbb{R}} \quad \frac{\mu}{2}\|f\|_\mathcal{K}^2 + \frac{1}{m}\sum_{i=1}^{m} L(1-y_i(f(x_i)+b)), \tag{1}$$

where $\mathcal{H}_\mathcal{K}$ is the Reproducing Kernel Hilbert Space (RKHS) induced by the Mercer kernel $\mathcal{K}$ with the norm $\|\cdot\|_\mathcal{K}$ (Aronszajn, 1950) and $\mu > 0$ is a trade-off parameter. The constant term b is called offset, which leads to much flexibility. The corresponding binary classifier is evaluated based on the optima of (1) by its sign function. Traditionally, the hinge loss $L_{\text{hinge}}(u) = \max\{u, 0\}$ is used. Besides, the squared hinge loss (Vapnik, 1998) and the least squares loss (Suykens and Vandewalle, 1999; Suykens et al., 2002) also have been widely applied. In classification and the related methodologies, robustness to outliers is always an important issue. The influence function (see, e.g., Steinwart and Christmann, 2008; De Brabanter et al., 2009) related to the hinge loss is bounded, which means that the effect of outliers on the result of minimizing the hinge loss is bounded. Though the effect is bounded, it can be significantly large since the penalty given to the outliers by the hinge loss is quite huge. In fact, any convex loss is unbounded. To remove the effect of outliers, researchers turn to some non-convex losses, such as the hard-margin loss, the normalized sigmoid loss (Mason et al., 2000), the ψ-learning loss (Shen et al., 2003), and the ramp loss (Collobert et al., 2006a,b). The ramp loss is defined as follows,

$$L_{\text{ramp}}(u) = \begin{cases} L_{\text{hinge}}(u), & u \le 1, \\ 1, & u > 1, \end{cases}$$

which is also called a truncated hinge loss in Wu and Liu (2007). The plots of the mentioned losses are illustrated in Figure 1, showing the robustness of these non-convex losses.

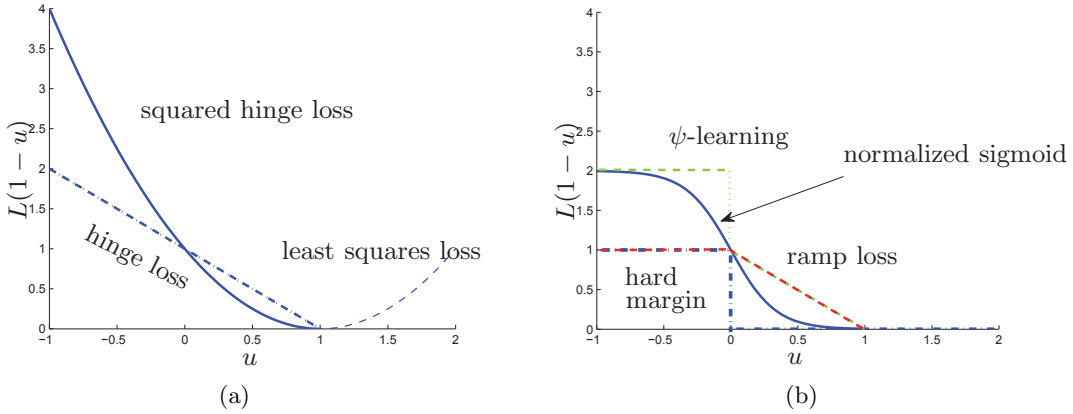


Figure 1: Plots of losses used for classification: (a) convex losses: the hinge loss (dash-dotted line), the squared hinge loss (solid line), and the least squares loss (dashed line); (b) robust but non-convex losses: the hard margin loss (blue dash-dotted line), the ψ-learning loss (green dashed line), the normalized sigmoid loss (blue solid line), and the ramp loss (red dashed line).

Among the mentioned robust but non-convex losses, the ramp loss is an attractive one. Using the ramp loss in (1), one obtains a ramp loss support vector machine (ramp-SVM). Because the ramp loss can be easily written as a difference of convex functions (DC), algorithms based on DC programming are applicable for ramp-SVM. The discussion about DC programming can be found in An et al. (1996), Horst and Thoai (1999), and An and Tao (2005). To apply DC programming in the ramp loss, we first observe the identity

$$L_{\text{ramp}}(u) = \min\{\max\{u, 0\}, 1\} = \max\{u, 0\} - \max\{u - 1, 0\}. \tag{2}$$

Therefore, SVM (1) with $L = L_{\text{ramp}}$ can be decomposed into the convex part $\frac{\mu}{2}\|f\|_{\mathcal{K}}^2 + \frac{1}{m}\sum_{i=1}^m \max\{1 - y_i(f(x_i) + b), 0\}$ and the concave part $-\frac{1}{m}\sum_{i=1}^m \max\{-y_i(f(x_i) + b), 0\}$. Hence DC programming can be used for finding a local minimizer of this problem, which has been applied by Collobert et al. (2006a), Wu and Liu (2007). DC programming for ramp-SVM is also referred to as a concave-convex procedure by Yuille and Rangarajan (2003). Besides the continuous optimization methods, ramp-SVM has been formulated as a mixed integer optimization problem by Brooks (2011) as below,

$$\begin{array}{ll} \min\limits_{f\in\mathcal{H}_{\mathcal{K}}, b\in\mathbb{R}, \omega} & \frac{\mu}{2}\|f\|_{\mathcal{K}}^2 + \frac{1}{m}\sum_{i=1}^m (e_i + \omega_i) \\ \text{s.t.} & \omega_i \in \{0, 1\}, \\ & 0 \le e_i \le 1, \quad i = 1, \cdots, m, \\ & y_i(f(x_i) + b) \ge 1 - e_i, \quad \text{if } \omega_i = 0. \end{array} \tag{3}$$

The optimization problem (3) should be solved over all possible binary vectors $\omega = [\omega_1, \ldots, \omega_m]^T \in \{0, 1\}^m$. Once the binary vector ω is given, this problem can be solved by quadratic programming. Consequently, when the size of the problem grows, the computation time explodes.

It is worth noting the case of taking $L = L_{\text{hinge}}$ in (1). It corresponds to the well-known C-SVM. One can solve C-SVM by its dual form, then the output function is represented as $\sum_{i=1}^m \nu_i^* y_i \mathcal{K}(x, x_i) + b^*$, where $[\nu_1^*, \cdots, \nu_m^*]^T$ is the optimal solution of

$$\begin{array}{ll} \min\limits_{\nu_i\in\mathbb{R}} & \frac{1}{2}\sum_{i,j=1}^m \nu_i\nu_j y_i y_j \mathcal{K}(x_i, x_j) - \sum_{i=1}^m \nu_i \\ \text{s.t.} & \sum_{i=1}^m \nu_i y_i = 0, \\ & 0 \le \nu_i \le \frac{1}{\mu m}, \quad i = 1, \cdots, m. \end{array}$$

The optimal offset b^* can be computed from the Karush-Kuhn-Tucker (KKT) conditions after $\{\nu_i^*\}_{i=1}^m$ is found (see, e.g., Suykens et al., 2002). From the dual form of C-SVM, we find that though we search the function f in a rather large space $\mathcal{H}_{\mathcal{K}}$, the optimal solution actually belongs to a finite-dimensional subspace given by $\mathcal{H}_{\mathcal{K},\mathbf{z}}^+$ with

$$\mathcal{H}_{\mathcal{K},\mathbf{z}}^+ = \left\{\sum_{i=1}^m \alpha_i y_i \mathcal{K}(x, x_i), \forall \alpha = [\alpha_1, \ldots, \alpha_m]^T \succeq 0\right\}.$$

Here the notation $\succeq 0$ means all the elements of the vector being non-negative.

To enhance the sparsity in the output function, the linear programming support vector machine (LPSVM) directly minimizes the data fitting term $\frac{1}{m}\sum_{i=1}^m L_{\text{hinge}}(1 - y_i(f(x_i) + b))$ with a ℓ_1-penalty term (see Vapnik, 1998; Smola et al., 1999). Given $f \in \mathcal{H}_{\mathcal{K},\mathbf{z}}^+$, the ℓ_1-penalty is defined as

$$\Omega(f) = \sum_{i=1}^m \alpha_i, \text{ for } f = \sum_{i=1}^m \alpha_i y_i \mathcal{K}(x, x_i), \tag{4}$$

which is the ℓ_1-penalty of the combinatorial coefficients of f. Then LPSVM can be formulated as follows,

$$\min_{f\in\mathcal{H}_{\mathcal{K}}^{+},b\in\mathbb{R}} \quad \mu\Omega(f)+\frac{1}{m}\sum_{i=1}^{m}L_{\text{hinge}}(1-y_i(f(x_i)+b)). \tag{5}$$

LPSVM is also related to 1-norm SVM proposed by Zhu et al. (2004), which searches a linear combination of basis functions and does not consider the non-negative constraint. The properties of LPSVM have been demonstrated in the literature (e.g., Bradley and Mangasarian, 2000; Kecman and Hadzic, 2000). Generalization error analysis for LPSVM can be found in Wu and Zhou (2005).

For problem (1), one can choose different penalty terms and different loss functions. For example, using $\|f\|_{\mathcal{K}}$ together with the hinge loss, we obtain C-SVM. The property of C-SVM can be observed from the properties of $\|f\|_{\mathcal{K}}$ and the hinge loss: since $\|f\|_{\mathcal{K}}$ is a quadratic function and the hinge loss is piecewise linear (pwl), the objective function of C-SVM is piecewise quadratic (pwq) and can be solved by constrained quadratic programming. For LPSVM, which consists of the ℓ_1-penalty and the hinge loss, the objective function is convex piecewise linear and hence can be minimized by linear programming. In Table 1, we summarize the properties of several penalties and losses.

	$\|f\|_{\mathcal{K}}$	$\Omega(f)$	hinge	squared hinge	least squares	ψ-learning	normalized sigmoid	ramp
function type	quadratic	pwl	pwl	pwq	quadratic	discontinuous	log	pwl
convexity	√	√	√	√	√	×	×	×
continuity	√	√	√	√	√	×	√	√
smoothness	√	×	×	√	√	×	√	×
sparsity	×	√	√	√	×	√	×	√
bounded influence fun.	—	—	√	×	×	√	√	√
bounded penalty value	—	—	×	×	×	√	√	√

* "pwl" stands for piecewise linear; "pwq" stands for piecewise quadratic.

Table 1: Properties of Different Penalties and Losses

The ramp loss gives a constant penalty for any large outlier and it is obviously robust. From Table 1, we observe that both $\Omega(f)$ and the ramp loss are continuous piecewise linear. It follows that if we choose $\Omega(f)$ and the ramp loss, the objective function of (1) is continuous piecewise linear and can be minimized by linear programming. Besides, minimizing $\Omega(f)$ enhances the sparsity. Motivated by this observation, in this paper we study the binary classifiers generated by minimizing the ramp loss and the ℓ_1-penalty, which is called a ramp loss linear programming support vector machine (ramp-LPSVM). The ramp-LPSVM has the following formulation,

$$(f^*_{\mathbf{z},\mu},b^*_{\mathbf{z},\mu})=\operatorname*{argmin}_{f\in\mathcal{H}_{\mathcal{K},\mathbf{z}}^{+},b\in\mathbb{R}} \quad \mu\Omega(f)+\frac{1}{m}\sum_{i=1}^{m}L_{\text{ramp}}(1-y_i(f(x_i)+b)), \tag{6}$$

where $\Omega(\cdot)$ is the ℓ_1-penalty defined by (4). And the induced classifier is given by $\mathrm{sgn}(f^*_{\mathbf{z},\mu} + b^*_{\mathbf{z},\mu})$. We call (6) ramp-LPSVM, which implies that the algorithm proposed later involves linear programming problems. Similarly to ramp-SVM, the proposed ramp-LPSVM enjoys robustness. Moreover, it can give a sparser solution. In addition to enhancing the sparsity, replacing the $\|\cdot\|_{\mathcal{K}}$-penalty in ramp-SVM by the ℓ_1-penalty is mainly motivated by the fact that both the ramp loss and the ℓ_1-penalty are piecewise linear, which helps developing more efficient algorithms.

Resulting from the identity (2), the problem related to ramp-LPSVM leads to a polyhedral concave problem, which minimizes a concave function on one polyhedron. A polyhedral concave problem is easier to handle than a regular non-convex problem and some efficient methods were reviewed by Horst and Hoang (1996). Moreover, ramp-LPSVM (6) has a piecewise linear objective function. For such kind of problems, a hill detouring technique proposed by Huang et al. (2012a) has shown good global search capability. As the name suggests, the hill detouring method searches on the level set to escape from a local optimum. One contribution of this paper is that we establish algorithms for solving ramp-LPSVM (6), including DC programming for local minimization and hill detouring for global search. Additionally, we investigate the asymptotic performance of ramp-LPSVM under the framework of statistical learning theory. Our analysis implies that ramp-LPSVM has a similar misclassification error bound and similar convergence behavior as C-SVM. Moreover, one can expect that the output binary classifier of algorithm (6) is robust, due to the ramp loss, and has a sparse representation, due to the ℓ_1-penalty.

The remainder of the paper is organized as follows: some statistical properties for the proposed ramp-LPSVM are discussed in Section 2. In Section 3, we establish problem-solving algorithms including DC programming for local minimization, and hill detouring for escaping from local optima. The proposed algorithms are tested then on numerical experiments in Section 4. Section 5 ends the paper with concluding remarks.

2. Theoretical Properties

In this section, we establish the theoretical analysis for ramp-LPSVM under the framework of statistical learning theory. In the following, we first show that the ramp loss is classification calibrated; see Proposition 1. In other works, we prove that minimizing the ramp loss results in the Bayes classifier. After that, an inequality is presented in Theorem 2 to bound the difference between the risk of the Bayes classifier and that of the classifier induced from minimizing the ramp loss. Finally, we obtain the convergence behavior of ramp-LPSVM, which is given in Theorem 5. To prove Theorem 5, error decomposition theorems for ramp-SVM and ramp-LPSVM are discussed. The analysis on the ramp loss is closely related to the properties of the hinge loss, because the ramp loss can be regarded as a truncated hinge loss. In our analysis, the global minimizer of the ramp loss plays an important role, which motivates us to establish a global search strategy in the next section.

To this end, we assume that the sample $\mathbf{z} = \{x_i, y_i\}_{i=1}^m$ is independently drawn from a probability measure ρ on $X \times Y$. The misclassification error for a binary classifier $\mathcal{C} : X \to Y$ is defined as the probability of the event $\mathcal{C}(x) \neq y$:

$$\mathcal{R}(\mathcal{C}) = \int_{X\times Y} \mathcal{I}_{y\neq\mathcal{C}(x)} d\rho = \int_X \rho(y \neq \mathcal{C}(x)|x) d\rho_X,$$

where $\mathcal{I}$ is the indicator function, ρ_X is the marginal distribution of ρ on X, and $\rho(y|x)$ is the conditional distribution of ρ at given x. It should be pointed out that $\rho(y|x)$ is a binary distribution, which is given by $\text{Prob}(y = 1|x)$ and $\text{Prob}(y = -1|x)$. The classifier that minimizes the misclassification error is the Bayes rule f_c, which is defined as,

$$f_c = \arg \min_{\mathcal{C}:X\to Y} \mathcal{R}(\mathcal{C}).$$

The Bayes rule can be evaluated as

$$f_c(x) = \begin{cases} 1, & \text{if } \text{Prob}(y = 1|x) \geq \text{Prob}(y = -1|x), \\ -1, & \text{if } \text{Prob}(y = 1|x) < \text{Prob}(y = -1|x). \end{cases}$$

The performance of a binary classifier induced by a real-valued function f is measured by the excess misclassification error $\mathcal{R}(\text{sgn}(f)) - \mathcal{R}(f_c)$. Let $f_{\mathbf{z},\mu} = f^*_{\mathbf{z},\mu} + b^*_{\mathbf{z},\mu}$ with $(f^*_{\mathbf{z},\mu}, b^*_{\mathbf{z},\mu})$ being the global minimizer of ramp-LPSVM (6). The purpose of the theoretical analysis is to estimate $\mathcal{R}(\text{sgn}(f_{\mathbf{z},\mu})) - \mathcal{R}(f_c)$ as the sample size m tends to infinity. Convergence rates will be derived under the choice of the parameter μ and conditions on the distribution ρ.

As an important ingredient in classification algorithms, the loss function L is used to model the target function of interest. Concretely, the target function denoted as $f_{L,\rho}$ minimizes the expected L-risk

$$\mathcal{R}_{L,\rho}(f) = \int_{X\times Y} L(1 - yf(x))d\rho$$

over all possible functions $f : X \to \mathbb{R}$ and can be defined pointwisely as below,

$$f_{L,\rho}(x) = \arg\min_{t\in\mathbb{R}} \int_Y L\,(1 - yt)\, d\rho(y|x),\ \forall x \in X.$$

The basic idea on designing algorithms is to replace the unknown true risk $\mathcal{R}_{L,\rho}$ by the empirical L-risk

$$\mathcal{R}_{L,\mathbf{z}}(f) = \frac{1}{m}\sum_{i=1}^{m} L(1 - y_i f(x_i)), \tag{7}$$

and to minimize this empirical risk (or its penalized version) over a suitable function class. When the hard margin loss, which counts the number of misclassification,

$$L_{\text{mis}}(u) = \begin{cases} 0, & u \geq 0, \\ 1, & u < 0, \end{cases}$$

is used, one can check that for any binary classifier $\mathcal{C} : X \to Y$, there holds $\mathcal{R}(\mathcal{C}) = \mathcal{R}_{L_{\text{mis}},\rho}(\mathcal{C})$. Therefore, the excess misclassification error can be written as

$$\mathcal{R}_{L_{\text{mis}},\rho}(\text{sgn}(f)) - \mathcal{R}_{L_{\text{mis}},\rho}(f_c).$$

However, the empirical algorithms based on L_{mis} will lead to NP-hard optimization problems, and thus it is not computationally realizable. One way to resolve this issue is to use surrogate loss functions as discussed in Section 1, and then to minimize the empirical

risk associated with the used surrogate loss. Among these losses, the hinge loss plays an important role, since one has $f_{L_{\text{hinge}},\rho} = f_c$.

Now we investigate the ramp loss. For a given $x \in X$, a simple calculation shows that

$$\begin{aligned} & \int_Y L_{\text{ramp}}\,(1-yt)\,d\rho(y|x) \\ = \;& L_{\text{ramp}}(1-t)\text{Prob}(y=1|x) + L_{\text{ramp}}(1+t)\text{Prob}(y=-1|x) \\ = \;& \begin{cases} \text{Prob}(y=1|x), & t \le -1, \\ \text{Prob}(y=1|x) + (1+t)\text{Prob}(y=-1|x), & -1 < t \le 0, \\ (1-t)\text{Prob}(y=1|x) + \text{Prob}(y=-1|x), & 0 \le t < 1, \\ \text{Prob}(y=-1|x), & t \ge 1. \end{cases} \end{aligned}$$

Obviously, when $\text{Prob}(y=1|x) > \text{Prob}(y=-1|x)$, the minimal value is $\text{Prob}(y=-1|x)$, which is achieved by $t=1$. When $\text{Prob}(y=1|x) < \text{P}(y=-1|x)$, the minimal value is $\text{Prob}(y=1|x)$, which is achieved by $t=-1$. Therefore, the corresponding target function $f_{L_{\text{ramp}},\rho}$ that minimizes the expected L_{ramp}-risk is the Bayes rule. The discussion above can be concluded in the following proposition.

Proposition 1 *For any measurable function $f : X \to \mathbb{R}$, there holds*

$$\mathcal{R}_{L_{\text{ramp}},\rho}(f) \ge \mathcal{R}_{L_{\text{ramp}},\rho}(f_c).$$

That is, the Bayes rule f_c is a minimizer of the expected L_{ramp}-risk.

Next, for a real-valued function $f : X \to \mathbb{R}$, we consider bounding the excess misclassification error by the generalization error $\mathcal{R}_{L_{\text{ramp}},\rho}(f) - \mathcal{R}_{L_{\text{ramp}},\rho}(f_{L_{\text{ramp}},\rho})$. Such kind of bound plays an essential role in error analysis of classification algorithms. When the loss function is convex and satisfies some regularity conditions, the corresponding bound is the so-called self-calibration inequality and has been established by Bartlett et al. (2006) and Steinwart (2007). For example, a typical result presented in Cucker and Zhou (2007) claims that, if a general loss function satisfies the following conditions:

- $L(1-u)$ is convex with respect to u;
- $L(1-u)$ is differentiable at $u=0$ and $\frac{dL(1-u)}{du}|_{u=0} < 0$;
- $\min\{u : L(1-u) = 0\} = 1$;
- $\frac{d^2L(1-u)}{du^2}|_{u=1} > 0$,

then there exists a constant $c_L > 0$ such that for any measurable function $f : X \to \mathbb{R}$,

$$\mathcal{R}_{L_{\text{mis}},\rho}(\text{sgn}(f)) - \mathcal{R}_{L_{\text{mis}},\rho}(f_c) \le c_L\sqrt{\mathcal{R}_{L,\rho}(f) - \mathcal{R}_{L,\rho}(f_{L,\rho})}. \tag{8}$$

This inequality holds for many loss functions, such as the hinge loss, the squared hinge loss, and the least squares loss. For the hinge loss L_{hinge}, Zhang (2004) gave a tighter bound by the following inequality,

$$\mathcal{R}_{L_{\text{mis}},\rho}(\text{sgn}(f)) - \mathcal{R}_{L_{\text{mis}},\rho}(f_c) \leq \mathcal{R}_{L_{\text{hinge}},\rho}(f) - \mathcal{R}_{L_{\text{hinge}},\rho}(f_{L_{\text{hinge}},\rho}).$$

The improvement is mainly due to the property that $\mathcal{R}_{L_{\text{hinge}},\rho}(f_{L_{\text{hinge}},\rho}) = \mathcal{R}_{L_{\text{hinge}},\rho}(f_c)$.

For the ramp loss L_{ramp}, we cannot directly use the conclusion given by (8), since the loss is not convex. However, as L_{ramp} can be considered as a truncated hinge loss and maintains the same property due to Proposition 1, one thus can establish a similar inequality for the ramp loss.

Theorem 2 *For any probability measure ρ and any measurable function $f : X \to \mathbb{R}$,*

$$\mathcal{R}_{L_{\text{mis}},\rho}(\text{sgn}(f)) - \mathcal{R}_{L_{\text{mis}},\rho}(f_c) \leq \mathcal{R}_{L_{\text{ramp}},\rho}(f) - \mathcal{R}_{L_{\text{ramp}},\rho}(f_{L_{\text{ramp}},\rho}). \tag{9}$$

Proof By Proposition 1, we have $\mathcal{R}_{L_{\text{ramp}},\rho}(f_{L_{\text{ramp}},\rho}) = \mathcal{R}_{L_{\text{ramp}},\rho}(f_c)$. Since y and $f_c(x)$ belong to $\{-1, 1\}$, $1-yf_c(x)$ takes value of 0 or 2. We hence have $\mathcal{R}_{L_{\text{mis}},\rho}(f_c) = \mathcal{R}_{L_{\text{ramp}},\rho}(f_c)$, which comes from the fact that

$$L_{\text{mis}}(0) = L_{\text{ramp}}(0) \quad \text{and} \quad L_{\text{mis}}(2) = L_{\text{ramp}}(2).$$

Thus, to prove (9), we need to show that

$$\mathcal{R}_{L_{\text{mis}},\rho}(\text{sgn}(f)) \leq \mathcal{R}_{L_{\text{ramp}},\rho}(f), \tag{10}$$

which is equivalent to

$$\int_{X\times Y} L_{\text{mis}}\Big(1 - y\text{sgn}(f(x))\Big) - L_{\text{ramp}}\Big(1 - yf(x)\Big)d\rho \leq 0.$$

For any y and $f(x)$, if $yf(x) \leq 0$, then $y\text{sgn}(f(x)) \leq 0$, which follows that $L_{\text{mis}}(1 - y\text{sgn}(f(x))) = L_{\text{ramp}}(1 - yf(x)) = 1$. If $yf(x) > 0$, then we have $y\text{sgn}(f(x)) = 1$ and $L_{\text{mis}}(1 - y\text{sgn}(f(x))) = 0$. Since $L_{\text{ramp}}(1 - yf(x))$ is always nonnegative, we have $L_{\text{mis}}(1 - y\text{sgn}(f(x))) - L_{\text{ramp}}(1 - yf(x)) \leq 0$ for this case.

Summarizing the above discussion, we prove (10) and then Theorem 2. ■

From Theorem 2, in order to estimate $\mathcal{R}_{L_{\text{mis}},\rho}(\text{sgn}(f_{\mathbf{z},\mu})) - \mathcal{R}_{L_{\text{mis}},\rho}(f_c)$, we turn to bound $\mathcal{R}_{L_{\text{ramp}},\rho}(f_{\mathbf{z},\mu}) - \mathcal{R}_{L_{\text{ramp}},\rho}(f_c)$. We thus need an error decomposition for the latter. This decomposition process is well-developed in the literature for RKHS-based regularization schemes (see, e.g., Cucker and Zhou, 2007; Steinwart and Christmann, 2008). To explain the details, we take ramp-SVM below as an example. For $\mathbf{z} = \{x_i, y_i\}_{i=1}^m$ and $\lambda > 0$, let $\tilde{f}_{\mathbf{z},\lambda} = \tilde{f}^*_{\mathbf{z},\lambda} + \tilde{b}^*_{\mathbf{z},\lambda}$, where

$$(\tilde{f}^*_{\mathbf{z},\lambda}, \tilde{b}^*_{\mathbf{z},\lambda}) = \underset{f\in\mathcal{H}_K, b\in\mathbb{R}}{\text{argmin}} \ \frac{\lambda}{2}\|f\|_K^2 + \frac{1}{m}\sum_{i=1}^m L_{\text{ramp}}(1 - y_i(f(x_i) + b)). \tag{11}$$

Then the following decomposition holds true:

$$\begin{aligned}\mathcal{R}_{L_{\text{ramp}},\rho}(\tilde{f}_{\mathbf{z},\lambda}) - \mathcal{R}_{L_{\text{ramp}},\rho}(f_c) \;\leq\; & \Big\{\mathcal{R}_{L_{\text{ramp}},\rho}(\tilde{f}_{\mathbf{z},\lambda}) - \mathcal{R}_{L_{\text{ramp}},\mathbf{z}}(\tilde{f}_{\mathbf{z},\lambda})\Big\} \\ & + \big\{\mathcal{R}_{L_{\text{ramp}},\mathbf{z}}(f_\lambda) - \mathcal{R}_{L_{\text{ramp}},\rho}(f_\lambda)\big\} + \mathcal{A}(\lambda),\end{aligned}$$

where $\mathcal{R}_{L_{\text{ramp}},\mathbf{z}}(f)$ is the empirical L_{ramp}-risk given by (7). The function f_λ depends on λ and is defined by the data-free limit of (11), that is $f_\lambda = f_\lambda^* + b_\lambda^*$ with

$$(f_\lambda^*, b_\lambda^*) = \operatorname*{argmin}_{f\in\mathcal{H}_\mathcal{K}, b\in\mathbb{R}} \; \frac{\lambda}{2}\|f\|_\mathcal{K}^2 + \mathcal{R}_{\text{ramp},\rho}(f+b). \tag{12}$$

The term $\mathcal{A}(\lambda)$ measures the approximation power of the system $(\mathcal{K},\rho)$ and is defined by

$$\mathcal{A}(\lambda) = \inf_{f\in\mathcal{H}_\mathcal{K}, b\in\mathbb{R}} \; \frac{\lambda}{2}\|f\|_\mathcal{K}^2 + \mathcal{R}_{\text{ramp},\rho}(f+b) - \mathcal{R}_{\text{ramp},\rho}(f_c), \quad \forall \lambda > 0. \tag{13}$$

It is easy to establish such kind of decomposition if one notices the fact that both $\tilde{f}_{\mathbf{z},\lambda}$ and f_λ lie in the same function space. However, it is not the case for ramp-LPSVM. The data-dependent nature of $\mathcal{H}_{\mathcal{K},\mathbf{z}}^+$ leads to an essential difficulty in the error analysis. Motivated by Wu and Zhou (2005), we shall establish the error decomposition for ramp-LPSVM (6) with the aid of $\tilde{f}_{\mathbf{z},\lambda}$. To this end, we first show some properties of $\tilde{f}_{\mathbf{z},\lambda}$, which play an important role in our analysis.

Proposition 3 *For any $\lambda > 0$, $(\tilde{f}_{\mathbf{z},\lambda}^*, \tilde{b}_{\mathbf{z},\lambda}^*)$ is given by (11) and $\tilde{f}_{\mathbf{z},\lambda} = \tilde{f}_{\mathbf{z},\lambda}^* + \tilde{b}_{\mathbf{z},\lambda}^*$. Then $\tilde{f}_{\mathbf{z},\lambda}^* \in \mathcal{H}_{\mathcal{K},\mathbf{z}}^+$ and*

$$\Omega(\tilde{f}_{\mathbf{z},\lambda}^*) \leq \lambda^{-1}\mathcal{R}_{L_{\text{ramp}},\mathbf{z}}(\tilde{f}_{\mathbf{z},\lambda}) + \|\tilde{f}_{\mathbf{z},\lambda}^*\|_\mathcal{K}^2. \tag{14}$$

Proof Following the idea of Brooks (2011), one can formulate the minimization problem (11) as a mixed integer optimization problem, which is given by (3) with $\mu = \lambda$. We first show that if the binary vector $\omega^* = [\omega_1^*, \cdots, \omega_m^*]^T \in \{0,1\}^m$ is optimal for the optimization problem (3), then the global minimizer of (11) can be obtained by solving the following minimization problem

$$\begin{array}{ll}\min\limits_{f\in\mathcal{H}_\mathcal{K}, e_i, b\in\mathbb{R}} & \frac{\lambda}{2}\|f\|_\mathcal{K}^2 + \frac{1}{m}\sum_{i=1}^m e_i \\ \text{s.t.} & e_i \geq 0, \quad i = 1,\cdots,m, \\ & y_i(f(x_i)+b) \geq 1 - e_i, \quad \text{if } \omega_i^* = 0.\end{array} \tag{15}$$

In fact, when the optimal ω^* is given, the global minimizer of (11) can be solved by the optimization problem (3), which is reduced to

$$\begin{array}{ll}\min\limits_{f\in\mathcal{H}_\mathcal{K}, e_i, b\in\mathbb{R}} & \frac{\lambda}{2}\|f\|_\mathcal{K}^2 + \frac{1}{m}\sum_{i=1}^m e_i \\ \text{s.t.} & 0 \leq e_i \leq 1, \quad i = 1,\cdots,m, \\ & y_i(f(x_i)+b) \geq 1 - e_i, \quad \text{if } \omega_i^* = 0.\end{array} \tag{16}$$

Let $e^* = [e_1^*, \cdots, e_m^*]^T$ be the optimal slack variables in the above minimization problem. Then the triple $(\tilde{f}_{\mathbf{z},\lambda}^*, \tilde{b}_{\mathbf{z},\lambda}^*, e^*)$ is the optimal solution of minimization problem (16). Correspondingly, denote $(\tilde{f}_{\mathbf{z},\lambda}^1, \tilde{b}_{\mathbf{z},\lambda}^1, e^{*1})$ as the optimal solution of minimization problem (15)

with $e^{*1} = [e_1^{*1}, \cdots, e_m^{*1}]^T$. As the constraints in problem (16) is a subset of that in problem (15), we thus have

$$\frac{\lambda}{2}\|\tilde{f}^1_{\mathbf{z},\lambda}\|^2_{\mathcal{K}} + \frac{1}{m}\sum_{i=1}^m e_i^{*1} \leq \frac{\lambda}{2}\|\tilde{f}^*_{\mathbf{z},\lambda}\|^2_{\mathcal{K}} + \frac{1}{m}\sum_{i=1}^m e_i^*.$$

To prove our claim, we just need to verify that $0 \leq e_i^{*1} \leq 1$ for $i = 1, \cdots, m$. For $\omega_i^* = 1$, it is easy to see that $e_i^{*1} = 0$. Next we prove the conclusion for the case $\omega_i^* = 0$. Define an index set as $I := \{i \in \{1, \cdots, m\} : \omega_i^* = 0 \text{ and } e_i^{*1} > 1\}$. If I is an non-empty set, we further define a binary vector ω' with $\omega'_i = 1$ for $i \in I$ and $\omega'_i = \omega_i^*$ otherwise. As $\omega_i = 1$ implies the corresponding optimal e_i should equal 0, we then define e'_i as $e'_i = 0$ if $\omega'_i = 1$ and $e'_i = e_i^{*1}$ otherwise. One can check that

$$\frac{\lambda}{2}\|\tilde{f}^1_{\mathbf{z},\lambda}\|^2_{\mathcal{K}} + \frac{1}{m}\sum_{i=1}^m (e'_i + \omega'_i) < \frac{\lambda}{2}\|\tilde{f}^1_{\mathbf{z},\lambda}\|^2_{\mathcal{K}} + \frac{1}{m}\sum_{i=1}^m (e_i^{*1} + \omega_i^*) \leq \frac{\lambda}{2}\|\tilde{f}^*_{\mathbf{z},\lambda}\|^2_{\mathcal{K}} + \frac{1}{m}\sum_{i=1}^m (e_i^* + \omega_i^*).$$

We thus derive a contradiction to the assumption that $(\tilde{f}^*_{\mathbf{z},\lambda}, \tilde{b}^*_{\mathbf{z},\lambda}, e^*, \omega^*)$ is a global optimal solution for problem (3) and the conclusion follows.

Now we can prove our desired result based on the optimization problem (15). Let $I_0 = \{i : \omega_i^* = 0\}$ and $I_1 = \{i : \omega_i^* = 1\}$. Since the triple $(\tilde{f}^*_{\mathbf{z},\lambda}, \tilde{b}^*_{\mathbf{z},\lambda}, e^*)$ is the optimal solution of problem (15), from the KKT condition, there exist constants $\{\tilde{\alpha}_i^*\}_{i\in I_0}$, such that

$$\begin{aligned}
&\tilde{f}^*_{\mathbf{z},\lambda}(x) = \sum_{i\in I_0} \tilde{\alpha}_i^* y_i K(x_i, x) \text{ with } 0 \leq \tilde{\alpha}_i^* \leq \frac{1}{\lambda m},\\
&\sum_{i\in I_0} \tilde{\alpha}_i^* y_i = 0,\\
&1 - y_i(\tilde{f}^*_{\mathbf{z},\lambda}(x_i) + \tilde{b}^*_{\mathbf{z},\lambda}) \leq 0, \quad \text{if } i \in I_0 \text{ and } \tilde{\alpha}_i^* = 0,\\
&0 \leq e_i^* = 1 - y_i(\tilde{f}^*_{\mathbf{z},\lambda}(x_i) + \tilde{b}^*_{\mathbf{z},\lambda}) \leq 1, \quad \text{if } i \in I_0 \text{ and } \tilde{\alpha}_i^* \neq 0.
\end{aligned}$$

We also have $e_i^* = 0$, if $i \in I_1$. Moreover, by the same argument used in the proof about the equivalence of problems (15) and (16), one can find that when $i \in I_1$, we must have $1 - y_i(\tilde{f}^*_{\mathbf{z},\lambda}(x_i) + \tilde{b}^*_{\mathbf{z},\lambda}) > 1$ or $1 - y_i(\tilde{f}^*_{\mathbf{z},\lambda}(x_i) + \tilde{b}^*_{\mathbf{z},\lambda}) < 0$ due to the optimality of ω^*.

From the expression of $\tilde{f}^*_{\mathbf{z},\lambda}$, we can write $\tilde{f}^*_{\mathbf{z},\lambda}$ as $\sum_{i=1}^m \alpha_i^* y_i K(x_i, x)$ with $\alpha_i^* = \tilde{\alpha}_i^*$ if $i \in I_0$ and $\alpha_i^* = 0$ otherwise. Then $\tilde{f}^*_{\mathbf{z},\lambda} \in \mathcal{H}^+_{\mathcal{K},\mathbf{z}}$. Furthermore, the relation $\sum_{i\in I_0} \tilde{\alpha}_i^* y_i = 0$ implies $\sum_{i\in I_0} \tilde{\alpha}_i^* y_i \tilde{b}^*_{\mathbf{z},\lambda} = 0$. Then we have

$$\Omega(\tilde{f}^*_{\mathbf{z},\lambda}) = \sum_{i\in I_0} \tilde{\alpha}_i^* = \sum_{i\in I_0} \tilde{\alpha}_i^*(1 - y_i(\tilde{f}^*_{\mathbf{z},\lambda}(x_i) + \tilde{b}^*_{\mathbf{z},\lambda})) + \sum_{i\in I_0} \tilde{\alpha}_i^* y_i \tilde{f}^*_{\mathbf{z},\lambda}(x_i).$$

Note that $\tilde{f}^*_{\mathbf{z},\lambda}(x) = \sum_{i\in I_0} \tilde{\alpha}_i^* y_i K(x_i, x)$. By the definition of $\|\cdot\|_{\mathcal{K}}$-norm, it follows that

$$\sum_{i\in I_0} \tilde{\alpha}_i^* y_i \tilde{f}^*_{\mathbf{z},\lambda}(x_i) = \sum_{i,j\in I_0} \tilde{\alpha}_i^* y_i \tilde{\alpha}_j^* y_j K(x_i, x_j) = \|\tilde{f}^*_{\mathbf{z},\lambda}\|^2_{\mathcal{K}}.$$

Additionally, based on our analysis, we also have

$$\sum_{i\in I_0}\tilde{\alpha}_i^*(1-y_i(\tilde{f}^*_{\mathbf{z},\lambda}(x_i)+\tilde{b}^*_{\mathbf{z},\lambda}))=\sum_{i\in I_0}\tilde{\alpha}_i^* L_{\mathrm{ramp}}(y_i(\tilde{f}^*_{\mathbf{z},\lambda}(x_i)+\tilde{b}^*_{\mathbf{z},\lambda}))\le\lambda^{-1}\mathcal{R}_{L_{\mathrm{ramp}},\mathbf{z}}(\tilde{f}_{\mathbf{z},\lambda}).$$

Hence the bound for $\Omega(\tilde{f}^*_{\mathbf{z},\lambda})$ follows. ■

Now we are in the position to make an error decomposition for ramp-LPSVM.

Theorem 4 *For $0<\mu\le\lambda\le 1$, let $\eta=\frac{\mu}{\lambda}$. Recall that $f_{\mathbf{z},\mu}=f^*_{\mathbf{z},\mu}+b^*_{\mathbf{z},\mu}$ where $(f^*_{\mathbf{z},\mu},b^*_{\mathbf{z},\mu})$ is a global minimizer of ramp-LPSVM (6) and $f_\lambda=f^*_\lambda+b^*_\lambda$ with $(f^*_\lambda,b^*_\lambda)$ given by (12). Define the sample error $\mathcal{S}(m,\mu,\lambda)$ as below,*

$$\mathcal{S}(m,\mu,\lambda)=\left\{\mathcal{R}_{L_{\mathrm{ramp}},\rho}(f_{\mathbf{z},\mu})-\mathcal{R}_{L_{\mathrm{ramp}},\mathbf{z}}(f_{\mathbf{z},\mu})\right\}+(1+\eta)\left\{\mathcal{R}_{L_{\mathrm{ramp}},\mathbf{z}}(f_\lambda)-\mathcal{R}_{L_{\mathrm{ramp}},\rho}(f_\lambda)\right\}.$$

Then there holds

$$\mathcal{R}_{L_{\mathrm{ramp}},\rho}(f_{\mathbf{z},\mu})-\mathcal{R}_{\mathrm{ramp},\rho}(f_c)+\mu\Omega(f^*_{\mathbf{z},\mu})\le\eta\mathcal{R}_{L_{\mathrm{ramp}},\rho}(f_c)+\mathcal{S}(m,\mu,\lambda)+2\mathcal{A}(\lambda),\qquad(17)$$

where $\mathcal{A}(\lambda)$ is the approximation error given by (13).

Proof Recall that for any $\lambda>0$, $\tilde{f}_{\mathbf{z},\lambda}=\tilde{f}^*_{\mathbf{z},\lambda}+\tilde{b}^*_{\mathbf{z},\lambda}$ where $(\tilde{f}^*_{\mathbf{z},\lambda},\tilde{b}^*_{\mathbf{z},\lambda})$ is given by (11). Due to the definition of $f_{\mathbf{z},\mu}$ and the fact $\tilde{f}^*_{\mathbf{z},\lambda}\in\mathcal{H}^+_{\mathcal{K},\mathbf{z}}$, we have

$$\mathcal{R}_{L_{\mathrm{ramp}},\mathbf{z}}(f_{\mathbf{z},\mu})+\mu\Omega(f^*_{\mathbf{z},\mu})\le\mathcal{R}_{L_{\mathrm{ramp}},\mathbf{z}}(\tilde{f}_{\mathbf{z},\lambda})+\mu\Omega(\tilde{f}^*_{\mathbf{z},\lambda}).$$

Proposition 3 gives

$$\Omega(\tilde{f}^*_{\mathbf{z},\lambda})\le\lambda^{-1}\mathcal{R}_{L_{\mathrm{ramp}},\mathbf{z}}(\tilde{f}_{\mathbf{z},\lambda})+\|\tilde{f}^*_{\mathbf{z},\lambda}\|^2_{\mathcal{K}}.$$

Hence,

$$\mathcal{R}_{L_{\mathrm{ramp}},\mathbf{z}}(f_{\mathbf{z},\mu})+\mu\Omega(f^*_{\mathbf{z},\mu})\le\left(1+\frac{\mu}{\lambda}\right)\mathcal{R}_{L_{\mathrm{ramp}},\mathbf{z}}(\tilde{f}_{\mathbf{z},\lambda})+\mu\|\tilde{f}^*_{\mathbf{z},\lambda}\|^2_{\mathcal{K}}.$$

This enables us to bound $\mathcal{R}_{L_{\mathrm{ramp}},\rho}(f_{\mathbf{z},\mu})+\mu\Omega(f^*_{\mathbf{z},\mu})$ as

$$\begin{aligned}\mathcal{R}_{L_{\mathrm{ramp}},\rho}(f_{\mathbf{z},\mu})+\mu\Omega(f^*_{\mathbf{z},\mu}) &\le \left\{\mathcal{R}_{L_{\mathrm{ramp}},\rho}(f_{\mathbf{z},\mu})-\mathcal{R}_{L_{\mathrm{ramp}},\mathbf{z}}(f_{\mathbf{z},\mu})\right\}\\ &\quad+\left(1+\frac{\mu}{\lambda}\right)\mathcal{R}_{L_{\mathrm{ramp}},\mathbf{z}}(\tilde{f}_{\mathbf{z},\lambda})+\mu\|\tilde{f}^*_{\mathbf{z},\lambda}\|^2_{\mathcal{K}}.\end{aligned}$$

Next we use the definitions of $\tilde{f}_{\mathbf{z},\lambda}$ and f_λ to analyze the last two terms of the above bound:

$$\begin{aligned}&\left(1+\frac{\mu}{\lambda}\right)\mathcal{R}_{L_{\mathrm{ramp}},\mathbf{z}}(\tilde{f}_{\mathbf{z},\lambda})+\mu\|\tilde{f}^*_{\mathbf{z},\lambda}\|^2_{\mathcal{K}}\\ &\le\left(1+\frac{\mu}{\lambda}\right)\left(\mathcal{R}_{L_{\mathrm{ramp}},\mathbf{z}}(\tilde{f}_{\mathbf{z},\lambda})+\lambda\|\tilde{f}^*_{\mathbf{z},\lambda}\|^2_{\mathcal{K}}\right)\\ &\le\left(1+\frac{\mu}{\lambda}\right)\left(\mathcal{R}_{L_{\mathrm{ramp}},\mathbf{z}}(f_\lambda)+\lambda\|f^*_\lambda\|^2_{\mathcal{K}}\right)\\ &=\left(1+\frac{\mu}{\lambda}\right)\left(\mathcal{R}_{L_{\mathrm{ramp}},\mathbf{z}}(f_\lambda)-\mathcal{R}_{L_{\mathrm{ramp}},\rho}(f_\lambda)+\mathcal{R}_{L_{\mathrm{ramp}},\rho}(f_\lambda)+\lambda\|f^*_\lambda\|^2_{\mathcal{K}}\right).\end{aligned}$$

Combining the above estimates, we find that $\mathcal{R}_{L_{\rm ramp},\rho}(f_{\mathbf{z},\mu}) - \mathcal{R}_{{\rm ramp},\rho}(f_c) + \mu\Omega(f^*_{\mathbf{z},\mu})$ can be bounded by

$$\left\{\mathcal{R}_{L_{\rm ramp},\rho}(f_{\mathbf{z},\mu}) - \mathcal{R}_{L_{\rm ramp},\mathbf{z}}(f_{\mathbf{z},\mu})\right\} + \left(1+\frac{\mu}{\lambda}\right)\left\{\mathcal{R}_{L_{\rm ramp},\mathbf{z}}(f_\lambda) - \mathcal{R}_{L_{\rm ramp},\rho}(f_\lambda)\right\}$$
$$+ \left(1+\frac{\mu}{\lambda}\right)\left\{\mathcal{R}_{L_{\rm ramp},\rho}(f_\lambda) - \mathcal{R}_{L_{\rm ramp},\rho}(f_c) + \lambda\|f^*_\lambda\|^2_{\mathcal{K}}\right\} + \frac{\mu}{\lambda}\mathcal{R}_{L_{\rm ramp},\rho}(f_c).$$

Recalling the definition of f_λ, one has $\mathcal{A}(\lambda) = \mathcal{R}_{L_{\rm ramp},\rho}(f_\lambda) - \mathcal{R}_{L_{\rm ramp},\rho}(f_c) + \lambda\|f^*_\lambda\|^2_{\mathcal{K}}$. Hence the desired result follows. ■

With the help of Theorem 4, the generalization error is estimated by bounding $\mathcal{S}(m,\mu,\lambda)$ and $\mathcal{A}(\lambda)$ respectively. As the ramp loss is Lipschitz continuous, one can show that

$$\mathcal{R}_{{\rm ramp},\rho}(f) - \mathcal{R}_{{\rm ramp},\rho}(f_c) \le \|f - f_c\|_{L^1_{\rho_X}}.$$

Hence the approximation error $\mathcal{A}(\lambda)$ can be estimated by the approximation in a weighted L^1 space with the norm $\|f\|_{L^1_{\rho_X}} = \int_X |f(x)|d\rho_X$, as done in Smale and Zhou (2003). The following assumption is standard in the literature of learning theory (see, e.g., Cucker and Zhou, 2007; Steinwart and Christmann, 2008).

Assumption 1 *For any $0 < \beta \le 1$ and $c_\beta > 0$, the approximation error satisfies*

$$\mathcal{A}(\lambda) \le c_\beta \lambda^\beta, \qquad \forall \lambda > 0. \tag{18}$$

We also expect that the sample error $\mathcal{S}(m,\lambda,\mu)$ will tend to zero at a certain rate as the sample size tends to infinity. The asymptotical behaviors of $\mathcal{S}(m,\lambda,\mu)$ can be illustrated by the convergence of the empirical mean $\frac{1}{m}\sum_{i=1}^m \varsigma_i$ to its expectation $\mathbb{E}\varsigma_i$, where $\{\varsigma_i\}_{i=1}^m$ are independent random variables defined as

$$\varsigma_i = L_{\rm ramp}(y_i f(x_i)). \tag{19}$$

At the end of this section, we present our main theorem to illustrate the convergence behavior of ramp-LPSVM (6).

Theorem 5 *Suppose that Assumption 1 holds with $0 < \beta \le 1$. Take $\mu = m^{-\frac{\beta+1}{4\beta+2}}$ and $f_{\mathbf{z},\mu} = f^*_{\mathbf{z},\mu} + b^*_{\mathbf{z},\mu}$ with $(f^*_{\mathbf{z},\mu}, b^*_{\mathbf{z},\mu})$ being the global minimizer of ramp-LPSVM (6). Then for any $0 < \delta < 1$, with probability at least $1-\delta$, there holds*

$$\mathcal{R}_{L_{\rm mis},\rho}({\rm sgn}(f_{\mathbf{z},\mu})) - \mathcal{R}_{L_{\rm mis},\rho}(f_c) \le \tilde{c}\left(\log\frac{4}{\delta}\right)^{1/2} m^{-\frac{\beta}{4\beta+2}}, \tag{20}$$

where $\tilde{c}$ is a constant independent δ or m.

This theorem will be proved in Appendix by concentration techniques developed by Bartlett and Mendelson (2003). Based on the decomposition formula (17) established for ramp-LPSVM, one can also derive sharp convergence results under the framework applied by Wu and Zhou (2005). Here we use ramp-SVM (11) to conduct an error decomposition

for ramp-LPSVM (6), so the derived convergence rates of the latter are essentially no worse than those of ramp-SVM. Actually, also from our discussion in this section, ramp-SVM and C-SVM should have almost the same error bounds. One thus can expect that ramp-LPSVM enjoys similar asymptotic behaviors as C-SVM. It also should be pointed that, throughout our analysis, the global optimality plays an important role. Therefore, to guarantee the performance of ramp-LPSVM, a global search strategy is necessary.

3. Problem-solving Algorithms

In the previous section, we discussed theoretical properties for ramp-LPSVM. Its robustness and sparsity can be expected, if a good solution of ramp-LPSVM (6) can be obtained. However, (6) is non-convex. Therefore, in this paper, we propose a downhill method for local minimization and a heuristic for escaping a local minimum. Difference of convex function (DC) programming proposed by An et al. (1996) and An and Tao (2005) has been applied for ramp loss minimization problems (see Wu and Liu, 2007; Wang et al., 2010). By Yuille and Rangarajan (2003), Collobert et al. (2006b), Zhao and Sun (2008), this type of methods is also called a concave-convex procedure. For the proposed ramp-LPSVM, the DC technique is applicable as well.

Let $\alpha = [\alpha_1, \cdots, \alpha_m]^T \in \mathbb{R}^m$. Based on the identity (2), ramp-LPSVM (6) can be written as follows,

$$\begin{aligned} \min_{\alpha \succeq 0, b} \quad & \mu \sum_{i=1}^{m} \alpha_i + \frac{1}{m} \sum_{i=1}^{m} \max \left\{ 1 - y_i \left(\sum_{j=1}^{m} \alpha_j y_j \mathcal{K}(x_i, x_j) + b \right), 0 \right\} \\ & - \frac{1}{m} \sum_{i=1}^{m} \max \left\{ -y_i \left(\sum_{j=1}^{m} \alpha_j y_j \mathcal{K}(x_i, x_j) + b \right), 0 \right\}. \end{aligned} \tag{21}$$

We let $\zeta = [\alpha^T, b]^T$ stand for the optimization variable and $D(\zeta)$ for the feasible set of (21). Denote the convex part (the first line of) as $g(\zeta)$, and the concave part (the second line of (21)) as $h(\zeta)$. After that, (21) can be written as $\min_{\zeta \in D(\zeta)} g(\zeta) - h(\zeta)$. Then DC programming developed by Horst and Thoai (1999) and An and Tao (2005) is applicable. We give the following algorithm for local minimization for ramp-LPSVM.

Algorithm 1: DC programming for ramp-LPSVM from $\hat{\alpha}, \hat{b}$

- Set $\delta > 0$, $k := 0$ and $\zeta_0 := [\hat{\alpha}^T, \hat{b}]^T$;

repeat

- Select $\eta_k \in \partial h(\zeta_k)$;
- $\zeta_{k+1} := \arg \min_{\zeta \in D(\zeta)} g(\zeta) - \big(h(\zeta_k) + (\zeta - \zeta_k)^T \eta_k\big)$;
- Set $k := k + 1$;

until $\|\zeta_k - \zeta_{k-1}\| < \delta$;

- Algorithm ends and returns ζ_k.

Since $g(\zeta)$ is convex and piecewise linear, Algorithm 1 involves only LP, which can be effectively solved. One noticeable point is that $h(\zeta)$ is not differentiable at some points.

The non-differentiability of $h(\zeta)$ comes from $\max\{u, 0\}$, of which the sub-gradient at $u = 0$ is in the interval $[0, 1]$:

$$\frac{\partial \max\{u, 0\}}{\partial u}\Big|_{u=0} \in [0, 1].$$

In our algorithm, we choose 0.5 as the value of the above sub-gradient and then $\eta_k \in \partial h(\zeta_k)$ is uniquely defined. The local optimality condition for DC problems has been investigated by An and Tao (2005) and references therein. For a differentiable function, one can use the gradient information to check whether the solution is locally optimal. However, ramp-LPSVM is non-smooth and a sub-gradient technique should be considered. The local minimizer of a non-smooth objective function should meet the local optimality condition for all vectors in its sub-gradient set. In Algorithm 1, we only consider one value of the sub-gradient, thus, the result of the above process is not necessarily a local minimum. The rigorous local optimality condition and the related algorithm can be found in Huang et al. (2012b). However, because of the effectiveness of DC programming, we suggest Algorithm 1 for ramp-LPSVM in this paper.

As a local search algorithm, DC programming can effectively decrease the objective value of (21). The main difficulty of solving (21) is that it is non-convex and hence we may be trapped in a local optimum. To escape from a local optimum, we introduce slack variable $c = [c_1, \cdots, c_m]^T$ and transform (21) into the following concave minimization problem,

$$\begin{aligned} \min_{\alpha,b,c} \quad & \mu \sum_{i=1}^{m} \alpha_i + \frac{1}{m} \sum_{i=1}^{m} c_i - \frac{1}{m} \sum_{i=1}^{m} \max \left\{ -y_i \left(\sum_{j=1}^{m} \alpha_j y_j \mathcal{K}(x_i, x_j) + b \right), 0 \right\} \\ \text{s.t.} \quad & c_i \geq 1 - y_i \left(\sum\nolimits_{j=1}^{m} \alpha_j y_j \mathcal{K}(x_i, x_j) + b \right), \quad i = 1, 2, \ldots, m, \\ & c_i \geq 0, \quad i = 1, 2, \ldots, m, \\ & \alpha_i \geq 0, \quad i = 1, 2, \ldots, m. \end{aligned} \tag{22}$$

This is a concave minimization problem constrained in a polyhedron, which is called a polyhedral concave problem by Horst and Hoang (1996). Generally, among non-convex problems, a polyhedral concave problem is relatively easy to deal with. Various techniques, such as γ-extension, vertex enumeration, partition algorithm, concavity cutting, have been discussed insightfully in Horst and Hoang (1996) and successfully applied (see, e.g., Porembski, 2004; Mangasarian, 2007; Shu and Karimi, 2009). Moreover, the objective function of (22) is piecewise linear, which makes the hill detouring method proposed by Huang et al. (2012a) applicable. In the following, we first introduce the basic idea of the hill detouring method and then establish a global search algorithm for ramp-LPSVM.

For notational convenience, we use $\xi = [\alpha^T, b, c^T]^T$ to denote the optimization variable of (22). The objective function is continuous piecewise linear and is denoted as $p(\xi)$. The feasible set, which is a polyhedron, can be written as $A\xi \leq q$. Then (22) is compactly represented as the following polyhedral concave problem, of which the objective function is piecewise linear:

$$\min\nolimits_{\xi} \quad p(\xi), \quad \text{s.t.} \quad A\xi \leq q. \tag{23}$$

Assume that we are trapped in a local optimum $\tilde{\xi}$ with value $\tilde{p} = p(\tilde{\xi})$ and we are trying to escape from it. We observe that (in a non-degenerated case): i) the local optimum $\tilde{\xi}$ is a

vertex of the feasible set; ii) any level set $\{\xi : p(\xi) = u\}, \forall u$ is the boundary of a polyhedron. The first property can be derived from the concavity of the objective function. The second property comes from the piecewise linearity of $p(\xi)$. These properties imply a new method searching on the level set to find another feasible solution $\hat{\xi}$ with the same objective value $p(\hat{\xi}) = \tilde{p}$. If such $\hat{\xi}$ is found, we escape from $\tilde{\xi}$ and a downhill method can be used to find a new local optimum. Otherwise, if such $\hat{\xi}$ does not exist, one can conclude that $\tilde{\xi}$ is the optimal solution. Searching on the level set of $p(\xi) = \tilde{p}$ will not decrease neither increase the objective value and it is hence called hill detouring. In practice, in order to avoid to find $\tilde{\xi}$ again, we search on $\{p(\xi) = \tilde{p} - \varepsilon\}$ with a small positive ε for computational convenience. If $\{p(\xi) = \tilde{p} - \varepsilon\} = \emptyset$, we know that $\tilde{\xi}$ is ε-optimal. The performance of hill detouring is not sensitive to the ε value, when ε is small (but large enough to distinguish $\tilde{p} - \epsilon$ and $\tilde{p}$). In this paper, we set $\varepsilon = 10^{-6}$.

Hill detouring, which is to solve the feasibility problem

$$\text{find} \quad \xi, \quad \text{s.t.} \quad p(\xi) = \tilde{p} - \varepsilon, \ A\xi \leq q, \tag{24}$$

is a natural idea for global optimization but it is hard to implement for a regular concave minimization functions. The main difficulty is the nonlinear equation $p(\xi) = \tilde{p} - \varepsilon$. In ramp-LPSVM, the objective function of (22) is continuous and piecewise linear, thus, $p(\xi) = \tilde{p} - \varepsilon$ can be transformed into (finite) linear equations. That means (24) can be written as a series of LP feasibility problems, which makes line search on $\{\xi : p(\xi) = \tilde{p} - \varepsilon\}$ possible.

To investigate the property of (23) and the corresponding hill detouring technique, we consider a 2-dimensional problem. In this intuitive example, the objective function is $p(\xi) = a_0^T \xi + b_0 - \sum_{i=1}^{6} \max\{0, a_i^T \xi + b_i\}$, where

$$a_0 = \begin{bmatrix} 0.05 \\ -0.1 \end{bmatrix} \quad a_1 = \begin{bmatrix} -1 \\ -0.4 \end{bmatrix} \quad a_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad a_3 = \begin{bmatrix} 0.5 \\ 0.1 \end{bmatrix} \quad a_4 = \begin{bmatrix} -0.9 \\ 0.4 \end{bmatrix} \quad a_5 = \begin{bmatrix} -0.6 \\ -1 \end{bmatrix} \quad a_6 = \begin{bmatrix} 0.9 \\ 0.9 \end{bmatrix}$$
$$b_0 = -0.2 \quad b_1 = 0.8 \quad b_2 = -0.2 \quad b_3 = -0.5 \quad b_4 = 0.2 \quad b_5 = 1 \quad b_6 = 0.8.$$

The feasible domain is an octagon, of which the vertices are $[2, 1]^T, [1, 2]^T, \ldots, [1, -2]^T$. The plots of $p(\xi)$ and the feasible set are shown in Figure 2, where $\tilde{\xi} = [2, 1]^T$ is a local optimum and the global optimum is $\xi^\star = [-2, -1]^T$.

Now we try to escape from $\tilde{\xi}$ by hill detouring. In other words, we search on the level set $\{\xi : p(\xi) = \tilde{p} - \varepsilon\}$ to find a feasible solution. The level set is displayed by the green dashed line in Figure 3. According to the property that $\tilde{\xi}$ is a vertex of the feasible domain, we can first search along the corresponding active edges, which are shown by the black solid lines, to find the γ-extensions. The definition of γ-extension was given by Horst and Hoang (1996) and is reviewed below.

Definition 6 *Suppose f is a concave function, ξ is a given point, γ is a scalar with $\gamma \leq f(\xi)$, and θ_0 is a positive number large enough. Let $d \neq 0$ be a direction and $\theta = \min\{\theta_0, \sup\{t : f(\xi + td) \geq \gamma\}\}$, then $\xi + \theta d$ is called the γ-extension of $f(\xi)$ from ξ along d.*

Set $\gamma = \tilde{p} - \varepsilon$. γ-extensions from $\tilde{\xi}$ can be easily found by bisection according to the concavity of $p(\xi)$. For any direction d, we set $t_1 = 0$ and t_2 as a large enough positive number. If $p(\tilde{\xi} + t_2 d) > \gamma$, there is no γ-extension along this direction. Otherwise, after the following bisection scheme, $\frac{1}{2}(t_1 + t_2)$ is the γ-extension from $\tilde{\xi}$ along d,

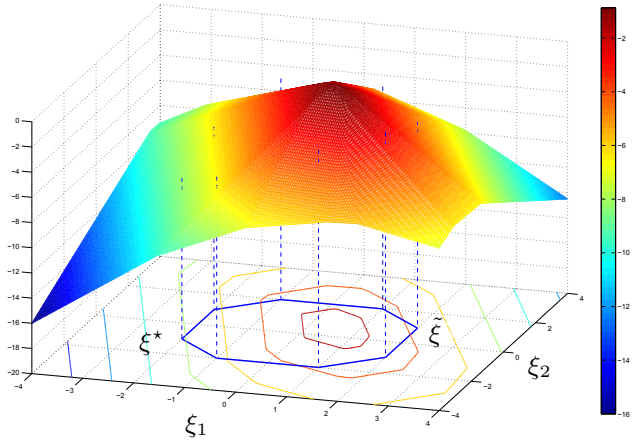


Figure 2: Plots of the objective function $p(\xi)$ and the feasible domain $A\xi \leq q$, of which the boundary is shown by the blue solid line. $\tilde{\xi} = [2, 1]$ is a local optimum and $\tilde{p} = p(\tilde{\xi}) = -4.5$; $\xi^\star = [-2, -1]$ with $p(\xi^\star) = -8.2$ is the global optimum.

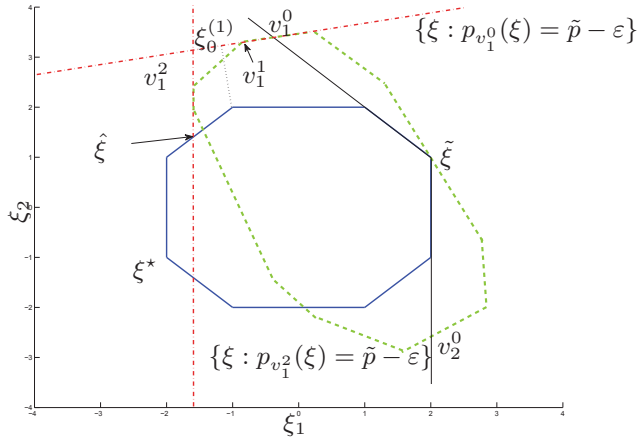


Figure 3: Hill detouring method. From a local optimum $\tilde{\xi}$, we can find v_1^0, which is the γ-extension along the active edge. Searching in the hyperplane of the level set, we arrive at v_1^1, v_1^2, and $\hat{\xi}$, successively. $\hat{\xi}$ is feasible and has a less objective value than $p(\tilde{\xi})$, then we successfully escape from the local optimum $\tilde{\xi}$.

While $t_2 - t_1 > 10^{-6}$

If $f(\tilde{\xi} + \frac{1}{2}(t_1+t_2)d)) > \gamma$, set $t_1 = \frac{1}{2}(t_1+t_2)$; **Else** set $t_2 = \frac{1}{2}(t_1+t_2)$.

For the concerned example, along the edges of the feasible set, which are active at $\tilde{\xi}$, we find the γ-extensions, denoted by v_1^0 and v_2^0. If the convex hull of v_1^0, v_2^0 and $\tilde{\xi}$ covers the feasible set, $\tilde{\xi}$ is ε-optimal for (23). Otherwise, these extensions provide good initial points for hill detouring.

The objective function $p(x)$ is piecewise linear and there exist a finite number of subregions, in each of which, $p(\xi)$ becomes a linear function. Therefore, for any given ξ_0, we can find a subregion, denoted by D_{ξ_0}, such that $\xi_0 \in D_{\xi_0}$ and there is a corresponding linear function, denoted by $p_{\xi_0}(\xi)$, satisfying: $p(\xi) = p_{\xi_0}(\xi), \forall \xi \in D_{\xi_0}$. Constrained in the region related to ξ_0, the feasibility problem (24) becomes

$$\begin{aligned} \text{find} \quad & \xi \\ \text{s.t.} \quad & p_{\xi_0}(\xi) = \tilde{p} - \varepsilon, \quad \xi \in D_{\xi_0} \\ & A\xi \le q. \end{aligned} \tag{25}$$

Since $p(\xi)$ is concave and $p_{\xi_0}(\xi)$ is essentially the first order Taylor expansion of $p(\xi)$, we know that $p(\xi) \le p_{\xi_0}(\xi), \forall \xi_0, \xi$, where the equality holds when $\xi \in D_{\xi_0}$. For a solution ξ' satisfying $p_{\xi_0}(\xi') = \tilde{p} - \varepsilon$ but outside D_{ξ_0}, we have $p(\xi') < \tilde{p} - \varepsilon$. If ξ' is feasible ($A\xi' \le q$), then a better solution is found. Therefore, in hill detouring method, we ignore the constraint $\xi \in D_{\xi_0}$ in (25) and consider the following optimization problem,

$$\begin{aligned} \min_{\xi^{(1)}, \xi^{(2)}} \quad & \|\xi^{(1)} - \xi^{(2)}\|_\infty \\ \text{s.t.} \quad & p_{\xi_0}(\xi^{(1)}) = \tilde{p} - \varepsilon \\ & A\xi^{(2)} \le q, \end{aligned} \tag{26}$$

for which $\xi^{(1)} = \xi_0, \xi^{(2)} = \tilde{\xi}$ provides a feasible solution. Notice that after introducing a slack variable $s \in \mathbb{R}$, minimizing $\|\xi^{(1)} - \xi^{(2)}\|_\infty$ is equivalently to minimize s with the constraint that each component of $\xi^{(1)} - \xi^{(2)}$ is between $-s$ and s. Then (26) is essentially an LP problem. Starting from v_1^0, we set $\xi_0 = v_1^0$ and solve (26), of which the solution is denoted by $\xi_0^{(1)}, \xi_0^{(2)}$. As displayed in Figure 3, $\xi_0^{(1)}$ is the point which is closest to the feasible domain among all the points in hyperplane $p_{v_1^0}(\xi) = \tilde{p} - \varepsilon$. Heuristically, we search on the level set towards $\xi_0^{(1)}$: going along the direction $d_0 = \xi_0^{(1)} - \xi_0$ and finding point v_1^1, where $p(\xi)$ becomes another linear function. v_1^1 is also a vertex of the level set $\{\xi : p(\xi) = \tilde{p} - \varepsilon\}$. Then we construct a new linear function $p_{v_1^1}(\xi)$, which is different to $p_{v_1^0}(\xi)$. Repeating the above process, we can get v_1^2. After that, solving (26) for $\xi_0 = v_1^2$ leads to $\hat{\xi}$, which is feasible and has a objective value $\tilde{p} - \varepsilon$, then we successfully escape from $\tilde{\xi}$ by hill detouring.

We have shown the basic idea of the hill detouring method by one 2-dimensional problem. For ramp-LPSVM, the hill detouring method for (22) is similar to the above process. Specifically, the local linear function for a given $\xi_0 = [\alpha_0^T, b_0, c_0^T]^T$ is below,

$$p_{\xi_0}(\xi) = \mu \sum_{i=1}^{m} \alpha_i + \frac{1}{m} \sum_{i=1}^{m} c_i + \frac{1}{m} \sum_{i \in \mathcal{M}_{\xi_0}} y_i \left(\sum_{j=1}^{m} \alpha_j y_j \mathcal{K}(x_i, x_j) + b \right), \tag{27}$$

where $\mathcal{M}_{\xi_0}$ is a union of $\mathcal{M}^+_{\xi_0}$ and any subset of $\mathcal{M}^0_{\xi_0}$ and the related sets are defined below,

$$\begin{aligned}\mathcal{M}^+_\xi &= \left\{i : -y_i\left(\sum_{j=1}^m \alpha_j y_j \mathcal{K}(x_i,x_j)+b\right) > 0\right\},\\ \mathcal{M}^0_\xi &= \left\{i : -y_i\left(\sum_{j=1}^m \alpha_j y_j \mathcal{K}(x_i,x_j)+b\right) = 0\right\}.\end{aligned}$$

The above choice means $\mathcal{M}^+_{\xi_0} \subseteq \mathcal{M}_{\xi_0} \subseteq \mathcal{M}^+_{\xi_0} \bigcup \mathcal{M}^0_{\xi_0}$. For a random ξ, $\mathcal{M}^0_\xi$ is usually empty. For a point like v_1^1 in Figure 3, which is a vertex of the level set, $\mathcal{M}^0_{v_1^1} \neq \emptyset$. In this case, there are multiple choices for p_{ξ_0} and we select $\mathcal{M}_{\xi_0}$ which has not been considered. Summarizing the discussions, we give the following algorithm for ramp-LPSVM (6).

Algorithm 2: Global Search for ramp-LPSVM

initialize

- Set δ (the threshold of convergence for DC programming), ε (the difference value in hill detouring), K_{step} (the maximal number of hill detouring steps)
- Give an initial feasible solution $\hat{\alpha}, \hat{b}$;

repeat

- Use Algorithm 1 from $\hat{\alpha}, \hat{b}$ to obtain locally optimal solution $\tilde{\alpha}, \tilde{b}$;
- Compute $\tilde{c}_i := \max\left\{-y_i\left(\sum_{j=1}^m \tilde{\alpha}_j y_j \mathcal{K}(x_i,x_j)+\tilde{b}\right), 0\right\}$;
- Set $\tilde{\xi} := [\tilde{\alpha}^T, \tilde{b}, \tilde{c}^T]^T$, $\gamma := p(\tilde{\xi}) - \varepsilon$, where $p(\xi)$ is the object of (22), and compute the γ-extensions for edges active at $\tilde{\xi}$. We denote the γ-extensions as $v_1, v_2, \ldots$ and the distance of v_i to the feasible set of (22) as dist_i;
- Let $k := 0$ and $\mathcal{S}_\mathcal{M} := \emptyset$;
- **repeat**
  - Let $k := k+1$, select $i_0 := \arg\min_i \text{dist}_i$, and set $\xi_0 := v_{i_0}$;
  - Select $\mathcal{M}_{\xi_0}$ according to $\mathcal{M}^+_{\xi_0}, \mathcal{M}^0_{\xi_0}$ such that $\mathcal{M}_{\xi_0} \notin \mathcal{S}_\mathcal{M}$;
  - Set $\mathcal{S}_\mathcal{M} := \mathcal{S}_\mathcal{M} \bigcup \{\mathcal{M}_{\xi_0}\}$;
  - Construct $p_{\xi_0}(\xi)$ and solve LP (26), of which the solution is $\xi_0^{(1)}, \xi_0^{(2)}$;
  - **if** $\xi_0^{(1)} = \xi_0^{(2)}$ **then**
    - Set $\hat{\alpha}, \hat{b}$ according to $\xi_0^{(1)}$ and terminate the inner loop;
  - **else**
    - Let $d := \xi_0^{(1)} - \xi_0$ and find $\theta := \max\{\theta : p(\xi_0 + \theta d) = p_{\xi_0}(\xi_0 + \theta d)\}$;
    - Set $v_{i_0} := \xi_0 + \theta d$ and update dist_{i_0};
  - **end**
- **until** $k \geq K_{\text{step}}$;

until $\tilde{\alpha} = \hat{\alpha}, \tilde{b} = \hat{b}$;

- Algorithm ends and returns $\tilde{\alpha}, \tilde{b}$.

4. Numerical Experiments

In the numerical experiments, we evaluate the performance of ramp-LPSVM (6) and its problem-solving algorithms. We first report the optimization performance and then discuss

the robustness and the sparsity compared with C-SVM, LPSVM (5), and ramp-SVM (11). C-SVM and LPSVM are convex problems, which are solved by the Matlab optimization toolbox. For ramp-SVM, we apply the algorithm proposed by Collobert et al. (2006a). The data are downloaded from the UCI Machine Learning Repository given by Frank and Asuncion (2010). In data sets "Spect", "Monk1", "Monk2", and "Monk3", the training and the testing sets are provided. For the others, we randomly partition the data into two parts: half data are used for training and the remaining data are for testing. In this paper, we focus on outliers and hence we contaminate the training data set by randomly selecting some instances in class -1 and changing their labels. Since there are random factors in sampling and adding outliers, we repeat the above process 10 times for each data set and report the average accuracy on the testing data. In our experiments, we apply a Gaussian kernel $\mathcal{K}(x_i, x_j) = \exp\left(-\|x_i - x_j\|^2/\sigma^2\right)$. The training data are normalized to $[0,1]^n$ and then the regularization coefficient μ and the kernel parameter σ are tuned by 10-fold cross-validation for each method. In the tuning phase, grid search using logarithmic scale is applied. The range of possible μ value is $[10^{-2}, 10^3]$ and the range of σ value is between 10^{-3} and 10^2. For ramp-LPSVM, since the global search needs more computation time, the parameters tuning by cross-validation is conducted based on Algorithm 1. The experiments are done in Matlab R2011a in Core 2-2.83 GHz, 2.96G RAM.

Intuitively, ramp-LPSVM can provide a sparse and robust result, if a good solution for (6) can be obtained. Hence, we first consider the optimization performance of the proposed algorithms. To evaluate them, we set $\mu = 1/10, \sigma = 1$ and use the four data sets for which the training data are provided. The result of ramp-LPSVM is sparse, we hence use $\hat{\alpha} = 0$, which is optimal when μ is large sufficiently, as the initial solution. When $\hat{\alpha} = 0$, simply calculating shows that $\hat{b} = 1$ is optimal to (6) if there are more training data in class $+1$ than in class -1 ($\#\{i : y_i = 1\} \geq \#\{i : y_i = -1\}$). Otherwise, we set $\hat{b} = -1$. From $\hat{\alpha}, \hat{b}$, we apply Algorithm 2 to minimize (6). Basically, Algorithm 2 in turn applies DC programming for local minimization and hill detouring for escaping local optima. In Table 2, we report the objective values of the obtained local optima and the corresponding computation time. The superscript indicates the sequence and f^1 is the result of Algorithm 1.

Data		f^1	f^2	f^3	f^4	f^5	f^6	GA
Spect	objective value	36.59	9.36	7.38	6.41	5.43	5.40	8.78
	time (s)	0.298	2.64	2.89	5.46	4.63	19.84	39.6
Monk1	objective value	9.94	8.96	7.11	—	—	—	10.10
	time (s)	4.04	8.14	26.8	—	—	—	66.34
Monk2	objective value	9.17	8.24	7.31	5.48	—	—	12.66
	time (s)	12.3	20.9	43.1	43.5	—	—	108.1
Monk3	objective value	4.92	4.02	—	—	—	—	11.38
	time (s)	3.93	32.7	—	—	—	—	69.21

Table 2: Global Search Performance of Algorithm 2 ($\delta = 10^{-6}, \varepsilon = 10^{-6}, K_{\text{step}} = 50$)

From the reported results, one can see the effectiveness of hill detouring for escaping from local optima. Another observation is that with the increasing quality of the local optimum, the hill detouring needs more time for escaping. When the initial point is not good, the

computation time for hill detouring is also small, which means that the performance of Algorithm 2 is not sensitive to the initial solution. To evaluate the global search capability, we also use the Genetic Algorithm (GA) toolbox developed by Chipperfield et al. (1994). The result of GA is random and we run GA algorithm repeatedly in the similar computing time of Algorithm 2. Then we select the best one and report it in Table 2. The comparison illustrates the global search capability of Algorithm 2. The basic elements of Algorithm 1 and Algorithm 2 are both to iteratively solve LPs. For large-scale problems, some fast methods for LP, especially the techniques designed for LPSVM by Bradley and Mangasarian (2000), Fung and Mangasarian (2004), and Mangasarian (2006), are applicable to speed up the solving procedure, which can be potential future work for ramp-LPSVM.

In the experiments above, the proposed algorithms show good minimization capability for ramp-LPSVM (6). Then one can expect good performance of the proposed model and algorithms, according to the robustness, sparsity, and other statistical properties discussed in Section 3. For each training set, we randomly select some data from class -1 and change their labels to be $+1$. The ratio of the outliers, denoted by r, is set to be $r = 0.0, 0.05, 0.10$. Based on the contaminated training set, we use C-SVM, LPSVM (5), ramp-SVM (11), and ramp-LPSVM (6) (solved by Algorithm 1 and Algorithm 2, respectively) to train the classifier and calculate the classification accuracy on the testing data. The above process is repeated 10 times. The average testing accuracy and the average number of support vectors (the corresponding $|\alpha_i|$ is larger than 10^{-6}) are reported in Table 3, where the data dimension n and the size of training data m are reported as well. The best results in the view of classification accuracy are underlined and the sparsest results are given in bold.

From Table 3, we observe that when there are no outliers, C-SVM performs well and LPSVM also provides good classifiers. The number of support vectors of LPSVM is always smaller than that of C-SVM, which relates to the property of ℓ_1 minimization. With an increasing number of outliers, the accuracy of C-SVM and LPSVM decreases. In contrast, the results of ramp-SVM and ramp-LPSVM are more stable, showing the robustness of the ramp loss. The ramp loss also brings some sparsity, since when $y_i f(x_i) \geq 0$, the ramp loss gives a constant penalty, which corresponds to a zero dual variable. The proposed ramp-LPSVM consists of the ℓ_1-penalty and the ramp loss, both of which can enhance the sparsity. Hence, the sparsity of the result of ramp-LPSVM is significant. Comparing the two algorithms for ramp-LPSVM, we find that Algorithm 2, which pursues a global solution, results in a more robust classifier. But the computation time of Algorithm 2 is significantly larger, as illustrated in Table 2. Generally, if there are heavy outliers and plenty allowable computation time, it is worth considering Algorithm 2 to find a good classifier. Otherwise, solving ramp-LPSVM by Algorithm 1 is a good choice.

5. Conclusion

In this paper, we proposed a robust classification method, called ramp-LPSVM. It consists of the ℓ_1-penalty and the ramp loss, which correspond to sparsity and robustness, respectively. The consistency and error bound for ramp-LPSVM have been discussed. Ramp-LPSVM trains a classifier by minimizing the ramp loss together with the ℓ_1-penalty, both of which are piecewise linear. According to the piecewise linearity, a local optimization method using DC programming and a global search strategy using hill detouring technique have been

Data	n	m	r	C-SVM		LPSVM		ramp-SVM		ramp-LPSVM (Algorithm 1)		ramp-LPSVM (Algorithm 2)	
Spect	21	80	0.00	86.03%	#80	88.77%	**#22**	88.77%	#75	88.77%	**#22**	88.77%	**#22**
	21	80	0.05	83.96%	#80	86.90%	**#18**	87.70%	#76	88.77%	#21	88.77%	#21
	21	80	0.10	84.49%	#79	84.33%	#20	85.56%	#74	87.71%	#18	88.37%	**#18**
Monk1	6	124	0.00	85.70%	#70	84.68%	#44	83.25%	#65	83.09%	#39	84.40%	**#38**
	6	124	0.05	82.70%	#72	80.61%	#47	81.17%	#61	81.92%	**#36**	82.07%	#37
	6	124	0.10	76.77%	#73	73.52%	#38	78.66%	#57	79.20%	**#31**	79.91%	#33
Monk2	6	169	0.00	83.80%	#96	81.05%	#62	83.80%	#86	82.62%	#57	82.86%	**#53**
	6	169	0.05	77.78%	#95	79.05%	#59	77.78%	#85	80.05%	**#54**	80.24%	#61
	6	169	0.10	71.76%	#96	75.88%	#58	74.68%	#75	78.53%	**#52**	79.84%	**#52**
Monk3	6	122	0.00	90.15%	#55	91.76%	#34	91.32%	#53	88.73%	**#29**	89.34%	#30
	6	122	0.05	87.13%	#61	88.47%	#33	88.68%	#47	87.42%	**#31**	86.80%	**#31**
	6	122	0.10	81.13%	#69	83.49%	#34	85.00%	#50	84.35%	#32	84.94%	**#30**
Breast	10	350	0.00	96.68%	#87	95.14%	#28	96.78%	#73	96.25%	#18	96.45%	**#17**
	10	350	0.05	95.63%	#90	93.41%	#25	95.90%	#71	96.20%	**#16**	96.07%	**#16**
	10	350	0.10	90.43%	#84	84.66%	#22	91.59%	#79	93.54%	**#16**	95.77%	#18
Pima	8	385	0.00	76.04%	#233	72.53%	#47	75.98%	#67	75.59%	#41	74.22%	**#39**
	8	385	0.05	75.78%	#230	74.31%	**#33**	75.29%	#68	74.56%	#40	74.40%	#42
	8	385	0.10	74.01%	#228	74.28%	**#31**	73.26%	#69	74.35%	#37	74.67%	#37
Trans.	4	375	0.00	76.33%	#199	75.86%	#22	77.01%	#32	77.11%	**#5**	77.11%	**#6**
	4	375	0.05	74.62%	#285	74.97%	#19	76.20%	#31	76.28%	**#6**	76.69%	#7
	4	375	0.10	73.06%	#274	72.90%	#12	76.73%	#30	75.28%	**#8**	76.28%	**#8**
Haber.	3	154	0.00	74.77%	#86	73.69%	#10	74.31%	#57	75.05%	**#5**	74.92%	**#5**
	3	154	0.05	74.38%	#81	73.25%	#9	73.26%	#68	73.79%	**#8**	73.97%	**#8**
	3	154	0.10	71.66%	#75	72.54%	**#11**	73.56%	#57	73.79%	**#11**	73.86%	**#11**
Ionos.	33	176	0.00	93.81%	#93	92.41%	**#29**	93.64%	#92	93.10%	#32	93.01%	#34
	33	176	0.05	92.22%	#97	89.26%	#32	92.89%	#95	92.33%	#32	93.03%	**#31**
	33	176	0.10	90.13%	#98	89.41%	#32	92.63%	#87	92.22%	**#27**	92.94%	#28

Table 3: Classification Accuracy on Testing Data and Number of Support Vectors

proposed. The proposed algorithms have good optimization capability and ramp-LPSVM has shown robustness and sparsity in numerical experiments.

Acknowledgments

The authors are grateful to the anonymous reviewers for insightful comments.
This work was supported in part by the scholarship of the Flemish Government; Research Council KUL: GOA/11/05 Ambiorics, GOA/10/09 MaNet, CoE EF/05/006 Optimization in Engineering (OPTEC), IOF-SCORES4CHEM, several PhD/postdoc & fellow grants; Flemish Government: FWO: PhD/postdoc grants, projects: G0226.06 (cooperative systems and optimization), G.0302.07 (SVM/Kernel), G.0320.08 (convex MPC), G.0558.08 (Robust MHE), G.0557.08 (Glycemia2), G.0588.09 (Brain-machine) research communities (WOG: ICCoS, ANMMM, MLDM); G.0377.09 (Mechatronics MPC), G.0377.12 (Structured models), IWT: PhD Grants, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, O&O-Dsquare; Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007-2011); IBBT; EU: ERNSI; ERC AdG A-DATADRIVE-B, FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCON (ICT-248940); Contract Research: AMINAL; Other: Helmholtz: viCERP, ACCM, Bauknecht, Hoerbiger. L. Shi is also supported by the National Natural Science Foundation of China (No. 11201079) and the Fundamental Research Funds for the Central Universities of China (No. 20520133238, No. 20520131169). Johan Suykens is a professor at KU Leuven, Belgium.

Appendix A.

In this appendix, we prove Theorem 5 in Section 2. First, we bound the offset by the following lemma.

Lemma 7 *For any $\mu > 0$, $m \in \mathbb{N}$, and $\mathbf{z} = \{x_i, y_i\}_{i=1}^m$, we can find a solution $(f^*_{\mathbf{z},\mu}, b^*_{\mathbf{z},\mu})$ of equation (6) satisfying $\min_{1\le i\le m} |f_{\mathbf{z},\mu}(x_i)| \le 1$, where $f_{\mathbf{z},\mu} = f^*_{\mathbf{z},\mu} + b^*_{\mathbf{z},\mu}$. Hence, $|b^*_{\mathbf{z},\mu}| \le 1 + \|f^*_{\mathbf{z},\mu}\|_\infty$.*

Proof Suppose a minimizer $f_{\mathbf{z},\mu} = f^*_{\mathbf{z},\mu} + b^*_{\mathbf{z},\mu}$ of (6) satisfies

$$r := \min_{1\le i\le m} |f_{\mathbf{z},\mu}(x_i)| = |f_{\mathbf{z},\mu}(x_{i_0})| > 1.$$

Then for each i, either $y_i f_{\mathbf{z},\mu}(x_i) \ge r > 1$ or $y_i f_{\mathbf{z},\mu}(x_i) \le -r < -1$. We consider a function $f^d_{\mathbf{z},\mu} := f_{\mathbf{z},\mu} - d$ with $d = (r-1)\mathrm{sgn}(f_{\mathbf{z},\mu}(x_{i_0}))$. Then $f^d_{\mathbf{z},\mu}$ satisfies $|f^d_{\mathbf{z},\mu}(x_{i_0})| = 1$ and $|f^d_{\mathbf{z},\mu}(x_{i_0})| \ge 1$. When $y_i f_{\mathbf{z},\mu}(x_i) > 1$, one can check that $y_i f^d_{\mathbf{z},\mu}(x_i) \ge 1$. Similarly, if $y_i f_{\mathbf{z},\mu}(x_i) < -1$, one still has $y_i f^d_{\mathbf{z},\mu}(x_i) \le -1$. Then $L_{\mathrm{ramp},\mathbf{z}}(f_{\mathbf{z},\mu}) = L_{\mathrm{ramp},\mathbf{z}}(f^d_{\mathbf{z},\mu})$. Therefore, $f^d_{\mathbf{z},\mu}$ is also a solution of equation (6) and satisfies our requirement.

Now if $f_{\mathbf{z},\mu} = f^*_{\mathbf{z},\mu} + b^*_{\mathbf{z},\mu}$ satisfies

$$|f_{\mathbf{z},\mu}(x_{i_0})| = \min_{1\le i\le m} |f_{\mathbf{z},\mu}(x_i)| \le 1,$$

we then have

$$|b^*_{\mathbf{z},\mu}| \le 1 + |f^*_{\mathbf{z},\mu}(x_{i_0})| \le 1 + \|f^*_{\mathbf{z},\mu}\|_\infty.$$

In this way, we complete the proof. ■

In the following, we shall always choose $f_{\mathbf{z},\mu}$ as in lemma 7. According to our proof, such kind of solutions can be easily constructed even though the obtained ones from the algorithm do not meet the requirement. Next, we find a function space covering $f_{\mathbf{z},\mu}$ when $\mathbf{z}$ runs over all possible samples.

Lemma 8 *For every $\mu > 0$, we have $f^*_{\mathbf{z},\mu} \in \mathcal{H}_K$ and*

$$\|f^*_{\mathbf{z},\mu}\|_K \le \kappa\Omega(f^*_{\mathbf{z},\mu}) \le \frac{\kappa}{\mu},$$

where $\kappa = \sup_{x,y\in X}\sqrt{|K(x,y)|}$.

Proof It is trivial that $f^*_{\mathbf{z},\mu} \in \mathcal{H}_K$. By the reproducing property (see Aronszajn, 1950), for $f^*_{\mathbf{z},\mu} = \sum_{i=1}^m \alpha^*_{i,\mathbf{z}} y_i K(x, x_i)$,

$$\|f^*_{\mathbf{z},\mu}\|_K = \left(\sum_{i,j=1}^m \alpha^*_{i,\mathbf{z}}\alpha^*_{j,\mathbf{z}}K(x_i,x_j)\right)^{1/2} \le \kappa\left(\sum_{i,j=1}^m \alpha_{i,\mathbf{z}}\alpha_{j,\mathbf{z}}\right)^{1/2} = \kappa\Omega(f^*_{\mathbf{z},\mu}).$$

Due to the definition of $f^*_{\mathbf{z},\mu}$, we have

$$\mathcal{R}_{\mathrm{ramp},\mathbf{z}}(f_{\mathbf{z},\mu}) + \mu\Omega(f^*_{\mathbf{z},\mu}) \le \mathcal{R}_{\mathrm{ramp},\mathbf{z}}(0) + \mu\Omega(0) \le 1.$$

This gives $\Omega(f^*_{\mathbf{z},\mu}) \le \frac{1}{\mu}$, and completes the proof. ■

From Lemma 7, Lemma 8 and the relation

$$\|f\|_\infty \le \kappa\|f\|_K, \quad \forall f \in \mathcal{H}_K,$$

we know that $f_{\mathbf{z},\mu}$ lies in

$$\mathcal{F}_\mu = \left\{f = f^* + b^* : \|f^*\|_K \le \frac{\kappa}{\mu} \text{ and } |b^*| \le 1 + \frac{\kappa^2}{\mu}\right\}. \tag{28}$$

Now we are in the position to prove the main theorem in Section 2. Our analysis mainly focus on estimating the sample error $\mathcal{S}(m,\mu,\lambda)$.

Proof of Theorem 5. We fist estimate $\mathcal{R}_{L_{\mathrm{ramp}},\mathbf{z}}(f_\lambda) - \mathcal{R}_{L_{\mathrm{ramp}},\rho}(f_\lambda)$ by considering the random variable ς_i defined by (19) with $f = f_\lambda$. As $L_{\mathrm{ramp}} : \mathbb{R} \to [0,1]$, there holds $|\varsigma_i - \mathbb{E}\varsigma_i| \le 2$. Then by the Hoeffding inequality (see, e.g., Cucker and Zhou, 2007, Corollary 3.6), with probability at least $1-\delta/2$, we have

$$\mathcal{R}_{L_{\mathrm{ramp}},\mathbf{z}}(f_\lambda) - \mathcal{R}_{L_{\mathrm{ramp}},\rho}(f_\lambda) \le \sqrt{\frac{8\log\frac{2}{\delta}}{m}}. \tag{29}$$

For the term $\mathcal{R}_{L_{\text{ramp}},\rho}(f_{\mathbf{z},\mu}) - \mathcal{R}_{L_{\text{ramp}},\mathbf{z}}(f_{\mathbf{z},\mu})$, note that $f_{\mathbf{z},\mu}$ varies with samples. In order to obtain the corresponding upper bound, we shall apply the uniform concentration inequality to the function set $\mathcal{F}_\mu$. One can directly use Theorem 8 in Bartlett and Mendelson (2003) to deal with this term and find with probability at least $1 - \delta/2$,

$$\mathcal{R}_{L_{\text{ramp}},\rho}(f_{\mathbf{z},\mu}) - \mathcal{R}_{L_{\text{ramp}},\mathbf{z}}(f_{\mathbf{z},\mu}) \leq \mathbb{E}_{\mathbf{z}}\mathbb{E}_\sigma \left[\sup_{g\in\tilde{\mathcal{F}}} \left| \frac{2}{m}\sum_{i=1}^m \sigma_i g(x_i, y_i)\right|\right] + \sqrt{\frac{8\log\frac{4}{\delta}}{m}} \tag{30}$$

where $\tilde{\mathcal{F}} := \{(x,y) \to L_{\text{ramp}}(yf(x)) - L_{\text{ramp}}(0) : f \in \mathcal{F}\}$ and $\sigma_1, \cdots, \sigma_m$ are independent uniform $\{-1, +1\}$-valued random variables. As the ramp loss is Lipschitz with constant 1, we further bound the first term in the right-hand side by the result of Bartlett and Mendelson (2003, Theorem 12) as

$$\begin{aligned}
& \mathbb{E}_{\mathbf{z}}\mathbb{E}_\sigma \left[\sup_{g\in\tilde{\mathcal{F}}} \left| \frac{2}{m}\sum_{i=1}^m \sigma_i g(x_i, y_i)\right|\right] \\
\leq\ & 2\mathbb{E}_{\mathbf{z}}\mathbb{E}_\sigma \left[\sup_{f\in\mathcal{F}} \left| \frac{2}{m}\sum_{i=1}^m \sigma_i f(x_i)\right|\right] \\
\leq\ & 2\mathbb{E}_{\mathbf{z}}\mathbb{E}_\sigma \left[\sup_{\{f^*\in\mathcal{H}_\mathcal{K}:\|f^*\|_\mathcal{K}\leq\frac{\kappa}{\mu}\}} \left| \frac{2}{m}\sum_{i=1}^m \sigma_i f^*(x_i)\right|\right] + \frac{2}{\sqrt{m}} + \frac{2\kappa^2}{\mu\sqrt{m}} \\
\leq\ & \frac{6\kappa^2}{\mu\sqrt{m}} + \frac{2}{\sqrt{m}}.
\end{aligned}$$

Here, the last inequality is from Lemma 22 in Bartlett and Mendelson (2003). Combining the above bound and (29), (30), we then have with probability at least $1 - \delta$,

$$\mathcal{S}(m, \mu, \lambda) \leq (2 + \eta)\sqrt{\frac{8\log\frac{4}{\delta}}{m}} + \frac{6\kappa^2}{\mu\sqrt{m}} + \frac{2}{\sqrt{m}}.$$

Finally, we let $\mu = m^{-\frac{\beta+1}{4\beta+2}}$ and $\lambda = m^{-\frac{1}{4\beta+2}}$. Then $\eta = \frac{\mu}{\lambda} = m^{-\frac{\beta}{4\beta+2}} \leq 1$. Therefore, by Theorem 2 and Theorem 4, we can derive the bound (20) with $\tilde{c} = 15 + 2c_\beta + 6\kappa^2$. This completes our proof. ■

References

L.T.H. An and P.D. Tao. The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. *Annals of Operations Research*, 133(1):23–46, 2005.

L.T.H. An, P.D. Tao, and L.D. Muu. Numerical solution for optimization over the efficient set by DC optimization algorithms. *Operations Research Letters*, 19(3):117–128, 1996.

N. Aronszajn, Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

P.L. Bartlett, M.I. Jordan, and J.D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

P.L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2003.

P.S. Bradley and O.L. Mangasarian. Massive data discrimination via linear support vector machines. *Optimization Methods and Software*, 13(1):1–10, 2000.

J.P. Brooks. Support vector machines with the ramp loss and the hard margin loss. *Operations Research*, 59(2):467–479, 2011.

A. Chipperfield, P. Fleming, H. Pohlheim, and C. Fonseca. Genetic algorithm toolbox user's guide. *Research Report*, 1994.

R. Collobert, F. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 201–208. ACM, 2006a.

R. Collobert, F. Sinz, J. Weston, and L. Bottou. Large scale transductive SVMs. *Journal of Machine Learning Research*, 7:1687–1712, 2006b.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

F. Cucker and D.X. Zhou. *Learning Theory: an Approximation Theory Viewpoint*. Cambridge University Press, 2007.

K. De Brabanter, K. Pelckmans, J. De Brabanter, M. Debruyne, J.A.K. Suykens, M. Hubert, and B. De Moor, Robustness of kernel based regression: a comparison of iterative weighting schemes. In *Proceedings of the 19th International Conference on Artificial Neural Networks,* pages 100–110, 2009.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

R. Horst and T. Hoang. *Global Optimization: Deterministic Approaches*. Springer Verlag, 1996.

F.M. Fung and O.L. Mangasarian. A feature selection Newton method for support vector machine classification. *Computational Optimization and Applications*, 28(2):185–202, 2004.

R. Horst and N.V. Thoai. DC programming: overview. *Journal of Optimization Theory and Applications*, 103(1):1–43, 1999.

X. Huang, J. Xu, X. Mu, and S. Wang. The hill detouring method for minimizing hinging hyperplanes functions. *Computers & Operations Research*, 39(7):1763–1770, 2012a.

X. Huang, J. Xu, and S. Wang. Exact penalty and optimality condition for nonseparable continuous piecewise linear programming. *Journal of Optimization Theory and Applications*, 155(1):145–164, 2012b.

V. Kecman and I. Hadzic. Support vectors selection by linear programming. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 5, pages 193–198. IEEE, 2000.

Y. Lin. A note on margin-based loss functions in classification. *Statistics & Probability Letters*, 68(1):73–82, 2004.

O.L. Mangasarian. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *Journal of Machine Learning Research*, 7:1517–1530, 2006.

O.L. Mangasarian. Absolute value equation solution via concave minimization. *Optimization Letters*, 1(1):3–8, 2007.

L. Mason, J. Baxter, P.L. Bartlett, and M. Frean. Boosting algorithms as gradient descent in function space. In S.A. Solla, T.K. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, 12:512–518, Cambridge, MA, MIT Press, 2000.

M. Porembski. Cutting planes for low-rank-like concave minimization problems. *Operations Research*, pages 942–953, 2004.

S. Smale and D.X. Zhou. Estimating the approximation error in learning theory. *Analysis and Applications*, 1(1):17–41, 2003.

A. Smola, B. Schölkopf, and G. Rätsch. Linear programs for automatic accuracy control in regression In *Proceedings of the 9th International Conference on Artificial Nerual Networks*, No. 470, pages 575–580, 1999.

X. Shen, G.C. Tseng, X. Zhang, and W. Wong. On ψ-learning. *Journal of the American Statistical Association*, 98(463):724–734, 2003.

J. Shu and I.A. Karimi. Efficient heuristics for inventory placement in acyclic networks. *Computers & Operations Research*, 36(11):2899–2904, 2009.

I. Steinwart. How to compare different loss functions and their risks. *Constructive Approximation*, 26(2):225–287, 2007.

I. Steinwart and A. Christmann. *Support Vector Machines*. New York: Springer, 2008.

J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

K. Wang, P. Zhong, and Y. Zhao. Training robust support vector regression via DC program. *Journal of Information and Computational Science*, 7(12):23852394, 2010.

Q. Wu and D.X. Zhou. SVM soft margin classifiers: linear programming versus quadratic programming. *Neural Computation*, 17(5):1160–1187, 2005.

Y. Wu and Y. Liu. Robust truncated hinge loss support vector machines. *Journal of the American Statistical Association*, 102(479):974–983, 2007.

A.L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4): 915–936, 2003.

T. Zhang. Statistical analysis of some multi-category large margin classification methods. *Journal of Machine Learning Research*, 5:1225–1251, 2004.

Y. Zhao and J. Sun. Robust support vector regression in the primal. *Neural Networks*, 21(10): 1548–1555, 2008.

J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani, 1-norm Support Vector Machines. In S. Thrun, L.K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, 16:49–56, Cambridge, MA, MIT Press, 2004.

Journal of Machine Learning Research 15 (2014) 2213-2238 Submitted 12/11; Revised 2/13; Published 6/14

Clustering Partially Observed Graphs via Convex Optimization

Yudong Chen YDCHEN@UTEXAS.EDU
Ali Jalali ALIJ@MAIL.UTEXAS.EDU
Sujay Sanghavi SANGHAVI@MAIL.UTEXAS.EDU
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712, USA

Huan Xu MPEXUH@NUS.EDU.SG
Department of Mechanical Engineering
National University of Singapore
Singapore 117575, SINGAPORE

Editor: Marina Meila

Abstract

This paper considers the problem of clustering a partially observed unweighted graph—i.e., one where for some node pairs we know there is an edge between them, for some others we know there is no edge, and for the remaining we do not know whether or not there is an edge. We want to organize the nodes into disjoint clusters so that there is relatively dense (observed) connectivity within clusters, and sparse across clusters.

We take a novel yet natural approach to this problem, by focusing on finding the clustering that minimizes the number of "disagreements"—i.e., the sum of the number of (observed) missing edges within clusters, and (observed) present edges across clusters. Our algorithm uses convex optimization; its basis is a reduction of disagreement minimization to the problem of recovering an (unknown) low-rank matrix and an (unknown) sparse matrix from their partially observed sum. We evaluate the performance of our algorithm on the classical Planted Partition/Stochastic Block Model. Our main theorem provides sufficient conditions for the success of our algorithm as a function of the minimum cluster size, edge density and observation probability; in particular, the results characterize the tradeoff between the observation probability and the edge density gap. When there are a constant number of clusters of equal size, our results are optimal up to logarithmic factors.

Keywords: graph clustering, convex optimization, sparse and low-rank decomposition

1. Introduction

This paper is about the following task: given partial observation of an undirected unweighted graph, partition the nodes into disjoint clusters so that there are dense connections within clusters, and sparse connections across clusters. By partial observation, we mean that for some node pairs we know if there is an edge or not, and for the other node pairs we do not know—these pairs are *unobserved*. This problem arises in several fields across science and engineering. For example, in sponsored search, each cluster is a submarket that represents a specific group of advertisers that do most of their spending on a group of query phrases—see e.g., Yahoo!-Inc (2009) for such a project at Yahoo. In VLSI and design automation, it is useful in minimizing signaling between components (Kernighan and Lin, 1970). In social networks, clusters may represent groups of people with

similar interest or background; finding clusters enables better recommendations and link prediction (Mishra et al., 2007). In the analysis of document databases, clustering the citation graph is often an essential and informative first step (Ester et al., 1995). In this paper, we will focus not on specific application domains, but rather on the basic graph clustering problem itself.

Partially observed graphs appear in many applications. For example, in online social networks like Facebook, we observe an edge/no edge between two users when they accept each other as a friend or explicitly decline a friendship suggestion. For the other user pairs, however, we simply have no friendship information between them, which are thus unobserved. More generally, we have partial observations whenever obtaining similarity data is difficult or expensive (e.g., because it requires human participation). In these applications, it is often the case that *most* pairs are unobserved, which is the regime we are particularly interested in.

As with any clustering problem, we need a precise mathematical definition of the clustering criterion with potentially a guaranteed performance. There is relatively few existing results with provable performance guarantees for graph clustering with partially observed node pairs. Many existing approaches to clustering fully observed graphs either require an additional input (e.g., the number of clusters k required for spectral or k-means clustering methods), or do not guarantee the performance of the clustering. We review existing related work in Section 1.2.

1.1 Our Approach

We focus on a natural formulation, one that does not require any other extraneous input besides the graph itself. It is based on minimizing *disagreements*, which we now define. Consider any candidate clustering; this will have (a) observed node pairs that are in different clusters, but have an edge between them, and (b) observed node pairs that are in the same cluster, but do not have an edge between them. The total number of node pairs of types (a) and (b) is the number of disagreements between the clustering and the given graph. We focus on the problem of finding the optimal clustering—one that minimizes the number of disagreements. Note that we do *not* pre-specify the number of clusters. For the special case of fully observed graphs, this formulation is exactly the same as the problem of *correlation clustering*, first proposed by Bansal et al. (2002). They show that exact minimization of the above objective is NP-hard in the worst case—we survey and compare with this and other related work in Section 1.2. As we will see, our approach and results are different.

We aim to achieve the combinatorial disagreement minimization objective using matrix splitting via convex optimization. In particular, as we show in Section 2 below, one can represent the adjacency matrix of the given graph as the sum of an unknown low-rank matrix (corresponding to "ideal" clusters) and a sparse matrix (corresponding to disagreements from this "ideal" in the given graph). Our algorithm either returns a clustering, which is guaranteed to be disagreement minimizing, or returns a "failure"—it never returns a sub-optimal clustering. For our main analytical result, we evaluate our algorithm's performance on the natural and classical *planted partition/stochastic block model* with partial observations. Our analysis provides stronger guarantees than are current results on general matrix splitting (Candès et al., 2011; Hsu et al., 2011; Li, 2013; Chen et al., 2013). The algorithm, model and results are given in Section 2. We prove our theoretical results in Section 3 and provide empirical results in Section 4.

1.2 Related Work

Our problem can be interpreted in the general clustering context as one in which the presence of an edge between two points indicates a "similarity", and the lack of an edge means "no similarity". The general field of clustering is of course vast, and a detailed survey of all methods therein is beyond our scope here. We focus instead on the three sets of papers most relevant to the problem here: the work on correlation clustering, on the planted partition/stochastic block model, and on graph clustering with partial observations.

1.2.1 CORRELATION CLUSTERING

As mentioned, for a completely observed graph, our problem is mathematically precisely the same as correlation clustering formulated in Bansal et al. (2002); in particular a "+" in correlation clustering corresponds to an edge in the graph, a "-" to the lack of an edge, and disagreements are defined in the same way. Thus, this paper can equivalently be considered as an algorithm, and guarantees, for *correlation clustering under partial observations*. Since correlation clustering is NP-hard, there has been much work on devising alternative approximation algorithms (Bansal et al., 2002; Emmanuel and Fiat, 2003). Approximations using convex optimization, including LP relaxation (Charikar et al., 2003; Demaine and Immorlica, 2003; Demaine et al., 2006) and SDP relaxation (Swamy, 2004; Mathieu and Schudy, 2010), possibly followed by rounding, have also been developed. We emphasize that we use a different convex relaxation, and we focus on understanding when our convex program yields an optimal clustering without further rounding.

We note that Mathieu and Schudy (2010) use a convex formulation with constraints enforcing positive semi-definiteness, triangle inequalities and fixed diagonal entries. For the fully observed case, their relaxation is at least as tight as ours, and since they add more constraints, it is possible that there are instances where their convex program works and ours does not. However, this seems hard to prove/disprove. Indeed, in the full observation setting they consider, their exact recovery guarantee is no better than ours. Moreover, as we argue in the next section, our guarantees are order-wise optimal in some important regimes and thus cannot be improved even with a tighter relaxation. Practically, our method is faster since, to the best of our knowledge, there is no low-complexity algorithm to deal with the $\Theta(n^3)$ triangle inequality constraints required by Mathieu and Schudy (2010). This means that our method can handle large graphs while their result is practically restricted to small ones (~ 100 nodes). In summary, their approach has higher computational complexity, and does not provide significant and characterizable performance gain in terms of exact cluster recovery.

1.2.2 PLANTED PARTITION MODEL

The planted partition model, also known as the stochastic block-model (Condon and Karp, 2001; Holland et al., 1983), assumes that the graph is generated with in-cluster edge probability p and inter-cluster edge probability q (where $p > q$) and fully observed. The goal is to recover the latent cluster structure. A class of this model with $\tau \triangleq \max\{1-p, q\} < \frac{1}{2}$ is often used as benchmark for *average case* performance for correlation clustering (see, e.g., Mathieu and Schudy, 2010). Our theoretical results are applicable to this model and thus directly comparable with existing work in this area. A detailed comparison is provided in Table 1. For fully observed graphs, our result matches the previous best bounds in both the minimum cluster size and the difference between in-cluster/inter-cluster densities. We would like to point out that nuclear norm minimization has

been used to solve the closely related planted clique problem (Alon et al., 1998; Ames and Vavasis, 2011).

Paper	Cluster size K	Density difference $(1-2\tau)$
Boppana (1987)	$n/2$	$\tilde{\Omega}(\frac{1}{\sqrt{n}})$
Jerrum and Sorkin (1998)	$n/2$	$\tilde{\Omega}(\frac{1}{n^{1/6-\epsilon}})$
Condon and Karp (2001)	$\tilde{\Omega}(n)$	$\tilde{\Omega}(\frac{1}{n^{1/2-\epsilon}})$
Carson and Impagliazzo (2001)	$n/2$	$\tilde{\Omega}(\frac{1}{\sqrt{n}})$
Feige and Kilian (2001)	$n/2$	$\tilde{\Omega}(\frac{1}{\sqrt{n}})$
McSherry (2001)	$\tilde{\Omega}(n^{2/3})$	$\tilde{\Omega}(\sqrt{\frac{n^2}{K^3}})$
Bollobás and Scott (2004)	$\tilde{\Omega}(n)$	$\tilde{\Omega}(\sqrt{\frac{1}{n}})$
Giesen and Mitsche (2005)	$\tilde{\Omega}(\sqrt{n})$	$\tilde{\Omega}(\frac{\sqrt{n}}{K})$
Shamir and Tsur (2007)	$\tilde{\Omega}(\sqrt{n})$	$\tilde{\Omega}(\frac{\sqrt{n}}{K})$
Mathieu and Schudy (2010)	$\tilde{\Omega}(\sqrt{n})$	$\tilde{\Omega}(1)$
Rohe et al. (2011)	$\tilde{\Omega}(n^{3/4})$	$\tilde{\Omega}(\frac{n^{3/4}}{K})$
Oymak and Hassibi (2011)	$\tilde{\Omega}(\sqrt{n})$	$\tilde{\Omega}(\frac{\sqrt{n}}{K})$
Chaudhuri et al. (2012)	$\tilde{\Omega}(\sqrt{n})$	$\tilde{\Omega}(\frac{\sqrt{n}}{K})$
This paper	$\tilde{\Omega}(\sqrt{n})$	$\tilde{\Omega}(\frac{\sqrt{n}}{K})$

Table 1: *Comparison with literature.* This table shows the lower-bound requirements on the minimum cluster size K and the density difference $p-q=1-2\tau$ that existing literature needs for exact recovery of the planted partitions, when the graph is fully observed and $\tau \triangleq \max\{1-p,q\} = \Theta(1)$. Some of the results in the table only guarantee recovering the membership of most, instead of all, nodes. To compare with these results, we use the soft-Ω notation $\tilde{\Omega}(\cdot)$, which hides the logarithmic factors that are necessary for recovering all nodes, which is the goal of this paper.

1.2.3 PARTIALLY OBSERVED GRAPHS

The previous work listed in Table 1, except Oymak and Hassibi (2011), does not handle partial observations directly. One natural way to proceed is to impute the missing observations with no-edge, or random edges with symmetric probabilities, and then apply any of the results in Table 1. This approach, however, leads to sub-optimal results. Indeed, this is done explicitly by Oymak and Hassibi (2011). They require the probability of observation p_0 to satisfy $p_0 \gtrsim \frac{\sqrt{K_{\min}}}{n}$, where n is the number of nodes and $K_{\min}$ is the minimum cluster size; in contrast, our approach only needs $p_0 \gtrsim \frac{n}{K_{\min}^2}$ (both right hand sides have to be less than 1, requiring $K_{\min} \gtrsim \sqrt{n}$, so the right hand side of our condition is order-wise smaller and thus less restrictive.) Shamir and Tishby (2011) deal with partial observations directly and shows that $p_0 \gtrsim \frac{1}{n}$ suffices for recovering two clusters of size $\Omega(n)$. Our result is applicable to much smaller clusters of size $\tilde{\Omega}(\sqrt{n})$. In addition, a nice feature of

our result is that it explicitly characterizes the tradeoffs between the three relevant parameters: p_0, τ, and $K_{\min}$; theoretical result like this is not available in previous work.

There exists other work that considers partial observations, but under rather different settings. For example, Balcan and Gupta (2010), Voevodski et al. (2010) and Krishnamurthy et al. (2012) consider the clustering problem where one samples the rows/columns of the adjacency matrix rather than its entries. Hunter and Strohmer (2010) consider partial observations in the features rather than in the similarity graph. Eriksson et al. (2011) show that $\tilde{\Omega}(n)$ actively selected pairwise similarities are sufficient for recovering a hierarchical clustering structure. Their results seem to rely on the hierarchical structure. When disagreements are present, the first split of the cluster tree can recovers clusters of size $\Omega(n)$; our results allow smaller clusters. Moreover, they require active control over the observation process, while we assume random observations.

2. Main Results

Our setup for the graph clustering problem is as follows. We are given a partially observed graph of n nodes, whose adjacency matrix is $\mathbf{A} \in \mathbb{R}^{n\times n}$, which has $a_{i,j} = 1$ if there is an edge between nodes i and j, $a_{i,j} = 0$ if there is no edge, and $a_{i,j} =$"?" if we do not know. (Here we follow the convention that $a_{i,i} = 0$ for all i.) Let $\Omega_{\text{obs}} \triangleq \{(i,j) : a_{i,j} \neq ?\}$ be the set of observed node pairs. The goal to find the optimal clustering, i.e., the one that has the minimum number of disagreements (defined in Section 1.1) in Ω_{obs}.

In the rest of this section, we present our algorithm for the above task and analyze its performance under the planted partition model with partial observations. We also study the optimality of the performance of our algorithm by deriving a necessary condition for any algorithm to succeed.

2.1 Algorithm

Our algorithm is based on convex optimization, and either (a) outputs a clustering that is guaranteed to be the one that minimizes the number of observed disagreements, or (b) declares "failure". In particular, it never produces a suboptimal clustering.[1] We now briefly present the main idea and then describe the algorithm.

Consider first the fully observed case, i.e., every $a_{i,j} = 0$ or 1. Suppose also that the graph is already ideally clustered—i.e., there is a partition of the nodes such that there is no edge between clusters, and each cluster is a clique. In this case, the matrix $\mathbf{A} + \mathbf{I}$ is now a *low-rank* matrix, with the rank equal to the number of clusters. This can be seen by noticing that if we re-order the rows and columns so that clusters appear together, the result would be a *block-diagonal* matrix, with each block being an all-ones submatrix—and thus rank one. Of course, this re-ordering does not change the rank of the matrix, and hence $\mathbf{A} + \mathbf{I}$ is exactly low-rank.

Consider now any given graph, still fully observed. In light of the above, we are looking for a decomposition of its $\mathbf{A} + \mathbf{I}$ into a low-rank part $\mathbf{K}^*$ (of block-diagonal all-ones, one block for each cluster) and a remaining $\mathbf{B}^*$ (the disagreements), such that the number of non-zero entries in $\mathbf{B}^*$ is as small as possible; i.e., $\mathbf{B}^*$ is sparse. Finally, the problem we look at is recovery of the best $\mathbf{K}^*$ when we do not observe all entries of $\mathbf{A} + \mathbf{I}$. The idea is depicted in Figure 1.

1. In practice, one might be able to use the "failed" output with rounding as an approximate solution. In this paper, we focus on the performance of the unrounded algorithm.

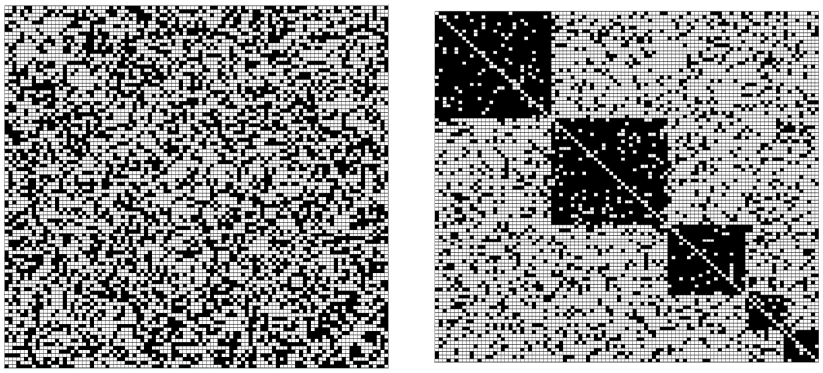


Figure 1: The adjacency matrix of a graph drawn from the planted partition model before and after proper reordering (i.e., clustering) of the nodes. The figure on the right is indicative of the matrix as a superposition of a sparse matrix and a low-rank block diagonal one.

We propose to perform the matrix splitting using convex optimization (Chandrasekaran et al., 2011; Candès et al., 2011). Our approach consists of dropping any additional structural requirements, and just looking for a decomposition of the given $\mathbf{A}+\mathbf{I}$ as the sum of a sparse matrix $\mathbf{B}$ and a low-rank matrix $\mathbf{K}$. Recall that Ω_{obs} is the set of observed entries, i.e., the set of elements of $\mathbf{A}$ that are known to be 0 or 1; we use the following convex program:

$$\begin{aligned} \min_{\mathbf{B},\mathbf{K}} \quad & \lambda \|\mathbf{B}\|_1 + \|\mathbf{K}\|_* \\ \text{s.t.} \quad & \mathcal{P}_{\Omega_{obs}}(\mathbf{B}+\mathbf{K}) = \mathcal{P}_{\Omega_{obs}}(\mathbf{A}+\mathbf{I}). \end{aligned} \tag{1}$$

Here, for any matrix $\mathbf{M}$, the term $\mathcal{P}_{\Omega_{\text{obs}}}(\mathbf{M})$ keeps all elements of $\mathbf{M}$ in Ω_{obs} unchanged, and sets all other elements to 0; the constraints thus state that the sparse and low-rank matrix should in sum be consistent with the observed entries of $\mathbf{A}+\mathbf{I}$. The term $\|\mathbf{B}\|_1 = \sum_{i,j}|b_{i,j}|$ is the ℓ_1 norm of the entries of the matrix $\mathbf{B}$, which is well-known to be a convex surrogate for the number of non-zero entries $\|\mathbf{B}\|_0$. The second term $\|\mathbf{K}\|_* = \sum_s \sigma_s(\mathbf{K})$ is the nuclear norm (also known as the trace norm), defined as the sum of the singular values of $\mathbf{K}$. This has been shown to be the tightest convex surrogate for the rank function for matrices with unit spectral norm (Fazel, 2002). Thus our objective function is a convex surrogate for the (natural) combinatorial objective $\lambda\|\mathbf{B}\|_0 + \text{rank}(\mathbf{K})$. The optimization problem (1) is, in fact, a semidefinite program (SDP) (Chandrasekaran et al., 2011).

We remark on the above formulation. (a) This formulation does not require specifying the number of clusters; this parameter is effectively learned from the data. The tradeoff parameter λ is artificial and can be easily determined: since any desired $\mathbf{K}^*$ has trace exactly equal to n, we simply choose the smallest λ such that the trace of the optimal solution is at least n. This can be done by, e.g., bisection, which is described below. (b) It is possible to obtain tighter convex relaxations by adding more constraints, such as the diagonal entry constraints $k_{i,i} = 1, \forall i$, the positive semi-definite constraint $\mathbf{K} \succeq 0$, or even the triangular inequalities $k_{i,j} + k_{j,k} - k_{i,k} \leq 1$. Indeed, this is done by Mathieu and Schudy (2010). Note that the guarantees for our formulation (to be presented in the next subsection) automatically imply guarantees for any other tighter relaxations. We choose to focus on our (looser) formulation for two reasons. First, and most importantly, even with the extra

constraints, Mathieu and Schudy (2010) do not deliver better exact recovery guarantees (cf. Table 1). In fact, we show in Section 2.3 that our results are near optimal in some important regimes, so tighter relaxations do not seem to provide additional benefits in exact recovery. Second, our formulation can be solved efficiently using existing Augmented Lagrangian Multiplier methods (Lin et al., 2009). This is no longer the case with the $\Theta(n^3)$ triangle inequality constraints enforced by Mathieu and Schudy (2010), and solving it as a standard SDP is only feasible for small graphs.

We are interested in the case when the convex program (1) produces an optimal solution $\mathbf{K}$ that is a block-diagonal matrix and corresponds to an ideal clustering.

Definition 1 (Validity) *The convex program* (1) *is said to produce a* valid *output if the low-rank matrix part* $\mathbf{K}$ *of the optimal solution corresponds to a graph of disjoint cliques; i.e., its rows and columns can be re-ordered to yield a block-diagonal matrix with all ones for each block.*

Validity of a given $\mathbf{K}$ can be easily checked via elementary re-ordering operations.[2] Our first simple but useful insight is that whenever the convex program (1) yields a valid solution, it is the disagreement minimizer.

Theorem 2 *For any* $\lambda > 0$*, if the solution of* (1) *is valid, then it is the clustering that minimizes the number of observed disagreements.*

Our complete clustering procedure is given as Algorithm 1. It takes the adjacency matrix of the graph $\mathbf{A}$ and outputs either the optimal clustering or declares failure. Setting the parameter λ is done via binary search. The initial value of λ is not crucial; we use $\lambda = \frac{1}{32\sqrt{\bar{p}_0 n}}$ based on our theoretical analysis in the next sub-section, where $\bar{p}_0$ is the empirical fraction of observed pairs. To solve the optimization problem (1), we use the fast algorithm developed by Lin et al. (2009), which is tailored for matrix splitting and takes advantage of the sparsity of the observations. By Theorem 2, whenever the algorithm results in a valid $\mathbf{K}$, we have found the optimal clustering.

Algorithm 1 Optimal-Cluster($\mathbf{A}$)

$\lambda \leftarrow \frac{1}{32\sqrt{\bar{p}_0 n}}$
while not terminated **do**
  Solve (1) to obtain the solution $(\mathbf{B}, \mathbf{K})$
  if $\mathbf{K}$ is valid **then**
    Output the clustering in $\mathbf{K}$ and EXIT.
  else if trace$(\mathbf{K}) > n$ **then**
    $\lambda \leftarrow \lambda/2$
  else if trace$(\mathbf{K}) < n$ **then**
    $\lambda \leftarrow 2\lambda$
  end if
end while
Declare Failure.

2. If we re-order a valid $\mathbf{K}$ such that identical rows and columns appear together, it will become block-diagonal.

2.2 Performance Analysis

For the main analytical contribution of this paper, we provide conditions under which the above algorithm will find the clustering that minimizes the number of disagreements among the observed entries. In particular, we characterize its performance under the standard and classical planted partition/stochastic block model with partial observations, which we now describe.

Definition 3 (Planted Partition Model with Partial Observations) *Suppose that n nodes are partitioned into r clusters, each of size at least $K_{\min}$. Let $\mathbf{K}^*$ be the low-rank matrix corresponding to this clustering (as described above). The adjacency matrix $\mathbf{A}$ of the graph is generated as follows: for each pair of nodes (i,j) in the same cluster, $a_{i,j} =?$ with probability $1-p_0$, $a_{i,j}=1$ with probability $p_0 p$, or $a_{ij}=0$ otherwise, independent of all others; similarly, for (i,j) in different clusters, $a_{i,j} =?$ with probability $1-p_0$, $a_{i,j}=1$ with probability $p_0 q$, or $a_{i,j}=0$ otherwise.*

Under the above model, the graph is observed at locations chosen at random with probability p_0. In expectation a fraction of $1-p$ of the in-cluster observations are disagreements; similarly, the fraction of disagreements in the across-cluster observations is q. Let $\mathbf{B}^* = \mathcal{P}_{\Omega_{\text{obs}}}(\mathbf{A}+\mathbf{I}-\mathbf{K}^*)$ be the matrix of observed disagreements for the original clustering; note that the support of $\mathbf{B}^*$ is contained in Ω_{obs}. The following theorem provides a sufficient condition for our algorithm to recover the original clustering $(\mathbf{B}^*, \mathbf{K}^*)$ with high probability. Combined with Theorem 2, it also shows that under the same condition, the original clustering is disagreement minimizing with high probability.

Theorem 4 *Let $\tau = \max\{1-p, q\}$. There exist universal positive constants c and C such that, with probability at least $1-cn^{-10}$, the original clustering $(\mathbf{B}^*, \mathbf{K}^*)$ is the unique optimal solution of (1) with $\lambda = \frac{1}{32\sqrt{np_0}}$ provided that*

$$p_0\,(1-2\tau)^2 \geq C\frac{n\log^2 n}{K_{\min}^2}. \tag{2}$$

Note that the quantity τ is (an upper bound of) the probability of having a disagreement, and $1-2\tau$ is (a lower bound of) the density gap $p-q$. The sufficient condition in the theorem is given in terms of the three parameters that define problem: the minimum cluster size $K_{\min}$, the density gap $1-2\tau$, and the observation probability p_0. We remark on these parameters.

- *Minimum cluster size* $K_{\min}$. Since the left hand side of the condition (2) in Theorem 4 is no more than 1, it imposes a lower-bound $K_{\min} = \tilde{\Omega}(\sqrt{n})$ on the cluster sizes. This means that our method can handle a growing number $\tilde{O}(\sqrt{n})$ of clusters. The lower-bound on $K_{\min}$ is attained when $1-2\tau$ and p_0 are both $\Theta(1)$, i.e., not decreasing as n grows. Note that all relevant works require a lower-bound at least as strong as ours (cf. Table 1).

- *Density gap* $1-2\tau$. When $p_0 = \Theta(1)$, our result allows this gap to be vanishingly small, i.e., $\tilde{\Omega}\left(\frac{\sqrt{n}}{K_{\min}}\right)$, where a larger $K_{\min}$ allows for a smaller gap. As we mentioned in Section 1.2, this matches the best available results (cf. Table 1), including those in Mathieu and Schudy (2010) and Oymak and Hassibi (2011), which use tighter convex relaxations that are more computationally demanding. We note that directly applying existing results in the low-rank-plus-sparse literature (Candès et al., 2011; Li, 2013) leads to weaker results, where the gap be bounded below by a constant.

- *Observation probability* p_0. When $1-2\tau = \Theta(1)$, our result only requires a vanishing fraction of observations, i.e., p_0 can be as small as $\tilde{\Theta}\left(\frac{n}{K_{\min}^2}\right)$; a larger $K_{\min}$ allows for a smaller p_0. As mentioned in Section 1.2, this scaling is better than prior results we know of.
- *Tradeoffs.* A novel aspect of our result is that it shows an explicit tradeoff between the observation probability p_0 and the density gap $1-2\tau$. The left hand side of (2) is linear in p_0 and quadratic in $1-2\tau$. This means if the number of observations is four times larger, then we can handle a 50% smaller density gap. Moreover, p_0 can go to zero quadratically faster then $1-2\tau$. Consequently, treating missing observations as disagreements would lead to quadratically weaker results. This agrees with the intuition that handling missing entries with known locations is easier than correcting disagreements whose locations are unknown.

We would like to point out that our algorithm has the capability to handle outliers. Suppose there are some isolated nodes which do not belong to any cluster, and they connect to each other and each node in the clusters with probability at most τ, with τ obeying the condition (2) in Theorem 4. Our algorithm will classify all these edges as disagreements, and hence automatically reveal the identity of each outlier. In the output of our algorithm, the low rank part $\mathbf{K}$ will have all zeros in the columns and rows corresponding to outliers—all their edges will appear in the disagreement matrix $\mathbf{B}$.

2.3 Lower Bounds

We now discuss the tightness of Theorem 4. Consider first the case where $K_{\min} = \Theta(n)$, which means there are a constant number of clusters. We establish a fundamental lower bound on the density gap $1-2\tau$ and the observation probability p_0 that are required for *any* algorithm to correctly recover the clusters.

Theorem 5 *Under the planted partition model with partial observations, suppose the true clustering is chosen uniformly at random from all possible clusterings with equal cluster size K. If $K = \Theta(n)$ and $\tau = 1-p = q > 1/100$, then for any algorithm to correctly identify the clusters with probability at least $\frac{3}{4}$, we need*

$$p_0(1-2\tau)^2 \geq C\frac{1}{n},$$

where $C > 0$ is an absolute constant.

Theorem 5 generalizes a similar result in Chaudhuri et al. (2012), which does not consider partial observations. The theorem applies to any algorithm regardless of its computational complexity, and characterizes the fundamental tradeoff between p_0 and $1-2\tau$. It shows that when $K_{\min} = \Theta(n)$, the requirement for $1-2\tau$ and p_0 in Theorem 4 is optimal up to logarithmic factors, and cannot be significantly improved by using more complicated methods.

For the general case with $K_{\min} = O(n)$, only part of the picture is known. Using non-rigorous arguments, Decelle et al. (2011) show that $1-2\tau \gtrsim \frac{\sqrt{n}}{K_{\min}}$ is necessary when $\tau = \Theta(1)$ and the graph is fully observed; otherwise recovery is impossible or computationally hard. According to this lower-bound, our requirement on the density gap $1-2\tau$ is probably tight (up to log factors) for all $K_{\min}$. However, a rigorous proof of this claim is still lacking, and seems to be a difficult problem. Similarly, the tightness of our condition on p_0 and the tradeoff between p_0 and τ is also unclear in this regime.

3. Proofs

In this section, we prove Theorems 2 and 4. The proof of Theorem 5 is deferred to Appendix B.

3.1 Proof of Theorem 2

We first prove Theorem 2, which says that if the optimization problem (1) produces a valid matrix, i.e., one that corresponds to a clustering of the nodes, then this is the disagreement minimizing clustering. Consider the following non-convex optimization problem

$$\begin{aligned} \min_{\mathbf{B},\mathbf{K}} \quad & \lambda \left\|\mathbf{B}\right\|_1 + \left\|\mathbf{K}\right\|_* \\ \text{s.t.} \quad & \mathcal{P}_{\Omega_{\text{obs}}}(\mathbf{B}+\mathbf{K}) = \mathcal{P}_{\Omega_{\text{obs}}}(\mathbf{I}+\mathbf{A}), \\ & \mathbf{K} \text{ is valid,} \end{aligned} \tag{3}$$

and let $(\mathbf{B}, \mathbf{K})$ be any feasible solution. Since $\mathbf{K}$ represents a valid clustering, it is positive semidefinite and has all ones along its diagonal. Therefore, it obeys $\left\|\mathbf{K}\right\|_* = \text{trace}(\mathbf{K}) = n$. On the other hand, because both $\mathbf{K}-\mathbf{I}$ and $\mathbf{A}$ are adjacency matrices, the entries of $\mathbf{B} = \mathbf{I}+\mathbf{A}-\mathbf{K}$ in Ω_{obs} must be equal to -1, 1 or 0 (i.e., it is a disagreement matrix). Clearly any optimal $\mathbf{B}$ must have zeros at the entries in Ω_{obs}^c. Hence $\left\|\mathbf{B}\right\|_1 = \left\|\mathcal{P}_{\Omega_{\text{obs}}}(\mathbf{B})\right\|_0$ when $\mathbf{K}$ is valid. We thus conclude that the above optimization problem (3) is equivalent to minimizing $\left\|\mathcal{P}_{\Omega_{\text{obs}}}(\mathbf{B})\right\|_0$ subject to the same constraints. This is exactly the minimization of the number of disagreements on the observed edges. Now notice that (1) is a relaxed version of (3). Therefore, if the optimal solution of (1) is valid and thus feasible to (3), then it is also optimal to (3), the disagreement minimization problem.

3.2 Proof of Theorem 4

We now turn to the proof of Theorem 4, which provides guarantees for when the convex program (1) recovers the true clustering $(\mathbf{B}^*, \mathbf{K}^*)$.

3.2.1 PROOF OUTLINE AND PRELIMINARIES

We overview the main steps in the proof of Theorem 4; details are provided in Sections 3.2.2–3.2.4 to follow. We would like to show that the pair $(\mathbf{B}^*, \mathbf{K}^*)$ corresponding to the true clustering is the unique optimal solution to our convex program (1). This involves the following three steps.

Step 1: We show that it suffices to consider an equivalent model for the observation and disagreements. This model is easier to handle, especially when the observation probability and density gap are vanishingly small, which is the regime of interest in this paper.

Step 2: We write down the sub-gradient based first-order sufficient conditions that need to be satisfied for $(\mathbf{B}^*, \mathbf{K}^*)$ to be the unique optimum of (1). In our case, this involves showing the existence of a matrix $\mathbf{W}$—the *dual certificate*—that satisfies certain properties. This step is technical—requiring us to deal with the intricacies of sub-gradients since our convex function is not smooth—but otherwise standard. Luckily for us, this has been done previously (Chandrasekaran et al., 2011; Candès et al., 2011; Li, 2013).

Step 3: Using the assumptions made on the true clustering $\mathbf{K}^*$ and its disagreements $\mathbf{B}^*$, we construct a candidate dual certificate $\mathbf{W}$ that meets the requirements in step 2, and thus certify $(\mathbf{B}^*, \mathbf{K}^*)$ as being the unique optimum.

The crucial Step 3 is where we go beyond the existing literature on matrix splitting (Chandrasekaran et al., 2011; Candès et al., 2011; Li, 2013). These results assume the observation probability and/or density gap is at least a constant, and hence do not apply to our setting. Here we provide a refined analysis, which leads to better performance guarantees than those that could be obtained via a direct application of existing sparse and low-rank matrix splitting results.

Next, we introduce some notations used in the rest of the proof of the theorem. The following definitions related to $\mathbf{K}^*$ are standard. By symmetry, the SVD of $\mathbf{K}^*$ has the form $\mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$, where $\mathbf{U} \in \mathbb{R}^{n\times r}$ contains the singular vectors of $\mathbf{K}^*$. We define the subspace

$$\mathcal{T} \triangleq \left\{\mathbf{U}\mathbf{X}^T + \mathbf{Y}\mathbf{U}^T : \mathbf{X}, \mathbf{Y} \in \mathbb{R}^{n\times r}\right\},$$

which is spanned of all matrices that share either the same column space or the same row space as $\mathbf{K}^*$. For any matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$, its orthogonal projection to the space $\mathcal{T}$ is given by $\mathcal{P}_{\mathcal{T}}(\mathbf{M}) = \mathbf{U}\mathbf{U}^T\mathbf{M} + \mathbf{M}\mathbf{U}\mathbf{U}^T - \mathbf{U}\mathbf{U}^T\mathbf{M}\mathbf{U}\mathbf{U}^T$. The projection onto $\mathcal{T}^\perp$, the complement orthogonal space of $\mathcal{T}$, is given by $\mathcal{P}_{\mathcal{T}^\perp}(\mathbf{M}) = \mathbf{M} - P_{\mathcal{T}}(\mathbf{M})$.

The following definitions are related to $\mathbf{B}^*$ and partial observations. Let $\Omega^* = \{(i,j) : b^*_{i,j} \neq 0\}$ be the set of matrix entries corresponding to the disagreements. Recall that Ω_{obs} is the set of observed entries. For any matrix $\mathbf{M}$ and entry set Ω_0, we let $\mathcal{P}_{\Omega_0}(\mathbf{M}) \in \mathbb{R}^{n\times n}$ be the matrix obtained from $\mathbf{M}$ by setting all entries not in the set Ω_0 to zero. We write $\Omega_0 \sim \text{Ber}_0(p)$ if the entry set Ω_0 does not contain the diagonal entries, and each pair (i,j) and (j,i) $(i \neq j)$ is contained in Ω_0 with probability p, independent all others; $\Omega_0 \sim \text{Ber}_1(p)$ is defined similarly except that Ω_0 contains all the diagonal entries. Under our partially observed planted partition model, we have $\Omega_{\text{obs}} \sim \text{Ber}_1(p_0)$ and $\Omega^* \sim \text{Ber}_0(\tau)$.

Several matrix norms are used. $\|\mathbf{M}\|$ and $\|\mathbf{M}\|_F$ represent the spectral and Frobenius norms of the matrix $\mathbf{M}$, respectively, and $\|\mathbf{M}\|_\infty \triangleq \max_{i,j}|m_{i,j}|$ is the matrix infinity norm.

3.2.2 STEP 1: EQUIVALENT MODEL FOR OBSERVATIONS AND DISAGREEMENTS

It is easy show that increasing p or decreasing q can only make the probability of success higher, so without loss of generality we assume $1 - p = q = \tau$. Observe that the probability of success is completely determined by the distribution of $(\Omega_{\text{obs}}, \mathbf{B}^*)$ under the planted partition model with partial observations. The first step is to show that it suffices to consider an equivalent model for generating $(\Omega_{\text{obs}}, \mathbf{B}^*)$, which results in the same distribution but is easier to handle. This is in the same spirit as Candès et al. (2011, Theorems 2.2 and 2.3) and Li (2013, Section 4.1). In particular, we consider the following procedure:

1. Let $\Gamma \sim \text{Ber}_1\left(p_0(1-2\tau)\right)$, and $\Omega \sim \text{Ber}_0\left(\frac{2p_0\tau}{1-p_0+2p_0\tau}\right)$. Let $\Omega_{\text{obs}} = \Gamma \cup \Omega$.

2. Let $\mathbf{S}$ be a symmetric random matrix whose upper-triangular entries are independent and satisfy $\mathbb{P}(s_{i,j} = 1) = \mathbb{P}(s_{i,j} = -1) = \frac{1}{2}$.

3. Define $\Omega' \subseteq \Omega$ as $\Omega' = \left\{(i,j) : (i,j) \in \Omega, s_{i,j} = 1 - 2k^*_{i,j}\right\}$. In other words, Ω' is the entries of $\mathbf{S}$ whose signs are consistent with a disagreement matrix.

4. Define $\Omega^* = \Omega' \backslash \Gamma$, and $\tilde{\Gamma} = \Omega_{\text{obs}} \backslash \Omega^*$.

5. Let $\mathbf{B}^* = \mathcal{P}_{\Omega^*}(\mathbf{S})$.

It is easy to verify that $(\Omega_{\text{obs}}, \mathbf{B}^*)$ has the same distribution as in the original model. In particular, we have $\mathbb{P}[(i,j) \in \Omega_{\text{obs}}] = p_0$, $\mathbb{P}[(i,j) \in \Omega^*, (i,j) \in \Omega_{\text{obs}}] = p_0\tau$ and $\mathbb{P}[(i,j) \in \Omega^*, (i,j) \notin \Omega_{\text{obs}}] = 0$, and observe that given $\mathbf{K}^*$, $\mathbf{B}^*$ is completely determined by its support Ω^*.

The advantage of the above model is that Γ and Ω are independent of each other, and $\mathbf{S}$ has random signed entries. This facilitates the construction of the dual certificate, especially in the regime of vanishing p_0 and $\left(\frac{1}{2} - \tau\right)$ considered in this paper. We use this equivalent model in the rest of the proof.

3.2.3 STEP 2: SUFFICIENT CONDITIONS FOR OPTIMALITY

We state the first-order conditions that guarantee $(\mathbf{B}^*, \mathbf{K}^*)$ to be the unique optimum of (1) with high probability. Here and henceforth, by *with high probability* we mean with probability at least $1 - cn^{-10}$ for some universal constant $c > 0$. The following lemma follows from Theorem 4.4 in Li (2013) and the discussion thereafter.

Lemma 6 (Optimality Condition) *Suppose* $\left\| \frac{1}{(1-2\tau)p_0} \mathcal{P}_{\mathcal{T}} \mathcal{P}_{\Gamma} \mathcal{P}_{\mathcal{T}} - \mathcal{P}_{\mathcal{T}} \right\| \leq \frac{1}{2}$. *Then* $(\mathbf{B}^*, \mathbf{K}^*)$ *is the unique optimal solution to* (1) *with high probability if there exists* $\mathbf{W} \in \mathbb{R}^{n \times n}$ *such that*

1. $\left\| \mathcal{P}_{\mathcal{T}}(\mathbf{W} + \lambda \mathcal{P}_{\Omega} \mathbf{S} - \mathbf{U}\mathbf{U}^{\top}) \right\|_F \leq \frac{\lambda}{n^2}$,
2. $\| \mathcal{P}_{\mathcal{T}^{\perp}}(\mathbf{W} + \lambda \mathcal{P}_{\Omega} \mathbf{S}) \| \leq \frac{1}{4}$,
3. $\mathcal{P}_{\Gamma^c}(\mathbf{W}) = 0$,
4. $\| \mathcal{P}_{\Gamma}(\mathbf{W}) \|_{\infty} \leq \frac{\lambda}{4}$.

Lemma 9 in the appendix guarantees that the condition $\left\| \frac{1}{(1-2\tau)p_0} \mathcal{P}_{\mathcal{T}} \mathcal{P}_{\Gamma} \mathcal{P}_{\mathcal{T}} - \mathcal{P}_{\mathcal{T}} \right\| \leq \frac{1}{2}$ is satisfied with high probability under the assumption of Theorem 4. It remains to show the existence of a desired dual certificate $\mathbf{W}$ which satisfies the four conditions in Lemma 6 with high probability.

3.2.4 STEP 3: DUAL CERTIFICATE CONSTRUCTION

We use a variant of the so-called *golfing scheme* (Candès et al., 2011; Gross, 2011) to construct $\mathbf{W}$. Our application of golfing scheme, as well as its analysis, is different from previous work and leads to stronger guarantees. In particular, we go beyond existing results by allowing the fraction of observed entries and the density gap to be vanishing.

By definition in Section 3.2.2, Γ obeys $\Gamma \sim \text{Ber}_1(p_0(1-2\tau))$. Observe that Γ may be considered to be generated by $\Gamma = \bigcup_{1 \leq k \leq k_0} \Gamma_k$, where the sets $\Gamma_k \sim \text{Ber}_1(t)$ are independent; here the parameter t obeys $p_0(1-2\tau) = 1 - (1-t)^{k_0}$, and k_0 is chosen to be $\lceil 5 \log n \rceil$. This implies $t \geq p_0(1-2\tau)/k_0 \geq C_0 \frac{n \log n}{K_{\min}^2}$ for some constant C_0, with the last inequality holds under the assumption of Theorem 4. For any random entry set $\Omega_0 \sim \text{Ber}_1(\rho)$, define the operator $\mathcal{R}_{\Omega_0} : \mathbb{R}^{n \times n} \mapsto \mathbb{R}^{n \times n}$ by

$$\mathcal{R}_{\Omega_0}(\mathbf{M}) = \sum_{i=1}^{n} m_{i,i} e_i e_i^T + \rho^{-1} \sum_{1 \leq i < j \leq n} \delta_{ij} m_{i,j} \left(e_i e_j^T + e_j e_i^T \right),$$

where δ_{ij} is the indicator random variable with $\delta_{ij} = 1$ if $(i,j) \in \Omega_0$ and 0 otherwise, and e_i is the i-th standard basis in $\mathbb{R}^{n \times n}$, i.e., the column vector with 1 in its i-th entry and 0 elsewhere.

We now define our dual certificate. Let $\mathbf{W} = \mathbf{W}_{k_0}$, where $\mathbf{W}_{k_0}$ is defined recursively by setting $\mathbf{W}_0 = 0$ and for all $k = 1, 2, \ldots, k_0$,

$$\mathbf{W}_k = \mathbf{W}_{k-1} + \mathcal{R}_{\Gamma_k}\mathcal{P}_{\mathcal{T}}\left(\mathbf{U}\mathbf{U}^T - \lambda\mathcal{P}_{\mathcal{T}}(\mathcal{P}_{\Omega}(\mathbf{S})) - \mathbf{W}_{k-1}\right).$$

Clearly the equality condition in Lemma 6 is satisfied. It remains to show that $\mathbf{W}$ also satisfies the inequality conditions with high probability. The proof makes use of the technical Lemmas 9–12 given in the appendix. For convenience of notation, we define the quantity $\Delta_k = \mathbf{U}\mathbf{U}^T - \lambda\mathcal{P}_{\mathcal{T}}\left(\mathcal{P}_{\Omega}(\mathbf{S})\right) - \mathcal{P}_{\mathcal{T}}(\mathbf{W}_k)$, and use the notation

$$\prod_{i=1}^{k}(\mathcal{P}_{\mathcal{T}} - \mathcal{P}_{\mathcal{T}}\mathcal{R}_{\Gamma_i}\mathcal{P}_{\mathcal{T}}) = (\mathcal{P}_{\mathcal{T}} - \mathcal{P}_{\mathcal{T}}\mathcal{R}_{\Gamma_k}\mathcal{P}_{\mathcal{T}})\cdots(\mathcal{P}_{\mathcal{T}} - \mathcal{P}_{\mathcal{T}}\mathcal{R}_{\Gamma_1}\mathcal{P}_{\mathcal{T}}),$$

where the order of multiplication is important. Observe that by construction of $\mathbf{W}$, we have

$$\Delta_k = \prod_{i=1}^{k}(\mathcal{P}_{\mathcal{T}} - \mathcal{P}_{\mathcal{T}}\mathcal{R}_{\Gamma_i}\mathcal{P}_{\mathcal{T}})(\mathbf{U}\mathbf{U}^T - \lambda\mathcal{P}_{\mathcal{T}}\mathcal{P}_{\Omega}(\mathbf{S})), k = 1, \ldots, k_0, \tag{4}$$

$$\mathbf{W}_{k_0} = \sum_{k=1}^{k_0}\mathcal{R}_{\Gamma_k}\Delta_{k-1}. \tag{5}$$

We are ready to prove that $\mathbf{W}$ satisfies inequalities 1, 2 and 4 in Lemma 6.

Inequality 1: Thanks to (4), we have the following geometric convergence :

$$\begin{aligned}
\left\|\mathcal{P}_{\mathcal{T}}(\mathbf{W} + \lambda\mathcal{P}_{\Omega}\mathbf{S} - \mathbf{U}\mathbf{U}^{\top})\right\|_F &= \|\Delta_{k_0}\|_F \\
&\leq \left(\prod_{k=1}^{k_0}\|\mathcal{P}_{\mathcal{T}} - \mathcal{P}_{\mathcal{T}}\mathcal{R}_{\Gamma_k}\mathcal{P}_{\mathcal{T}}\|\right)\left\|\mathbf{U}\mathbf{U}^T - \lambda\mathcal{P}_{\mathcal{T}}\mathcal{P}_{\Omega}(\mathbf{S})\right\|_F \\
&\overset{(a)}{\leq} e^{-k_0}(\left\|\mathbf{U}\mathbf{U}^T\right\|_F + \lambda\left\|\mathcal{P}_{\mathcal{T}}\mathcal{P}_{\Omega}(\mathbf{S})\right\|_F) \\
&\overset{(b)}{\leq} n^{-5}(n + \lambda\cdot n) \leq (1+\lambda)n^{-4} \overset{(c)}{\leq} \frac{1}{2n^2}\lambda.
\end{aligned}$$

Here, the inequality (a) follows Lemma 9 with $\epsilon_1 = e^{-1}$, (b) follows from our choices of λ and k_0 and the fact that $\|\mathcal{P}_{\mathcal{T}}\mathcal{P}_{\Omega}(\mathbf{S})\|_F \leq \|\mathcal{P}_{\Omega}(\mathbf{S})\|_F \leq n$, and (c) holds under the assumption $\lambda \geq \frac{1}{32\sqrt{n}}$ in the theorem.

Inequality 4: We have

$$\|\mathcal{P}_{\Gamma}(\mathbf{W})\|_{\infty} = \|\mathcal{P}_{\Gamma}(\mathbf{W}_{k_0})\|_{\infty} \leq \sum_{k=1}^{k_0}\|\mathcal{R}_{\Gamma_i}\Delta_{i-1}\|_{\infty} \leq t^{-1}\sum_{k=1}^{k_0}\|\Delta_{k-1}\|_{\infty},$$

where the first inequality follows from (5) and the triangle inequality. We proceed to obtain

$$\begin{aligned}\sum_{k=1}^{k_0}\|\Delta_{k-1}\|_\infty &\overset{(a)}{=} \sum_{k=1}^{k_0}\left\|\prod_{i=1}^{k-1}(\mathcal{P}_\mathcal{T}-\mathcal{P}_\mathcal{T}\mathcal{R}_{\Gamma_i}\mathcal{P}_\mathcal{T})(\mathbf{U}\mathbf{U}^T-\lambda\mathcal{P}_\mathcal{T}\mathcal{P}_\Omega(\mathbf{S}))\right\|_\infty \\ &\overset{(b)}{\leq} \sum_{k=1}^{k_0}\left(\frac{1}{2}\right)^k\left\|\mathbf{U}\mathbf{U}^T-\lambda\mathcal{P}_\mathcal{T}\mathcal{P}_\Omega(\mathbf{S})\right\|_\infty \\ &\overset{(c)}{\leq} \frac{1}{K_{\min}}+\lambda\sqrt{\frac{p_0 n\log n}{K_{\min}^2}}, \end{aligned} \tag{6}$$

where (a) follows from (4), (b) follows from Lemma 11 and (c) follows from Lemma 12. It follows that

$$\begin{aligned}\|\mathcal{P}_\Gamma(\mathbf{W})\|_\infty &\leq \frac{1}{t}\left(\frac{1}{K_{\min}}+\frac{n\log n}{K_{\min}^2}\lambda\right) \\ &\leq \frac{k_0}{p_0(1-2\tau)}\left(\frac{1}{K_{\min}}+\lambda\sqrt{\frac{p_0 n\log n}{K_{\min}^2}}\right)\leq\frac{1}{4}\lambda,\end{aligned}$$

where the last inequality holds under the assumption of Theorem 4

Inequality 2: Observe that by the triangle inequality, we have

$$\|\mathcal{P}_{\mathcal{T}^\perp}(\mathbf{W}+\lambda\mathcal{P}_\Omega(\mathbf{S}))\|\leq\lambda\|\mathcal{P}_{\mathcal{T}^\perp}(\mathcal{P}_\Omega(\mathbf{S}))\|+\|\mathcal{P}_{\mathcal{T}^\perp}(\mathbf{W}_{k_0})\|.$$

For the first term, standard results on the norm of a matrix with i.i.d. entries (e.g., see Vershynin 2010) give

$$\lambda\|\mathcal{P}_{\mathcal{T}^\perp}(\mathcal{P}_\Omega(\mathbf{S}))\|\leq\lambda\|\mathcal{P}_\Omega(\mathbf{S})\|\leq\frac{1}{32\sqrt{p_0 n}}\cdot 4\sqrt{\frac{2p_0\tau n}{1-p_0+2p_0\tau}}\leq\frac{1}{8}$$

It remains to show that the second term is bounded by $\frac{1}{8}$. To this end, we observe that

$$\begin{aligned}\|\mathcal{P}_{\mathcal{T}^\perp}(\mathbf{W}_{k_0})\| &\overset{(a)}{=} \sum_{k=1}^{k_0}\|\mathcal{P}_{\mathcal{T}^\perp}\left(\mathcal{R}_{\Gamma_k}\Delta_{k-1}-\Delta_{k-1}\right)\| \\ &\leq \sum_{k=1}^{k_0}\|\left(\mathcal{R}_{\Gamma_k}-\mathcal{I}\right)\Delta_{k-1}\| \\ &\overset{(b)}{\leq} C\sqrt{\frac{n\log n}{t}}\sum_{k=1}^{k_0}\|\Delta_{k-1}\|_\infty \\ &\overset{(c)}{\leq} C\sqrt{\frac{k_0 n\log n}{p_0(1-2\tau)}}\left(\frac{1}{K_{\min}}+\lambda\sqrt{\frac{p_0 n\log n}{K_{\min}^2}}\right)\leq\frac{1}{8},\end{aligned}$$

where in (a) we use (5) and the fact that $\Delta_k\in\mathcal{T}$, (b) follows from Lemma 10, and (c) follows from (6). This completes the proof of Theorem 4.

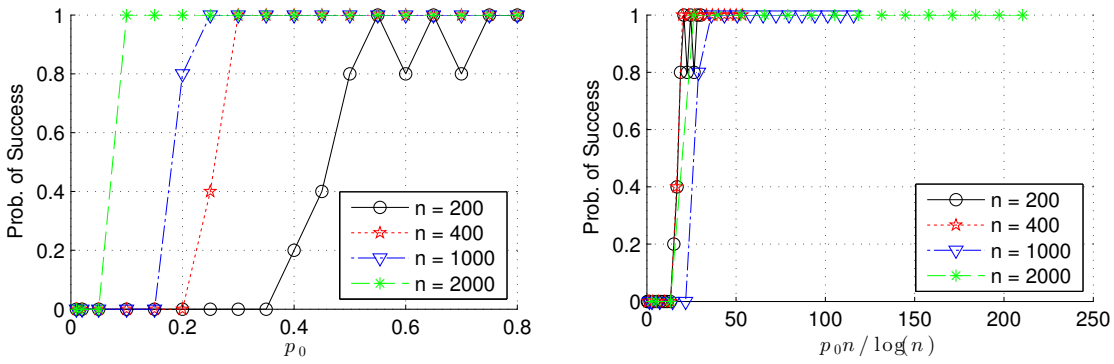


Figure 2: Simulation results verifying the performance of our algorithm as a function of the observation probability p_0 and the graph size n. The left pane shows the probability of successful recovery under different p_0 and n with fixed $\tau = 0.2$ and $K_{\min} = n/4$; each point is an average over 5 trials. After proper rescaling of the x-axis, the curves align as shown in the right pane, indicating a good match with our theoretical results.

4. Experimental Results

We explore via simulation the performance of our algorithm as a function of the values of the model parameters $(n, K_{\min}, p_0, \tau)$. We see that the performance matches well with the theory.

In the experiment, each test case is constructed by generating a graph with n nodes divided into clusters of equal size $K_{\min}$, and then placing a disagreement on each pair of node with probability τ independently. Each node pair is then observed with probability p_0. We then run Algorithm 1, where the optimization problem (1) is solved using the fast algorithm in Lin et al. (2009).. We check if the algorithm successfully outputs a solution that equals to the underlying true clusters. In the first set of experiments, we fix $\tau = 0.2$ and $K_{\min} = n/4$ and vary p_0 and n. For each (p_0, n), we repeat the experiment for 5 times and plot the probability of success in the left pane of Figure 2.

One observes that our algorithm has better performance with larger p_0 and n, and the success probability exhibits a phase transition. Theorem 4 predicts that, with τ fixed and $K_{\min} = n/4$, the transition occurs at $p_0 \propto \frac{n \log^2 n}{K_{\min}^2} \propto \frac{\log^2 n}{n}$; in particular, if we plot the success probability against the control parameter $\frac{p_0 n}{\log^2 n}$, all curves should align with each other. Indeed, this is precisely what we see in the right pane of Figure 2 where we use $\frac{p_0 n}{\log n}$ as the control parameter. This shows that Theorem 4 gives the correct scaling between p_0 and n up to an extra log factor.

In a similar fashion, we run another three sets of experiments with the following settings: (1) $n = 1000$ and $\tau = 0.2$ with varying $(p_0, K_{\min})$; (2) $K_{\min} = n/4$ and $p_0 = 0.2$ with varying (τ, n); (3) $n = 1000$ and $p_0 = 0.6$ with varying $(\tau, K_{\min})$. The results are shown in Figures 3, 4 and 5; note that each x-axis corresponds to a control parameter chosen according to the scaling predicted by Theorem 4. Again we observe that all the curves roughly align, indicating a good match with the theory. In particular, by comparing Figures 2 and 4 (or Figures 3 and 5), one verifies the quadratic tradeoff between observations and disagreements (i.e., p_0 vs. $1 - 2\tau$) as predicted by Theorem 4.

Finally, we compare the performance of our method with spectral clustering, a popular method for graph clustering. For spectral clustering, we first impute the missing entries of the adjacency

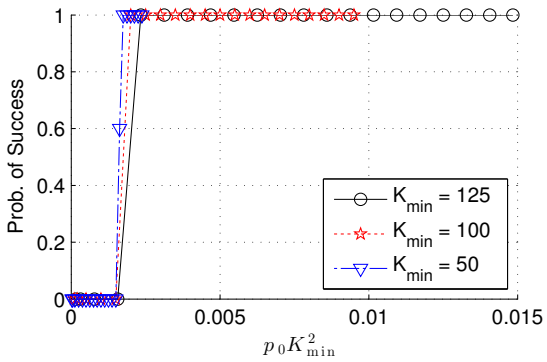


Figure 3: Simulation results verifying the performance of our algorithm as a function of the observation probability p_0 and the cluster size $K_{\min}$, with $n = 1000$ and $\tau = 0.2$ fixed.

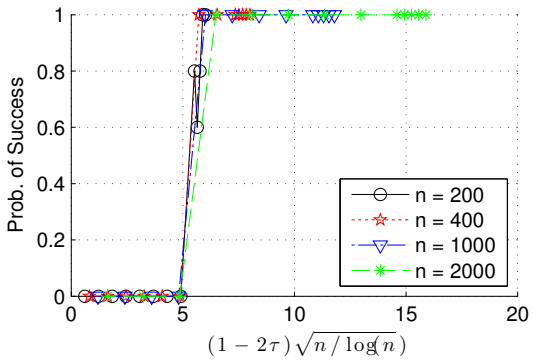


Figure 4: Simulation results verifying the performance of our algorithm as a function of the disagreement probability τ and the graph size n, with $p_0 = 0.2$ and $K_{\min} = n/4$ fixed.

matrix with either zeros or random $1/0$'s. We then compute the first k principal components of the adjacency matrix, and run k-means clustering on the principal components (von Luxburg, 2007); here we set k equal to the number of clusters. The adjacency matrix is generated in the same fashion as before using the parameters $n = 2000$, $K_{\min} = 200$ and $\tau = 0.1$. We vary the observation probability p_0 and plot the success probability in Figure 6. It can be observed that our method outperforms spectral clustering with both imputation schemes; in particular, it requires fewer observations.

5. Conclusion

We proposed a convex optimization formulation, based on a reduction to decomposing low-rank and sparse matrices, to address the problem of clustering partially observed graphs. We showed that under a wide range of parameters of the planted partition model with partial observations, our method is guaranteed to find the optimal (disagreement-minimizing) clustering. In particular, our method succeeds under higher levels of noise and/or missing observations than existing methods in

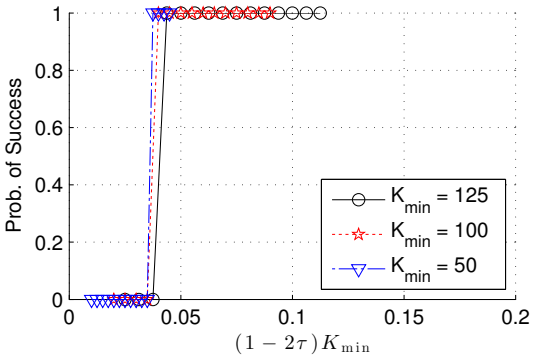


Figure 5: Simulation results verifying the performance of our algorithm as a function of the disagreement probability τ and the cluster size $K_{\min}$, with $n = 1000$ and $p_0 = 0.6$ fixed.

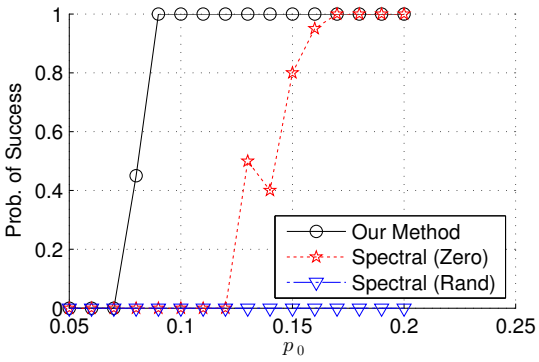


Figure 6: Comparison of our method and spectral clustering under different observation probabilities p_0, with $n = 2000$, $K_{\min} = 200$ and $\tau = 0.1$. For spectral clustering, two imputation schemes are considered: (a) Spectral (Zero), where the missing entries are imputed with zeros, and (b) Spectral (Rand), where they are imputed with $0/1$ random variables with symmetric probabilities. The result shows that our method recovers the underlying clusters with fewer observations.

this setting. The effectiveness of the proposed method and the scaling of the theoretical results are validated by simulation studies.

This work is motivated by graph clustering applications where obtaining similarity data is expensive and it is desirable to use as few observations as possible. As such, potential directions for future work include considering different sampling schemes such as active sampling, as well as dealing with sparse graphs with very few connections.

Acknowledgments

S. Sanghavi would like to acknowledge DTRA grant HDTRA1-13-1-0024 and NSF grants 1302435, 1320175 and 0954059. H. Xu is partially supported by the Ministry of Education of Singapore

through AcRF Tier Two grant R-265-000-443-112. The authors are grateful to the anonymous reviewers for their thorough reviews of this work and valuable suggestions on improving the manuscript.

Appendix A. Technical Lemmas

In this section, we provide several auxiliary lemmas required in the proof of Theorem 4. We will make use of the non-commutative Bernstein inequality. The following version is given by Tropp (2012).

Lemma 7 (Tropp, 2012) *Consider a finite sequence $\{\mathbf{M}_i\}$ of independent, random $n \times n$ matrices that satisfy the assumption $\mathbb{E}\mathbf{M}_i = 0$ and $\|\mathbf{M}_i\| \leq D$ almost surely. Let*

$$\sigma^2 = \max\left\{ \left\| \sum_i \mathbb{E}\left[\mathbf{M}_i \mathbf{M}_i^\top\right] \right\|, \left\| \sum_i \mathbb{E}\left[\mathbf{M}_i^\top \mathbf{M}_i\right] \right\| \right\}.$$

Then for all $\theta > 0$ we have

$$\begin{aligned} \mathbb{P}\left[\left\|\sum \mathbf{M}_i\right\| \geq \theta\right] &\leq 2n \exp\left(-\frac{\theta^2}{2\sigma^2 + 2D\theta/3}\right). \\ &\leq \begin{cases} 2n \exp\left(-\frac{3\theta^2}{8\sigma^2}\right), & \text{for } \theta \leq \frac{\sigma^2}{D}; \\ 2n \exp\left(-\frac{3\theta}{8D}\right), & \text{for } \theta \geq \frac{\sigma^2}{D}. \end{cases} \end{aligned} \tag{7}$$

Remark 8 *When $n = 1$, this becomes the standard two-sided Bernstein inequality.*

We will also make use of the following estimate, which follows from the structure of $\mathbf{U}$.

$$\left\|\mathcal{P}_{\mathcal{T}}(e_i e_j^\top)\right\|_F^2 = \left\|\mathbf{U}\mathbf{U}^T e_i\right\|^2 + \left\|\mathbf{U}\mathbf{U}^T e_j\right\|^2 - \left\|\mathbf{U}\mathbf{U}^T e_i\right\|^2 \left\|\mathbf{U}\mathbf{U}^T e_j\right\|^2 \leq \frac{2n}{K_{\min}^2}, \quad \forall 1 \leq i, j \leq n.$$

The first auxiliary lemma controls the operator norm of certain random operators. A similar result was given in Candès et al. (2011, Theorem 4.1). Our proof is different from theirs.

Lemma 9 *Suppose Ω_0 is a set of entries obeying $\Omega_0 \sim Ber_1(\rho)$. Consider the operator $P_{\mathcal{T}} - P_{\mathcal{T}}\mathcal{R}_{\Omega_0}P_{\mathcal{T}}$. For some constant $C_0 > 0$, we have*

$$\|\mathcal{P}_{\mathcal{T}} - \mathcal{P}_{\mathcal{T}}\mathcal{R}_{\Omega_0}\mathcal{P}_{\mathcal{T}}\| < \epsilon_1$$

with high probability provided that $\rho \geq C_0 \frac{n \log n}{\epsilon_1^2 K_{\min}^2}$ and $\epsilon_1 \leq 1$.

Proof For each (i, j), define the indicator random variable $\delta_{ij} = \mathbf{1}_{\{(i,j)\in\Omega_0\}}$. We observe that for any matrix $\mathbf{M} \in \mathcal{T}$,

$$\begin{aligned} \left(\mathcal{P}_{\mathcal{T}}\mathcal{R}_{\Omega_0}\mathcal{P}_{\mathcal{T}} - \mathcal{P}_{\mathcal{T}}\right)\mathbf{M} &= \sum_{1\leq i<j\leq n} \mathcal{S}_{ij}(\mathbf{M}) \\ &\triangleq \sum_{1\leq i<j\leq n} \left(\rho^{-1}\delta_{ij} - 1\right) \left\langle \mathcal{P}_{\mathcal{T}}(e_i e_j^\top),\ \mathbf{M} \right\rangle \mathcal{P}_{\mathcal{T}}(e_i e_j^\top + e_j e_i^\top). \end{aligned}$$

Here $\mathcal{S}_{ij} : \mathbb{R}^{n\times n} \mapsto \mathbb{R}^{n\times n}$ is a linear self-adjoint operator with $\mathbb{E}[\mathcal{S}_{ij}] = 0$. Using the fact that $\mathcal{P}_{\mathcal{T}}(e_i e_j^\top) = \left(\mathcal{P}_{\mathcal{T}}(e_j e_i^\top)\right)^\top$ and $\mathbf{M}$ is symmetric, we obtain the bounds

$$\begin{aligned}\|\mathcal{S}_{ij}\| &\le \rho^{-1}\left\|\mathcal{P}_{\mathcal{T}}(e_i e_j^\top)\right\|_F \left\|\mathcal{P}_{\mathcal{T}}(e_i e_j^\top + e_j e_i^\top)\right\|_F \\ &\le \rho^{-1}\cdot 2\left\|\mathcal{P}_{\mathcal{T}}(e_i e_j^\top)\right\|_F^2 \le \frac{4n}{K_{\min}^2\rho},\end{aligned}$$

and

$$\begin{aligned}&\left\|\mathbb{E}\left[\sum_{1\le i<j\le n}\mathcal{S}_{ij}^2(\mathbf{M})\right]\right\|_F \\ &= \left\|\sum_{1\le i<j\le n}\mathbb{E}\left[(\rho^{-1}\delta_{ij}^{(k)}-1)^2\right]\left\langle\mathcal{P}_{\mathcal{T}}(e_i e_j^\top),\,\mathbf{M}\right\rangle\left\langle\mathcal{P}_{\mathcal{T}}(e_i e_j^\top + e_j e_i^\top),\,e_i e_j^\top\right\rangle\mathcal{P}_{\mathcal{T}}(e_i e_j^\top + e_j e_i^\top)\right\|_F \\ &= \left(\rho^{-1}-1\right)\left\|\sum_{1\le i<j\le n} 2\left\|\mathcal{P}_{\mathcal{T}}(e_i e_j^\top)\right\|_F^2 m_{i,j}\mathcal{P}_{\mathcal{T}}(e_i e_j^\top + e_j e_i^\top)\right\|_F \\ &\le \left(\rho^{-1}-1\right)\left\|\sum_{1\le i<j\le n} 2\left\|\mathcal{P}_{\mathcal{T}}(e_i e_j^\top)\right\|_F^2 m_{i,j}(e_i e_j^\top + e_j e_i^\top)\right\|_F \\ &\le \left(\rho^{-1}-1\right)\frac{4n}{K_{\min}^2}\left\|\sum_{1\le i<j\le n} m_{i,j}(e_i e_j^\top + e_j e_i^\top)\right\|_F \\ &= \left(\rho^{-1}-1\right)\frac{4n}{K_{\min}^2}\|\mathbf{M}\|_F,\end{aligned}$$

which means $\left\|\mathbb{E}\left[\sum_{1\le i<j\le n}\mathcal{S}_{ij}^2\right]\right\| \le \frac{4n}{K_{\min}^2\rho}$. Applying the first inequality in the Bernstein inequality (7) gives

$$\mathbb{P}\left[\left\|\sum_{1\le i<j\le n}\mathcal{S}_{ij}\right\| \ge \epsilon_1\right] \le 2n^{2-2\beta}$$

provided $\rho \ge \frac{64\beta n\log n}{3K_{\min}^2\epsilon_1^2}$ and $\epsilon_1 < 1$. ■

The next lemma bounds the spectral norm of certain symmetric random matrices. A related result for non-symmetric matrices appeared in Candès and Recht (2009, Theorem 6.3).

Lemma 10 *Suppose Ω_0 is a set of entries obeying $\Omega_0 \sim Ber_1(\rho)$, and $\mathbf{M}$ is a fixed $n\times n$ symmetric matrix. Then for some constant $C_0 > 0$, we have*

$$\|(\mathcal{I}-\mathcal{R}_{\Omega_0})\mathbf{M}\| < \sqrt{C_0\frac{n\log n}{\rho}}\|\mathbf{M}\|_\infty,$$

with high probability provided that $\rho \ge C_0\frac{\log n}{n}$.

Proof Define δ_{ij} as before. Notice that

$$\mathcal{R}_{\Omega_0}(\mathbf{M}) - \mathbf{M} = \sum_{i<j} S_{ij} \triangleq \sum_{i<j} (\rho^{-1}\delta_{ij} - 1) m_{i,j} \left(e_i e_j^\top + e_j e_i^\top \right).$$

Here the symmetric matrix $S_{ij} \in \mathbb{R}^{n\times n}$ satisfies $\mathbb{E}\left[S_{ij}\right] = 0$, $\|S_{ij}\| \le 2\rho^{-1} \|\mathbf{M}\|_\infty$ and the bound

$$\begin{aligned}
\left\| \mathbb{E}\left[\sum_{i<j} S_{ij}^2 \right] \right\| &= \left(\rho^{-1} - 1\right) \left\| \sum_{i<j} m_{i,j}^2 \left(e_i e_i^\top + e_j e_j^\top \right) \right\| \\
&\le \left(\rho^{-1} - 1\right) \left\| \text{diag}\left(\sum_j m_{1,j}^2, \ldots, \sum_j m_{n,j}^2 \right) \right\| \\
&\le \left(\rho^{-1} - 1\right) n \|\mathbf{M}\|_\infty^2 \le 2\rho^{-1} n \|\mathbf{M}\|_\infty^2 .
\end{aligned}$$

When $\rho \ge \frac{16\beta \log n}{3n}$, we apply the first inequality in the Bernstein inequality (7) to obtain

$$\mathbb{P}\left[\left\| \sum_{i<j} S_{ij} \right\| \ge \sqrt{\frac{16\beta n \log n}{3\rho}} \|\mathbf{M}\|_\infty \right] \le 2n \exp\left(- \frac{3 \cdot \frac{16\beta n \log n}{3\rho} \|\mathbf{M}\|_\infty^2}{8 \cdot \frac{2n}{\rho} \|\mathbf{M}\|_\infty^2} \right) \le 2n^{1-\beta}.$$

The conclusion follows by choosing a sufficiently large constant β. ■

The third lemma bounds the infinity norm of certain random symmetric matrices. A related result is given in Candès et al. (2011, Lemma 3.1).

Lemma 11 *Suppose Ω_0 is a set of entries obeying $\Omega_0 \sim Ber_1(\rho)$, and $\mathbf{M} \in \mathcal{T}$ is a fixed symmetric $n \times n$ matrix. Then for some constant $C_0 > 0$, we have*

$$\|(\mathcal{P}_\mathcal{T} - \mathcal{P}_\mathcal{T}\mathcal{R}_{\Omega_0}\mathcal{P}_\mathcal{T})\mathbf{M}\|_\infty < \epsilon_3 \|\mathbf{M}\|_\infty,$$

with high probability provided that $\rho \ge C_0 \frac{n \log n}{\epsilon_3^2 K_{\min}^2}$ and $\epsilon_3 \le 1$.

Proof Define δ_{ij} as before. Fix an entry index (a, b). Notice that

$$(\mathcal{P}_\mathcal{T}\mathcal{R}_{\Omega_0}\mathcal{P}_\mathcal{T}\mathbf{M} - \mathcal{P}_\mathcal{T}\mathbf{M})_{a,b} = \sum_{i<j} \xi_{ij} \triangleq \sum_{i<j} \left\langle (\rho^{-1}\delta_{ij}^{(k)} - 1) m_{i,j} \mathcal{P}_\mathcal{T}\left(e_i e_j^\top + e_j e_i^\top \right),\ e_a e_b^\top \right\rangle.$$

The random variable ξ_{ij} satisfies $\mathbb{E}\left[\xi_{ij}\right] = 0$ and obeys the bounds

$$|\xi_{ij}| \le 2p^{-1} \left\| \mathcal{P}_\mathcal{T}(e_i e_j^\top) \right\|_F \left\| \mathcal{P}_\mathcal{T}(e_a e_b^\top) \right\|_F |m_{i,j}| \le \frac{4n}{K_{\min}^2 \rho} \|\mathbf{M}\|_\infty$$

and

$$\begin{aligned}\left|\mathbb{E}\left[\sum_{i<j}\xi_{ij}^2\right]\right| &= \left|\sum_{i<j}\mathbb{E}\left[(\rho^{-1}\delta_{ij}^{(k)}-1)^2\right]m_{i,j}^2\left\langle\mathcal{P}_{\mathcal{T}}\left(e_ie_j^\top+e_je_i^\top\right),\ e_ae_b^\top\right\rangle^2\right| \\ &\le \left(\rho^{-1}-1\right)\|\mathbf{M}\|_\infty^2\sum_{i<j}\left\langle e_ie_j^\top+e_je_i^\top,\ \mathcal{P}_{\mathcal{T}}(e_ae_b^\top)\right\rangle^2 \\ &\le 2\left(\rho^{-1}-1\right)\|\mathbf{M}\|_\infty^2\left\|\mathcal{P}_{\mathcal{T}}(e_ae_b^\top)\right\|_F^2 \\ &\le 2\left(\rho^{-1}-1\right)\frac{2n}{K_{\min}^2}\|\mathbf{M}\|_\infty^2 \\ &\le \frac{4n}{K_{\min}^2\rho}\|\mathbf{M}\|_\infty^2.\end{aligned}$$

When $\rho \ge \frac{64\beta n\log n}{3K_{\min}^2\epsilon_3^2}$ and $\epsilon_3 \le 1$, we apply the first inequality in the Bernstein inequality (7) with $n=1$ to obtain

$$\mathbb{P}\left[\left|\left(\mathcal{P}_{\mathcal{T}}\mathcal{R}_{\Omega_0}\mathcal{P}_{\mathcal{T}}\mathbf{M}-\mathcal{P}_{\mathcal{T}}\mathbf{M}\right)_{a,b}\right| \ge \epsilon_3\|\mathbf{M}\|_\infty\right] \le 2\exp\left(-\frac{3\epsilon_3^2\|\mathbf{M}\|_\infty^2}{8\frac{4n}{K_{\min}^2\rho}\|\mathbf{M}\|_\infty^2}\right) \le 2n^{-2\beta}.$$

Applying the union bound then yields

$$\mathbb{P}\left[\left\|\mathcal{P}_{\mathcal{T}}\mathcal{R}_{\Omega_0}\mathcal{P}_{\mathcal{T}}\mathbf{M}-\mathcal{P}_{\mathcal{T}}\mathbf{M}\right\|_\infty \ge \epsilon_3\|\mathbf{M}\|_\infty\right] \le 2n^{2-2\beta}.$$

■

The last lemma bounds the matrix infinity norm of $\mathcal{P}_{\mathcal{T}}\mathcal{P}_\Omega(\mathbf{S})$ for a ±1 random matrix $\mathbf{S}$.

Lemma 12 *Suppose $\Omega \sim Ber_0\left(\frac{2p_0\tau}{1-p_0+2p_0\tau}\right)$ and $\mathbf{S}\in\mathbb{R}^{n\times n}$ has i.i.d. symmetric ±1 entries. Under the assumption of Theorem 4, for some constant C_0, we have with high probability*

$$\|\mathcal{P}_{\mathcal{T}}\mathcal{P}_\Omega(\mathbf{S})\|_\infty \le C_0\sqrt{\frac{p_0n\log n}{K_{\min}^2}}.$$

Proof By triangle inequality, we have

$$\|\mathcal{P}_{\mathcal{T}}\mathcal{P}_\Omega(\mathbf{S})\|_\infty \le \left\|\mathbf{U}\mathbf{U}^T\mathcal{P}_\Omega(\mathbf{S})\right\|_\infty + \left\|\mathcal{P}_\Omega(\mathbf{S})\mathbf{U}\mathbf{U}^T\right\|_\infty + \left\|\mathbf{U}\mathbf{U}^T\mathcal{P}_\Omega(\mathbf{S})\mathbf{U}\mathbf{U}^T\right\|_\infty,$$

so it suffices to show that each of these three terms are bounded by $C\sqrt{\frac{p_0n\log n}{K_{\min}^2}}$ w.h.p. for some constant C. Under the assumption on Ω and $\mathbf{S}$ in the lemma statement, each pair of symmetric entries of $\mathcal{P}_\Omega(\mathbf{S})$ equals ±1 with probability $\rho \triangleq \frac{p_0\tau}{1-p_0+2p_0\tau}$ and 0 otherwise; notice that $\rho \le \frac{p_0}{2}$ since $\tau \le \frac{1}{2}$. Let $\left(s^{(i)}\right)^T$ be the ith row of $\mathbf{U}\mathbf{U}^T$. From the structure of $\mathbf{U}$, we know that for all i and j,

$$\left|s_j^{(i)}\right| \le \frac{1}{K_{\min}},$$

and for all i,

$$\sum_{j=1}^{n} \left(s_j^{(i)}\right)^2 \leq \frac{1}{K_{\min}}.$$

We now bound $\left\|\mathbf{U}\mathbf{U}^T\mathcal{P}_\Omega(\mathbf{S})\right\|_\infty$. For simplicity, we focus on the $(1,1)$ entry of $\mathbf{U}\mathbf{U}^T\mathcal{P}_\Omega(\mathbf{S})$ and denote this random variable as X. We may write X as $X = \sum_i s_i^{(1)} \left(\mathcal{P}_\Omega(\mathbf{S})\right)_{i,1}$, for which we have

$$\begin{aligned}
\mathbb{E}\left[s_i^{(1)}\left(\mathcal{P}_\Omega(\mathbf{S})\right)_{i,1}\right] &= 0,\\
\left|s_i^{(1)}\left(\mathcal{P}_\Omega(\mathbf{S})\right)_{i,1}\right| &\leq \left|s_i^{(1)}\right| \leq \frac{1}{K_{\min}}, \quad a.s.\\
\mathrm{Var}\left(X\right) &= \sum_{i:(i,1)\in\Omega} (s_i^{(1)})^2 \cdot 2\rho \leq \frac{p_0}{K_{\min}}.
\end{aligned}$$

Applying the standard Bernstein inequality then gives

$$\mathbb{P}\left[|X| > C\sqrt{\frac{p_0 n\log n}{K_{\min}^2}}\right] \leq 2\exp\left[-\left(C^2\frac{p_0 n\log n}{K_{\min}^2}\right)/\left(2\frac{p_0}{K_{\min}} + \frac{2C\sqrt{p_0 n\log n}}{3K_{\min}^2}\right)\right].$$

Under the assumption of Theorem 4, the right hand side above is bounded by $2n^{-12}$. It follows from the union bound that $\left\|\mathbf{U}\mathbf{U}^T\mathcal{P}_\Omega(\mathbf{S})\right\|_\infty \leq C\sqrt{\frac{p_0 n\log n}{K_{\min}^2}}$ w.h.p. Clearly, the same bound holds for $\left\|\mathcal{P}_\Omega(\mathbf{S})\mathbf{U}\mathbf{U}^T\right\|_\infty$. Finally, let K be the size of the cluster that node j is in. Observe that due to the structure of $\mathbf{U}\mathbf{U}^\top$, we have

$$\left(\mathbf{U}\mathbf{U}^T\mathcal{P}_\Omega(\mathbf{S})\mathbf{U}\mathbf{U}^T\right)_{i,j} = \sum_l \left(\mathbf{U}\mathbf{U}^\top\mathcal{P}_\Omega(\mathbf{S})\right)_{i,l}\left(\mathbf{U}\mathbf{U}^\top\right)_{l,j} \leq \frac{1}{K}\cdot K\cdot\left\|\mathbf{U}\mathbf{U}^\top\mathcal{P}_\Omega(\mathbf{S})\right\|_\infty,$$

which implies $\left\|\mathbf{U}\mathbf{U}^T\mathcal{P}_\Omega(\mathbf{S})\mathbf{U}\mathbf{U}^T\right\|_\infty \leq \left\|\mathbf{U}\mathbf{U}^\top\mathcal{P}_\Omega(\mathbf{S})\right\|_\infty$. This completes the proof of the lemma. ■

Appendix B. Proof of Theorem 5

We use a standard information theoretical argument, which improves upon a related proof by Chaudhuri et al. (2012). Let K be the size of the clusters (which are assumed to have equal size). For simplicity we assume n/K is an integer. Let $\mathcal{F}$ be the set of all possible partition of n nodes into n/K clusters of equal size K. Using Stirling's approximation, we have

$$M \triangleq |\mathcal{F}| = \frac{1}{(n/K)!}\binom{n}{K}\binom{n-K}{K}\cdots\binom{K}{K} \geq \left(\frac{n}{3K}\right)^{n(1-\frac{1}{K})} \geq c_1^{\frac{1}{2}n},$$

which holds for $K = \Theta(n)$.

Suppose the clustering $\mathbf{Y}$ is chosen uniformly at random from $\mathcal{F}$, and the graph $\mathbf{A}$ is generated from $\mathbf{Y}$ according to the planted partition model with partial observations, where we use $a_{ij} =?$ for

unobserved pairs. We use $\mathbb{P}_{\mathbf{A}|\mathbf{Y}}$ to denote the distribution of $\mathbf{A}$ given $\mathbf{Y}$. Let $\hat{\mathbf{Y}}$ be any measurable function of the observation $\mathbf{A}$. A standard application of Fano's inequality and the convexity of the mutual information (Yang and Barron, 1999) gives

$$\sup_{Y \in \mathcal{F}} \mathbb{P}\left[\hat{\mathbf{Y}} \neq \mathbf{Y} | \mathbf{Y}\right] \geq 1 - \frac{M^{-2} \sum_{\mathbf{Y}^{(1)}, \mathbf{Y}^{(2)} \in \mathcal{F}} D\left(\mathbb{P}_{\mathbf{A}|\mathbf{Y}^{(1)}} \| \mathbb{P}_{\mathbf{A}|\mathbf{Y}^{(2)}}\right) + \log 2}{\log M}, \tag{8}$$

where $D(\cdot\|\cdot)$ denotes the KL-divergence. We now upper bound this divergence. Given $\mathbf{Y}^{(l)}$, $l = 1, 2$, the $a_{i,j}$'s are independent of each other, so we have

$$D\left(\mathbb{P}_{\mathbf{A}|\mathbf{Y}^{(1)}} \| \mathbb{P}_{\mathbf{A}|\mathbf{Y}^{(2)}}\right) = \sum_{i,j} D\left(\mathbb{P}_{a_{i,j}|\mathbf{Y}^{(1)}} \| \mathbb{P}_{a_{i,j}|\mathbf{Y}^{(2)}}\right).$$

For each pair (i, j), the KL-divergence is zero if $y^{(1)}_{i,j} = y^{(2)}_{i,j}$, and otherwise satisfies

$$\begin{aligned} D\left(\mathbb{P}_{a_{i,j}|\mathbf{Y}^{(1)}} \| \mathbb{P}_{a_{i,j}|\mathbf{Y}^{(2)}}\right) &\leq p_0(1-\tau) \log \frac{p_0(1-\tau)}{p_0\tau} + p_0\tau \log \frac{p_0\tau}{p_0(1-\tau)} + (1-p_0) \log \frac{1-p_0}{1-p_0} \\ &= p_0(1-2\tau) \log \frac{1-\tau}{\tau} \\ &\leq p_0(1-2\tau) \left(\frac{1-\tau}{\tau} - 1\right) \\ &\leq c_2 p_0 (1-2\tau)^2, \end{aligned}$$

where $c_2 > 0$ is a universal constant and the last inequality holds under the assumption $\tau > 1/100$. Let N be the number of pairs (i, j) such that $y^{(1)}_{i,j} \neq y^{(2)}_{i,j}$. When $K = \Theta(n)$, we have

$$N \leq |\{(i,j) : y^{(1)}_{i,j} = 1\} \cup \{(i,j) : y^{(2)}_{i,j} = 1\}| \leq n^2.$$

It follows that $D\left(\mathbb{P}_{\mathbf{A}|\mathbf{Y}^{(1)}} \| \mathbb{P}_{\mathbf{A}|\mathbf{Y}^{(2)}}\right) \leq N \cdot c_2 p_0 (1-2\tau)^2 \leq c_2 n^2 p_0 (1-2\tau)^2$. Combining pieces, for the left hand side of (8) to be less than $1/4$, we must have $p_0(1-2\tau)^2 \geq C\frac{1}{n}$.

References

N. Alon, M. Krivelevich, and B. Sudakov. Finding a large hidden clique in a random graph. In *Proceedings of the 9th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 457–466, 1998.

B. Ames and S. Vavasis. Nuclear norm minimization for the planted clique and biclique problems. *Mathematical Programming*, 129(1):69–89, 2011.

M. F. Balcan and P. Gupta. Robust hierarchical clustering. In *Proceedings of the Conference on Learning Theory (COLT)*, 2010.

N. Bansal, A. Blum, and S. Chawla. Correlation clustering. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, 2002.

B. Bollobás and A. D. Scott. Max cut for random graphs with a planted partition. *Combinatorics, Probability and Computing*, 13(4-5):451–474, 2004.

R. B. Boppana. Eigenvalues and graph bisection: An average-case analysis. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, pages 280–285, 1987.

E. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

E. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):11, 2011.

T. Carson and R. Impagliazzo. Hill-climbing finds random planted bisections. In *Proceedings of the 12th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 903–909, 2001.

V. Chandrasekaran, S. Sanghavi, P. Parrilo, and A. Willsky. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21(2):572–596, 2011.

M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 524–533, 2003.

K. Chaudhuri, F. Chung, and A. Tsiatas. Spectral clustering of graphs with general degrees in the extended planted partition model. *Journal of Machine Learning Research*, 2012:1–23, 2012.

Y. Chen, A. Jalali, S. Sanghavi, and C. Caramanis. Low-rank matrix recovery from errors and erasures. *IEEE Transactions on Information Theory*, 59(7):4324–4337, 2013.

A. Condon and R. M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18(2):116–140, 2001.

A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84(6):066106, 2011.

E. D. Demaine and N. Immorlica. Correlation clustering with partial information. *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, pages 71–80, 2003.

E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2):172–187, 2006.

D. Emmanuel and A. Fiat. Correlation clustering minimizing disagreements on arbitrary weighted graphs. In *Proceedings of the 11th Annual European Symposium on Algorithms*, pages 208–220, 2003.

B. Eriksson, G. Dasarathy, A. Singh, and R. Nowak. Active clustering: Robust and efficient hierarchical clustering using adaptively selected similarities. *Arxiv preprint arXiv:1102.3887*, 2011.

M. Ester, H. Kriegel, and X. Xu. A database interface for clustering in large spatial databases. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 94–99, 1995.

M. Fazel. *Matrix Rank Minimization with Applications*. PhD thesis, Stanford University, 2002.

U. Feige and J. Kilian. Heuristics for semirandom graph problems. *Journal of Computer and System Sciences*, 63(4):639–671, 2001.

J. Giesen and D. Mitsche. Reconstructing many partitions using spectral techniques. In *Fundamentals of Computation Theory*, pages 433–444. Springer, 2005.

D. Gross. Recovering low-rank matrices from few coefficients in any basis. *IEEE Transactions on Information Theory*, 57(3):1548–1566, 2011.

P. W. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: Some first steps. *Social networks*, 5(2):109–137, 1983.

D. Hsu, S. M. Kakade, and T. Zhang. Robust matrix decomposition with sparse corruptions. *IEEE Transactions on Information Theory*, 57(11):7221–7234, 2011.

B. Hunter and T. Strohmer. Spectral clustering with compressed, incomplete and inaccurate measurements. Available at `https://www.math.ucdavis.edu/~strohmer/papers/2010/SpectralClustering.pdf`, 2010.

M. Jerrum and G. B. Sorkin. The metropolis algorithm for graph bisection. *Discrete Applied Mathematics*, 82(1-3):155–175, 1998.

B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, 1970.

A. Krishnamurthy, S. Balakrishnan, M. Xu, and A. Singh. Efficient active algorithms for hierarchical clustering. In *Proceedings of the 29th International Conference on Machine Learning*, pages 887–894, 2012.

X. Li. Compressed sensing and matrix completion with constant proportion of corruptions. *Constructive Approximation*, 37(1):73–99, 2013.

Z. Lin, M. Chen, L. Wu, and Y. Ma. The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices. *UIUC Technical Report UILU-ENG-09-2215*, 2009.

C. Mathieu and W. Schudy. Correlation clustering with noisy input. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 712–728, 2010.

F. McSherry. Spectral partitioning of random graphs. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 529–537, 2001.

N. Mishra, I. Stanton R. Schreiber, and R. E. Tarjan. Clustering social networks. In *Algorithms and Models for Web-Graph*, pages 56–67. Springer, 2007.

S. Oymak and B. Hassibi. Finding dense clusters via low rank + sparse decomposition. Available on arXiv:1104.5186v1, 2011.

K. Rohe, S. Chatterjee, and B. Yu. Spectral clustering and the high-dimensional stochastic blockmodel. *The Annals of Statistics*, 39(4):1878–1915, 2011.

O. Shamir and N. Tishby. Spectral clustering on a budget. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 661–669, 2011.

R. Shamir and D. Tsur. Improved algorithms for the random cluster graph model. *Random Structures & Algorithms*, 31(4):418–449, 2007.

C. Swamy. Correlation clustering: maximizing agreements via semidefinite programming. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2004.

J. A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2012.

R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. *Arxiv preprint arxiv:1011.3027*, 2010.

K. Voevodski, M. F. Balcan, H. Roglin, S. H. Teng, and Y. Xia. Efficient clustering with limited distance information. *arXiv preprint arXiv:1009.5168*, 2010.

U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

Yahoo!-Inc. Graph partitioning. Available at http://research.yahoo.com/project/2368, 2009.

Y. Yang and A. Barron. Information-theoretic determination of minimax rates of convergence. *The Annals of Statistics*, 27(5):1564–1599, 1999.

Journal of Machine Learning Research 1 (2014) 2239-2312 Submitted 7/13; Revised 11/13; Published 10/00

A Tensor Approach to Learning Mixed Membership Community Models

Animashree Anandkumar A.ANANDKUMAR@UCI.EDU
Department of Electrical Engineering & Computer Science,
University of California Irvine,
Irvine, CA 92697, USA

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

Daniel Hsu DJHSU@CS.COLUMBIA.EDU
Department of Computer Science,
Columbia University
116th Street and Broadway,
New York, NY 10027, USA

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

Editor: Nathan Srebro

Abstract

Community detection is the task of detecting hidden communities from observed interactions. Guaranteed community detection has so far been mostly limited to models with non-overlapping communities such as the stochastic block model. In this paper, we remove this restriction, and provide guaranteed community detection for a family of probabilistic network models with overlapping communities, termed as the mixed membership Dirichlet model, first introduced by Airoldi et al. (2008). This model allows for nodes to have fractional memberships in multiple communities and assumes that the community memberships are drawn from a Dirichlet distribution. Moreover, it contains the stochastic block model as a special case. We propose a unified approach to learning these models via a tensor spectral decomposition method. Our estimator is based on low-order moment tensor of the observed network, consisting of 3-star counts. Our learning method is fast and is based on simple linear algebraic operations, e.g., singular value decomposition and tensor power iterations. We provide guaranteed recovery of community memberships and model parameters and present a careful finite sample analysis of our learning method. As an important special case, our results match the best known scaling requirements for the (homogeneous) stochastic block model.

Keywords: community detection, spectral methods, tensor methods, moment-based estimation, mixed membership models

1. Introduction

Studying communities forms an integral part of social network analysis. A community generally refers to a group of individuals with shared interests (e.g., music, sports), or relationships (e.g., friends, co-workers). Community formation in social networks has been studied by many sociologists, (e.g., Moreno, 1934; Lazarsfeld et al., 1954; McPherson et al., 2001; Currarini et al., 2009), starting with the seminal work of Moreno (1934). They posit various factors such as *homophily*[1] among the individuals to be responsible for community formation. Various probabilistic and non-probabilistic network models attempt to explain community formation. In addition, they also attempt to quantify interactions and the extent of overlap between different communities, relative sizes among the communities, and various other network properties. Studying such community models are also of interest in other domains, e.g., in biological networks.

While there exists a vast literature on community models, learning these models is typically challenging, and various heuristics such as Markov Chain Monte Carlo (MCMC) or variational expectation maximization (EM) are employed in practice. Such heuristics tend to scale poorly for large networks. On the other hand, community models with guaranteed learning methods tend to be restrictive. A popular class of probabilistic models, termed as *stochastic blockmodels*, have been widely studied and enjoy strong theoretical learning guarantees, (e.g., White et al., 1976; Holland et al., 1983; Fienberg et al., 1985; Wang and Wong, 1987; Snijders and Nowicki, 1997; McSherry, 2001). On the other hand, they posit that an individual belongs to a single community, which does not hold in most real settings (Palla et al., 2005).

In this paper, we consider a class of mixed membership community models, originally introduced by Airoldi et al. (2008), and recently employed by Xing et al. (2010) and Gopalan et al. (2012). The model has been shown to be effective in many real-world settings, but so far, no learning approach exists with provable guarantees. In this paper, we provide a novel learning approach for learning these mixed membership models and prove that these methods succeed under a set of sufficient conditions.

The mixed membership community model of Airoldi et al. (2008) has a number of attractive properties. It retains many of the convenient properties of the stochastic block model. For instance, conditional independence of the edges is assumed, given the community memberships of the nodes in the network. At the same time, it allows for communities to overlap, and for every individual to be fractionally involved in different communities. It includes the stochastic block model as a special case (corresponding to zero overlap among the different communities). This enables us to compare our learning guarantees with existing works for stochastic block models and also study how the extent of overlap among different communities affects the learning performance.

1.1 Summary of Results

We now summarize the main contributions of this paper. We propose a novel approach for learning mixed membership community models of Airoldi et al. (2008). Our approach is a method of moments estimator and incorporates tensor spectral decomposition. We provide

1. The term *homophily* refers to the tendency that individuals belonging to the same community tend to connect more than individuals in different communities.

guarantees for our approach under a set of sufficient conditions. Finally, we compare our results to existing ones for the special case of the stochastic block model, where nodes belong to a single community.

1.1.1 Learning Mixed Membership Models

We present a tensor-based approach for learning the mixed membership stochastic block model (MMSB) proposed by Airoldi et al. (2008). In the MMSB model, the community membership vectors are drawn from the Dirichlet distribution, denoted by $\mathrm{Dir}(\alpha)$, where α is known the Dirichlet concentration vector. Employing the Dirichlet distribution results in sparse community memberships in certain regimes of α, which is realistic. The extent of overlap between different communities under the MMSB model is controlled (roughly) via a single scalar parameter, $\alpha_0 := \sum_i \alpha_i$, where $\alpha := [\alpha_i]$ is the Dirichlet concentration vector. When $\alpha_0 \to 0$, the mixed membership model degenerates to a stochastic block model and we have non-overlapping communities. When $\alpha_i < 1$ (and hence, $\alpha_0 < k$), the generated vectors tend to be sparse and we focus on this regime in this paper.

We propose a unified tensor-based learning method for the MMSB model and establish recovery guarantees under a set of sufficient conditions. These conditions are in in terms of the network size n, the number of communities k, extent of community overlaps (through α_0), and the average edge connectivity across various communities. Below, we present an overview of our guarantees for the special case of equal sized communities (each of size n/k) and homogeneous community connectivity: let p be the probability for any intra-community edge to occur, and q be the probability for any inter-community edge. Let Π be the community membership matrix, where $\Pi^{(i)}$ denotes the i^{th} row, which is the vector of membership weights of the nodes for the i^{th} community. Let P be the community connectivity matrix such that $P(i,i) = p$ and $P(i,j) = q$ for $i \neq j$.

Theorem 1 (Main Result) *For an MMSB model with network size n, number of communities k, connectivity parameters p, q and community overlap parameter α_0, when*[2]

$$n = \tilde{\Omega}(k^2(\alpha_0+1)^2), \qquad \frac{p-q}{\sqrt{p}} = \tilde{\Omega}\left(\frac{(\alpha_0+1)k}{n^{1/2}}\right), \tag{1}$$

our estimated community membership matrix $\hat{\Pi}$ and the edge connectivity matrix $\hat{P}$ satisfy with high probability (w.h.p.)

$$\frac{\varepsilon_{\pi,\ell_1}}{n} := \frac{1}{n}\max_{i\in[n]} \|\hat{\Pi}^i - \Pi^i\|_1 = \tilde{O}\left(\frac{(\alpha_0+1)^{3/2}\sqrt{p}}{(p-q)\sqrt{n}}\right) \tag{2}$$

$$\varepsilon_P := \max_{i,j\in[k]} |\hat{P}_{i,j} - P_{i,j}| = \tilde{O}\left(\frac{(\alpha_0+1)^{3/2}k\sqrt{p}}{\sqrt{n}}\right). \tag{3}$$

Further, our support estimates $\hat{S}$ satisfy w.h.p.,

$$\Pi(i,j) \geq \xi \Rightarrow \hat{S}(i,j) = 1 \quad \text{and} \quad \Pi(i,j) \leq \frac{\xi}{2} \Rightarrow \hat{S}(i,j) = 0, \quad \forall i \in [k], j \in [n], \tag{4}$$

where Π is the true community membership matrix and the threshold is chosen as $\xi = \Omega(\epsilon_P)$.

2. The notation $\tilde{\Omega}(\cdot), \tilde{O}(\cdot)$ denotes $\Omega(\cdot), O(\cdot)$ up to poly-log factors.

Remark: *Note that the scaling condition in* (1) *ensures that* $\frac{\varepsilon_{\pi,\ell_1}}{n}$ *in* (2) *is decaying, since we assume that* $\alpha_0 < k$ *(sparse regime). However, if we want the estimation error* ε_P *in* (3) *to decay, we require a slightly stronger condition with respect to* α_0 *that*

$$n = \tilde{\Omega}(k^2(\alpha_0+1)^3), \qquad \frac{p-q}{\sqrt{p}} = \tilde{\Omega}\left(\frac{(\alpha_0+1)^{1.5}k}{n^{1/2}}\right).$$

The complete details are in Section 4. We first provide some intuitions behind the sufficient conditions in (1). We require the network size n to be large enough compared to the number of communities k, and for the separation $p-q$ to be large enough, so that the learning method can distinguish the different communities. This is natural since a zero separation ($p=q$) implies that the communities are indistinguishable. Moreover, we see that the scaling requirements become more stringent as α_0 increases. This is intuitive since it is harder to learn communities with more overlap, and we quantify this scaling. For the Dirichlet distribution, it can be shown that the number of "significant" entries is roughly $O(\alpha_0)$ with high probability, and in many settings of practical interest, nodes may have significant memberships in only a few communities, and thus, α_0 is a constant (or growing slowly) in many instances.

In addition, we quantify the error bounds for estimating various parameters of the mixed membership model in (2) and (3). These errors decay under the sufficient conditions in (1). Lastly, we establish zero-error guarantees for support recovery in (4): our learning method correctly identifies (w.h.p) all the significant memberships of a node and also identifies the set of communities where a node does not have a strong presence, and we quantify the threshold ξ in Theorem 1. Further, we present the results for a general (non-homogeneous) MMSB model in Section 4.2.

1.1.2 Identifiability Result for the MMSB Model

A byproduct of our analysis yields novel identifiability results for the MMSB model based on low order graph moments. We establish that the MMSB model is identifiable, given access to third order moments in the form of counts of 3-star subgraphs, i.e., a star subgraph consisting of three leaves, for each triplet of leaves, when the community connectivity matrix P is full rank. Our learning approach involves decomposition of this third order tensor. Previous identifiability results required access to high order moments and were limited to the stochastic block model setting; see Section 1.3 for details.

1.1.3 Implications on Learning Stochastic Block Models

Our results have implications for learning stochastic block models, which is a special case of the MMSB model with $\alpha_0 \to 0$. In this case, the sufficient conditions in (1) reduce to

$$n = \tilde{\Omega}(k^2), \qquad \frac{p-q}{\sqrt{p}} = \tilde{\Omega}\left(\frac{k}{n^{1/2}}\right), \tag{5}$$

The scaling requirements in (5) match with the best known bounds[3] (up to poly-log factors) for learning uniform stochastic block models and were previously achieved by Chen et al. (2012) via convex optimization involving semi-definite programming (SDP). In contrast, we propose an iterative non-convex approach involving tensor power iterations and linear algebraic techniques, and obtain similar guarantees. For a detailed comparison of learning guarantees under various methods for learning (homogeneous) stochastic block models, see Chen et al. (2012).

Thus, we establish learning guarantees explicitly in terms of the extent of overlap among the different communities for general MMSB models. Many real-world networks involve sparse community memberships and the total number of communities is typically much larger than the extent of membership of a single individual, e.g., hobbies/interests of a person, university/company networks that a person belongs to, the set of transcription factors regulating a gene, and so on. Thus, we see that in this regime of practical interest, where $\alpha_0 = \Theta(1)$, the scaling requirements in (1) match those for the stochastic block model in (5) (up to polylog factors) without any degradation in learning performance. Thus, we establish that learning community models with sparse community memberships is akin to learning stochastic block models and we present a unified approach and analysis for learning these models.

To the best of our knowledge, this work is the first to establish polynomial time learning guarantees for probabilistic network models with overlapping communities and we provide a fast and an iterative learning approach through linear algebraic techniques and tensor power iterations. While the results of this paper are mostly limited to a theoretical analysis of the tensor method for learning overlapping communities, we note recent results which show that this method (with improvements and modifications) is very accurate in practice on real datasets from social networks, and is scalable to graphs with millions of nodes (Huang et al., 2013).

1.2 Overview of Techniques

We now describe the main techniques employed in our learning approach and in establishing the recovery guarantees.

1.2.1 Method of Moments and Subgraph Counts

We propose an efficient learning algorithm based on low order moments, viz., counts of small subgraphs. Specifically, we employ a third-order tensor which counts the number of 3-stars in the observed network. A 3-star is a star graph with three leaves (see Figure 1) and we count the occurrences of such 3-stars across different partitions. We establish that (an adjusted) 3-star count tensor has a simple relationship with the model parameters, when the network is drawn from a mixed membership model. We propose a multi-linear transformation using edge-count matrices (also termed as the process of whitening), which reduces the problem of learning mixed membership models to the *canonical polyadic (CP) decomposition* of an orthogonal symmetric tensor, for which tractable decomposition exists,

3. There are many methods which achieve the best known scaling for n in (5), but have worse scaling for the separation $p - q$. This includes variants of the spectral clustering method, (e.g., Chaudhuri et al., 2012). See Chen et al. (2012) for a detailed comparison.

as described below. Note that the decomposition of a general tensor into its rank-one components is referred to as its CP decomposition (Kolda and Bader, 2009) and is in general NP-hard (Hillar and Lim, 2013). However, the decomposition is tractable in the special case of an orthogonal symmetric tensor considered here.

1.2.2 Tensor Spectral Decomposition via Power Iterations

Our tensor decomposition method is based on the popular power iterations (see Anandkumar et al., 2012a, e.g.,). It is a simple iterative method to compute the stable eigen-pairs of a tensor. In this paper, we propose various modifications to the basic power method to strengthen the recovery guarantees under perturbations. For instance, we introduce adaptive deflation techniques (which involves subtracting out the eigen-pairs previously estimated). Moreover, we initialize the tensor power method with (whitened) neighborhood vectors from the observed network, as opposed to random initialization. In the regime, where the community overlaps are small, this leads to an improved performance. Additionally, we incorporate thresholding as a post-processing operation, which again, leads to improved guarantees for sparse community memberships, i.e., when the overlap among different communities is small. We theoretically establish that all these modifications lead to improvement in performance guarantees and we discuss comparisons with the basic power method in Section 4.4.

1.2.3 Sample Analysis

We establish that our learning approach correctly recovers the model parameters and the community memberships of all nodes under exact moments. We then carry out a careful analysis of the empirical graph moments, computed using the network observations. We establish tensor concentration bounds and also control the perturbation of the various quantities used by our learning algorithm via matrix Bernstein's inequality (Tropp, 2012, thm. 1.4) and other inequalities. We impose the scaling requirements in (1) for various concentration bounds to hold.

1.3 Related Work

There is extensive work on modeling communities and various algorithms and heuristics for discovering them. We mostly limit our focus to works with theoretical guarantees.

1.3.1 Method of Moments

The method of moments approach dates back to Pearson (1894) and has been applied for learning various community models. Here, the moments correspond to counts of various subgraphs in the network. They typically consist of aggregate quantities, e.g., number of star subgraphs, triangles etc. in the network. For instance, Bickel et al. (2011) analyze the moments of a stochastic block model and establish that the subgraph counts of certain structures, termed as "wheels" (a family of trees), are sufficient for identifiability under some natural non-degeneracy conditions. In contrast, we establish that moments up to third order (corresponding to edge and 3-star counts) are sufficient for identifiability of the stochastic block model, and also more generally, for the mixed membership Dirichlet model. We

employ subgraph count tensors, corresponding to the number of subgraphs (such as stars) over a set of labeled vertices, while the work of Bickel et al. (2011) considers only aggregate (i.e., scalar) counts. Considering tensor moments allows us to use simple subgraphs (edges and 3 stars) corresponding to low order moments, rather than more complicated graphs (e.g., wheels considered by Bickel et al. (2011)) with larger number of nodes, for learning the community model.

The method of moments is also relevant for the family of random graph models termed as *exponential random graph models* (Holland and Leinhardt, 1981; Frank and Strauss, 1986). Subgraph counts of fixed graphs such as stars and triangles serve as sufficient statistics for these models. However, parameter estimation given the subgraph counts is in general NP-hard, due to the normalization constant in the likelihood (the partition function) and the model suffers from degeneracy issues; see Rinaldo et al. (2009); Chatterjee and Diaconis (2011) for detailed discussion. In contrast, we establish in this paper that the mixed membership model is amenable to simple estimation methods through linear algebraic operations and tensor power iterations using subgraph counts of 3-stars.

1.3.2 Stochastic Block Models

Many algorithms provide learning guarantees for stochastic block models. For a detailed comparison of these methods, see the recent work by Chen et al. (2012). A popular method is based on spectral clustering (McSherry, 2001), where community memberships are inferred through projection onto the spectrum of the Laplacian matrix (or its variants). This method is fast and easy to implement (via singular value decomposition). There are many variants of this method, e.g., the work of Chaudhuri et al. (2012) employs normalized Laplacian matrix to handle degree heterogeneities. In contrast, the work of Chen et al. (2012) uses convex optimization techniques via semi-definite programming learning block models. For a detailed comparison of learning guarantees under various methods for learning stochastic block models, see Chen et al. (2012).

1.3.3 Non-probabilistic Approaches

The classical approach to community detection tries to directly exploit the properties of the graph to define communities, without assuming a probabilistic model. Girvan and Newman (2002) use betweenness to remove edges until only communities are left. However, Bickel and Chen (2009) show that these algorithms are (asymptotically) biased and that using modularity scores can lead to the discovery of an incorrect community structure, even for large graphs. Jalali et al. (2011) define community structure as the structure that satisfies the maximum number of edge constraints (whether two individuals like/dislike each other). However, these models assume that every individual belongs to a single community.

Recently, some non-probabilistic approaches have been introduced with overlapping community models by Arora et al. (2012) and Balcan et al. (2012). The analysis of Arora et al. (2012) is mostly limited to dense graphs (i.e., $\Theta(n^2)$ edges for a n node graph), while our analysis provides learning guarantees for much sparser graphs (as seen by the scaling requirements in (1)). Moreover, the running time of the method of Arora et al. (2012) is *quasipolynomial time* (i.e., $O(n^{\log n})$) for the general case, and is based on a combinatorial learning approach. In contrast, our learning approach is based on simple linear algebraic

techniques and the running time is a low-order polynomial (roughly it is $O(n^2k)$ for a n node network with k communities under a serial computation model and $O(n+k^3)$ under a parallel computation model). The work of Balcan et al. (2012) assumes endogenously formed communities, by constraining the fraction of edges within a community compared to the outside. They provide a polynomial time algorithm for finding all such "self-determined" communities and the running time is $n^{O(\log 1/\alpha)/\alpha}$, where α is the fraction of edges within a self-determined community, and this bound is improved to linear time when $\alpha > 1/2$. On the other hand, the running time of our algorithm is mostly independent of the parameters of the assumed model, (and is roughly $O(n^2k)$). Moreover, both these works are limited to homophilic models, where there are more edges within each community, than between any two different communities. However, our learning approach is not limited to this setting and also does not assume homogeneity in edge connectivity across different communities (but instead it makes probabilistic assumptions on community formation). In addition, we provide improved guarantees for homophilic models by considering additional post-processing steps in our algorithm. Recently, Abraham et al. (2012) provide an algorithm for approximating the parameters of an Euclidean log-linear model in polynomial time. However, there setting is considerably different than the one in this paper.

1.3.4 Inhomogeneous Random Graphs, Graph Limits and Weak Regularity Lemma

Inhomogeneous random graphs have been analyzed in a variety of settings (e.g., Bollobás et al., 2007; Lovász, 2009) and are generalizations of the stochastic block model. Here, the probability of an edge between any two nodes is characterized by a general function (rather than by a $k \times k$ matrix as in the stochastic block model with k blocks). Note that the mixed membership model considered in this work is a special instance of this general framework. These models arise as the limits of convergent (dense) graph sequences and for this reason, the functions are also termed as "graphons" or graph limits (Lovász, 2009). A deep result in this context is the regularity lemma and its variants. The weak regularity lemma proposed by Frieze and Kannan (1999), showed that any convergent dense graph can be approximated by a stochastic block model. Moreover, they propose an algorithm to learn such a block model based on the so-called d_2 distance. The d_2 distance between two nodes measures similarity with respect to their "two-hop" neighbors and the block model is obtained by thresholding the d_2 distances. However, the method is limited to learning block models and not overlapping communities.

1.3.5 Learning Latent Variable Models (Topic Models)

The community models considered in this paper are closely related to the probabilistic topic models (Blei, 2012), employed for text modeling and document categorization. Topic models posit the occurrence of words in a corpus of documents, through the presence of multiple latent topics in each document. Latent Dirichlet allocation (LDA) is perhaps the most popular topic model, where the topic mixtures are assumed to be drawn from the Dirichlet distribution. In each document, a topic mixture is drawn from the Dirichlet distribution, and the words are drawn in a conditional independent manner, given the topic mixture. The mixed membership community model considered in this paper can be interpreted as

a generalization of the LDA model, where a node in the community model can function both as a document and a word. For instance, in the directed community model, when the outgoing links of a node are considered, the node functions as a document, and its outgoing neighbors can be interpreted as the words occurring in that document. Similarly, when the incoming links of a node in the network are considered, the node can be interpreted as a word, and its incoming links, as documents containing that particular word. In particular, we establish that certain graph moments under the mixed membership model have similar structure as the observed word moments under the LDA model. This allows us to leverage the recent developments from Anandkumar et. al. (Anandkumar et al., 2012c,a,b) for learning topic models, based on the method of moments. These works establish guaranteed learning using second- and third-order observed moments through linear algebraic and tensor-based techniques. In particular, in this paper, we exploit the tensor power iteration method of Anandkumar et al. (2012b), and propose additional improvements to obtain stronger recovery guarantees. Moreover, the sample analysis is quite different (and more challenging) in the community setting, compared to topic models analyzed in Anandkumar et al. (2012c,a,b). We clearly spell out the similarities and differences between the community model and other latent variable models in Section 4.4.

1.3.6 Lower Bounds

The work of Feldman et al. (2012) provides lower bounds on the complexity of statistical algorithms, and shows that for cliques of size $O(n^{1/2-\delta})$, for any constant $\delta > 0$, at least $n^{\Omega(\log \log n)}$ queries are needed to find the cliques. There are works relating the hardness of finding hidden cliques and the use of higher order moment tensors for this purpose. Frieze and Kannan (2008) relate the problem of finding a hidden clique to finding the top eigenvector of the third order tensor, corresponding to the maximum spectral norm. Charles and Vempala (2009) extend the result to arbitrary r^{th}-order tensors and the cliques have to be size $\Omega(n^{1/r})$ to enable recovery from r^{th}-order moment tensors in a n node network. However, this problem (finding the top eigenvector of a tensor) is known to be NP-hard in general (Hillar and Lim, 2013). Thus, tensors are useful for finding smaller hidden cliques in network (albeit by solving a computationally hard problem). In contrast, we consider tractable tensor decomposition through reduction to orthogonal tensors (under the scaling requirements of (1)), and our learning method is a fast and an iterative approach based on tensor power iterations and linear algebraic operations. Mossel et al. (2012) provide lower bounds on the separation $p - q$, the edge connectivity between intra-community and inter-community, for identifiability of communities in stochastic block models in the sparse regime (when $p, q \sim n^{-1}$), when the number of communities is a constant $k = O(1)$. Our method achieves the lower bounds on separation of edge connectivity up to poly-log factors.

1.3.7 Likelihood-based Approaches to Learning MMSB

Another class of approaches for learning MMSB models are based on optimizing the observed likelihood. Traditional approaches such as Gibbs sampling or expectation maximization (EM) can be too expensive apply in practice for MMSB models. Variational approaches which optimize the so-called evidence lower bound (Hoffman et al., 2012; Gopalan et al., 2012), which is a lower bound on the marginal likelihood of the observed data (typically by

applying a mean-field approximation), are efficient for practical implementation. Stochastic versions of the variational approach provide even further gains in efficiency and are state-of-art practical learning methods for MMSB models (Gopalan et al., 2012). However, these methods lack theoretical guarantees; since they optimize a bound on the likelihood, they are not guaranteed to recover the underlying communities consistently. A recent work (Celisse et al., 2012) establishes consistency of maximum likelihood and variational estimators for stochastic block models, which are special cases of the MMSB model. However, it is not known if the results extend to general MMSB models. Moreover, the framework of Celisse et al. (2012) assumes a fixed number of communities and growing network size, and provide only asymptotic consistency guarantees. Thus, they do not allow for high-dimensional settings, where the parameters of the learning problem also grow as the observed dimensionality grows. In contrast, in this paper, we allow for the number of communities to grow, and provide precise constraints on the scaling bounds for consistent estimation under finite samples. It is an open problem to obtain such bounds for maximum likelihood and variational estimators. On the practical side, a recent work deploying the tensor approach proposed in this paper by Huang et al. (2013) shows that the tensor approach is more than an order of magnitude faster in recovering the communities than the variational approach, is scalable to networks with millions of nodes, and also has better accuracy in recovering the communities.

2. Community Models and Graph Moments

In the first part of section, we describe the mixed membership community model based on Dirichlet priors for the community draws by the individuals. Then in Section 2.2, we define and analyze the graph moments for these models.

2.1 Community Membership Models

We first introduce the special case of the popular stochastic block model, where each node belongs to a single community.

2.1.1 Notation

We consider networks with n nodes and let $[n] := \{1, 2, \ldots, n\}$. Let G be the $\{0, 1\}$ adjacency[4] matrix for the random network and let $G_{A,B}$ be the submatrix of G corresponding to rows $A \subseteq [n]$ and columns $B \subseteq [n]$. We consider models with k underlying (hidden) communities. For node i, let $\pi_i \in \mathbb{R}^k$ denote its *community membership vector*, i.e., the vector is supported on the communities to which the node belongs. In the special case of the popular stochastic block model described below, π_i is a basis coordinate vector, while the more general mixed membership model relaxes this assumption and a node can be in multiple communities with fractional memberships. Define $\Pi := [\pi_1|\pi_2|\cdots|\pi_n] \in \mathbb{R}^{k\times n}$. and let $\Pi_A := [\pi_i : i \in A] \in \mathbb{R}^{k\times|A|}$ denote the set of column vectors restricted to $A \subseteq [n]$. For a matrix M, let $(M)_i$ and $(M)^i$ denote its i^{th} column and row respectively. For a matrix M with singular value decomposition (SVD) $M = UDV^\top$, let $(M)_{k-svd} := U\tilde{D}V^\top$ denote the k-rank SVD of M, where $\tilde{D}$ is limited to top-k singular values of M. Let $M^\dagger$ denote

4. Our analysis can easily be extended to weighted adjacency matrices with bounded entries.

the MoorePenrose pseudo-inverse of M. Let $\mathbb{I}(\cdot)$ be the indicator function. Let $\mathrm{Diag}(v)$ denote a diagonal matrix with diagonal entries given by a vector v. We use the term high probability to mean with probability $1 - n^{-c}$ for any constant $c > 0$.

2.1.2 Stochastic Block Model (special case)

In this model, each individual is independently assigned to a single community, chosen at random: each node i chooses community j independently with probability $\widehat{\alpha}_j$, for $i \in [n], j \in [k]$, and we assign $\pi_i = e_j$ in this case, where $e_j \in \{0,1\}^k$ is the j^{th} coordinate basis vector. Given the community assignments Π, every directed[5] edge in the network is independently drawn: if node u is in community i and node v is in community j (and $u \neq v$), then the probability of having the edge (u, v) in the network is $P_{i,j}$. Here, $P \in [0,1]^{k \times k}$ and we refer to it as the *community connectivity matrix*. This implies that given the community membership vectors π_u and π_v, the probability of an edge from u to v is $\pi_u^\top P \pi_v$ (since when $\pi_u = e_i$ and $\pi_v = e_j$, we have $\pi_u^\top P \pi_v = P_{i,j}$.). The stochastic model has been extensively studied and can be learnt efficiently through various methods, e.g., spectral clustering (McSherry, 2001), convex optimization (Chen et al., 2012). and so on. Many of these methods rely on conditional independence assumptions of the edges in the block model for guaranteed learning.

2.1.3 Mixed Membership Model

We now consider the extension of the stochastic block model which allows for an individual to belong to multiple communities and yet preserves some of the convenient independence assumptions of the block model. In this model, the community membership vector π_u at node u is a probability vector, i.e., $\sum_{i \in [k]} \pi_u(i) = 1$, for all $u \in [n]$. Given the community membership vectors, the generation of the edges is identical to the block model: given vectors π_u and π_v, the probability of an edge from u to v is $\pi_u^\top P \pi_v$, and the edges are independently drawn. This formulation allows for the nodes to be in multiple communities, and at the same time, preserves the conditional independence of the edges, given the community memberships of the nodes.

2.1.4 Dirichlet Prior for Community Membership

The only aspect left to be specified for the mixed membership model is the distribution from which the community membership vectors Π are drawn. We consider the popular setting of Airoldi et al. (2008), where the community vectors $\{\pi_u\}$ are i.i.d. draws from the Dirichlet distribution, denoted by $\mathrm{Dir}(\alpha)$, with parameter vector $\alpha \in \mathbb{R}^k_{>0}$. The probability density function of the Dirichlet distribution is given by

$$\mathbb{P}[\pi] = \frac{\prod_{i=1}^k \Gamma(\alpha_i)}{\Gamma(\alpha_0)} \prod_{i=1}^k \pi_i^{\alpha_i - 1}, \quad \pi \sim \mathrm{Dir}(\alpha), \alpha_0 := \sum_i \alpha_i, \tag{6}$$

where $\Gamma(\cdot)$ is the Gamma function and the ratio of the Gamma function serves as the normalization constant.

5. We limit our discussion to directed networks in this paper, but note that the results also hold for undirected community models, where P is a symmetric matrix, and an edge (u, v) is formed with probability $\pi_u^\top P \pi_v = \pi_v^\top P \pi_u$.

The Dirichlet distribution is widely employed for specifying priors in Bayesian statistics, e.g., latent Dirichlet allocation (Blei et al., 2003). The Dirichlet distribution is the conjugate prior of the multinomial distribution which makes it attractive for Bayesian inference.

Let $\widehat{\alpha}$ denote the normalized parameter vector α/α_0, where $\alpha_0 := \sum_i \alpha_i$. In particular, note that $\widehat{\alpha}$ is a probability vector: $\sum_i \widehat{\alpha}_i = 1$. Intuitively, $\widehat{\alpha}$ denotes the relative expected sizes of the communities (since $\mathbb{E}[n^{-1} \sum_{u \in [n]} \pi_u[i]] = \widehat{\alpha}_i$). Let $\widehat{\alpha}_{\max}$ be the largest entry in $\widehat{\alpha}$, and $\widehat{\alpha}_{\min}$ be the smallest entry. Our learning guarantees will depend on these parameters.

The stochastic block model is a limiting case of the mixed membership model when the Dirichlet parameter is $\alpha = \alpha_0 \cdot \widehat{\alpha}$, where the probability vector $\widehat{\alpha}$ is held fixed and $\alpha_0 \to 0$. In the other extreme when $\alpha_0 \to \infty$, the Dirichlet distribution becomes peaked around a single point, for instance, if $\alpha_i \equiv c$ and $c \to \infty$, the Dirichlet distribution is peaked at $k^{-1} \cdot \vec{1}$, where $\vec{1}$ is the all-ones vector. Thus, the parameter α_0 serves as a measure of the average sparsity of the Dirichlet draws or equivalently, of how concentrated the Dirichlet measure is along the different coordinates. This in effect, controls the extent of overlap among different communities.

2.1.5 Sparse Regime of Dirichlet Distribution

When the Dirichlet parameter vector satisfies[6] $\alpha_i < 1$, for all $i \in [k]$, the Dirichlet distribution $\mathrm{Dir}(\alpha)$ generates "sparse" vectors with high probability;[7]; see Telgarsky (2012) (and in the extreme case of the block model where $\alpha_0 \to 0$, it generates 1-sparse vectors). Many real-world settings involve sparse community membership and the total number of communities is typically much larger than the extent of membership of a single individual, e.g., hobbies/interests of a person, university/company networks that a person belongs to, the set of transcription factors regulating a gene, and so on. Our learning guarantees are limited to the sparse regime of the Dirichlet model.

2.2 Graph Moments under Mixed Membership Models

Our approach for learning a mixed membership community model relies on the form of the graph moments[8] under the mixed membership model. We now describe the specific graph moments used by our learning algorithm (based on 3-star and edge counts) and provide explicit forms for the moments, assuming draws from a mixed membership model.

2.2.1 Notation

Recall that G denotes the adjacency matrix and that $G_{X,A}$ denotes the submatrix corresponding to edges going from X to A. Recall that $P \in [0,1]^{k \times k}$ denotes the community connectivity matrix. Define

$$F := \Pi^\top P^\top = [\pi_1 | \pi_2 | \cdots | \pi_n]^\top P^\top. \tag{7}$$

6. The assumption that the Dirichlet distribution be in the sparse regime is not strictly needed. Our results can be extended to general Dirichlet distributions, but with worse scaling requirements on the network size n for guaranteed learning.
7. Roughly the number of entries in π exceeding a threshold τ is at most $O(\alpha_0 \log(1/\tau))$ with high probability, when $\pi \sim \mathrm{Dir}(\alpha)$.
8. We interchangeably use the term first order moments for edge counts and third order moments for 3-star counts.

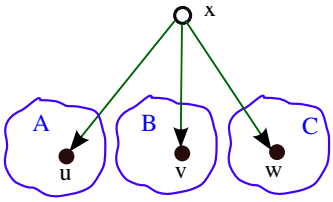


Figure 1: Our moment-based learning algorithm uses 3-star count tensor from set X to sets A, B, C (and the roles of the sets are interchanged to get various estimates). Specifically, T is a third order tensor, where $\mathrm{T}(u, v, w)$ is the normalized count of the 3-stars with u, v, w as leaves over all $x \in X$.

For a subset $A \subseteq [n]$ of individuals, let $F_A \in \mathbb{R}^{|A| \times k}$ denote the submatrix of F corresponding to nodes in A, i.e., $F_A := \Pi_A^\top P^\top$. We will subsequently show that F_A is linear map which takes any community vector π_i as input and outputs the corresponding neighborhood vector $G_{i,A}^\top$ in expectation.

Our learning algorithm uses moments up to the third-order, represented as a tensor. A third-order tensor T is a three-dimensional array whose (p, q, r)-th entry denoted by $T_{p,q,r}$. The symbol $\otimes$ denotes the standard Kronecker product: if u, v, w are three vectors, then

$$(u \otimes v \otimes w)_{p,q,r} := u_p \cdot v_q \cdot w_r. \tag{8}$$

A tensor of the form $u \otimes v \otimes w$ is referred to as a rank-one tensor. The decomposition of a general tensor into a sum of its rank-one components is referred to as *canonical polyadic (CP) decomposition* Kolda and Bader (2009). We will subsequently see that the graph moments can be expressed as a tensor and that the CP decomposition of the graph-moment tensor yields the model parameters and the community vectors under the mixed membership community model.

2.2.2 Graph Moments under Stochastic Block Model

We first analyze the graph moments in the special case of a stochastic block model (i.e., $\alpha_0 = \sum_i \alpha_i \to 0$ in the Dirichlet prior in (6)) and then extend it to general mixed membership model. We provide explicit expressions for the graph moments corresponding to edge counts and 3-star counts. We later establish in Section 3 that these moments are sufficient to learn the community memberships of the nodes and the model parameters of the block model.

2.2.3 3-star Counts

The primary quantity of interest is a third-order tensor which counts the number of 3-stars. A 3-star is a star graph with three leaves $\{a, b, c\}$ and we refer to the internal node x of the star as its "head", and denote the structure by $x \to \{a, b, c\}$ (see Figure 1). We partition the network into four[9] parts and consider 3-stars such that each node in the 3-star belongs to a different partition. This is necessary to obtain a simple form of the moments, based on the conditional independence assumptions of the block model, see Proposition 2. Specifically,

9. For sample complexity analysis, we require dividing the graph into more than four partitions to deal with statistical dependency issues, and we outline it in Section 3.

consider[10] a partition A, B, C, X of the network. We count the number of 3-stars from X to A, B, C and our quantity of interest is

$$\mathrm{T}_{X\to\{A,B,C\}} := \frac{1}{|X|}\sum_{i\in X}[G_{i,A}^\top \otimes G_{i,B}^\top \otimes G_{i,C}^\top], \tag{9}$$

where $\otimes$ is the Kronecker product, defined in (8) and $G_{i,A}$ is the row vector supported on the set of neighbors of i belonging to set A. $\mathrm{T} \in \mathbb{R}^{|A|\times|B|\times|C|}$ is a third order tensor, and an element of the tensor is given by

$$\mathrm{T}_{X\to\{A,B,C\}}(a,b,c) = \frac{1}{|X|}\sum_{x\in X} G(x,a)G(x,b)G(x,c), \quad \forall a\in A, b\in B, c\in C, \tag{10}$$

which is the normalized count of the number of 3-stars with leaves a, b, c such that its "head" is in set X.

We now relate the tensor $\mathrm{T}_{X\to\{A,B,C\}}$ to the parameters of the stochastic block model, viz., the community connectivity matrix P and the community probability vector $\widehat{\alpha}$, where $\widehat{\alpha}_i$ is the probability of choosing community i.

Proposition 2 (Moments in Stochastic Block Model) *Given partitions A, B, C, X, and $F := \Pi^\top P^\top$, where P is the community connectivity matrix and Π is the matrix of community membership vectors, we have*

$$\mathbb{E}[G_{X,A}^\top|\Pi_A,\Pi_X] = F_A\Pi_X, \tag{11}$$

$$\mathbb{E}[\mathrm{T}_{X\to\{A,B,C\}}\,|\Pi_A,\Pi_B,\Pi_C] = \sum_{i\in[k]}\widehat{\alpha}_i(F_A)_i\otimes(F_B)_i\otimes(F_C)_i, \tag{12}$$

where $\widehat{\alpha}_i$ is the probability for a node to select community i.

Remark: *In Equation* (11)*, we see that the edge generation occurs under a linear model, and more precisely, the matrix $F_A \in \mathbb{R}^{|A|\times k}$ is a linear map which takes a community vector $\pi_i \in \mathbb{R}^k$ to a neighborhood vector $G_{i,A}^\top \in \mathbb{R}^{|A|}$ in expectation.*

Remark: *(Identifiability under third order moments)) Note the form of the* 3*-star count tensor* T *in* (12)*. It provides a CP decomposition of* T *since each term in the summation, viz., $\widehat{\alpha}_i(F_A)_i\otimes(F_B)_i\otimes(F_C)_i$, is a rank one tensor. Thus, we can learn the matrices F_A, F_B, F_C and the vector $\widehat{\alpha}$ through CP decomposition of tensor* T*. Once these parameters are learnt, learning the communities is straight-forward under exact moments: by exploiting* (11)*, we find Π_X as*

$$\Pi_X = F_A^\dagger \cdot \mathbb{E}[G_{X,A}^\top|\Pi_A,\Pi_X].$$

Similarly, we can consider another tensor consisting of 3*-stars from A to X, B, C, and obtain matrices F_X, F_B and F_C through a CP decomposition, and so on. Once we obtain matrices F and Π for the entire set of nodes in this manner, we can obtain the community connectivity matrix P, since $F := \Pi^\top P^\top$. Thus, in principle, we are able to learn all the*

10. To establish our theoretical guarantees, we assume that the partitions A, B, C, X are randomly chosen and are of size $\Theta(n)$.

model parameters ($\widehat{\alpha}$ and P) and the community membership matrix Π under the stochastic block model, given exact moments. This establishes identifiability of the model given moments up to third order and forms a high-level approach for learning the communities. When only samples are available, we establish that the empirical versions are close to the exact moments considered above, and we modify the basic learning approach to obtain robust guarantees. See Section 3 for details.

Remark: *(Significance of conditional independence relationships) The main property exploited in proving the tensor form in* (12) *is the conditional-independence assumption under the stochastic block model: the realization of the edges in each 3-star, say in $x \to \{a, b, c\}$, is conditionally independent given the community membership vector π_x, when $x \neq a \neq b \neq c$. This is because the community membership vectors Π are assumed to be drawn independently at the different nodes and the edges are drawn independently given the community vectors.*

Considering 3-stars from X to A, B, C where X, A, B, C form a partition ensures that this conditional independence is satisfied for all the 3-stars in tensor T.

Proof: Recall that the probability of an edge from u to v given π_u, π_v is

$$\mathbb{E}[G_{u,v}|\pi_u, \pi_v] = \pi_u^\top P \pi_v = \pi_v^\top P^\top \pi_u = F_v \pi_u,$$

and $\mathbb{E}[G_{X,A}|\Pi_A, \Pi_X] = \Pi_X^\top P \Pi_A = \Pi_X^\top F_A^\top$ and thus (11) holds. For the tensor form, first consider an element of the tensor, with $a \in A, b \in B, c \in C$,

$$\mathbb{E}\left[\mathrm{T}_{X \to \{A,B,C\}}(a, b, c)|\pi_a, \pi_b, \pi_c, \pi_x\right] = \frac{1}{|X|} \sum_{x \in X} F_a \pi_x \cdot F_b \pi_x \cdot F_c \pi_x,$$

The equation follows from the conditional-independence assumption of the edges (assuming $a \neq b \neq c$). Now taking expectation over the nodes in X, we have

$$\begin{aligned}
\mathbb{E}\left[\mathrm{T}_{X \to \{A,B,C\}}(a, b, c)|\pi_a, \pi_b, \pi_c\right] &= \frac{1}{|X|} \sum_{x \in X} \mathbb{E}\left[F_a \pi_x \cdot F_b \pi_x \cdot F_c \pi_x | \pi_a, \pi_b, \pi_c\right] \\
&= \mathbb{E}\left[F_a \pi \cdot F_b \pi \cdot F_c \pi | \pi_a, \pi_b, \pi_c\right] \\
&= \sum_{j \in [k]} \widehat{\alpha}_j (F_a)_j \cdot (F_b)_j \cdot (F_c)_j,
\end{aligned}$$

where the last step follows from the fact that $\pi = e_j$ with probability $\widehat{\alpha}_j$ and the result holds when $x \neq a, b, c$. Recall that $(F_a)_j$ denotes the j^{th} column of F_a (since $F_a e_j = (F_a)_j$). Collecting all the elements of the tensor, we obtain the desired result. ■

2.2.4 Graph Moments under Mixed Membership Dirichlet Model

We now analyze the graph moments for the general mixed membership Dirichlet model. Instead of the raw moments (i.e., edge and 3-star counts), we consider modified moments to obtain similar expressions as in the case of the stochastic block model.

Let $\mu_{X \to A} \in \mathbb{R}^{|A|}$ denote a vector which gives the normalized count of edges from X to A:

$$\mu_{X \to A} := \frac{1}{|X|} \sum_{i \in X} [G_{i,A}^\top]. \tag{13}$$

We now define a modified adjacency matrix[11] $G_{X,A}^{\alpha_0}$ as

$$G_{X,A}^{\alpha_0} := \left(\sqrt{\alpha_0+1}G_{X,A} - (\sqrt{\alpha_0+1}-1)\vec{1}\mu_{X\to A}^\top\right). \tag{14}$$

In the special case of the stochastic block model ($\alpha_0 \to 0$), $G_{X,A}^{\alpha_0} = G_{X,A}$ is the submatrix of the adjacency matrix G. Similarly, we define modified third-order statistics,

$$\begin{aligned}\mathrm{T}_{X\to\{A,B,C\}}^{\alpha_0} &:= (\alpha_0+1)(\alpha_0+2)\,\mathrm{T}_{X\to\{A,B,C\}} + 2\,\alpha_0^2\,\mu_{X\to A}\otimes\mu_{X\to B}\otimes\mu_{X\to C} \\ &- \frac{\alpha_0(\alpha_0+1)}{|X|}\sum_{i\in X}\left[G_{i,A}^\top\otimes G_{i,B}^\top\otimes\mu_{X\to C} + G_{i,A}^\top\otimes\mu_{X\to B}\otimes G_{i,C}^\top + \mu_{X\to A}\otimes G_{i,B}^\top\otimes G_{i,C}^\top\right],\end{aligned} \tag{15}$$

and it reduces to (a scaled version of) the 3-star count $\mathrm{T}_{X\to\{A,B,C\}}$ defined in (9) for the stochastic block model ($\alpha_0 \to 0$). The modified adjacency matrix and the 3-star count tensor can be viewed as a form of "centering" of the raw moments which simplifies the expressions for the moments. The following relationships hold between the modified graph moments $G_{X,A}^{\alpha_0}$, T^{α_0} and the model parameters P and $\widehat{\alpha}$ of the mixed membership model.

Proposition 3 (Moments in Mixed Membership Model) *Given partitions A, B, C, X and $G_{X,A}^{\alpha_0}$ and T^{α_0}, as in (14) and (15), normalized Dirichlet concentration vector $\widehat{\alpha}$, and $F := \Pi^\top P^\top$, where P is the community connectivity matrix and Π is the matrix of community memberships, we have*

$$\mathbb{E}[(G_{X,A}^{\alpha_0})^\top|\Pi_A,\Pi_X] = F_A\,\mathrm{Diag}(\widehat{\alpha}^{1/2})\Psi_X, \tag{16}$$

$$\mathbb{E}[\mathrm{T}_{X\to\{A,B,C\}}^{\alpha_0}\,|\Pi_A,\Pi_B,\Pi_C] = \sum_{i=1}^k \widehat{\alpha}_i(F_A)_i\otimes(F_B)_i\otimes(F_C)_i, \tag{17}$$

where $(F_A)_i$ corresponds to i^{th} column of F_A and Ψ_X relates to the community membership matrix Π_X as

$$\Psi_X := \mathrm{Diag}(\widehat{\alpha}^{-1/2})\left(\sqrt{\alpha_0+1}\Pi_X - (\sqrt{\alpha_0+1}-1)\left(\frac{1}{|X|}\sum_{i\in X}\pi_i\right)\vec{1}^\top\right).$$

Moreover, we have that

$$|X|^{-1}\mathbb{E}_{\Pi_X}[\Psi_X\Psi_X^\top] = I. \tag{18}$$

Remark: *The 3-star count tensor T^{α_0} is carefully chosen so that the CP decomposition of the tensor directly yields the matrices F_A, F_B, F_C and $\widehat{\alpha}_i$, as in the case of the stochastic block model. Similarly, the modified adjacency matrix $(G_{X,A}^{\alpha_0})^\top$ is carefully chosen to eliminate second-order correlation in the Dirichlet distribution and we have that $|X|^{-1}\mathbb{E}_{\Pi_X}[\Psi\Psi^\top] = I$ is the identity matrix. These properties will be exploited by our learning algorithm in Section 3.*

11. To compute the modified moments G^{α_0}, and T^{α_0}, we need to know the value of the scalar $\alpha_0 := \sum_i \alpha_i$, which is the concentration parameter of the Dirichlet distribution and is a measure of the extent of overlap between the communities. We assume its knowledge here.

Remark: *Recall that α_0 quantifies the extent of overlap among the communities. The computation of the modified moment T^{α_0} requires the knowledge of α_0, which is assumed to be known. Since this is a scalar quantity, in practice, we can easily tune this parameter via cross validation.*

Proof: The proof is on lines of Proposition 2 for stochastic block models ($\alpha_0 \to 0$) but more involved due to the form of Dirichlet moments. Recall $\mathbb{E}[G_{i,A}^\top|\pi_i, \Pi_A] = F_A\pi_i$ for a mixed membership model, and $\mu_{X\to A} := \frac{1}{|X|}\sum_{i\in X} G_{i,A}^\top$, therefore $\mathbb{E}[\mu_{X\to A}|\Pi_A, \Pi_X] = F_A\left(\frac{1}{|X|}\sum_{i\in X}\pi_i\right)\vec{1}^\top$. Equation (16) follows directly. For Equation (18), we note the Dirichlet moment, $\mathbb{E}[\pi\pi^\top] = \frac{1}{\alpha_0+1}\operatorname{Diag}(\widehat{\alpha}) + \frac{\alpha_0}{\alpha_0+1}\widehat{\alpha}\widehat{\alpha}^\top$, when $\pi \sim \operatorname{Dir}(\alpha)$ and

$$\begin{aligned}
|X|^{-1}\mathbb{E}[\Psi_X\Psi_X^\top] &= \operatorname{Diag}(\widehat{\alpha}^{-1/2})\Big[(\alpha_0+1)\mathbb{E}[\pi\pi^\top] + (-2\sqrt{\alpha_0+1}(\sqrt{\alpha_0+1}-1) \\
&\quad +(\sqrt{\alpha_0+1}-1)^2)\mathbb{E}[\pi]\mathbb{E}[\pi]^\top\Big]\operatorname{Diag}(\widehat{\alpha}^{-1/2}) \\
&= \operatorname{Diag}(\widehat{\alpha}^{-1/2})\left(\operatorname{Diag}(\widehat{\alpha}) + \alpha_0\widehat{\alpha}\widehat{\alpha}^\top + (-\alpha_0)\widehat{\alpha}\widehat{\alpha}^\top\right)\operatorname{Diag}(\widehat{\alpha}^{-1/2}) \\
&= I.
\end{aligned}$$

On lines of the proof of Proposition 2 for the block model, the expectation in (17) involves multi-linear map of the expectation of the tensor products $\pi\otimes\pi\otimes\pi$ among other terms. Collecting these terms, we have that

$$\begin{aligned}
&(\alpha_0+1)(\alpha_0+2)\mathbb{E}[\pi\otimes\pi\otimes\pi] - (\alpha_0)(\alpha_0+1)(\mathbb{E}[\pi\otimes\pi\otimes\mathbb{E}[\pi]] \\
&+\mathbb{E}[\pi\otimes\mathbb{E}[\pi]\otimes\pi] + \mathbb{E}[\mathbb{E}[\pi]\otimes\pi\otimes\pi]) + 2\alpha_0^2\mathbb{E}[\pi]\otimes\mathbb{E}[\pi]\otimes\mathbb{E}[\pi]
\end{aligned}$$

is a diagonal tensor, in the sense that its (p,p,p)-th entry is $\widehat{\alpha}_p$, and its (p,q,r)-th entry is 0 when p,q,r are not all equal. With this, we have (17). ■

Note the nearly identical forms of the graph moments for the stochastic block model in (11), (12) and for the general mixed membership model in (16), (17). In other words, the modified moments $G_{X,A}^{\alpha_0}$ and T^{α_0} have similar relationships to underlying parameters as the raw moments in the case of the stochastic block model. This enables us to use a unified learning approach for the two models, outlined in the next section.

3. Algorithm for Learning Mixed Membership Models

The simple form of the graph moments derived in the previous section is now utilized to recover the community vectors Π and model parameters $P, \widehat{\alpha}$ of the mixed membership model. The method is based on the so-called tensor power method, used to obtain a tensor decomposition. We first outline the basic tensor decomposition method below and then demonstrate how the method can be adapted to learning using the graph moments at hand. We first analyze the simpler case when exact moments are available in Section 3.2 and then extend the method to handle empirical moments computed from the network observations in Section 3.3.

3.1 Overview of Tensor Decomposition through Power Iterations

In this section, we review the basic method for tensor decomposition based on power iterations for a special class of tensors, viz., symmetric orthogonal tensors. Subsequently, in Section 3.2 and 3.3, we modify this method to learn the mixed membership model from graph moments, described in the previous section. For details on the tensor power method, refer to Anandkumar et al. (2012a); Kolda and Mayo (2011).

Recall that a third-order tensor T is a three-dimensional array and we use $T_{p,q,r}$ to denote the (p,q,r)-th entry of the tensor T. The standard symbol $\otimes$ is used to denote the Kronecker product, and $(u \otimes v \otimes w)$ is a rank one tensor. The decomposition of a tensor into its rank one components is called the CP decomposition.

3.1.1 Multi-linear Maps

We can view a tensor $T \in \mathbb{R}^{d \times d \times d}$ as a multilinear map in the following sense: for a set of matrices $\{V_i \in \mathbb{R}^{d \times m_i} : i \in [3]\}$, the (i_1, i_2, i_3)-th entry in the three-way array representation of $T(V_1, V_2, V_3) \in \mathbb{R}^{m_1 \times m_2 \times m_3}$ is

$$[T(V_1, V_2, V_3)]_{i_1,i_2,i_3} := \sum_{j_1,j_2,j_3 \in [d]} T_{j_1,j_2,j_3} \, [V_1]_{j_1,i_1} \, [V_2]_{j_2,i_2} \, [V_3]_{j_3,i_3}.$$

The term multilinear map arises from the fact that the above map is linear in each of the coordinates, e.g., if we replace V_1 by $aV_1 + bW_1$ in the above equation, where W_1 is a matrix of appropriate dimensions, and a, b are any scalars, the output is a linear combination of the outputs under V_1 and W_1 respectively. We will use the above notion of multi-linear transforms to describe various tensor operations. For instance, $T(I, I, v)$ yields a matrix, $T(I, v, v)$, a vector, and $T(v, v, v)$, a scalar.

3.1.2 Symmetric Tensors and Orthogonal Decomposition

A special class of tensors are the symmetric tensors $T \in \mathbb{R}^{d \times d \times d}$ which are invariant to permutation of the array indices. Symmetric tensors have CP decomposition of the form

$$T = \sum_{i \in [r]} \lambda_i v_i \otimes v_i \otimes v_i = \sum_{i \in [r]} \lambda_i v_i^{\otimes 3}, \tag{19}$$

where r denotes the tensor CP rank and we use the notation $v_i^{\otimes 3} := v_i \otimes v_i \otimes v_i$. It is convenient to first analyze methods for decomposition of symmetric tensors and we then extend them to the general case of asymmetric tensors.

Further, a sub-class of symmetric tensors are those which possess a decomposition into orthogonal components, i.e., the vectors $v_i \in \mathbb{R}^d$ are orthogonal to one another in the above decomposition in (19) (without loss of generality, we assume that vectors $\{v_i\}$ are orthonormal in this case). An orthogonal decomposition implies that the tensor rank $r \leq d$ and there are tractable methods for recovering the rank-one components in this setting. We limit ourselves to this setting in this paper.

3.1.3 Tensor Eigen Analysis

For symmetric tensors T possessing an orthogonal decomposition of the form in (19), each pair (λ_i, v_i), for $i \in [r]$, can be interpreted as an eigen-pair for the tensor T, since

$$T(I, v_i, v_i) = \sum_{j \in [r]} \lambda_j \langle v_i, v_j \rangle^2 v_j = \lambda_i v_i, \quad \forall i \in [r],$$

due to the fact that $\langle v_i, v_j \rangle = \delta_{i,j}$. Thus, the vectors $\{v_i\}_{i \in [r]}$ can be interpreted as fixed points of the map

$$v \mapsto \frac{T(I, v, v)}{\|T(I, v, v)\|}, \tag{20}$$

where $\|\cdot\|$ denotes the spectral norm (and $\|T(I, v, v)\|$ is a vector norm), and is used to normalize the vector v in (20).

3.1.4 Basic Tensor Power Iteration Method

A straightforward approach to computing the orthogonal decomposition of a symmetric tensor is to iterate according to the fixed-point map in (20) with an arbitrary initialization vector. This is referred to as the tensor power iteration method. Additionally, it is known that the vectors $\{v_i\}_{i \in [r]}$ are the only *stable* fixed points of the map in (20). In other words, the set of initialization vectors which converge to vectors other than $\{v_i\}_{i \in [r]}$ are of measure zero. This ensures that we obtain the correct set of vectors through power iterations and that no spurious answers are obtained. See Anandkumar et al. (2012b, Thm. 4.1) for details. Moreover, after an approximately fixed point is obtained (after many power iterations), the estimated eigen-pair can be subtracted out (i.e., *deflated*) and subsequent vectors can be similarly obtained through power iterations. Thus, we can obtain all the stable eigen-pairs $\{\lambda_i, v_i\}_{i \in [r]}$ which are the components of the orthogonal tensor decomposition. The method needs to be suitably modified when the tensor T is perturbed (e.g., as in the case when empirical moments are used) and we discuss it in Section 3.3.

3.2 Learning Mixed Membership Models under Exact Moments

We first describe the learning approach when exact moments are available. In Section 3.3, we suitably modify the approach to handle perturbations, which are introduced when only empirical moments are available.

We now employ the tensor power method described above to obtain a CP decomposition of the graph moment tensor T^{α_0} in (15). We first describe a "symmetrization" procedure to convert the graph moment tensor T^{α_0} to a symmetric orthogonal tensor through a multi-linear transformation of T^{α_0}. We then employ the power method to obtain a symmetric orthogonal decomposition. Finally, the original CP decomposition is obtained by reversing the multi-linear transform of the symmetrization procedure. This yields a guaranteed method for obtaining the decomposition of graph moment tensor T^{α_0} under exact moments. We note that this symmetrization approach has been earlier employed in other contexts, e.g., for learning hidden Markov models (Anandkumar et al., 2012b, Sec. 3.3).

3.2.1 Reduction of the Graph-moment Tensor to Symmetric Orthogonal Form (Whitening)

Recall from Proposition 3 that the modified 3-star count tensor T^{α_0} has a CP decomposition as

$$\mathbb{E}[\mathrm{T}^{\alpha_0} | \Pi_A, \Pi_B, \Pi_C] = \sum_{i=1}^{k} \widehat{\alpha}_i (F_A)_i \otimes (F_B)_i \otimes (F_C)_i.$$

We now describe a symmetrization procedure to convert T^{α_0} to a symmetric orthogonal tensor through a multi-linear transformation using the modified adjacency matrix G^{α_0}, defined in (14). Consider the singular value decomposition (SVD) of the modified adjacency matrix G^{α_0} under exact moments:

$$|X|^{-1/2} \mathbb{E}[(G^{\alpha_0}_{X,A})^\top | \Pi] = U_A D_A V_A^\top.$$

Define $W_A := U_A D_A^{-1}$, and similarly define W_B and W_C using the corresponding matrices $G^{\alpha_0}_{X,B}$ and $G^{\alpha_0}_{X,C}$ respectively. Now define

$$R_{A,B} := \frac{1}{|X|} W_B^\top \mathbb{E}[(G^{\alpha_0}_{X,B})^\top | \Pi] \cdot \mathbb{E}[(G^{\alpha_0}_{X,A}) | \Pi] W_A, \quad \tilde{W}_B := W_B R_{A,B}, \tag{21}$$

and similarly define $\tilde{W}_C$. We establish that a multilinear transformation (as defined in (3.1.1)) of the graph-moment tensor T^{α_0} using matrices $W_A, \tilde{W}_B$, and $\tilde{W}_C$ results in a symmetric orthogonal form.

Lemma 4 (Orthogonal Symmetric Tensor) *Assume that the matrices F_A, F_B, F_C and Π_X have rank k, where k is the number of communities. We have an orthogonal symmetric tensor form for the modified 3-star count tensor T^{α_0} in (15) under a multilinear transformation using matrices $W_A, \tilde{W}_B$, and $\tilde{W}_C$:*

$$\mathbb{E}[\mathrm{T}^{\alpha_0}(W_A, \tilde{W}_B, \tilde{W}_C) | \Pi_A, \Pi_B, \Pi_C] = \sum_{i \in [k]} \lambda_i (\Phi)_i^{\otimes 3} \in \mathbb{R}^{k \times k \times k}, \tag{22}$$

where $\lambda_i := \widehat{\alpha}_i^{-0.5}$ and $\Phi \in \mathbb{R}^{k \times k}$ is an orthogonal matrix, given by

$$\Phi := W_A^\top F_A \, \mathrm{Diag}(\widehat{\alpha}^{0.5}). \tag{23}$$

Remark: *Note that the matrix W_A orthogonalizes F_A under exact moments, and is referred to as a* whitening matrix. *Similarly, the matrices $\tilde{W}_B = R_{A,B} W_B$ and $\tilde{W}_C = R_{A,C} W_C$ consist of whitening matrices W_B and W_C, and in addition, the matrices $R_{A,B}$ and $R_{A,C}$ serve to symmetrize the tensor. We can interpret $\{\lambda_i, (\Phi)_i\}_{i \in [k]}$ as the stable eigenpairs of the transformed tensor (henceforth, referred to as the* whitened and symmetrized tensor*).*

Remark: *The full rank assumption on matrix $F_A = \Pi_A^\top P^\top \in \mathbb{R}^{|A| \times k}$ implies that $|A| \geq k$, and similarly $|B|, |C|, |X| \geq k$. Moreover, we require the community connectivity matrix $P \in \mathbb{R}^{k \times k}$ to be of full rank*[12] *(which is a natural non-degeneracy condition). In this case,*

12. In the work of McSherry (2001), where spectral clustering for stochastic block models is analyzed, rank deficient P is allowed as long as the neighborhood vectors generated by any pair of communities are sufficiently different. On the other hand, our method requires P to be full rank. We argue that this is a mild restriction since we allow for mixed memberships while McSherry (2001) limit to the stochastic block model.

we can reduce the graph-moment tensor T^{α_0} *to a* k*-rank orthogonal symmetric tensor, which has a unique decomposition. This implies that the mixed membership model is identifiable using* 3*-star and edge count moments, when the network size* $n = |A|+|B|+|C|+|X| \geq 4k$*, matrix* P *is full rank and the community membership matrices* $\Pi_A, \Pi_B, \Pi_C, \Pi_X$ *each have rank* k*. On the other hand, when only empirical moments are available, roughly, we require the network size* $n = \Omega(k^2(\alpha_0+1)^2)$ *(where* $\alpha_0 := \sum_i \alpha_i$ *is related to the extent of overlap between the communities) to provide guaranteed learning of the community membership and model parameters. See Section 4 for a detailed sample analysis.*

Proof: Recall that the modified adjacency matrix G^{α_0} satisfies

$$\mathbb{E}[(G^{\alpha_0}_{X,A})^\top | \Pi_A, \Pi_X] = F_A \operatorname{Diag}(\widehat{\alpha}^{1/2}) \Psi_X,$$
$$\Psi_X := \operatorname{Diag}(\widehat{\alpha}^{-1/2}) \left(\sqrt{\alpha_0+1}\Pi_X - (\sqrt{\alpha_0+1}-1) \left(\frac{1}{|X|} \sum_{i \in X} \pi_i \right) \vec{1}^\top \right).$$

From the definition of Ψ_X above, we see that it has rank k when Π_X has rank k. Using the Sylvester's rank inequality, we have that the rank of $F_A \operatorname{Diag}(\widehat{\alpha}^{1/2})\Psi_X$ is at least $2k-k=k$. This implies that the whitening matrix W_A also has rank k. Notice that

$$|X|^{-1} W_A^\top \mathbb{E}[(G^{\alpha_0}_{X,A})^\top | \Pi] \cdot \mathbb{E}[(G^{\alpha_0}_{X,A}) | \Pi] W_A = D_A^{-1} U_A^\top U_A D_A^2 U_A^\top U_A D_A^{-1} = I \in \mathbb{R}^{k \times k},$$

or in other words, $|X|^{-1} M M^\top = I$, where $M := W_A^\top F_A \operatorname{Diag}(\widehat{\alpha}^{1/2}) \Psi_X$. We now have that

$$\begin{aligned} I = |X|^{-1} \mathbb{E}_{\Pi_X}\left[M M^\top \right] &= |X|^{-1} W_A^\top F_A \operatorname{Diag}(\widehat{\alpha}^{1/2}) \mathbb{E}[\Psi_X \Psi_X^\top] \operatorname{Diag}(\widehat{\alpha}^{1/2}) F_A^\top W_A \\ &= W_A^\top F_A \operatorname{Diag}(\widehat{\alpha}) F_A^\top W_A, \end{aligned}$$

since $|X|^{-1}\mathbb{E}_{\Pi_X}[\Psi_X \Psi_X^\top] = I$ from (18), and we use the fact that the sets A and X do not overlap. Thus, W_A whitens $F_A \operatorname{Diag}(\widehat{\alpha}^{1/2})$ under exact moments (up on taking expectation over Π_X) and the columns of $W_A^\top F_A \operatorname{Diag}(\widehat{\alpha}^{1/2})$ are orthonormal. Now note from the definition of $\tilde{W}_B$ that

$$\tilde{W}_B^\top \mathbb{E}[(G^{\alpha_0}_{X,B})^\top | \Pi] = W_A^\top \mathbb{E}[(G^{\alpha_0}_{X,A})^\top | \Pi],$$

since W_B satisfies

$$|X|^{-1} W_B^\top \mathbb{E}[(G^{\alpha_0}_{X,B})^\top | \Pi] \cdot \mathbb{E}[(G^{\alpha_0}_{X,B}) | \Pi] W_B = I,$$

and similar result holds for $\tilde{W}_C$. The final result in (22) follows by taking expectation of tensor T^{α_0} over Π_X. ■

3.2.2 Overview of the Learning Approach under Exact Moments

With the above result in place, we are now ready to describe the high-level approach for learning the mixed membership model under exact moments. First, symmetrize the graph-moment tensor T^{α_0} as described above and then apply the tensor power method described in the previous section. This enables us to obtain the vector of eigenvalues $\lambda := \widehat{\alpha}^{-1/2}$ and the matrix of eigenvectors $\Phi = W_A^\top F_A \operatorname{Diag}(\widehat{\alpha}^{0.5})$ using tensor power iterations. We can

then recover the community membership vectors of set A^c (i.e., nodes not in set A) under exact moments as

$$\Pi_{A^c} \leftarrow \mathrm{Diag}(\lambda)^{-1}\Phi^\top W_A^\top \mathbb{E}[G_{A^c,A}^\top|\Pi],$$

since $\mathbb{E}[G_{A^c,A}^\top|\Pi] = F_A\Pi_{A^c}$ (since A and A^c do not overlap) and $\mathrm{Diag}(\lambda)^{-1}\Phi^\top W_A^\top = \mathrm{Diag}(\widehat{\alpha})F_A^\top W_A W_A^\top$ under exact moments. In order to recover the community membership vectors of set A, viz., Π_A, we can use the edge-set $G_{A,B}$. Once all the community membership vectors Π are obtained, we can obtain the community connectivity matrix P, using the relationship: $\Pi^\top P\Pi = \mathbb{E}[G|\Pi]$ and noting that we assume Π to be of rank k. Thus, we are able to learn the community membership vectors Π and the model parameters $\widehat{\alpha}$ and P of the mixed membership model using edge counts and the 3-star count tensor. We now describe modifications to this approach to handle empirical moments.

3.3 Learning Algorithm under Empirical Moments

In the previous section, we explored a tensor-based approach for learning mixed membership model under exact moments. However, in practice, we only have samples (i.e., the observed network), and the method needs to be robust to perturbations when empirical moments are employed.

Algorithm 1 $\{\hat{\Pi}, \hat{P}, \widehat{\alpha}\} \leftarrow \mathrm{LearnMixedMembership}(G, k, \alpha_0, N, \tau)$

Input: Adjacency matrix $G \in \mathbb{R}^{n\times n}$, k is the number of communities, $\alpha_0 := \sum_i \alpha_i$, where α is the Dirichlet parameter vector, N is the number of iterations for the tensor power method, and τ is used for thresholding the estimated community membership vectors, specified in (29) in assumption A5. Let $A^c := [n] \setminus A$ denote the set of nodes not in A.

Output: Estimates of the community membership vectors $\Pi \in \mathbb{R}^{n\times k}$, community connectivity matrix $P \in [0,1]^{k\times k}$, and the normalized Dirichlet parameter vector $\widehat{\alpha}$.

Partition the vertex set $[n]$ into 5 parts X, Y, A, B, C.

Compute moments $G^{\alpha_0}_{X,A}$, $G^{\alpha_0}_{X,B}$, $G^{\alpha_0}_{X,C}$, $\mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}$ using (14) and (15).

$\{\hat{\Pi}, \widehat{\alpha}\} \leftarrow \mathrm{LearnPartitionCommunity}(G^{\alpha_0}_{X,A}, G^{\alpha_0}_{X,B}, G^{\alpha_0}_{X,C}, \mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}, G, N, \tau)$.

Define $\hat{Q}$ such that its i-th row is $\hat{Q}^i := (\alpha_0+1)\frac{\hat{\Pi}^i}{|\hat{\Pi}^i|_1} - \frac{\alpha_0}{n}\vec{1}^\top$. {We will establish that $\hat{Q} \approx (\Pi^\dagger)^\top$ under conditions A1-A5.}

Estimate $\hat{P} \leftarrow \hat{Q}G\hat{Q}^\top$. {Recall that $\mathbb{E}[G] = \Pi^\top P\Pi$ in our model.}

Return $\hat{\Pi}, \hat{P}, \widehat{\alpha}$

3.3.1 Pre-processing Step: Partitioning

In the previous section, we partitioned the nodes into four sets A, B, C, X for learning under exact moments. However, we require more partitions under empirical moments to avoid statistical dependency issues and obtain stronger reconstruction guarantees. We now divide the network into five non-overlapping sets A, B, C, X, Y. The set X is employed to compute whitening matrices $\hat{W}_A$, $\hat{W}_B$ and $\hat{W}_C$, described in detail subsequently, the set Y is employed to compute the 3-star count tensor T^{α_0} and sets A, B, C contain the leaves of

Procedure 1 $\{\hat{\Pi}, \widehat{\alpha}\} \leftarrow$ LearnPartitionCommunity($G^{\alpha_0}_{X,A}$, $G^{\alpha_0}_{X,B}$, $G^{\alpha_0}_{X,C}$, $\mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}$, G, N, τ)

Input: Require modified adjacency submatrices $G^{\alpha_0}_{X,A}$, $G^{\alpha_0}_{X,B}$, $G^{\alpha_0}_{X,C}$, 3-star count tensor $\mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}$, adjacency matrix G, number of iterations N for the tensor power method and threshold τ for thresholding estimated community membership vectors. Let $\mathrm{Thres}(A,\tau)$ denote the element-wise thresholding operation using threshold τ, i.e., $\mathrm{Thres}(A,\tau)_{i,j} = A_{i,j}$ if $A_{i,j} \geq \tau$ and 0 otherwise. Let e_i denote basis vector along coordinate i.

Output: Estimates of Π and $\widehat{\alpha}$.

Compute rank-k SVD: $(|X|^{-1/2}G^{\alpha_0}_{X,A})^{\top}_{k-svd} = U_A D_A V_A^{\top}$ and compute whitening matrices $\hat{W}_A := U_A D_A^{-1}$. Similarly, compute $\hat{W}_B, \hat{W}_C$ and $\hat{R}_{AB}, \hat{R}_{AC}$ using (24).

Compute whitened and symmetrized tensor $T \leftarrow \mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}(\hat{W}_A, \hat{W}_B\hat{R}_{AB}, \hat{W}_C\hat{R}_{AC})$.

$\{\hat{\lambda}, \hat{\Phi}\} \leftarrow$TensorEigen$(T, \{\hat{W}_A^{\top}G^{\top}_{i,A}\}_{i\notin A}, N)$. $\{\hat{\Phi}$ is a $k\times k$ matrix with each columns being an estimated eigenvector and $\hat{\lambda}$ is the vector of estimated eigenvalues.$\}$

$\hat{\Pi}_{A^c} \leftarrow \mathrm{Thres}(\mathrm{Diag}(\hat{\lambda})^{-1}\hat{\Phi}^{\top}\hat{W}_A^{\top}G^{\top}_{A^c,A}\,,\ \tau)$ and $\hat{\alpha}_i \leftarrow \hat{\lambda}_i^{-2}$, for $i \in [k]$.

$\hat{\Pi}_A \leftarrow \mathrm{Thres}(\mathrm{Diag}(\hat{\lambda})^{-1}\hat{\Phi}^{\top}\hat{R}_{AB}^{\top}\hat{W}_B^{\top}G^{\top}_{A,B}\,,\ \tau)$.

Return $\hat{\Pi}$ and $\hat{\alpha}$.

the 3-stars under consideration. The roles of the sets can be interchanged to obtain the community membership vectors of all the sets.

3.3.2 Pre-processing Step: Whitening

The whitening procedure is along the same lines as described in the previous section, except that now empirical moments are used. Specifically, consider the k-rank singular value decomposition (SVD) of the modified adjacency matrix G^{α_0} defined in (14),

$$(|X|^{-1/2}G^{\alpha_0}_{X,A})^{\top}_{k-svd} = U_A D_A V_A^{\top}.$$

Define $\hat{W}_A := U_A D_A^{-1}$, and similarly define $\hat{W}_B$ and $\hat{W}_C$ using the corresponding matrices $G^{\alpha_0}_{X,B}$ and $G^{\alpha_0}_{X,C}$ respectively. Now define

$$\hat{R}_{A,B} := \frac{1}{|X|}\hat{W}_B^{\top}(G^{\alpha_0}_{X,B})^{\top}_{k-svd}\cdot(G^{\alpha_0}_{X,A})_{k-svd}\hat{W}_A, \tag{24}$$

and similarly define $\hat{R}_{AC}$. The whitened and symmetrized graph-moment tensor is now computed as

$$\mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}(\hat{W}_A, \hat{W}_B\hat{R}_{AB}, \hat{W}_C\hat{R}_{AC}),$$

where T^{α_0} is given by (15) and the multi-linear transformation of a tensor is defined in (3.1.1).

3.3.3 Modifications to the Tensor Power Method

Recall that under exact moments, the stable eigen-pairs of a symmetric orthogonal tensor can be computed in a straightforward manner through the basic power iteration method in

(20), along with the deflation procedure. However, this is not sufficient to get good reconstruction guarantees under empirical moments. We now propose a robust tensor method, detailed in Procedure 2. The main modifications involve: *(i)* efficient initialization and *(ii)* adaptive deflation, which are detailed below. Employing these modifications allows us to tolerate a far greater perturbation of the third order moment tensor, than the basic tensor power procedure employed in Anandkumar et al. (2012b). See remarks following Theorem 11 in Appendix A for the precise comparison.

3.3.4 Modification 1: Efficient Initialization

Recall that the basic tensor power method incorporates generic initialization vectors and this procedure recovers all the stable eigenvectors correctly (except for initialization vectors over a set of measure zero). However, under empirical moments, we have a perturbed tensor, and here, it is advantageous to instead employ specific initialization vectors. For instance, to obtain one of the eigenvectors $(\Phi)_i$, it is advantageous to initialize with a vector in the neighborhood of $(\Phi)_i$. This not only reduces the number of power iterations required to converge (approximately), but more importantly, this makes the power method more robust to perturbations. See Theorem 11 in Appendix A.1 for a detailed analysis quantifying the relationship between initialization vectors, tensor perturbation and the resulting guarantees for recovery of the tensor eigenvectors.

For a mixed membership model in the sparse regime, recall that the community membership vectors Π are sparse (with high probability). Under this regime of the model, we note that the whitened neighborhood vectors contain good initializers for the power iterations. Specifically, in Procedure 2, we initialize with the whitened neighborhood vectors $\hat{W}_A^\top G_{i,A}^\top$, for $i \notin A$. The intuition behind this is as follows: for a suitable choice of parameters (such as the scaling of network size n with respect to the number of communities k), we expect neighborhood vectors $G_{i,A}^\top$ to concentrate around their mean values, viz., , $F_A \pi_i$. Since π_i is sparse (w.h.p) for the model regime under consideration, this implies that there exist vectors $\hat{W}_A^\top F_A \pi_i$, for $i \in A^c$, which concentrate (w.h.p) on only along a few eigen-directions of the whitened tensor, and hence, serve as an effective initializer.

3.3.5 Modification 2: Adaptive Deflation

Recall that in the basic power iteration procedure, we can obtain the eigen-pairs one after another through simple deflation: subtracting the estimates of the current eigen-pairs and running the power iterations again to obtain new eigenvectors. However, it turns out that we can establish better theoretical guarantees (in terms of greater robustness) when we adaptively deflate the tensor in each power iteration. In Procedure 2, among the estimated eigen-pairs, we only deflate those which "compete" with the current estimate of the power iteration. In other words, if the vector in the current iteration $\theta_t^{(\tau)}$ has a significant projection along the direction of an estimated eigen-pair ϕ_j, i.e.,

$$|\lambda_j \left\langle \theta_t^{(\tau)}, \phi_j \right\rangle| > \xi,$$

for some threshold ξ, then the eigen-pair is deflated; otherwise the eigenvector ϕ_j is not deflated. This allows us to carefully control the error build-up for each estimated eigenpair in our analysis. Intuitively, if an eigenvector does not have a good correlation with the

current estimate, then it does not interfere with the update of the current vector, while if the eigenvector has a good correlation, then it is pertinent that it be deflated so as to discourage convergence in the direction of the already estimated eigenvector. See Theorem 11 in Appendix A.1 for details.

Finally, we note that stabilization, as proposed by Kolda and Mayo (2011) for general tensor eigen-decomposition (as opposed to orthogonal decomposition in this paper), can be effective in improving convergence, especially on real data, and we defer its detailed analysis to future work.

Procedure 2 $\{\lambda, \Phi\} \leftarrow$TensorEigen$(T, \{v_i\}_{i\in[L]}, N)$

Input: Tensor $T \in \mathbb{R}^{k\times k\times k}$, L initialization vectors $\{v_i\}_{i\in L}$, number of iterations N.
Output: the estimated eigenvalue/eigenvector pairs $\{\lambda, \Phi\}$, where λ is the vector of eigenvalues and Φ is the matrix of eigenvectors.

```
for i = 1 to k do
  for τ = 1 to L do
    θ_0 ← v_τ.
    for t = 1 to N do
      T̃ ← T.
      for j = 1 to i − 1 (when i > 1) do
        if |λ_j ⟨θ_t^(τ), φ_j⟩| > ξ then
          T̃ ← T̃ − λ_j φ_j^⊗3.
        end if
      end for
      Compute power iteration update θ_t^(τ) := T̃(I, θ_{t−1}^(τ), θ_{t−1}^(τ)) / ‖T̃(I, θ_{t−1}^(τ), θ_{t−1}^(τ))‖
    end for
  end for
  Let τ* := arg max_{τ∈L} {T̃(θ_N^(τ), θ_N^(τ), θ_N^(τ))}.
  Do N power iteration updates starting from θ_N^(τ*) to obtain eigenvector estimate φ_i,
  and set λ_i := T̃(φ_i, φ_i, φ_i).
end for
return the estimated eigenvalue/eigenvectors (λ, Φ).
```

3.3.6 Reconstruction after Tensor Power Method

Recall that previously in Section 3.2, when exact moments are available, estimating the community membership vectors Π is straightforward, once we recover all the stable tensor eigen-pairs. However, in case of empirical moments, we can obtain better guarantees with the following modification: the estimated community membership vectors Π are further subject to thresholding so that the weak values are set to zero. Since we are limiting ourselves to the regime of the mixed membership model, where the community vectors Π are sparse (w.h.p), this modification strengthens our reconstruction guarantees. This thresholding step is incorporated in Algorithm 1.

Moreover, recall that under exact moments, estimating the community connectivity matrix P is straightforward, once we recover the community membership vectors since $P \leftarrow (\Pi^\top)^\dagger \mathbb{E}[G|\Pi]\Pi^\dagger$. However, when empirical moments are available, we are able to establish better reconstruction guarantees through a different method, outlined in Algorithm 1. We define $\hat{Q}$ such that its i-th row is

$$\hat{Q}^i := (\alpha_0 + 1)\frac{\hat{\Pi}^i}{|\hat{\Pi}^i|_1} - \frac{\alpha_0}{n}\vec{1}^\top,$$

based on estimates $\hat{\Pi}$, and the matrix $\hat{P}$ is obtained as $\hat{P} \leftarrow \hat{Q}G\hat{Q}^\top$. We subsequently establish that $\hat{Q}\hat{\Pi}^\top \approx I$, under a set of sufficient conditions outlined in the next section.

3.3.7 Improved Support Recovery Estimates in Homophilic Models

A sub-class of community model are those satisfying *homophily*. As discussed in Section 1, homophily or the tendency to form edges within the members of the same community has been posited as an important factor in community formation, especially in social settings. Many of the existing learning algorithms (e.g., Chen et al., 2012) require this assumption to provide guarantees in the stochastic block model setting. Moreover, our procedure described below can be easily modified to work in situations where the order of intra-connectivity and inter-connectivity among communities is reversed, i.e., in the community connectivity matrix $P \in [0,1]^{k\times k}$, $P(i,i) \equiv p < P(i,j) \equiv q$, for all $i \neq j$. For instance, in the k-coloring model (McSherry, 2001), $p = 0$ and $q > 0$.

We describe the post-processing method in Procedure 3 for models with community connectivity matrix P satisfying $P(i,i) \equiv p > P(i,j) \equiv q$ for all $i \neq j$. For such models, we can obtain improved estimates by averaging. Specifically, consider nodes in set C and edges going from C to nodes in B. First, consider the special case of the stochastic block model: for each node $c \in C$, compute the number of neighbors in B belonging to each community (as given by the estimate $\hat{\Pi}$ from Algorithm 1), and declare the community with the maximum number of such neighbors as the community of node c. Intuitively, this provides a better estimate for Π_C since we average over the edges in B. This method has been used before in the context of spectral clustering (McSherry, 2001).

The same idea can be extended to the general mixed membership (homophilic) models: declare communities to be significant if they exceed a certain threshold, as evaluated by the average number of edges to each community. The correctness of the procedure can be gleaned from the fact that if the true F matrix is input, it satisfies

$$F_{j,i} = q + \Pi_{i,j}(p - q), \quad \forall\, i \in [k], j \in [n],$$

and if the true P matrix is input, $H = p$ and $L = q$. Thus, under a suitable threshold ξ, the entries $F_{j,i}$ provide information on whether the corresponding community weight $\Pi_{i,j}$ is significant.

In the next section, we establish that in certain regime of parameters, this support recovery procedure can lead to zero-error support recovery of significant community memberships of the nodes and also rule out communities where a node does not have a strong presence.

Procedure 3 $\{\hat{S}\} \leftarrow \text{SupportRecoveryHomophilicModels}(G, k, \alpha_0, \xi, \hat{\Pi})$

Input: Adjacency matrix $G \in \mathbb{R}^{n \times n}$, k is the number of communities, $\alpha_0 := \sum_i \alpha_i$, where α is the Dirichlet parameter vector, ξ is the threshold for support recovery, corresponding to significant community memberships of an individual. Get estimate $\hat{\Pi}$ from Algorithm 1. Also asume the model is homophilic: $P(i,i) \equiv p > P(i,j) \equiv q$, for all $i \neq j$.

Output: $\hat{S} \in \{0,1\}^{n \times k}$ is the estimated support for significant community memberships (see Theorem 7 for guarantees).

Consider partitions A, B, C, X, Y as in Algorithm 1.

Define $\hat{Q}$ on lines of definition in Algorithm 1, using estimates $\hat{\Pi}$. Let the i-th row for set B be $\hat{Q}^i_B := (\alpha_0 + 1)\frac{\hat{\Pi}^i_B}{|\hat{\Pi}^i_B|_1} - \frac{\alpha_0}{n}\vec{1}^\top$. Similarly define $\hat{Q}^i_C$.

Estimate $\hat{F}_C \leftarrow G_{C,B}\hat{Q}_B^\top$, $\hat{P} \leftarrow \hat{Q}_C\hat{F}_C$.

if $\alpha_0 = 0$ (stochastic block model) **then**

  for $x \in C$ **do**

    Let $i^* \leftarrow \arg\max_{i \in [k]} \hat{F}_C(x,i)$ and $\hat{S}(i^*, x) \leftarrow 1$ and 0 o.w.

  end for

else

  Let H be the average of diagonals of $\hat{P}$, L be the average of off-diagonals of $\hat{P}$

  for $x \in C$, $i \in [k]$ **do**

    $\hat{S}(i,x) \leftarrow 1$ if $\hat{F}_C(x,i) \geq L + (H - L) \cdot \frac{3\xi}{4}$ and zero otherwise.{Identify large entries}

  end for

end if

Permute the roles of the sets A, B, C, X, Y to get results for remaining nodes.

3.3.8 Computational Complexity

We note that the computational complexity of the method, implemented naively, is $O(n^2k + k^{4.43}\widehat{\alpha}_{\min}^{-1})$ when $\alpha_0 > 1$ and $O(n^2k)$ when $\alpha_0 < 1$. This is because the time for computing whitening matrices is dominated by SVD of the top k singular vectors of $n \times n$ matrix, which takes $O(n^2k)$ time. We then compute the whitened tensor T which requires time $O(n^2k + k^3n) = O(n^2k)$, since for each $i \in Y$, we multiply $G_{i,A}, G_{i,B}, G_{i,C}$ with the corresponding whitening matrices, and this step takes $O(nk)$ time. We then average this $k \times k \times k$ tensor over different nodes $i \in Y$ to the result, which takes $O(k^3)$ time in each step.

For the tensor power method, the time required for a single iteration is $O(k^3)$. We need at most $\log n$ iterations per initial vector, and we need to consider $O(\widehat{\alpha}_{\min}^{-1}k^{0.43})$ initial vectors (this could be smaller when $\alpha_0 < 1$). Hence the total running time of tensor power method is $O(k^{4.43}\widehat{\alpha}_{\min}^{-1})$ (and when α_0 is small this can be improved to $O(k^4\widehat{\alpha}_{\min}^{-1})$ which is dominated by $O(n^2k)$.

In the process of estimating Π and P, the dominant operation is multiplying $k \times n$ matrix by $n \times n$ matrix, which takes $O(n^2k)$ time. For support recovery, the dominant operation is computing the "average degree", which again takes $O(n^2k)$ time. Thus, we have that the overall computational time is $O(n^2k + k^{4.43}\widehat{\alpha}_{\min}^{-1})$ when $\alpha_0 > 1$ and $O(n^2k)$ when $\alpha_0 < 1$.

Note that the above bound on complexity of our method nearly matches the bound for spectral clustering method (McSherry, 2001), since computing the k-rank SVD requires $O(n^2k)$ time. Another method for learning stochastic block models is based on convex optimization involving semi-definite programming (SDP) (Chen et al., 2012), and it provides the best scaling bounds (for both the network size n and the separation $p - q$ for edge connectivity) known so far. The specific convex problem can be solved via the method of *augmented Lagrange multipliers* (Lin et al., 2010), where each step consists of an SVD operation and q-linear convergence is established by Lin et al. (2010). This implies that the method has complexity $O(n^3)$, since it involves taking SVD of a general $n \times n$ matrix, rather than a k-rank SVD. Thus, our method has significant advantage in terms of computational complexity, when the number of communities is much smaller than the network size ($k \ll n$).

Further, a subsequent work provides a more sophisticated implementation of the proposed tensor method through parallelization and the use of stochastic gradient descent for tensor decomposition (Huang et al., 2013). Additionally, the k-rank SVD operations are approximated via randomized methods such as the Nystrom's method leading to more efficient implementations (Gittens and Mahoney, 2013). Huang et al. (2013) deploy the tensor approach for community detection and establish that it has a running time of $O(n + k^3)$ using nk cores under a parallel computation model (JáJá, 1992).

4. Sample Analysis for Proposed Learning Algorithm

In this section we analyze our algorithm when the moments are estimated from the sample. Unlike common sample complexity analysis, here we are only given one instance of the graph. We treat the edges in the graph as independent samples (conditioned on community membership), and the "sample complexity" will control how many communities we can learn in a graph with n vertices.

4.1 Homogeneous Mixed Membership Models

It is easier to first present the results for our proposed algorithm for the special case, where all the communities have the same expected size and the entries of the community connectivity matrix P are equal on diagonal and off-diagonal locations:

$$\widehat{\alpha}_i \equiv \frac{1}{k}, \qquad P(i,j) = p \cdot \mathbb{I}(i=j) + q \cdot \mathbb{I}(i \neq j), \quad p \geq q. \tag{25}$$

In other words, the probability of an edge according to P only depends on whether it is between two individuals of the same community or between different communities. The above setting is also well studied for stochastic block models ($\alpha_0 = 0$), allowing us to compare our results with existing ones. The results for general mixed membership models are deferred to Section 4.2.

[A1] Sparse regime of Dirichlet parameters: The community membership vectors are drawn from the Dirichlet distribution, $\mathrm{Dir}(\alpha)$, under the mixed membership model. We assume that $\alpha_i < 1$ for $i \in [k]$ (see Section 2.1 for an extended discussion on the sparse regime of the Dirichlet distribution) and that α_0 is known.

[A2] Condition on the network size: Given the concentration parameter of the Dirichlet distribution, $\alpha_0 := \sum_i \alpha_i$, we require that

$$n = \tilde{\Omega}(k^2(\alpha_0+1)^2), \tag{26}$$

and that the disjoint sets A, B, C, X, Y are chosen randomly and are of size $\Theta(n)$. Note that from assumption A1, $\alpha_i < 1$ which implies that $\alpha_0 < k$. Thus, in the worst-case, when $\alpha_0 = \Theta(k)$, we require[13] $n = \tilde{\Omega}(k^4)$, and in the best case, when $\alpha_0 = \Theta(1)$, we require $n = \tilde{\Omega}(k^2)$. The latter case includes the stochastic block model ($\alpha_0 = 0$), and thus, our results match the state-of-art bounds for learning stochastic block models.

[A3] Condition on edge connectivity: Recall that p is the probability of intra-community connectivity and q is the probability of inter-community connectivity. We require that

$$\frac{p-q}{\sqrt{p}} = \Omega\left(\frac{(\alpha_0+1)k}{n^{1/2}}\right) \tag{27}$$

The above condition is on the standardized separation between intra-community and inter-community connectivity (note that $\sqrt{p}$ is the standard deviation of a Bernoulli random variable). The above condition is required to control the perturbation in the whitened tensor (computed using observed network samples), thereby, providing guarantees on the estimated eigen-pairs through the tensor power method.

[A4] Condition on number of iterations of the power method: We assume that the number of iterations N of the tensor power method in Procedure 2 satisfies

$$N \geq C_2 \cdot \left(\log(k) + \log\log\left(\frac{p-q}{p}\right)\right), \tag{28}$$

for some constant C_2.

13. The notation $\tilde{\Omega}(\cdot), \tilde{O}(\cdot)$ denotes $\Omega(\cdot), O(\cdot)$ up to poly-log factors.

[A5] Choice of τ for thresholding community vector estimates: The threshold τ for obtaining estimates $\hat{\Pi}$ of community membership vectors in Algorithm 1 is chosen as

$$\tau = \begin{cases} \Theta\left(\frac{k\sqrt{\alpha_0}}{\sqrt{n}} \cdot \frac{\sqrt{p}}{p-q}\right), & \alpha_0 \neq 0, \quad (29) \\ 0.5, & \alpha_0 = 0, \quad (30) \end{cases}$$

For the stochastic block model ($\alpha_0 = 0$), since π_i is a basis vector, we can use a large threshold. For general models ($\alpha_0 \neq 0$), τ can be viewed as a regularization parameter and decays as $n^{-1/2}$ when other parameters are held fixed. We are now ready to state the error bounds on the estimates of community membership vectors Π and the block connectivity matrix P. $\hat{\Pi}$ and $\hat{P}$ are the estimates computed in Algorithm 1.

Recall that for a matrix M, $(M)^i$ and $(M)_i$ denote the i^{th} row and column respectively. We say that an event holds with high probability, if it occurs with probability $1 - n^{-c}$ for some constant $c > 0$.

Theorem 5 (Guarantees on Estimating P, Π) *Under assumptions A1-A5, we have with high probability*

$$\varepsilon_{\pi,\ell_1} := \max_{i\in[n]} \|\hat{\Pi}^i - \Pi^i\|_1 = \tilde{O}\left(\frac{(\alpha_0+1)^{3/2}\sqrt{np}}{(p-q)}\right) \quad (31)$$

$$\varepsilon_P := \max_{i,j\in[k]} |\hat{P}_{i,j} - P_{i,j}| = \tilde{O}\left(\frac{(\alpha_0+1)^{3/2}k\sqrt{p}}{\sqrt{n}}\right). \quad (32)$$

The proofs are given in the Appendix and a proof outline is provided in Section 4.3.

The main ingredient in establishing the above result is the tensor concentration bound and additionally, recovery guarantees under the tensor power method in Procedure 2. We now provide these results below.

Recall that $F_A := \Pi_A^\top P^\top$ and $\Phi = W_A^\top F_A \operatorname{Diag}(\widehat{\alpha}^{1/2})$ denotes the set of tensor eigenvectors under exact moments in (23), and $\hat{\Phi}$ is the set of estimated eigenvectors under empirical moments, obtained using Procedure 1. We establish the following guarantees.

Lemma 6 (Perturbation bound for estimated eigen-pairs) *Under the assumptions A1-A4, the recovered eigenvector-eigenvalue pairs $(\hat{\Phi}_i, \hat{\lambda}_i)$ from the tensor power method in Procedure 2 satisfies with high probability, for a permutation θ, such that*

$$\max_{i\in[k]} \|\hat{\Phi}_i - \Phi_{\theta(i)}\| \leq 8k^{-1/2}\varepsilon_T, \qquad \max_{i\in[k]} |\lambda_i - \widehat{\alpha}_{\theta(i)}^{-1/2}| \leq 5\varepsilon_T, \quad (33)$$

The tensor perturbation bound ε_T is given by

$$\varepsilon_T := \Big\| \mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}(\hat{W}_A, \hat{W}_B\hat{R}_{AB}, \hat{W}_C\hat{R}_{AC}) - \mathbb{E}[\mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}(W_A, W_BR_{AB}, W_CR_{AC})|\Pi_{A\cup B\cup C}] \Big\| \quad (34)$$

$$= \tilde{O}\left(\frac{(\alpha_0+1)k^{3/2}\sqrt{p}}{(p-q)\sqrt{n}}\right), \quad (35)$$

where $\|T\|$ for a tensor T refers to its spectral norm.

Remark: *(Stochastic Block Models ($\alpha_0 = 0$)) For stochastic block models, assumptions A2 and A3 reduce to*

$$n = \tilde{\Omega}(k^2), \qquad \zeta = \Theta\left(\frac{\sqrt{p}}{p-q}\right) = O\left(\frac{n^{1/2}}{k}\right). \tag{36}$$

This matches with the best known scaling (up to poly-log factors), and was previously achieved via convex optimization by Chen et al. (2012) for stochastic block models. However, our results in Theorem 5 do not provide zero error guarantees as in Chen et al. (2012). We strengthen our results to provide zero-error guarantees in Section 4.1.1 below and thus, match the scaling of Chen et al. (2012) for stochastic block models. Moreover, we also provide zero-error support recovery guarantees for recovering significant memberships of nodes in mixed membership models in Section 4.1.1.

Remark: *(Dependence on α_0) The guarantees degrade as α_0 increases, which is intuitive since the extent of community overlap increases. The requirement for scaling of n also grows as α_0 increases. Note that the guarantees on ε_π and ε_P can be improved by assuming a more stringent scaling of n with respect to α_0, rather than the one specified by A2.*

4.1.1 Zero-error Guarantees for Support Recovery

Recall that we proposed Procedure 3 as a post-processing step to provide improved support recovery estimates. We now provide guarantees for this method.

We now specify the threshold ξ for support recovery in Procedure 3.

[A6] Choice of ξ for support recovery: We assume that the threshold ξ in Procedure 3 satisfies

$$\xi = \Omega(\varepsilon_P),$$

where ε_P is specified in Theorem 5.

We now state the guarantees for support recovery.

Theorem 7 (Support recovery guarantees) *Assuming A1-A6 and (25) hold, the support recovery method in Procedure 3 has the following guarantees on the estimated support set $\hat{S}$: with high probability,*

$$\Pi(i,j) \geq \xi \Rightarrow \hat{S}(i,j) = 1 \quad \text{and} \quad \Pi(i,j) \leq \frac{\xi}{2} \Rightarrow \hat{S}(i,j) = 0, \quad \forall i \in [k], j \in [n], \tag{37}$$

where Π is the true community membership matrix.

Thus, the above result guarantees that the Procedure 3 correctly recovers all the "large" entries of Π and also correctly rules out all the "small" entries in Π. In other words, we can correctly infer all the significant memberships of each node and also rule out the set of communities where a node does not have a strong presence.

The only shortcoming of the above result is that there is a gap between the "large" and "small" values, and for an intermediate set of values (in $[\xi/2, \xi]$), we cannot guarantee correct inferences about the community memberships. Note this gap depends on ε_P, the error in estimating the P matrix. This is intuitive, since as the error ε_P decreases, we can infer the community memberships over a large range of values.

For the special case of stochastic block models (i.e., $\lim \alpha_0 \to 0$), we can improve the above result and give a zero error guarantee at all nodes (w.h.p). Note that we no longer require a threshold ξ in this case, and only infer one community for each node.

Corollary 8 (Zero error guarantee for block models) *Assuming A1-A5 and* (25) *hold, the support recovery method in Procedure 3 correctly identifies the community memberships for all nodes with high probability in case of stochastic block models* ($\alpha_0 \to 0$).

Thus, with the above result, we match the state-of-art results of Chen et al. (2012) for stochastic block models in terms of scaling requirements and recovery guarantees.

4.2 General (Non-homogeneous) Mixed Membership Models

In the previous sections, we provided learning guarantees for learning homogeneous mixed membership models. Here, we extend the results to learning general non-homogeneous mixed membership models under a sufficient set of conditions, involving scaling of various parameters such as network size n, number of communities k, concentration parameter α_0 of the Dirichlet distribution (which is a measure of overlap of the communities) and so on.
[B1] Sparse regime of Dirichlet parameters: The community membership vectors are drawn from the Dirichlet distribution, $\mathrm{Dir}(\alpha)$, under the mixed membership model. We assume that[14] $\alpha_i < 1$ for $i \in [k]$ $\alpha_i < 1$ (see Section 2.1 for an extended discussion on the sparse regime of the Dirichlet distribution).
[B2] Condition on the network size: Given the concentration parameter of the Dirichlet distribution, $\alpha_0 := \sum_i \alpha_i$, and $\widehat{\alpha}_{\min} := \alpha_{\min}/\alpha_0$, the expected size of the smallest community, define

$$\rho := \frac{\alpha_0 + 1}{\widehat{\alpha}_{\min}}. \tag{38}$$

We require that the network size scale as

$$n = \Omega\left(\rho^2 \log^2 k\right), \tag{39}$$

and that the sets A, B, C, X, Y are $\Theta(n)$. Note that from assumption B1, $\alpha_i < 1$ which implies that $\alpha_0 < k$. Thus, in the worst-case, when $\alpha_0 = \Theta(k)$, we require[15] $n = \tilde{\Omega}(k^4)$, assuming equal sizes: $\widehat{\alpha}_i = 1/k$, and in the best case, when $\alpha_0 = \Theta(1)$, we require $n = \tilde{\Omega}(k^2)$. The latter case includes the stochastic block model ($\alpha_0 = 0$), and thus, our results match the state-of-art bounds for learning stochastic block models. See Section 4.1 for an extended discussion.
[B3] Condition on relative community sizes and block connectivity matrix: Recall that $P \in [0,1]^{k \times k}$ denotes the block connectivity matrix. Define

$$\zeta := \left(\frac{\widehat{\alpha}_{\max}}{\widehat{\alpha}_{\min}}\right)^{1/2} \frac{\sqrt{(\max_i (P\widehat{\alpha})_i)}}{\sigma_{\min}(P)}, \tag{40}$$

14. The assumption B1 that the Dirichlet distribution be in the sparse regime is not strictly needed. Our results can be extended to general Dirichlet distributions, but with worse scaling requirements on n. The dependence of n is still polynomial in α_0, i.e., we require $n = \tilde{\Omega}((\alpha_0+1)^c \widehat{\alpha}_{\min}^{-2})$, where $c \geq 2$ is some constant.
15. The notation $\tilde{\Omega}(\cdot), \tilde{O}(\cdot)$ denotes $\Omega(\cdot), O(\cdot)$ up to log factors.

where $\sigma_{\min}(P)$ is the minimum singular value of P. We require that

$$\zeta = \begin{cases} O\left(\frac{n^{1/2}}{\rho}\right), & \alpha_0 < 1 \quad (41) \\ O\left(\frac{n^{1/2}}{\rho k \widehat{\alpha}_{\max}}\right) & \alpha_0 \geq 1. \quad (42) \end{cases}$$

Intuitively, the above condition requires the ratio of maximum and minimum expected community sizes to be not too large and for the matrix P to be well conditioned. The above condition is required to control the perturbation in the whitened tensor (computed using observed network samples), thereby, providing guarantees on the estimated eigen-pairs through the tensor power method. The above condition can be interpreted as a separation requirement between intra-community and inter-community connectivity in the special case considered in Section 4.1. Specifically, for the special case of homogeneous mixed membership model, we have

$$\sigma_{\min}(P) = \Theta(p-q), \quad \max_i (P\widehat{\alpha})_i = \frac{p}{k} + (k-1)\frac{q}{k} \leq p.$$

Thus, the assumptions A2 and A3 in Section 4.1 given by

$$n = \tilde{\Omega}(k^2(\alpha_0+1)^2), \qquad \zeta = \Theta\left(\frac{\sqrt{p}}{p-q}\right) = O\left(\frac{n^{1/2}}{(\alpha_0+1)k}\right)$$

are special cases of the assumptions B2 and B3 above.

[B4] Condition on number of iterations of the power method: We assume that the number of iterations N of the tensor power method in Procedure 2 satisfies

$$N \geq C_2 \cdot \left(\log(k) + \log\log\left(\frac{\sigma_{\min}(P)}{(\max_i (P\widehat{\alpha})_i)}\right)\right), \quad (43)$$

for some constant C_2.

[B5] Choice of τ for thresholding community vector estimates: The threshold τ for obtaining estimates $\hat{\Pi}$ of community membership vectors in Algorithm 1 is chosen as

$$\tau = \begin{cases} \Theta\left(\frac{\rho^{1/2} \cdot \zeta \cdot \widehat{\alpha}_{\max}^{1/2}}{n^{1/2} \cdot \widehat{\alpha}_{\min}}\right), & \alpha_0 \neq 0, \quad (44) \\ 0.5, & \alpha_0 = 0, \quad (45) \end{cases}$$

For the stochastic block model ($\alpha_0 = 0$), since π_i is a basis vector, we can use a large threshold. For general models ($\alpha_0 \neq 0$), τ can be viewed as a regularization parameter and decays as $n^{-1/2}$ when other parameters are held fixed. Moreover, when $n = \tilde{\Theta}(\rho^2)$, we have that $\tau \sim \rho^{-1/2}$ when other terms are held fixed. Recall that $\rho \propto (\alpha_0+1)$ when the expected community sizes $\widehat{\alpha}_i$ are held fixed. In this case, $\tau \sim \rho^{-1/2}$ allows for smaller values to be picked up after thresholding as α_0 is increased. This is intuitive since as α_0 increases, the community vectors π are more "spread out" across different communities and have smaller values.

We are now ready to state the error bounds on the estimates of community membership vectors Π and the block connectivity matrix P. $\hat{\Pi}$ and $\hat{P}$ are the estimates computed in Algorithm 1.

Recall that for a matrix M, $(M)^i$ and $(M)_i$ denote the i^{th} row and column respectively. We say that an event holds with high probability, if it occurs with probability $1 - n^{-c}$ for some constant $c > 0$.

Theorem 9 (Guarantees on estimating P, Π) *Under assumptions B1-B5, The estimates $\hat{P}$ and $\hat{\Pi}$ obtained from Algorithm 1 satisfy with high probability,*

$$\varepsilon_{\pi,\ell_1} := \max_{i\in[k]} |(\hat{\Pi})^i - (\Pi)^i|_1 = \tilde{O}\left(n^{1/2}\cdot\rho^{3/2}\cdot\zeta\cdot\widehat{\alpha}_{\max}\right) \tag{46}$$

$$\varepsilon_P := \max_{i,j\in[n]} |\hat{P}_{i,j} - P_{i,j}| = \tilde{O}\left(n^{-1/2}\cdot\rho^{5/2}\cdot\zeta\cdot\widehat{\alpha}_{\max}^{3/2}\cdot(P_{\max}-P_{\min})\right) \tag{47}$$

The proofs are in Appendix B and a proof outline is provided in Section 4.3.

The main ingredient in establishing the above result is the tensor concentration bound and additionally, recovery guarantees under the tensor power method in Procedure 2. We now provide these results below.

Recall that $F_A := \Pi_A^\top P^\top$ and $\Phi = W_A^\top F_A \operatorname{Diag}(\widehat{\alpha}^{1/2})$ denotes the set of tensor eigenvectors under exact moments in (23), and $\hat{\Phi}$ is the set of estimated eigenvectors under empirical moments, obtained using Procedure 1. We establish the following guarantees.

Lemma 10 (Perturbation bound for estimated eigen-pairs) *Under the assumptions B1-B4, the recovered eigenvector-eigenvalue pairs $(\hat{\Phi}_i, \hat{\lambda}_i)$ from the tensor power method in Procedure 2 satisfies with high probability, for a permutation θ, such that*

$$\max_{i\in[k]} \|\hat{\Phi}_i - \Phi_{\theta(i)}\| \le 8\widehat{\alpha}_{\max}^{1/2}\varepsilon_T, \qquad \max_i |\lambda_i - \widehat{\alpha}_{\theta(i)}^{-1/2}| \le 5\varepsilon_T, \tag{48}$$

The tensor perturbation bound ε_T is given by

$$\begin{aligned}\varepsilon_T &:= \Big\| \mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}(\hat{W}_A, \hat{W}_B\hat{R}_{AB}, \hat{W}_C\hat{R}_{AC}) \\ &\qquad -\mathbb{E}[\mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}(W_A, W_BR_{AB}, W_CR_{AC})|\Pi_{A\cup B\cup C}]\Big\| \qquad (49)\\ &= \tilde{O}\left(\frac{\rho}{\sqrt{n}}\cdot\frac{\zeta}{\widehat{\alpha}_{\max}^{1/2}}\right), \qquad (50)\end{aligned}$$

where $\|T\|$ for a tensor T refers to its spectral norm, ρ is defined in (38) *and ζ in* (40).

4.2.1 Application to Planted Clique Problem

The planted clique problem is a special case of the stochastic block model Condon and Karp (1999), and is arguably the simplest setting for the community problem. Here, a clique of size s is uniformly planted (or placed) in an Erdős-Rényi graph with edge probability 0.5. This can be viewed as a stochastic block model with $k = 2$ communities, where $\widehat{\alpha}_{\min} = s/n$ is the probability of a node being in a clique and $\widehat{\alpha}_{\max} = 1 - s/n$. The connectivity matrix

is $P = [1, q; q, q]$ with $q = 0.5$, since the probability of connectivity within the clique is 1 and the probability of connectivity for any other node pair is 0.5.

Since the planted clique setting has unequal sized communities, the general result in Section 9 is applicable, and we demonstrate how the assumptions $(B1)$-$(B5)$ simplify for the planted clique setting. We have that $\alpha_0 = 0$, since the communities are non-overlapping. For assumption $B2$, we have that

$$\rho = \frac{\alpha_0 + 1}{\widehat{\alpha}_{\min}} = \frac{n}{s}, \quad n = \tilde{\Omega}(\rho^2) \Rightarrow s = \tilde{\Omega}(\sqrt{n}). \tag{51}$$

For assumption $B3$, we have that $\sigma_{\min}(P) = \Theta(1)$ and that $\max_i (P\widehat{\alpha})_i \leq s/n + q \leq 2$, and thus the assumption $B3$ simplifies as

$$\zeta := \left(\frac{\widehat{\alpha}_{\max}}{\widehat{\alpha}_{\min}}\right)^{1/2} \frac{\sqrt{(\max_i (P\widehat{\alpha})_i)}}{\sigma_{\min}(P)} = \tilde{O}\left(\frac{\sqrt{n}}{\rho}\right) \Rightarrow s = \tilde{\Omega}\left(n^{2/3}\right). \tag{52}$$

The condition in (51) that $s = \tilde{\Omega}(n^{1/2})$ matches the computational lower bounds for recovering the clique (Feldman et al., 2012). Unfortunately, the condition in (52) that $s = \tilde{\Omega}\left(n^{2/3}\right)$ is worse. This is required for assumption $(B3)$ to hold, which is needed to ensure the success of the tensor power method. The whitening step is particularly sensitive to the condition number of the matrix to be whitened (i.e., matrices F_A, F_B, F_C in our case and the condition numbers for these matrices depend on the ratio of the community sizes), which results in a weaker guarantee. Thus, our method does not perform very well when the community sizes are drastically different. It remains an open question if our method can be improved in this setting. We conjecture that using "peeling" ideas similar to Ailon et al. (2013), where the communities are recovered one by one can improve our dependence on the ratio of community sizes.

4.3 Proof Outline

We now summarize the main techniques involved in proving Theorem 9. The details are in the Appendix. The main ingredient is the concentration of the adjacency matrix: since the edges are drawn independently conditioned on the community memberships, we establish that the adjacency matrix concentrates around its mean under the stated assumptions. See Appendix C.4 for details. With this in hand, we can then establish concentration of various quantities used by our learning algorithm.

Step 1: Whitening matrices. We first establish concentration bounds on the whitening matrices $\hat{W}_A, \hat{W}_B, \hat{W}_C$ computed using empirical moments, described in Section 3.3.1. With this in hand, we can approximately recover the span of matrix F_A since $\hat{W}_A^\top F \operatorname{Diag}(\widehat{\alpha}_i)^{1/2}$ is a rotation matrix. The main technique employed is the Matrix Bernstein's inequality (Tropp, 2012, thm. 1.4). See Appendix C.2 for details.

Step 2: Tensor concentration bounds. Recall that we use the whitening matrices to obtain a symmetric orthogonal tensor. We establish that the whitened and symmetrized tensor concentrates around its mean. (Note that the empirical third order tensor $T_{X \to A,B,C}$ tends to its expectation conditioned on Π_A, Π_B, Π_C when $|X| \to \infty$). This is done in several stages and we carefully control the tensor perturbation bounds. See Appendix C.1 for details.

Step 3: Tensor power method analysis. We analyze the performance of Procedure 2 under empirical moments. We employ the various improvements, detailed in Section 3.3.3 to establish guarantees on the recovered eigen-pairs. This includes coming up with a condition on the tensor perturbation bound, for the tensor power method to succeed. It also involves establishing that there exist good initializers for the power method among (whitened) neighborhood vectors. This allows us to obtain stronger guarantees for the tensor power method, compared to earlier analysis by Anandkumar et al. (2012b). This analysis is crucial for us to obtain state-of-art scaling bounds for guaranteed recovery (for the special case of stochastic block model). See Appendix A for details.

Step 4: Thresholding of estimated community vectors. In Step 3, we provide guarantees for recovery of each eigenvector in ℓ_2 norm. Direct application of this result only allows us to obtain ℓ_2 norm bounds for row-wise recovery of the community matrix Π. In order to strengthen the result to an ℓ_1 norm bound, we threshold the estimated Π vectors. Here, we exploit the sparsity in Dirichlet draws and carefully control the contribution of weak entries in the vector. Finally, we establish perturbation bounds on P through rather straightforward concentration bound arguments. See Appendix B.2 for details.

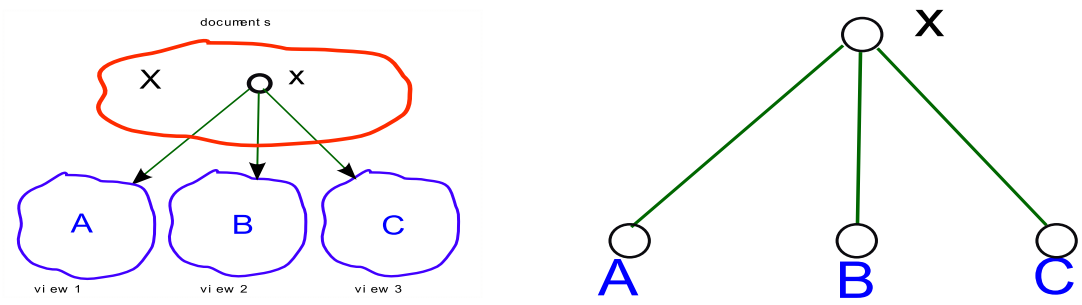
Step 5: Support recovery guarantees. To simplify the argument, consider the stochastic block model. Recall that Procedure 3 readjusts the community membership estimates based on degree averaging. For each vertex, if we count the average degree towards these "approximate communities", for the correct community the result is concentrated around value p and for the wrong community the result is around value q. Therefore, we can correctly identify the community memberships of all the nodes, when $p - q$ is sufficiently large, as specified by A3. The argument can be easily extended to general mixed membership models. See Appendix B.4 for details.

4.4 Comparison with Previous Results

We now compare the results of this paper to our previous work (Anandkumar et al., 2012b) on the use of tensor-based approaches for learning various latent variable models such as topic models, hidden Markov models (HMM) and Gaussian mixtures. At a high level, the tensor approach is exploited in a similar manner in all these models (including the community model in this paper), viz., that the conditional-independence relationships of the model result in a low rank tensor, constructed from low order moments under the given model. However, there are several important differences between the community model and the other latent variable models considered by Anandkumar et al. (2012b) and we list them below. We also precisely list the various algorithmic improvements proposed in this paper with respect to the tensor power method, and how they can be applicable to other latent variable models.

4.4.1 Topic Model vs. Community Model

Among the latent variable models studied by Anandkumar et al. (2012b), the topic model, viz., latent Dirichlet allocation (LDA), bears the closest resemblance to MMSB. In fact, the MMSB model was originally inspired by the LDA model. The analogy between the MMSB model and the LDA is direct under our framework and we describe it below.



(a) Community model as a topic model

(b) Graphical model representation

Figure 2: Casting the community model as a topic model, we obtain conditional independence of the three views.

Recall that for learning MMSBs, we consider a partition of the nodes $\{X, A, B, C\}$ and we consider the set of 3-stars from set X to A, B, C. We can construct an equivalent topic model as follows: the nodes in X form the "documents" and for each document $x \in X$, the neighborhood vectors $G_{xA}^\top, G_{xB}^\top, G_{xC}^\top$ form the three "words" or "views" for that document. In each document $x \in X$, the community vector π_x corresponds to the "topic vector" and the matrices F_A, F_B and F_C correspond to the topic-word matrices. Note that the three views $G_{xA}^\top, G_{xB}^\top, G_{xC}^\top$ are conditionally independent given the topic vector π_x. Thus, the community model can be cast as a topic model or a multi-view model. See Figure 2.

Although the community model can be viewed as a topic model, it has some important special properties which allows us to provide better guarantees. The topic-word matrices F_A, F_B, F_C are not arbitrary matrices. Recall that $F_A := \Pi_A^\top P^\top$ and similarly F_B, F_C are random matrices and we can provide strong concentration bounds for these matrices by appealing to random matrix theory. Moreover, each of the views in the community model has additional structure, viz., the vector $G_{x,A}^\top$ has independent Bernoulli entries conditioned on the community vector π_x, while in a general multi-view model, we only specify the conditional distribution of each view given the hidden topic vector. This further allows us to provide specialized concentration bounds for the community model. Importantly, we can recover the community memberships (or topic vectors) accurately while for a general multi-view model this cannot be guaranteed and we can only hope to recover the model parameters.

4.4.2 Improvements to Tensor Recovery Guarantees in This Paper

In this paper, we make modifications to the tensor power method of Anandkumar et al. (2012b) and obtain better guarantees for the community setting. Recall that the two modifications are adaptive deflation and initialization using whitened neighborhood vectors. The adaptive deflation leads to a weaker gap condition for an initialization vector to succeed in estimating a tensor eigenvector efficiently. Initialization using whitened neighborhood vectors allows us to tolerate more noise in the estimated 3-star tensor, thereby improving our sample complexity result. We make this improvement precise below.

If we directly apply the tensor power method of Anandkumar et al. (2012b), without considering the modifications, we require a stronger condition on the sample complexity and edge connectivity. For simplicity, consider the homogeneous setting of Section 4.1. The conditions $(A2)$ and $(A3)$ now need to be replaced with stronger conditions: **[A2']** **Sample complexity:** The number of samples satisfies

$$n = \tilde{\Omega}(k^4(\alpha_0 + 1)^2).$$

[A3'] **Edge connectivity:** The edge connectivity parameters p, q satisfy

$$\frac{p - q}{\sqrt{p}} = \Omega\left(\frac{(\alpha_0 + 1)k^2}{\sqrt{n}}\right).$$

Thus, we obtain significant improvements in recovery guarantees via algorithmic modifications and careful analysis of concentration bounds.

The guarantees derived in this paper are specific to the community setting, and we outlined previously the special properties of the community model when compared to a general multi-view model. However, when the documents of the topic model are sufficiently long, the word frequency vector within a document has good concentration, and our modified tensor method has better recovery guarantees in this setting as well. Thus, the improved tensor recovery guarantees derived in this paper are applicable in scenarios where we have access to better initialization vectors rather than simple random initialization.

5. Conclusion

In this paper, we presented a novel approach for learning overlapping communities based on a tensor decomposition approach. We established that our method is guaranteed to recover the underlying community memberships correctly, when the communities are drawn from a mixed membership stochastic block model (MMSB). Our method is also computationally efficient and requires simple linear algebraic operations and tensor iterations. Moreover, our method is tight for the special case of the stochastic block model (up to poly-log factors), both in terms of sample complexity and the separation between edge connectivity within a community and across different communities.

We now note a number of interesting open problems and extensions. While we obtained tight guarantees for MMSB models with uniform sized communities, our guarantees are weak when the community sizes are drastically different, such as in the planted clique setting where we do not match the computational lower bound (Feldman et al., 2012). The whitening step in the tensor decomposition method is particularly sensitive to the ratio of community sizes and it is interesting to see if modifications can be made to our algorithm to provide tight guarantees under unequal community sizes. While this paper mostly dealt with the theoretical analysis of the tensor method for community detection, we note recent experimental results where the tensor method is deployed on graphs with millions of nodes with very good accuracy and running times (Huang et al., 2013). In fact, the running times are more than an order of magnitude better than the state-of-art variational approach for learning MMSB models. The work of (Huang et al., 2013) makes an important modification to make the method scalable, viz., that the tensor decomposition is

carried out through stochastic updates in parallel unlike the serial batch updates considered here. Establishing theoretical guarantees for stochastic tensor decomposition is an important problem. Moreover, we have limited ourselves to the MMSB models, which assumes a linear model for edge formation, which is not applicable universally. For instance, exclusionary relationships, where two nodes cannot be connected because of their memberships in certain communities cannot be imposed in the MMSB model. Are there other classes of mixed membership models which do not suffer from this restriction, and yet are identifiable and are amenable for learning? Moreover, the Dirichlet distribution in the MMSB model imposes constraints on the memberships across different communities. Can we incorporate mixed memberships with arbitrary correlations? The answers to these questions will further push the boundaries of tractable learning of mixed membership communities models.

Acknowledgements

We thank the JMLR Action Editor Nathan Srebro and the anonymous reviewers for comments which significantly improved this manuscript. We thank Jure Leskovec for helpful discussions regarding various community models. Part of this work was done when AA, RG, and DH were at MSR New England. AA is supported in part by the Microsoft faculty fellowship, NSF Career award CCF-1254106, NSF Award CCF-1219234 and the ARO YIP Award W911NF-13-1-0084.

Appendix A. Tensor Power Method Analysis

In this section, we leverage on the perturbation analysis for tensor power method in Anandkumar et al. (2012b). As discussed in Section 3.3.3, we propose the following modifications to the tensor power method and obtain guarantees below for the modified method. The two main modifications are: (1) we modify the tensor deflation process in the robust power method in Procedure 2. Rather than a fixed deflation step after obtaining an estimate of the eigenvalue-eigenvector pair, in this paper, we deflate adaptively depending on the current estimate, and (2)rather than selecting random initialization vectors, as in Anandkumar et al. (2012b), we initialize with vectors obtained from adjacency matrix.

Below in Section A.1, we establish success of the modified tensor method under "good" initialization vectors, as defined below. This involves improved error bounds for the modified deflation procedure provided in Section A.2. In Section C.5, we subsequently establish that under the Dirichlet distribution (for small α_0), we obtain "good" initialization vectors.

A.1 Analysis under Good Initialization Vectors

We now show that when "good" initialization vectors are input to tensor power method in Procedure 2, we obtain good estimates of eigen-pairs under appropriate choice of number of iterations N and spectral norm ϵ of tensor perturbation.

Let $T = \sum_{i\in[k]} \lambda_i v_i$, where v_i are orthonormal vectors and $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_k$. Let $\tilde{T} = T + E$ be the perturbed tensor with $\|E\| \leq \epsilon$. Recall that N denotes the number of iterations of the tensor power method.

We call an initialization vector u to be (γ, R_0)-good if there exists v_i such that $\langle u, v_i \rangle > R_0$ and

$$|\langle u, v_i \rangle| - \max_{j<i} |\langle u, v_j \rangle| > \gamma |\langle u, v_i \rangle|. \tag{53}$$

Choose $\gamma = 1/100$.

Theorem 11 *There exists universal constants $C_1, C_2 > 0$ such that the following holds.*

$$\epsilon \leq C_1 \cdot \lambda_{\min} R_0^2, \qquad N \geq C_2 \cdot \left(\log(k) + \log\log\left(\frac{\lambda_{\max}}{\epsilon} \right) \right), \tag{54}$$

Assume there is at least one good initialization vector corresponding to each v_i, $i \in [k]$. The parameter ξ for choosing deflation vectors in each iteration of the tensor power method in Procedure 2 is chosen as $\xi \geq 25\epsilon$. We obtain eigenvalue-eigenvector pairs $(\hat{\lambda}_1, \hat{v}_1), (\hat{\lambda}_2, \hat{v}_2), \ldots, (\hat{\lambda}_k, \hat{v}_k)$ such that there exists a permutation π on $[k]$ with

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

and

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

Remark: *(need for adaptive deflation) We now compare the above result with the result in (Anandkumar et al., 2012b, Thm. 5.1), where similar guarantees are obtained for a simpler version of the tensor power method without any adaptive deflation and using random initialization. The main difference is in our requirement of the gap γ in (53) for an initialization vector is weaker than the gap requirement in (Anandkumar et al., 2012b, Thm. 5.1). This is due to the use of adaptive deflation in this paper.*

Remark: *(need for non-random initialization) In this paper, we employ whitened neighborhood vectors generated under the MMSB model for initialization, while (Anandkumar et al., 2012b, Thm. 5.1) assumes a random initialization. Under random initialization, we obtain $R_0 \sim 1/\sqrt{k}$ (with poly(k) trials), while for initialization using whitened neighborhood vectors, we subsequently establish that $R_0 = \Omega(1)$ is a constant, when number of samples n is large enough. We also establish that the gap requirement in (53) is satisfied for the choice of $\gamma = 1/100$ above. See Lemma 25 for details. Thus, we can tolerate much larger perturbation ϵ of the third order moment tensor, when non-random initializations are employed.*

Proof: The proof is on lines of the proof of (Anandkumar et al., 2012b, Thm. 5.1) but here, we consider the modified deflation procedure, which improves the condition on ϵ in (54). We provide the full proof below for completeness.

We prove by induction on i, the number of eigenpairs estimated so far by Procedure 2. Assume that 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. $D(u, i)$ is the set of deflated vectors given current estimate of the power method is $u \in S^{k-1}$:
$$D(u, i; \xi) := \{j : |\hat{\lambda}_i \hat{\theta}_i| \geq \xi\} \cap [i],$$
where $\hat{\theta}_i := \langle u, \hat{v}_i \rangle$.

3. The error tensor

$$\begin{aligned}\tilde{E}_{i+1,u} &:= \left(\hat{T} - \sum_{j \in D(u,i;\xi)} \hat{\lambda}_j \hat{v}_j^{\otimes 3}\right) - \sum_{j \notin D(u,i;\xi)} \lambda_{\pi(j)} v_{\pi(j)}^{\otimes 3} \\ &= E + \sum_{j \in D(u,i;\xi)} \left(\lambda_{\pi(j)} v_{\pi(j)}^{\otimes 3} - \hat{\lambda}_j \hat{v}_j^{\otimes 3}\right)\end{aligned}$$

satisfies

$$\|\tilde{E}_{i+1,u}(I,u,u)\| \leq 56\epsilon, \quad \forall u \in S^{k-1}; \tag{55}$$

$$\|\tilde{E}_{i+1,u}(I,u,u)\| \leq 2\epsilon, \quad \forall u \in S^{k-1} \text{ s.t. } \exists j \geq i+1 \centerdot (u^\top v_{\pi(j)})^2 \geq 1 - (168\epsilon/\lambda_{\pi(j)})^2. \tag{56}$$

We take $i = 0$ as the base case, so we can ignore the first assertion, and just observe that for $i = 0$, $D(u, 0; \xi) = \emptyset$ and thus

$$\tilde{E}_{1,u} = \hat{T} - \sum_{j=1}^{k} \lambda_i v_i^{\otimes 3} = E, \quad \forall\, u \in S^{k-1}.$$

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. The power iterations now take a subset of $j \in [i]$ for deflation, depending on the current estimate. Set

$$C_1 := \min\left\{(56 \cdot 9 \cdot 102)^{-1}, (100 \cdot 168)^{-1}, \Delta' \text{ from Lemma 12 with } \Delta = 1/50\right\}. \tag{57}$$

For all good initialization vectors which are γ-separated relative to $\pi(j_{\max})$, we have (i) $|\theta_{j_{\max},0}^{(\tau)}| \geq R_0$, and (ii) that by (Anandkumar et al., 2012b, 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 $\gamma_0 \geq 1 - 1/(1+\gamma) \geq 1/101$, thus it follows from the bounds on the other quantities that $\tilde{\epsilon} = 2p\epsilon \leq 56C_1 \cdot \lambda_{\min} R_0^2 < \frac{\gamma_0}{2(1+8\kappa)} \cdot \tilde{\lambda}_{\min} \cdot \theta_{i^*,0}^2$ 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,

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

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 $j \neq j^*$,

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

which in turn implies

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

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

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

Therefore by (Anandkumar et al., 2012b, 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

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

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 $\{\pi(j) : j \leq i-1\}$ (recall $j^* \geq i$).

We now argue that $\tilde{E}_{i+1,u}$ has the required properties to complete the inductive step. By Lemma 12 (using $\tilde{\epsilon} := 5\epsilon$, $\xi = 5\tilde{\epsilon} = 25\epsilon$ and $\Delta := 1/50$, the latter providing one upper bound on C_1 as per (57)), we have for any unit vector $u \in S^{k-1}$,

$$\left\| \left(\sum_{j \leq i} \Big(\lambda_{\pi(j)} v_{\pi(j)}^{\otimes 3} - \hat{\lambda}_j \hat{v}_j^{\otimes 3} \Big) \right) (I, u, u) \right\| \leq \left(1/50 + 100 \sum_{j=1}^{i} (u^\top v_{\pi(j)})^2 \right)^{1/2} 5\epsilon \leq 55\epsilon. \tag{58}$$

Therefore by the triangle inequality,

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

Thus the bound (55) holds.

To prove that (56) holds, for any unit vector $u \in S^{k-1}$ such that there exists $j' \geq i+1$ with $(u^\top v_{\pi(j')})^2 \geq 1 - (168\epsilon/\lambda_{\pi(j')})^2$. We have (via the second bound on C_1 in (57) and the corresponding assumed bound $\epsilon \leq C_1 \cdot \lambda_{\min} R_0^2$)

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

and therefore

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

By the triangle inequality, we have $\|\tilde{E}_{i+1}(I,u,u)\| \leq 2\epsilon$. Therefore (56) holds, so the second assertion of the inductive hypothesis holds. We conclude that by the induction principle, there exists a permutation π such that two assertions hold for $i = k$. From the last induction step ($i = k$), it is also clear from (58) that $\|T - \sum_{j=1}^{k}\hat{\lambda}_j\hat{v}_j^{\otimes 3}\| \leq 55\epsilon$. This completes the proof of the theorem. ■

A.2 Deflation Analysis

Lemma 12 (Deflation analysis) *Let $\tilde{\epsilon} > 0$ and let $\{v_1, \ldots, v_k\}$ be an orthonormal basis for $\mathbb{R}^k$ and $\lambda_i \geq 0$ for $i \in [k]$. Let $\{\hat{v}_1, \ldots, \hat{v}_k\} \in \mathbb{R}^k$ be a set of unit vectors and $\hat{\lambda}_i \geq 0$. Define third order tensor $\mathcal{E}_i$ such that*

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

For some $t \in [k]$ and a unit vector $u \in S^{k-1}$ such that $u = \sum_{i\in[k]}\theta_i v_i$ and $\hat{\theta}_i := \langle u, \hat{v}_i\rangle$, we have for $i \in [t]$,

$$\begin{aligned} |\hat{\lambda}_i\hat{\theta}_i| &\geq \xi \geq 5\tilde{\epsilon}, \\ |\hat{\lambda}_i - \lambda_i| &\leq \tilde{\epsilon}, \\ \|\hat{v}_i - v_i\| &\leq \min\{\sqrt{2},\ 2\tilde{\epsilon}/\lambda_i\}, \end{aligned}$$

then, the following holds

$$\left\|\sum_{i=1}^{t}\mathcal{E}_i(I,u,u)\right\|_2^2 \leq \left(4(5+11\tilde{\epsilon}/\lambda_{\min})^2 + 128(1+\tilde{\epsilon}/\lambda_{\min})^2(\tilde{\epsilon}/\lambda_{\min})^2\right)\tilde{\epsilon}^2\sum_{i=1}^{t}\theta_i^2 + 64(1+\tilde{\epsilon}/\lambda_{\min})^2\tilde{\epsilon}^2 + 2048(1+\tilde{\epsilon}/\lambda_{\min})^2\tilde{\epsilon}^2.$$

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

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

Proof: The proof is on lines of deflation analysis in (Anandkumar et al., 2012b, Lemma B.5), but we improve the bounds based on additional properties of vector u. From Anandkumar et al. (2012b), we have that for all $i \in [t]$, and any unit vector u,

$$\left\|\sum_{i=1}^{t}\mathcal{E}_i(I,u,u)\right\|_2^2 \leq \left(4(5+11\tilde{\epsilon}/\lambda_{\min})^2 + 128(1+\tilde{\epsilon}/\lambda_{\min})^2(\tilde{\epsilon}/\lambda_{\min})^2\right)\tilde{\epsilon}^2\sum_{i=1}^{t}\theta_i^2 + 64(1+\tilde{\epsilon}/\lambda_{\min})^2\tilde{\epsilon}^2\sum_{i=1}^{t}(\tilde{\epsilon}/\lambda_i)^2 + 2048(1+\tilde{\epsilon}/\lambda_{\min})^2\tilde{\epsilon}^2\left(\sum_{i=1}^{t}(\tilde{\epsilon}/\lambda_i)^3\right)^2. \tag{59}$$

Let $\hat{\lambda}_i = \lambda_i + \delta_i$ and $\hat{\theta}_i = \theta_i + \beta_i$. We have $\delta_i \leq \tilde{\epsilon}$ and $\beta_i \leq 2\tilde{\epsilon}/\lambda_i$, and that $|\hat{\lambda}_i \hat{\theta}_i| \geq \xi$.

$$\begin{aligned} ||\hat{\lambda}_i \hat{\theta}_i| - |\lambda_i \theta_i|| &\leq |\hat{\lambda}_i \hat{\theta}_i - \lambda_i \theta_i| \\ &\leq |(\lambda_i + \delta_i)(\theta_i + \beta_i) - \lambda_i \theta_i| \\ &\leq |\delta_i \theta_i + \lambda_i \beta_i + \delta_i \beta_i| \\ &\leq 4\tilde{\epsilon}. \end{aligned}$$

Thus, we have that $|\lambda_i \theta_i| \geq 5\tilde{\epsilon} - 4\tilde{\epsilon} = \tilde{\epsilon}$. Thus $\sum_{i=1}^{t} \tilde{\epsilon}^2/\lambda_i^2 \leq \sum_i \theta_i^2 \leq 1$. Substituting in (59), we have the result. ■

Appendix B. Proof of Theorem 9

We now prove the main results on error bounds claimed in Theorem 9 for the estimated community vectors $\hat{\Pi}$ and estimated block probability matrix $\hat{P}$ in Algorithm 1. Below, we first show that the tensor perturbation bounds claimed in Lemma 10 holds.

Let $\|T\|$ denote the spectral norm for a tensor T (or in special cases a matrix or a vector). Let $\|M\|_F$ denote the Frobenius norm. Let $|M_1|$ denote the operator ℓ_1 norm, i.e., the maximum ℓ_1 norm of its columns and $\|M\|_\infty$ denote the maximum ℓ_1 norm of its rows. Let $\kappa(M)$ denote the condition number, i.e., $\frac{\|M\|}{\sigma_{\min}(M)}$.

B.1 Proof of Lemma 10

From Theorem 11 in Appendix A, we see that the tensor power method returns eigenvalue-vector pair $(\hat{\lambda}_i, \hat{\Phi}_i)$ such that there exists a permutation θ with

$$\max_{i \in [k]} \|\hat{\Phi}_i - \Phi_{\theta(i)}\| \leq 8\widehat{\alpha}_{\max}^{1/2} \varepsilon_T, \tag{60}$$

and

$$\max_i |\lambda_i - \widehat{\alpha}_{\theta(i)}^{-1/2}| \leq 5\varepsilon_T, \tag{61}$$

when the perturbation of the tensor is small enough, according to

$$\varepsilon_T \leq C_1 \widehat{\alpha}_{\max}^{-1/2} r_0^2, \tag{62}$$

for some constant C_1, when initialized with a (γ, r_0) good vector.

With the above result, two aspects need to be established: (1) the whitened tensor perturbation ϵ_T is as claimed, (2) the condition in (62) is satisfied and (3) there exist good initialization vectors when whitened neighborhood vectors are employed. The tensor perturbation bound ϵ_T is established in Theorem 16 in Appendix C.1.

Lemma 25 establishes that when $\zeta = O(\sqrt{n} r_0^2/\rho)$, we have good initialization vectors with Recall $r_0^2 = \Omega(1/\widehat{\alpha}_{\max} k)$ when $\alpha_0 > 1$ and $r_0^2 = \Omega(1)$ for $\alpha_0 \leq 1$, and $\gamma = 1/100$ with probability $1 - 9\delta$ under Dirichlet distribution, when

$$n = \tilde{\Omega}\left(\alpha_{\min}^{-1} k^{0.43} \log(k/\delta)\right), \tag{63}$$

which is satisfied since we assume $\widehat{\alpha}_{\min}^{-2} < n$.

We now show that the condition in (62) is satisfied under the assumptions B1-B4. Since ϵ_T is given by

$$\varepsilon_T = \tilde{O}\left(\frac{\rho}{\sqrt{n}} \cdot \frac{\zeta}{\widehat{\alpha}_{\max}^{1/2}}\right),$$

the condition in (62) is equivalent to $\zeta = O(\sqrt{n} r_0^2/\rho)$. Therefore when $\zeta = O(\sqrt{n} r_0^2/\rho)$, the assumptions of Theorem 11 are satisfied.

B.2 Reconstruction of Π after Tensor Power Method

Let $(M)^i$ and $(M)_i$ denote the i^{th} row and i^{th} column in matrix M respectively. Let $Z \subseteq A^c$ denote any subset of nodes not in A, considered in Procedure LearnPartition Community. Define

$$\tilde{\Pi}_Z := \mathrm{Diag}(\lambda)^{-1} \Phi^\top \hat{W}_A^\top G_{Z,A}^\top. \tag{64}$$

Recall that the final estimate $\hat{\Pi}_Z$ is obtained by thresholding $\tilde{\Pi}_Z$ element-wise with threshold τ in Procedure 1. We first analyze perturbation of $\tilde{\Pi}_Z$.

Lemma 13 (Reconstruction Guarantees for $\tilde{\Pi}_Z$) *Assuming Lemma 10 holds and the tensor power method recovers eigenvectors and eigenvalues up to the guaranteed errors, we have with probability* $1 - 122\delta$,

$$\begin{aligned}\varepsilon_\pi := \max_{i \in Z} \|(\tilde{\Pi}_Z)^i - (\Pi_Z)^i\| &= O\left(\varepsilon_T \widehat{\alpha}_{\max}^{1/2} \left(\frac{\widehat{\alpha}_{\max}}{\widehat{\alpha}_{\min}}\right)^{1/2} \|\Pi_Z\|\right), \\ &= O\left(\rho \cdot \zeta \cdot \widehat{\alpha}_{\max}^{1/2} \left(\frac{\widehat{\alpha}_{\max}}{\widehat{\alpha}_{\min}}\right)^{1/2}\right)\end{aligned}$$

where ε_T *is given by* (72).

Proof: We have $(\tilde{\Pi}_Z)^i = \lambda_i^{-1}((\Phi)_i)^\top \hat{W}_A^\top G_{Z,A}^\top$. We will now use perturbation bounds for each of the terms to get the result.

The first term is

$$\begin{aligned}&\|\mathrm{Diag}(\lambda_i)^{-1} - \mathrm{Diag}(\widehat{\alpha}_i^{1/2})\| \cdot \|\mathrm{Diag}(\widehat{\alpha}^{1/2}) \tilde{F}_A^\top\| \cdot \|\tilde{F}_A\| \cdot \|\Pi_Z\| \\ &\leq 5\varepsilon_T \widehat{\alpha}_{\max} \widehat{\alpha}_{\min}^{-1/2} (1+\varepsilon_1)^2 \|\Pi_Z\|\end{aligned}$$

from the fact that $\|\mathrm{Diag}(\widehat{\alpha}^{1/2}) \tilde{F}_A^\top\| \leq 1 + \varepsilon_1$, where ε_1 is given by (87). The second term is

$$\begin{aligned}&\|\mathrm{Diag}(\widehat{\alpha}^{1/2})\| \cdot \|(\Phi)_i - \widehat{\alpha}_i^{1/2} (\tilde{F}_A)_i\| \cdot \|\tilde{F}_A\| \cdot \|\Pi_Z\| \\ &\leq 8\widehat{\alpha}_{\max} \varepsilon_T \widehat{\alpha}_{\min}^{-1/2} (1+\varepsilon_1) \|\Pi_Z\|\end{aligned}$$

The third term is

$$\begin{aligned}&\|\widehat{\alpha}_i^{1/2}\| \cdot \|(\hat{W}_A^\top - W_A^\top) F_A \Pi_Z\| \\ &\leq \widehat{\alpha}_{\max}^{1/2} \widehat{\alpha}_{\min}^{-1/2} \|\Pi_Z\| \epsilon_W &(65) \\ &\leq O\left(\left(\frac{\widehat{\alpha}_{\max}}{\widehat{\alpha}_{\min}}\right)^{1/2} \varepsilon_T \widehat{\alpha}_{\min}^{1/2} \|\Pi_Z\|\right), &(66)\end{aligned}$$

from Lemma 17 and finally, we have

$$\begin{aligned}
&\|\widehat{\alpha}_i^{1/2}\| \cdot \|W_A\| \cdot \|G_{Z,A}^\top - F_A \Pi_Z\| \\
&\leq O\left(\widehat{\alpha}_{\max}^{1/2} \frac{\sqrt{\alpha_0+1}}{\widehat{\alpha}_{\min}\sigma_{\min}(P)} \sqrt{(\max_i (P\widehat{\alpha})_i)(1+\varepsilon_2+\varepsilon_3)\log\frac{k}{\delta}}\right) \qquad (67)\\
&\leq O\left(\left(\frac{\widehat{\alpha}_{\max}}{\widehat{\alpha}_{\min}}\right)^{1/2} \varepsilon_T \sqrt{\alpha_0+1}(1+\varepsilon_2+\varepsilon_3)\sqrt{\frac{\log k}{\delta}}\right) \qquad (68)
\end{aligned}$$

from Lemma 22 and Lemma 23.

The third term in (66) dominates the last term in (68) since $(\alpha_0+1)\log k/\delta < n\widehat{\alpha}_{\min}$ (due to assumption B2 on scaling of n). ■

We now show that if we threshold the entries of $\tilde{\Pi}_Z$, the the resulting matrix $\hat{\Pi}_Z$ has rows close to those in Π_Z in ℓ_1 norm.

Lemma 14 (Guarantees after thresholding) *For $\hat{\Pi}_Z := \mathrm{Thres}(\tilde{\Pi}_Z, \tau)$, where τ is the threshold, we have with probability $1-2\delta$, that*

$$\begin{aligned}
\varepsilon_{\pi,\ell_1} := \max_{i\in[k]} |(\hat{\Pi}_Z)^i - (\Pi_Z)^i|_1 = O\Bigg(&\sqrt{n\eta}\,\varepsilon_\pi \sqrt{\log\frac{1}{2\tau}}\left(1-\sqrt{\frac{2\log(k/\delta)}{n\eta\log(1/2\tau)}}\right)\\
&+n\eta\tau + \sqrt{(n\eta+4\tau^2)\log\frac{k}{\delta}} + \frac{\varepsilon_\pi^2}{\tau}\Bigg),
\end{aligned}$$

where $\eta = \widehat{\alpha}_{\max}$ when $\alpha_0 < 1$ and $\eta = \alpha_{\max}$ when $\alpha_0 \in [1,k)$.

Remark: *The above guarantee on $\hat{\Pi}_Z$ is stronger than for $\tilde{\Pi}_Z$ in Lemma 13 since this is an ℓ_1 guarantee on the rows compared to ℓ_2 guarantee on rows for $\tilde{\Pi}_Z$.*
Remark: *When τ is chosen as*

$$\tau = \Theta(\frac{\varepsilon_\pi}{\sqrt{n\eta}}) = \Theta\left(\frac{\rho^{1/2}\cdot\zeta\cdot\widehat{\alpha}_{\max}^{1/2}}{n^{1/2}\cdot\widehat{\alpha}_{\min}}\right),$$

we have that

$$\begin{aligned}
\max_{i\in[k]} |(\hat{\Pi}_Z)^i - (\Pi_Z)^i|_1 &= \tilde{O}\left(\sqrt{n\eta}\cdot\varepsilon_\pi\right)\\
&= \tilde{O}\left(n^{1/2}\cdot\rho^{3/2}\cdot\zeta\cdot\widehat{\alpha}_{\max}\right)
\end{aligned}$$

Proof: Let $S_i := \{j : \hat{\Pi}_Z(i,j) > 2\tau\}$. For a vector v, let v_S denote the sub-vector by considering entries in set S. We now have

$$|(\hat{\Pi}_Z)^i - (\Pi_Z)^i|_1 \leq |(\hat{\Pi}_Z)^i_{S_i} - (\Pi_Z)^i_{S_i}|_1 + |(\Pi_Z)^i_{S_i^c}|_1 + |(\hat{\Pi}_Z)^i_{S_i^c}|_1$$

Case $\alpha_0 < 1$**:** From Lemma 26, we have $\mathbb{P}[\Pi(i,j) \geq 2\tau] \leq 8\widehat{\alpha}_i \log(1/2\tau)$. Since $\Pi(i,j)$ are independent for $j \in Z$, we have from multiplicative Chernoff bound (Kearns and Vazirani, 1994, Thm 9.2), that with probability $1-\delta$,

$$\max_{i\in[k]} |S_i| < 8n\widehat{\alpha}_{\max} \log\left(\frac{1}{2\tau}\right)\left(1 - \sqrt{\frac{2\log(k/\delta)}{n\widehat{\alpha}_i \log(1/2\tau)}}\right).$$

We have

$$|(\tilde{\Pi}_Z)^i_{S_i} - (\Pi_Z)^i_{S_i}|_1 \leq \varepsilon_\pi |S_i|^{1/2},$$

and the i^{th} rows of $\tilde{\Pi}_Z$ and $\hat{\Pi}_Z$ can differ on S_i, we have $|\tilde{\Pi}_Z(i,j) - \hat{\Pi}_Z(i,j)| \leq \tau$, for $j \in S_i$, and number of such terms is at most ε_π^2/τ^2. Thus,

$$|(\tilde{\Pi}_Z)^i_{S_i} - (\hat{\Pi}_Z)^i_{S_i}|_1 \leq \frac{\varepsilon_\pi^2}{\tau}.$$

For the other term, from Lemma 26, we have

$$\mathbb{E}[\Pi_Z(i,j) \cdot \delta(\Pi_Z(i,j) \leq 2\tau)] \leq \widehat{\alpha}_i(2\tau).$$

Applying Bernstein's bound we have with probability $1-\delta$

$$\max_{i\in[k]} \sum_{j\in Z} \Pi_Z(i,j) \cdot \delta(\Pi_Z(i,j) \leq 2\tau) \leq n\widehat{\alpha}_{\max}(2\tau) + \sqrt{2(n\widehat{\alpha}_{\max} + 4\tau^2)\log\frac{k}{\delta}}.$$

For $\hat{\Pi}^i_{S_i^c}$, we further divide S_i^c into T_i and U_i, where $T_i := \{j : \tau/2 < \Pi_Z(i,j) \leq 2\tau\}$ and $U_i := \{j : \Pi_Z(i,j) \leq \tau/2\}$.

In the set T_i, using similar argument we know $|(\Pi_Z)^i_{T_i} - (\tilde{\Pi}_Z)^i_{T_i}|_1 \leq O(\varepsilon_\pi\sqrt{n\widehat{\alpha}_{\max}\log 1/\tau})$, therefore

$$|\hat{\Pi}^i_{T_i}|_1 \leq |\tilde{\Pi}^i_{T_i}|_1 \leq |\Pi^i_{T_i} - \tilde{\Pi}^i_{T_i}|_1 + |\Pi^i_{S_i^c}|_1 \leq O(\varepsilon_\pi\sqrt{n\widehat{\alpha}_{\max}\log 1/\tau}).$$

Finally, for index $j \in U_i$, in order for $\hat{\Pi}_Z(i,j)$ be positive, it is required that $\tilde{\Pi}_Z(i,j) - \Pi_Z(i,j) \geq \tau/2$. In this case, we have

$$|(\hat{\Pi}_Z)^i_{U_i}|_1 \leq \frac{4}{\tau}\left\|(\tilde{\Pi}_Z)^i_{U_i} - \Pi^i_{U_i}\right\|^2 \leq \frac{4\varepsilon_\pi^2}{\tau}.$$

Case $\alpha_0 \in [1,k)$**:** From Lemma 26, we see that the results hold when we replace $\widehat{\alpha}_{\max}$ with $\alpha_{\max}$. ■

B.3 Reconstruction of P after Tensor Power Method

Finally we would like to use the community vectors Π and the adjacency matrix G to estimate the P matrix. Recall that in the generative model, we have $\mathbb{E}[G] = \Pi^\top P \Pi$. Thus, a straightforward estimate is to use $(\hat{\Pi}^\dagger)^\top G \hat{\Pi}^\dagger$. However, our guarantees on $\hat{\Pi}$ are not strong enough to control the error on $\hat{\Pi}^\dagger$ (since we only have row-wise ℓ_1 guarantees).

We propose an alternative estimator $\hat{Q}$ for $\hat{\Pi}^\dagger$ and use it to find $\hat{P}$ in Algorithm 1. Recall that the i-th row of $\hat{Q}$ is given by

$$\hat{Q}^i := (\alpha_0+1)\frac{\hat{\Pi}^i}{|\hat{\Pi}^i|_1} - \frac{\alpha_0}{n}\vec{1}^\top.$$

Define Q using exact communities, i.e.,

$$Q^i := (\alpha_0+1)\frac{\Pi^i}{|\Pi^i|_1} - \frac{\alpha_0}{n}\vec{1}^\top.$$

We show below that $\hat{Q}$ is close to $\Pi^\dagger$, and therefore, $\hat{P} := \hat{Q}^\top G \hat{Q}$ is close to P w.h.p.

Lemma 15 (Reconstruction of P) *With probability $1-5\delta$,*

$$\varepsilon_P := \max_{i,j\in[n]} |\hat{P}_{i,j} - P_{i,j}| \leq O\left(\frac{(\alpha_0+1)^{3/2}\varepsilon_\pi(P_{\max}-P_{\min})}{\sqrt{n}}\widehat{\alpha}_{\min}^{-1}\widehat{\alpha}_{\max}^{1/2}\log\frac{nk}{\delta}\right)$$

Remark: *If we define a new matrix Q' as $(Q')^i := \frac{\alpha_0+1}{n\widehat{\alpha}_i}\Pi^i - \frac{\alpha_0}{n}\vec{1}^\top$, then $\mathbb{E}_\Pi[Q'\Pi^\top] = I$. Below, we show that Q' is close to Q since $\mathbb{E}[|\Pi^i|_1] = n\widehat{\alpha}_i$ and thus the above result holds. We require Q to be normalized by $|\Pi^i|_1$ in order to ensure that the first term of Q has equal column norms, which will be used in our proofs subsequently.*

Proof: The proof goes in three steps:

$$P \approx Q\Pi^\top P\Pi Q^\top \approx QGQ^\top \approx \hat{Q}G\hat{Q}^\top.$$

Note that $\mathbb{E}_\Pi[\Pi Q^\top] = I$ and by Bernstein's bound, we can claim that $\Pi Q^\top$ is close to I and can show that the i-th row of $Q\Pi^\top$ satisfies

$$\Delta_i := |(Q\Pi^\top)^i - e_i^\top|_1 = O\left(k\sqrt{\log\left(\frac{nk}{\delta}\right)\frac{\widehat{\alpha}_{\max}}{\widehat{\alpha}_{\min}}}\frac{1}{\sqrt{n}}\right)$$

with probability $1-\delta$. Moreover,

$$\begin{aligned}|(\Pi^\top P\Pi Q^\top)_{i,j} - (\Pi^\top P)_{i,j}| &\leq |(\Pi^\top P)^i((Q)_j - e_j)| = |(\Pi^\top P)^i\Delta_j| \\ &\leq O\left(\frac{P_{\max}k\cdot\sqrt{\widehat{\alpha}_{\max}/\widehat{\alpha}_{\min}}}{\sqrt{n}}\sqrt{\log\frac{nk}{\delta}}\right).\end{aligned}$$

using the fact that $(\Pi^\top P)_{i,j} \leq P_{\max}$.

Now we claim that $\hat{Q}$ is close to Q and it can be shown that

$$|Q^i - \hat{Q}^i|_1 \leq O\left(\frac{\varepsilon_P}{P_{\max}-P_{\min}}\right) \tag{69}$$

Using (69), we have

$$\begin{aligned}|(\Pi^\top P\Pi Q^\top)_{i,j} - (\Pi^\top P\Pi\hat{Q}^\top)_{i,j}| &= |(\Pi^\top P\Pi)^i(Q^\top - \hat{Q}^\top)_j| \\ &= ((\Pi^\top P\Pi)^i - P_{\min}\vec{1}^\top)|(Q^\top - \hat{Q}^\top)_j|_1 \\ &\leq O((P_{\max}-P_{\min})|(Q^\top - \hat{Q}^\top)_j|_1) = O(\varepsilon_P).\end{aligned}$$

using the fact that $(Q^j - \hat{Q}^j)\vec{1} = 0$, due to the normalization.

Finally, $|(G\hat{Q}^\top)_{i,j}(\Pi^\top P \Pi \hat{Q}^\top)_{i,j}|$ are small by standard concentration bounds (and the differences are of lower order). Combining these $|\hat{P}_{i,j} - P_{i,j}| \leq O(\varepsilon_P)$.

■

B.4 Zero-error Support Recovery Guarantees

Recall that we proposed Procedure 3 to provide improved support recovery estimates in the special case of homophilic models (where there are more edges within a community than to any community outside). We limit our analysis to the special case of uniform sized communities ($\alpha_i = 1/k$) and matrix P such that $P(i,j) = p\mathbb{I}(i=j) + q\mathbb{I}(i \neq j)$ and $p \geq q$. In principle, the analysis can be extended to homophilic models with more general P matrix (with suitably chosen thresholds for support recovery).

We first consider analysis for the stochastic block model (i.e., $\alpha_0 \to 0$) and prove the guarantees claimed in Corollary 8.

Proof of Corollary 8: Recall the definition of $\tilde{\Pi}$ in (64) and $\hat{\Pi}$ is obtained by thresholding $\tilde{\Pi}$ with threshold τ. Since the threshold τ for stochastic block models is 0.5 (assumption B5), we have

$$|(\hat{\Pi})^i - (\Pi)^i|_1 = O(\varepsilon_\pi^2), \tag{70}$$

where ε_π is the row-wise ℓ_2 error for $\tilde{\Pi}$ in Lemma 13. This is because $\Pi(i,j) \in \{0,1\}$, and in order for our method to make a mistake, it takes $1/4$ in the ℓ_2^2 error.

In Procedure 3, for the stochastic block model ($\alpha_0 = 0$), for a node $x \in [n]$, we have

$$\hat{F}(x,i) = \sum_{y \in [n]} G_{x,y} \frac{\hat{\Pi}(i,y)}{|\hat{\Pi}^i|_1} \approx \sum_{y \in [n]} G_{x,y} \frac{\hat{\Pi}(i,y)}{|\Pi^i|_1} \approx \frac{k}{n} \sum_{y \in [n]} G_{x,y} \hat{\Pi}(i,y),$$

using (70) and the fact that the size of each community on average is n/k. In other words, for each vertex x, we compute the average number of edges from this vertex to all the estimated communities according to $\hat{\Pi}$, and set it to belong to the one with largest average degree. Note that the margin of error on average for each node to be assigned the correct community according to the above procedure is $(p-q)n/k$, since the size of each community is n/k and the average number of intra-community edges at a node is pn/k and edges to any different community at a node is qn/k. From (70), we have that the average number of errors made is $O((p-q)\varepsilon_\pi^2)$. Note that the degrees concentrate around their expectations according to Bernstein's bound and the fact that the edges used for averaging is independent from the edges used for estimating $\hat{\Pi}$. Thus, for our method to succeed in inferring the correct community at a node, we require,

$$O((p-q)\varepsilon_\pi^2) \leq (p-q)\frac{n}{k},$$

which implies

$$p - q \geq \tilde{\Omega}\left(\frac{\sqrt{p}k}{\sqrt{n}}\right).$$

■

We now prove the general result on support recovery.

Proof of Theorem 7: From Lemma 15,

$$|\hat{P}_{i,j} - P_{i,j}| \leq O(\varepsilon_P)$$

which implies bounds for the average of diagonals H and average of off-diagonals L:

$$|H - p| = O(\varepsilon_P), \quad |L - q| = O(\varepsilon_P).$$

On similar lines as the proof of Lemma 15 and from independence of edges used to define $\hat{F}$ from the edges used to estimate $\hat{\Pi}$, we also have

$$|\hat{F}(j,i) - F(j,i)| \leq O(\varepsilon_P).$$

Note that $F_{j,i} = q + \Pi_{i,j}(p-q)$. The threshold ξ satisfies $\xi = \Omega(\varepsilon_P)$, therefore, all the entries in F that are larger than $q + (p-q)\xi$, the corresponding entries in S are declared to be one, while none of the entries that are smaller than $q + (p-q)\xi/2$ are set to one in S. ■

Appendix C. Concentration Bounds

In this section we prove concentration bounds for the tensors and matrices appeared in the algorithm.

C.1 Main Result: Tensor Perturbation Bound

We now provide the main result that the third-order whitened tensor computed from samples concentrates. Recall that $\mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}$ denotes the third order moment computed using edges from partition Y to partitions A, B, C in (15). $\hat{W}_A, \hat{W}_B\hat{R}_{AB}, \hat{W}_C\hat{R}_{AC}$ are the whitening matrices defined in (24). The corresponding whitening matrices $W_A, W_B R_{AB}, W_C R_{AC}$ for exact moment third order tensor $\mathbb{E}[\mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}\,|\Pi]$ will be defined later. Recall that ρ is defined in (38) as $\rho := \frac{\alpha_0+1}{\widehat{\alpha}_{\min}}$. Given $\delta \in (0,1)$, throughout assume that

$$n = \Omega\left(\rho^2 \log^2 \frac{k}{\delta}\right), \tag{71}$$

as in Assumption $(B2)$.

Theorem 16 (Perturbation of whitened tensor) *When the partitions A, B, C, X, Y satisfy* (71)*, we have with probability $1 - 100\delta$,*

$$\begin{aligned}
\varepsilon_T &:= \left\| \mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}(\hat{W}_A, \hat{W}_B\hat{R}_{AB}, \hat{W}_C\hat{R}_{AC}) - \mathbb{E}[\mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}(W_A, \tilde{W}_B, \tilde{W}_C)|\Pi_A, \Pi_B, \Pi_C] \right\| \\
&= O\left(\frac{(\alpha_0+1)\sqrt{(\max_i (P\widehat{\alpha})_i)}}{n^{1/2}\widehat{\alpha}^{3/2}_{\min}\sigma_{\min}(P)} \cdot \left(1 + \left(\frac{\rho^2}{n}\log^2\frac{k}{\delta}\right)^{1/4}\right)\sqrt{\frac{\log k}{\delta}} \right) \\
&= \tilde{O}\left(\frac{\rho}{\sqrt{n}} \cdot \frac{\zeta}{\widehat{\alpha}^{1/2}_{\max}} \right).
\end{aligned} \tag{72}$$

C.1.1 Proof Overview

The proof of the above result follows. It consists mainly of the following steps: (1) Controlling the perturbations of the whitening matrices and (2) Establishing concentration of the third moment tensor (before whitening). Combining the two, we can then obtain perturbation of the whitened tensor. Perturbations for the whitening step is established in Appendix C.2. Auxiliary concentration bounds required for the whitening step, and for the claims below are in Appendix C.3 and C.4.

Proof of Theorem 16: In tensor T^{α_0} in (15), the first term is

$$(\alpha_0+1)(\alpha_0+2)\sum_{i\in Y}\left(G_{i,A}^\top \otimes G_{i,B}^\top \otimes G_{i,C}^\top\right).$$

We claim that this term dominates in the perturbation analysis since the mean vector perturbation is of lower order. We now consider perturbation of the whitened tensor

$$\Lambda_0 = \frac{1}{|Y|}\sum_{i\in Y}\left((\hat{W}_A^\top G_{i,A}^\top)\otimes(\hat{R}_{AB}^\top \hat{W}_B^\top G_{i,B}^\top)\otimes(\hat{R}_{AC}^\top \hat{W}_C^\top G_{i,C}^\top)\right).$$

We show that this tensor is close to the corresponding term in the expectation in three steps.

First we show it is close to

$$\Lambda_1 = \frac{1}{|Y|}\sum_{i\in Y}\left((\hat{W}_A^\top F_A\pi_i)\otimes(\hat{R}_{AB}^\top \hat{W}_B^\top F_B\pi_i)\otimes(\hat{R}_{AC}^\top \hat{W}_C^\top F_C\pi_i)\right).$$

Then this vector is close to the expectation over Π_Y.

$$\Lambda_2 = \mathbb{E}_{\pi\sim\mathrm{Dir}(\alpha)}\left((\hat{W}_A^\top F_A\pi)\otimes(\hat{R}_{AB}^\top \hat{W}_B^\top F_B\pi)\otimes(\hat{R}_{AC}^\top \hat{W}_C^\top F_C\pi)\right).$$

Finally we replace the estimated whitening matrix $\hat{W}_A$ with W_A, defined in (73), and note that W_A whitens the exact moments.

$$\Lambda_3 = \mathbb{E}_{\pi\sim\mathrm{Dir}(\alpha)}\left((W_A^\top F_A\pi)\otimes(\tilde{W}_B^\top F_B\pi)\otimes(\tilde{W}_C^\top F_C\pi)\right).$$

For $\Lambda_0-\Lambda_1$, the dominant term in the perturbation bound (assuming partitions A,B,C,X,Y are of size n) is (since for any rank 1 tensor, $\|u\otimes v\otimes w\| = \|u\|\cdot\|v\|\cdot\|w\|$),

$$O\left(\frac{1}{|Y|}\|\tilde{W}_B^\top F_B\|^2\left\|\sum_{i\in Y}\left(\hat{W}_A^\top G_{i,A}^\top - \hat{W}_A^\top F_A\pi_i\right)\right\|\right)$$
$$O\left(\frac{1}{|Y|}\widehat{\alpha}_{\min}^{-1}\cdot\frac{(\alpha_0+1)(\max_i(P\widehat{\alpha})_i)}{\widehat{\alpha}_{\min}\sigma_{\min}(P)}\cdot(1+\varepsilon_1+\varepsilon_2+\varepsilon_3)\sqrt{\log\frac{n}{\delta}}\right),$$

with probability $1-13\delta$ (Lemma 18). Since there are 7 terms in the third order tensor T^{α_0}, we have the bound with probability $1-91\delta$.

For $\Lambda_1 - \Lambda_2$, since $\hat{W}_A F_A \operatorname{Diag}(\widehat{\alpha})^{1/2}$ has spectral norm almost 1, by Lemma 20 the spectral norm of the perturbation is at most

$$\left\|\hat{W}_A F_A \operatorname{Diag}(\widehat{\alpha})^{1/2}\right\|^3 \left\|\frac{1}{|Y|}\sum_{i\in Y}(\operatorname{Diag}(\widehat{\alpha})^{-1/2}\pi_i)^{\otimes 3} - \mathbb{E}_{\pi\sim\mathrm{Dir}(\alpha)}(\operatorname{Diag}(\widehat{\alpha})^{-1/2}\pi_i)^{\otimes 3}\right\|$$
$$\leq O\left(\frac{1}{\widehat{\alpha}_{\min}\sqrt{n}}\cdot\sqrt{\log\frac{n}{\delta}}\right).$$

For the final term $\Lambda_2 - \Lambda_3$, the dominating term is

$$(\hat{W}_A - W_A)F_A \operatorname{Diag}(\widehat{\alpha})^{1/2}\|\Lambda_3\| \leq \varepsilon_{W_A}\|\Lambda_3\|$$
$$\leq O\left(\frac{(\alpha_0+1)\sqrt{\max_i(P\widehat{\alpha})_i}}{n^{1/2}\widehat{\alpha}_{\min}^{3/2}\sigma_{\min}(P)}(1+\varepsilon_1+\varepsilon_2+\varepsilon_3)\sqrt{\log\frac{n}{\delta}}\right).$$

Putting all these together, the third term $\|\Lambda_2 - \Lambda_3\|$ dominates. We know with probability at least $1 - 100\delta$, the perturbation in the tensor is at most

$$O\left(\frac{(\alpha_0+1)\sqrt{\max_i(P\widehat{\alpha})_i}}{n^{1/2}\widehat{\alpha}_{\min}^{3/2}\sigma_{\min}(P)}(1+\varepsilon_1+\varepsilon_2+\varepsilon_3)\sqrt{\log\frac{n}{\delta}}\right).$$

■

C.2 Whitening Matrix Perturbations

Consider rank-k SVD of $|X|^{-1/2}(G^{\alpha_0}_{X,A})^\top_{k-svd} = \hat{U}_A\hat{D}_A\hat{V}_A^\top$, and the whitening matrix is given by $\hat{W}_A := \hat{U}_A\hat{D}_A^{-1}$ and thus $|X|^{-1}\hat{W}_A^\top(G^{\alpha_0}_{X,A})^\top_{k-svd}(G^{\alpha_0}_{X,A})_{k-svd}\hat{W}_A = I$. Now consider the singular value decomposition of

$$|X|^{-1}\hat{W}_A^\top\mathbb{E}[(G^{\alpha_0}_{X,A})^\top|\Pi]\cdot\mathbb{E}[(G^{\alpha_0}_{X,A})|\Pi]\hat{W}_A = \Phi\tilde{D}\Phi^\top.$$

$\hat{W}_A$ does not whiten the exact moments in general. On the other hand, consider

$$W_A := \hat{W}_A\Phi_A\tilde{D}_A^{-1/2}\Phi_A^\top. \tag{73}$$

Observe that W_A whitens $|X|^{-1/2}\mathbb{E}[(G^{\alpha_0}_{X,A})|\Pi]$

$$|X|^{-1}W_A^\top\mathbb{E}[(G^{\alpha_0}_{X,A})^\top|\Pi]\mathbb{E}[(G^{\alpha_0}_{X,A})|\Pi]W_A = (\Phi_A\tilde{D}_A^{-1/2}\Phi_A^\top)^\top\Phi_A\tilde{D}_A\Phi_A^\top\Phi_A\tilde{D}_A^{-1/2}\Phi_A^\top = I$$

Now the ranges of W_A and $\hat{W}_A$ may differ and we control the perturbations below.

Also note that $\hat{R}_{A,B}$, $\hat{R}_{A,C}$ are given by

$$\hat{R}_{AB} := |X|^{-1}\hat{W}_B^\top(G^{\alpha_0}_{X,B})^\top_{k-svd}(G^{\alpha_0}_{X,A})_{k-svd}\hat{W}_A. \tag{74}$$
$$R_{AB} := |X|^{-1}W_B^\top\mathbb{E}[(G^{\alpha_0}_{X,B})^\top|\Pi]\cdot\mathbb{E}[G^{\alpha_0}_{X,A}|\Pi]\cdot W_A. \tag{75}$$

Recall ϵ_G is given by (80), and $\sigma_{\min}\left(\mathbb{E}[G^{\alpha_0}_{X,A}|\Pi]\right)$ is given in (23) and $|A| = |B| = |X| = n$.

Lemma 17 (Whitening matrix perturbations) *With probability* $1-\delta$,

$$\epsilon_{W_A} := \|\operatorname{Diag}(\widehat{\alpha})^{1/2} F_A^\top (\hat{W}_A - W_A)\| = O\left(\frac{(1-\varepsilon_1)^{-1/2}\epsilon_G}{\sigma_{\min}\left(\mathbb{E}[G^{\alpha_0}_{X,A}|\Pi]\right)}\right) \tag{76}$$

$$\epsilon_{\tilde{W}_B} := \|\operatorname{Diag}(\widehat{\alpha})^{1/2} F_B^\top (\hat{W}_B \hat{R}_{AB} - W_B R_{AB})\| = O\left(\frac{(1-\varepsilon_1)^{-1/2}\epsilon_G}{\sigma_{\min}\left(\mathbb{E}[G^{\alpha_0}_{X,B}|\Pi]\right)}\right) \tag{77}$$

Thus, with probability $1-6\delta$,

$$\epsilon_{W_A} = \epsilon_{\tilde{W}_B} = O\left(\frac{(\alpha_0+1)\sqrt{\max_i (P\widehat{\alpha})_i}}{n^{1/2}\widehat{\alpha}_{\min}\sigma_{\min}(P)} \cdot (1+\varepsilon_1+\varepsilon_2+\varepsilon_3)\right), \tag{78}$$

where ε_1, ε_2 *and* ε_3 *are given by* (86) *and* (87).

Remark: *Note that when partitions* X, A *satisfy* (71), $\varepsilon_1, \varepsilon_2, \varepsilon_3$ *are small. When* P *is well conditioned and* $\widehat{\alpha}_{\min} = \widehat{\alpha}_{\max} = 1/k$, *we have* $\epsilon_{W_A}, \epsilon_{\tilde{W}_B} = O(k/\sqrt{n})$.

Proof: Using the fact that $W_A = \hat{W}_A \Phi_A \tilde{D}_A^{-1/2} \Phi_A^\top$ or $\hat{W}_A = W_A \Phi_A \tilde{D}_A^{1/2} \Phi_A^\top$ we have that

$$\begin{aligned}
\|\operatorname{Diag}(\widehat{\alpha})^{1/2} F_A^\top (\hat{W}_A - W_A)\| &\le \|\operatorname{Diag}(\widehat{\alpha})^{1/2} F_A^\top W_A (I - \Phi_A \tilde{D}_A^{1/2} \Phi_A^\top)\| \\
&= \|\operatorname{Diag}(\widehat{\alpha})^{1/2} F_A^\top W_A (I - \tilde{D}_A^{1/2})\| \\
&\le \|\operatorname{Diag}(\widehat{\alpha})^{1/2} F_A^\top W_A (I - \tilde{D}_A^{1/2})(I + \tilde{D}_A^{1/2})\| \\
&\le \|\operatorname{Diag}(\widehat{\alpha})^{1/2} F_A^\top W_A\| \cdot \|I - \tilde{D}_A\|
\end{aligned}$$

using the fact that $\tilde{D}_A$ is a diagonal matrix.

Now note that W_A whitens $|X|^{-1/2}\mathbb{E}[G^{\alpha_0}_{X,A}|\Pi] = |X|^{-1/2} F_A \operatorname{Diag}(\alpha^{1/2}) \Psi_X$, where Ψ_X is defined in (85). Further it is shown in Lemma 23 that Ψ_X satisfies with probability $1-\delta$ that

$$\varepsilon_1 := \|I - |X|^{-1} \Psi_X \Psi_X^\top\| \le O\left(\sqrt{\frac{(\alpha_0+1)}{\widehat{\alpha}_{\min}|X|}} \cdot \log\frac{k}{\delta}\right)$$

Since $\varepsilon_1 \ll 1$ when X, A satisfy (71). We have that $|X|^{-1/2}\Psi_X$ has singular values around 1. Since W_A whitens $|X|^{-1/2}\mathbb{E}[G^{\alpha_0}_{X,A}|\Pi]$, we have

$$|X|^{-1} W_A^\top F_A \operatorname{Diag}(\alpha^{1/2}) \Psi_X \Psi_X^\top \operatorname{Diag}(\alpha^{1/2}) F_A^\top W_A = I.$$

Thus, with probability $1-\delta$,

$$\|\operatorname{Diag}(\widehat{\alpha})^{1/2} F_A^\top W_A\| = O((1-\varepsilon_1)^{-1/2}).$$

Let $\mathbb{E}[(G_{X,A}^{\alpha_0})|\Pi] = (G_{X,A}^{\alpha_0})_{k-svd} + \Delta$. We have

$$\begin{aligned}
\|I - \tilde{D}_A\| &= \|I - \Phi_A \tilde{D}_A \Phi_A^\top\| \\
&= \|I - |X|^{-1} \hat{W}_A^\top \mathbb{E}[(G_{X,A}^{\alpha_0})^\top|\Pi] \cdot \mathbb{E}[(G_{X,A}^{\alpha_0})|\Pi] \hat{W}_A\| \\
&= O\left(|X|^{-1} \|\hat{W}_A^\top \left(\Delta^\top (G_{X,A}^{\alpha_0})_{k-svd} + \Delta (G_{X,A}^{\alpha_0})_{k-svd}^\top\right) \hat{W}_A\|\right) \\
&= O\left(|X|^{-1/2} \|\hat{W}_A^\top \Delta^\top \hat{V}_A + \hat{V}_A^\top \Delta \hat{W}_A\|\right), \\
&= O\left(|X|^{-1/2} \|\hat{W}_A\| \|\Delta\|\right) \\
&= O\left(|X|^{-1/2} \|W_A\| \epsilon_G\right),
\end{aligned}$$

since $\|\Delta\| \leq \epsilon_G + \sigma_{k+1}(G_{X,A}^{\alpha_0}) \leq 2\epsilon_G$, using Weyl's theorem for singular value perturbation and the fact that $\epsilon_G \cdot \|W_A\| \ll 1$ and $\|W_A\| = |X|^{1/2}/\sigma_{\min}\left(\mathbb{E}[G_{X,A}^{\alpha_0}|\Pi]\right)$.

We now consider perturbation of $W_B R_{AB}$. By definition, we have that

$$\mathbb{E}[G_{X,B}^{\alpha_0}|\Pi] \cdot W_B R_{AB} = \mathbb{E}[G_{X,A}^{\alpha_0}|\Pi] \cdot W_A.$$

and

$$\|W_B R_{AB}\| = |X|^{1/2} \sigma_{\min}(\mathbb{E}[G_{X,B}^{\alpha_0}|\Pi])^{-1}.$$

Along the lines of previous derivation for ϵ_{W_A}, let

$$|X|^{-1} (\hat{W}_B \hat{R}_{AB})^\top \cdot \mathbb{E}[(G_{X,B}^{\alpha_0})^\top|\Pi] \cdot \mathbb{E}[G_{X,B}^{\alpha_0}|\Pi] \hat{W}_B \hat{R}_{AB} = \Phi_B \tilde{D}_B \Phi_B^\top.$$

Again using the fact that $|X|^{-1}\Psi_X \Psi_X^\top \approx I$, we have

$$\|\operatorname{Diag}(\widehat{\alpha})^{1/2} F_B^\top W_B R_{AB}\| \approx \|\operatorname{Diag}(\widehat{\alpha})^{1/2} F_A^\top W_A\|,$$

and the rest of the proof follows. ■

C.3 Auxiliary Concentration Bounds

Lemma 18 (Concentration of sum of whitened vectors) *Assuming all the partitions satisfy* (71)*, with probability* $1 - 7\delta$*,*

$$\begin{aligned}
\left\|\sum_{i \in Y} \left(\hat{W}_A^\top G_{i,A}^\top - \hat{W}_A^\top F_A \pi_i\right)\right\| &= O(\sqrt{|Y| \widehat{\alpha}_{\max}} \epsilon_{W_A}) \\
&= O\left(\frac{\sqrt{(\alpha_0+1)(\max_i (P\widehat{\alpha})_i)}}{\widehat{\alpha}_{\min} \sigma_{\min}(P)} \cdot (1 + \varepsilon_2 + \varepsilon_3) \sqrt{\log n/\delta}\right), \\
\left\|\sum_{i \in Y} \left((\hat{W}_B \hat{R}_{AB})^\top (G_{i,B}^\top - F_B \pi_i)\right)\right\| &= O\left(\frac{\sqrt{(\alpha_0+1)(\max_i (P\widehat{\alpha})_i)}}{\widehat{\alpha}_{\min} \sigma_{\min}(P)} \cdot (1 + \varepsilon_1 + \varepsilon_2 + \varepsilon_3) \sqrt{\log n/\delta}\right).
\end{aligned}$$

Remark: *Note that when P is well conditioned and $\widehat{\alpha}_{\min} = \widehat{\alpha}_{\max} = 1/k$, we have the above bounds as $O(k)$. Thus, when it is normalized with $1/|Y| = 1/n$, we have the bound as $O(k/n)$.*

Proof: Note that $\hat{W}_A$ is computed using partition X and $G_{i,A}$ is obtained from $i \in Y$. We have independence for edges across different partitions X and Y. Let $\Xi_i := \hat{W}_A^\top (G_{i,A}^\top - F_A \pi_i)$.Applying matrix Bernstein's inequality to each of the variables, we have

$$\begin{aligned}\|\Xi_i\| &\leq \|\hat{W}_A\| \cdot \|G_{i,A}^\top - F_A \pi_i\| \\ &\leq \|\hat{W}_A\| \sqrt{\|F_A\|_1},\end{aligned}$$

from Lemma 22. The variances are given by

$$\begin{aligned}\|\sum_{i\in Y} \mathbb{E}[\Xi_i \Xi_i^\top | \Pi]\| &\leq \sum_{i\in Y} \hat{W}_A^\top \operatorname{Diag}(F_A \pi_i) \hat{W}_A, \\ &\leq \|\hat{W}_A\|^2 \|F_Y\|_1 \\ &= O\left(\frac{|Y|}{|A|} \cdot \frac{(\alpha_0+1)(\max_i (P\widehat{\alpha})_i)}{\widehat{\alpha}_{\min}^2 \sigma_{\min}^2(P)} \cdot (1+\varepsilon_2+\varepsilon_3) \right),\end{aligned}$$

with probability $1-2\delta$ from (83) and (84), and $\varepsilon_2, \varepsilon_3$ are given by (87). Similarly, $\|\sum_{i\in Y} \mathbb{E}[\Xi_i^\top \Xi_i | \Pi]\| \leq \|\hat{W}_A\|^2 \|F_Y\|_1$. Thus, from matrix Bernstein's inequality, we have with probability $1-3\delta$

$$\begin{aligned}\|\sum_{i\in Y} \Xi_i\| &= O(\|\hat{W}_A\| \sqrt{\max(\|F_A\|_1, \|F_X\|_1)}). \\ &= O\left(\frac{\sqrt{(\alpha_0+1)(\max_i (P\widehat{\alpha})_i)}}{\widehat{\alpha}_{\min} \sigma_{\min}(P)} \cdot (1+\varepsilon_2+\varepsilon_3) \sqrt{\log n/\delta} \right)\end{aligned}$$

On similar lines, we have the result for B and C, and also use the independence assumption on edges in various partitions. ■

We now show that not only the sum of whitened vectors concentrates, but that each individual whitened vector $\hat{W}_A^\top G_{i,A}^\top$ concentrates, when A is large enough.

Lemma 19 (Concentration of a random whitened vector) *Conditioned on π_i, with probability at least $1/4$,*

$$\left\| \hat{W}_A^\top G_{i,A}^\top - W_A^\top F_A \pi_i \right\| \leq O(\varepsilon_{W_A} \widehat{\alpha}_{\min}^{-1/2}) = \tilde{O}\left(\frac{\sqrt{(\alpha_0+1)(\max_i (P\widehat{\alpha})_i)}}{n^{1/2} \widehat{\alpha}_{\min}^{3/2} \sigma_{\min}(P)} \right).$$

Remark: *The above result is not a high probability event since we employ Chebyshev's inequality to establish it. However, this is not an issue for us, since we will employ it to show that out of $\Theta(n)$ whitened vectors, there exists at least one good initialization vector corresponding to each eigen-direction, as required in Theorem 11 in Appendix A. See Lemma 25 for details.*

Proof We have

$$\left\| \hat{W}_A^\top G_{i,A}^\top - W_A^\top F_A \pi_i \right\| \leq \left\| (\hat{W}_A - W_A)^\top F_A \pi_i \right\| + \left\| \hat{W}_A^\top (G_{i,A}^\top - F_A \pi_i) \right\|.$$

The first term is satisfies satisfies with probability $1 - 3\delta$

$$\begin{aligned}\|(\hat{W}_A^\top - W_A^\top)F_A\pi_i\| &\leq \epsilon_{W_A}\widehat{\alpha}_{min}^{-1/2} \\ &= O\left(\frac{(\alpha_0+1)\widehat{\alpha}_{\max}^{1/2}\sqrt{(\max_i(P\widehat{\alpha})_i)}}{n^{1/2}\widehat{\alpha}_{\min}^{3/2}\sigma_{\min}(P)} \cdot (1+\varepsilon_1+\varepsilon_2+\varepsilon_3)\right)\end{aligned}$$

Now we bound the second term. Note that $G_{i,A}^\top$ is independent of $\hat{W}_A^\top$, since they are related to disjoint subset of edges. The whitened neighborhood vector can be viewed as a sum of vectors:

$$\hat{W}_A^\top G_{i,A}^\top = \sum_{j\in A} G_{i,j}(\hat{W}_A^\top)_j = \sum_{j\in A} G_{i,j}(\hat{D}_A\hat{U}_A^\top)_j = \hat{D}_A \sum_{j\in A} G_{i,j}(\hat{U}_A^\top)_j.$$

Conditioned on π_i and F_A, $G_{i,j}$ are Bernoulli variables with probability $(F_A\pi_i)_j$. The goal is to compute the variance of the sum, and then use Chebyshev's inequality noted in Proposition 32.

Note that the variance is given by

$$\|\mathbb{E}[(G_{i,A}^\top - F_A\pi_i)^\top \hat{W}_A\hat{W}_A^\top(G_{i,A}^\top - F_A\pi_i)]\| \leq \|\hat{W}_A\|^2 \sum_{j\in A}(F_A\pi_i)_j \left\|(\hat{U}_A^\top)_j\right\|^2.$$

We now bound the variance. By Wedin's theorem, we know the span of columns of $\hat{U}_A$ is $O(\epsilon_G/\sigma_{\min}(G_X^{\alpha_0}, A)) = O(\epsilon_{W_A})$ close to the span of columns of F_A. The span of columns of F_A is the same as the span of rows in Π_A. In particular, let $Proj_\Pi$ be the projection matrix of the span of rows in Π_A, we have

$$\left\|\hat{U}_A\hat{U}_A^\top - Proj_\Pi\right\| \leq O(\epsilon_{W_A}).$$

Using the spectral norm bound, we have the Frobenius norm

$$\left\|\hat{U}_A\hat{U}_A^\top - Proj_\Pi\right\|_F \leq O(\epsilon_{W_A}\sqrt{k})$$

since they are rank k matrices. This implies that

$$\sum_{j\in A}\left(\left\|(\hat{U}_A^\top)_j\right\| - \left\|Proj_\Pi^j\right\|\right)^2 = O(\epsilon_{W_A}^2 k).$$

Now

$$\|Proj_\Pi^j\| \leq \frac{\|\pi_j\|}{\sigma_{\min}(\Pi_A)} = O\left(\sqrt{\frac{(\alpha_0+1)}{n\widehat{\alpha}_{\min}}}\right),$$

from Lemma 23

Now we can bound the variance of the vectors $\sum_{j\in A} G_{i,j}(\hat{U}_A^\top)_j$, since the variance of $G_{i,j}$ is bounded by $(F_A\pi_i)_j$ (its probability), and the variance of the vectors is at most

$$\begin{aligned}\sum_{j\in A}(F_A\pi_i)_j\left\|(\hat{U}_A^\top)_j\right\|^2 &\le 2\sum_{j\in A}(F_A\pi_i)_j\left\|Proj_\Pi^j\right\|^2 + 2\sum_{j\in A}(F_A\pi_i)_j\left(\left\|(\hat{U}_A^\top)_j\right\| - \left\|Proj_\Pi^j\right\|\right)^2\\ &\le 2\sum_{j\in A}(F_A\pi_i)_j\max_{j\in A}\left(\left\|Proj_\Pi^j\right\|^2\right) + \max_{i,j}P_{i,j}\sum_{j\in A}\left(\left\|(\hat{U}_A^\top)_j\right\| - \left\|Proj_\Pi^j\right\|\right)^2\\ &\le O\left(\frac{|F_A|_1(\alpha_0+1)}{n\widehat{\alpha}_{\min}}\right)\end{aligned}$$

Now Chebyshev's inequality implies that with probability at least 1/4 (or any other constant),

$$\left\|\sum_{j\in A}(G_{i,j}-F_A\pi_i)(\hat{U}_A^\top)_j\right\|^2 \le O\left(\frac{|F_A|_1(\alpha_0+1)}{n\widehat{\alpha}_{\min}}\right).$$

And thus, we have

$$\hat{W}_A^\top(G_{i,A}-F_A\pi_i) \le \sqrt{\frac{|F_A|_1(\alpha_0+1)}{n\widehat{\alpha}_{\min}}}\cdot\left\|\hat{W}_A^\top\right\| \le O\left(\epsilon_{W_A}\widehat{\alpha}_{\min}^{-1/2}\right).$$

Combining the two terms, we have the result. ■

Finally, we establish the following perturbation bound between empirical and expected tensor under the Dirichlet distribution, which is used in the proof of Theorem 16.

Lemma 20 (Concentration of third moment tensor under Dirichlet distribution) *With probability* $1-\delta$, *for* $\pi_i \overset{iid}{\sim} \mathrm{Dir}(\alpha)$,

$$\begin{aligned}\left\|\frac{1}{|Y|}\sum_{i\in Y}(\mathrm{Diag}(\widehat{\alpha})^{-1/2}\pi_i)^{\otimes 3} - \mathbb{E}_{\pi\sim\mathrm{Dir}(\alpha)}(\mathrm{Diag}(\widehat{\alpha})^{-1/2}\pi)^{\otimes 3}\right\| &\le O\left(\cdot\frac{1}{\widehat{\alpha}_{\min}\sqrt{n}}\sqrt{\log\frac{n}{\delta}}\right)\\ &= \tilde{O}\left(\frac{1}{\widehat{\alpha}_{min}\sqrt{n}}\right)\end{aligned}$$

Proof The spectral norm of this tensor cannot be larger than the spectral norm of a $k\times k^2$ matrix that we obtain be "collapsing" the last two dimensions (by definitions of norms). Let $\phi_i := \mathrm{Diag}(\hat{\alpha})^{-1/2}\pi_i$ and the "collapsed" tensor is the matrix $\phi_i(\phi_i\otimes\phi_i)^\top$ (here we view $\phi_i\otimes\phi_i$ as a vector in $\mathbb{R}^{k^2}$). We apply Matrix Bernstein on the matrices $Z_i = \phi_i(\phi_i\otimes\phi_i)^\top$. Now

$$\left\|\sum_{i\in Y}\mathbb{E}[Z_iZ_i^\top]\right\| \le |Y|\max\|\phi\|^4\left\|\mathbb{E}[\phi\phi^\top]\right\| \le |Y|\widehat{\alpha}_{\min}^{-2}$$

since $\left\|\mathbb{E}[\phi\phi^\top]\right\| \le 2$. For the other variance term $\left\|\sum_{i\in Y}\mathbb{E}[Z_i^\top Z_i]\right\|$, we have

$$\left\|\sum_{i\in Y}\mathbb{E}[Z_i^\top Z_i]\right\| \le |Y|\widehat{\alpha}_{\min}\left\|\mathbb{E}[(\phi\otimes\phi)(\phi\otimes\phi)^\top]\right\|.$$

It remains to bound the norm of $\mathbb{E}[(\phi \otimes \phi)(\phi \otimes \phi)^\top]$. We have

$$\|\mathbb{E}[(\phi \otimes \phi)(\phi \otimes \phi)^\top]\| = \sup \left(\|\mathbb{E}[M^2]\|,\ s.t.\, M = \sum_{i,j} N_{i,j} \phi_i \phi_j^\top,\ \|N\|_F = 1 \right).$$

by definition. We now group the terms of $\mathbb{E}[M^2]$ and bound them separately.

$$\begin{aligned} M^2 = & \sum_i N_{i,i}^2 \phi_i \phi_i^\top \|\phi_i\|^2 + \sum_{i \neq j} N_{i,j}^2 \phi_i \phi_j^\top \langle \phi_i, \phi_j \rangle \\ & + \sum_{i \neq j \neq a} N_{i,i} N_{j,a} \phi_i \phi_a^\top \langle \phi_i, \phi_j \rangle + \sum_{i \neq j \neq a \neq b} N_{i,j} N_{a,b} \phi_i \phi_b^\top \langle \phi_j, \phi_a \rangle \end{aligned} \tag{79}$$

We bound the terms individually now.

$\|\phi(i)\|^4$ terms: By properties of Dirichlet distribution we know

$$\mathbb{E}[\|\phi(i)\|^4] = \Theta(\widehat{\alpha}_i^{-1}) \leq O(\widehat{\alpha}_{\min}^{-1}).$$

Thus, for the first term in (79), we have

$$\sup_{N:\|N\|_F=1} \| \sum_i \mathbb{E}[N_{i,i}^2 \phi_i \phi_i^\top \|\phi_i\|]\|^2\| = O(\widehat{\alpha}_{\min}^{-1}).$$

$\|\phi(i)\|^3 \cdot \|\phi(j)\|$ terms: We have

$$\|\mathbb{E}[\sum_{i,j} N_{i,i} N_{i,j} \phi(i)^3 \phi(j)]\| \leq \mathbb{E}[\|\phi_i\|^2 \cdot \|\phi_j\|] \leq O(\sqrt{\sum_{i,j} (N_{i,i}^2 \hat{\alpha}(j)) \sum_{i,j} N_{i,j}^2 \hat{\alpha}(i)^{-1}}) \leq O(\widehat{\alpha}_{\min}^{-1/2}).$$

$\|\phi(i)\|^2 \cdot \|\phi(j)\|^2$ terms: the total number of such terms is $O(k^2)$ and we have

$$\mathbb{E}[\|\phi(i)\|^2 \cdot \|\phi(j)\|^2] = \Theta(1),$$

and thus the Frobenius norm of these set of terms is smaller than $O(k)$

$\|\phi(i)\|^2 \cdot \|\phi(j)\| \cdot \|\phi(a)\|$ terms: there are $O(k^3)$ such terms, and we have

$$\|\mathbb{E}[\phi(i)\|^2 \cdot \|\phi(j)\| \cdot \|\phi(a)]\| = \Theta(\hat{\alpha}(i_2)^{1/2} \hat{\alpha}(i_3)^{1/2}).$$

The Frobenius norm of this part of matrix is bounded by

$$O\left(\sqrt{\sum_{i,j,a \in [k]} \hat{\alpha}(j) \hat{\alpha}(a)} \right) \leq O(\sqrt{k}) \sqrt{\sum_j \sum_a \widehat{\alpha}_j \widehat{\alpha}_a} \leq O(\sqrt{k}).$$

the rest: the sum is

$$\mathbb{E}[\sum_{i \neq j \neq a \neq b} N_{i,j} N_{a,b} \hat{\alpha}(i)^{1/2} \hat{\alpha}(j)^{1/2} \hat{\alpha}(a)^{1/2} \hat{\alpha}(b)^{1/2}].$$

It is easy to break the bounds into the product of two sums ($\sum_{i,j}$ and $\sum_{a,b}$) and then bound each one by Cauchy-Schwartz, the result is 1.

Hence the variance term in Matrix Bernstein's inequality can be bounded by $\sigma^2 \leq O(n\widehat{\alpha}_{\min}^{-2})$, each term has norm at most $\widehat{\alpha}_{\min}^{-3/2}$. When $\widehat{\alpha}_{\min}^{-2} < n$ we know the variance term dominates and the spectral norm of the difference is at most $O(\widehat{\alpha}_{\min}^{-1} n^{-1/2}\sqrt{\log n/\delta})$ with probability $1-\delta$.

■

C.4 Basic Results on Spectral Concentration of Adjacency Matrix

Let $n := \max(|A|, |X|)$.

Lemma 21 (Concentration of $G_{X,A}^{\alpha_0}$) *When $\pi_i \sim \mathrm{Dir}(\alpha)$, for $i \in V$, with probability $1-4\delta$,*

$$\epsilon_G := \|G_{X,A}^{\alpha_0} - \mathbb{E}[(G_{X,A}^{\alpha_0})^\top|\Pi]\| = O\left(\sqrt{(\alpha_0+1)n \cdot (\max_i (P\widehat{\alpha})_i)(1+\varepsilon_2)\log\frac{n}{\delta}}\right) \tag{80}$$

Proof: From definition of $G_{X,A}^{\alpha_0}$, we have

$$\epsilon_G \leq \sqrt{\alpha_0+1}\|G_{X,A} - \mathbb{E}[G_{X,A}|\Pi]\| + (\sqrt{\alpha_0+1}-1)\sqrt{|X|}\|\mu_{X,A} - \mathbb{E}[\mu_{X,A}|\Pi]\|.$$

We have concentration for $\mu_{X,A}$ and adjacency submatrix $G_{X,A}$ from Lemma 22. ■

We now provide concentration bounds for adjacency sub-matrix $G_{X,A}$ from partition X to A and the corresponding mean vector. Recall that $\mathbb{E}[\mu_{X\to A}|F_A, \pi_X] = F_A\pi_X$ and $\mathbb{E}[\mu_{X\to A}|F_A] = F_A\widehat{\alpha}$.

Lemma 22 (Concentration of adjacency submatrices) *When $\pi_i \overset{iid}{\sim} \mathrm{Dir}(\alpha)$ for $i \in V$, with probability $1-2\delta$,*

$$\|G_{X,A} - \mathbb{E}[G_{X,A}|\Pi]\| = O\left(\sqrt{n \cdot (\max(\max_i (P\widehat{\alpha})_i, \max_i (P^\top\widehat{\alpha})_i))(1+\varepsilon_2)\log\frac{n}{\delta}}\right). \tag{81}$$

$$\|\mu_A - \mathbb{E}[\mu_A|\Pi]\| = O\left(\frac{1}{|X|}\sqrt{n \cdot (\max(\max_i (P\widehat{\alpha})_i, \max_i (P^\top\widehat{\alpha})_i))(1+\varepsilon_2)\log\frac{n}{\delta}}\right), \tag{82}$$

where ε_2 is given by (87).

Proof: Recall $\mathbb{E}[G_{X,A}|\Pi] = F_A\Pi_X$ and $G_{A,X} = \mathrm{Ber}(F_A\Pi_X)$ where $\mathrm{Ber}(\cdot)$ denotes the Bernoulli random matrix with independent entries. Let

$$Z_i := (G_{i,A}^\top - F_A\pi_i)e_i^\top.$$

We have $G_{X,A}^\top - F_A\Pi_X = \sum_{i\in X} Z_i$. We apply matrix Bernstein's inequality.

We compute the variances $\sum_i \mathbb{E}[Z_iZ_i^\top|\Pi]$ and $\sum_i \mathbb{E}[Z_i^\top Z_i|\Pi]$. We have that $\sum_i \mathbb{E}[Z_iZ_i^\top|\Pi]$ only the diagonal terms are non-zero due to independence of Bernoulli variables, and

$$\mathbb{E}[Z_iZ_i^\top|\Pi] \leq \mathrm{Diag}(F_A\pi_i) \tag{83}$$

entry-wise. Thus,

$$\begin{aligned}\|\sum_{i\in X}\mathbb{E}[Z_iZ_i^\top|\Pi]\| &\le \max_{a\in A}\sum_{i\in X,b\in[k]}F_A(a,b)\pi_i(b)\\ &= \max_{a\in A}\sum_{i\in X,b\in[k]}F_A(a,b)\Pi_X(b,i)\\ &\le \max_{c\in[k]}\sum_{i\in X,b\in[k]}P(b,c)\Pi_X(b,i)\\ &= \|P^\top\Pi_X\|_\infty. \end{aligned} \tag{84}$$

Similarly $\|\sum_{i\in X}\mathbb{E}[Z_i^\top Z_i]\| = \sum_{i\in X}\mathrm{Diag}(\mathbb{E}[\|G_{i,A}^\top - F_A\pi_i\|^2]) \le \|P^\top\Pi_X\|_\infty$. On lines of Lemma 27, we have $\|P^\top\Pi_X\|_\infty = O(|X|\cdot(\max_i(P^\top\widehat{\alpha})_i))$ when $|X|$ satisfies (71).

We now bound $\|Z_i\|$. First note that the entries in $G_{i,A}$ are independent and we can use the vector Bernstein's inequality to bound $\|G_{i,A}-F_A\pi_i\|$. We have $\max_{j\in A}|G_{i,j}-(F_A\pi_i)_j| \le 2$ and $\sum_j\mathbb{E}[G_{i,j}-(F_A\pi_i)_j]^2 \le \sum_j(F_A\pi_i)_j \le \|F_A\|_1$. Thus with probability $1-\delta$, we have

$$\|G_{i,A}-F_A\pi_i\| \le (1+\sqrt{8\log(1/\delta)})\sqrt{\|F_A\|_1} + 8/3\log(1/\delta).$$

Thus, we have the bound that $\|\sum_i Z_i\| = O(\max(\sqrt{\|F_A\|_1}, \sqrt{\|P^\top\Pi_X\|_\infty}))$. The concentration of the mean term follows from this result. ■

We now provide spectral bounds on $\mathbb{E}[(G_{X,A}^{\alpha_0})^\top|\Pi]$. Define

$$\psi_i := \mathrm{Diag}(\hat{\alpha})^{-1/2}(\sqrt{\alpha_0+1}\pi_i - (\sqrt{\alpha_0+1}-1)\mu). \tag{85}$$

Let Ψ_X be the matrix with columns ψ_i, for $i\in X$. We have

$$\mathbb{E}[(G_{X,A}^{\alpha_0})^\top|\Pi] = F_A\,\mathrm{Diag}(\hat{\alpha})^{1/2}\Psi_X,$$

from definition of $\mathbb{E}[(G_{X,A}^{\alpha_0})^\top|\Pi]$.

Lemma 23 (Spectral bounds) *With probability* $1-\delta$,

$$\varepsilon_1 := \|I - |X|^{-1}\Psi_X\Psi_X^\top\| \le O\left(\sqrt{\frac{(\alpha_0+1)}{\widehat{\alpha}_{\min}|X|}}\cdot\log\frac{k}{\delta}\right) \tag{86}$$

With probability $1-2\delta$,

$$\begin{aligned}\|\mathbb{E}[(G_{X,A}^{\alpha_0})^\top|\Pi]\| &= O\left(\|P\|\widehat{\alpha}_{\max}\sqrt{|X||A|(1+\varepsilon_1+\varepsilon_2)}\right)\\ \sigma_{\min}\left(\mathbb{E}[(G_{X,A}^{\alpha_0})^\top|\Pi]\right) &= \Omega\left(\widehat{\alpha}_{\min}\sqrt{\frac{|A||X|}{\alpha_0+1}(1-\varepsilon_1-\varepsilon_3)}\cdot\sigma_{\min}(P)\cdot\right),\end{aligned}$$

where

$$\varepsilon_2 := O\left(\left(\frac{1}{|A|\widehat{\alpha}_{\max}^2}\log\frac{k}{\delta}\right)^{1/4}\right),\quad \varepsilon_3 := O\left(\left(\frac{(\alpha_0+1)^2}{|A|\widehat{\alpha}_{\min}^2}\log\frac{k}{\delta}\right)^{1/4}\right). \tag{87}$$

Remark: *When partitions X, A satisfy* (71)*, $\varepsilon_1, \varepsilon_2, \varepsilon_3$ are small.*
Proof: Note that ψ_i is a random vector with norm bounded by $O(\sqrt{(\alpha_0+1)/\widehat{\alpha}_{min}})$ from Lemma 27 and $\mathbb{E}[\psi_i \psi_i^\top] = I$. We now prove (86). using Matrix Bernstein Inequality. Each matrix $\psi_i\psi_i^\top/|X|$ has spectral norm at most $O((\alpha_0+1)/\widehat{\alpha}_{min}|X|)$. The variance σ^2 is bounded by

$$\left\| \frac{1}{|X|^2} \mathbb{E}[\sum_{i\in X} \|\psi_i\|^2 \psi_i\psi_i^\top] \right\| \leq \left\| \frac{1}{|X|^2} \max \|\psi_i\|^2 \mathbb{E}[\sum_{i\in X} \psi_i\psi_i^\top] \right\| \leq O((\alpha_0+1)/\widehat{\alpha}_{\min}|X|).$$

Since $O((\alpha_0+1)/\alpha_{\min}|X|) < 1$, the variance dominates in Matrix Bernstein's inequality.

Let $B := |X|^{-1}\Psi_X \Psi_X^\top$. We have with probability $1-\delta$,

$$\begin{aligned}\sigma_{\min}(\mathbb{E}[(G_{X,A}^{\alpha_0})^\top|\Pi]) &= \sqrt{|X|\sigma_{\min}(F_A \operatorname{Diag}(\hat{\alpha})^{1/2} B \operatorname{Diag}(\hat{\alpha})^{1/2} F_A^\top)}, \\ &= \Omega(\sqrt{\widehat{\alpha}_{\min}|X|(1-\epsilon_1)} \cdot \sigma_{\min}(F_A)).\end{aligned}$$

From Lemma 27, with probability $1-\delta$,

$$\sigma_{\min}(F_A) \geq \left(\sqrt{\frac{|A|\widehat{\alpha}_{\min}}{\alpha_0+1}} - O((|A|\log k/\delta)^{1/4}) \right) \cdot \sigma_{\min}(P).$$

Similarly other results follow. ■

C.5 Properties of Dirichlet Distribution

In this section, we list various properties of Dirichlet distribution.

C.5.1 Sparsity Inducing Property

We first note that the Dirichlet distribution $\mathrm{Dir}(\alpha)$ is sparse depending on values of α_i, which is shown in Telgarsky (2012).

Lemma 24 *Let reals $\tau \in (0,1]$, $\alpha_i > 0$, $\alpha_0 := \sum_i \alpha_i$ and integers $1 \leq s \leq k$ be given. Let $(X_i, \ldots, X_k) \sim \mathrm{Dir}(\alpha)$. Then*

$$\Pr\left[|\{i : X_i \geq \tau\}| \leq s\right] \geq 1 - \tau^{-\alpha_0} e^{-(s+1)/3} - e^{-4(s+1)/9},$$

when $s+1 < 3k$.

We now show that we obtain good initialization vectors under Dirichlet distribution.

Arrange the $\widehat{\alpha}_j$'s in ascending order, i.e., $\widehat{\alpha}_1 = \widehat{\alpha}_{\min} \leq \widehat{\alpha}_2 \ldots \leq \widehat{\alpha}_k = \widehat{\alpha}_{\max}$. Recall that columns vectors $\hat{W}_A^\top G_{i,A}^\top$, for $i \notin A$, are used as initialization vectors to the tensor power method. We say that $u_i := \frac{\hat{W}_A^\top G_{i,A}^\top}{\|\hat{W}_A^\top G_{i,A}^\top\|}$ is a (γ, R_0)-good initialization vector corresponding to $j \in [k]$ if

$$|\langle u_i, \Phi_j\rangle| \geq R_0, \quad |\langle u_i, \Phi_j\rangle| - \max_{m<j} |\langle u_i, \Phi_m\rangle| \geq \gamma\, |\langle u_i, \Phi_j\rangle|\,, \tag{88}$$

where $\Phi_j := \widehat{\alpha}_j^{1/2}(\tilde{F}_A)_j$, where $(\tilde{F}_A)_j$ is the j^{th} column of $\tilde{F}_A := W_A^\top F_A$. Note that the $\{\Phi_j\}$ are orthonormal and are the eigenvectors to be estimated by the tensor power method.

Lemma 25 (Good initialization vectors under Dirichlet distribution) *When* $\pi_i \overset{iid}{\sim} \mathrm{Dir}(\alpha)$, *and* $\alpha_j < 1$, *let*

$$\Delta := O\left(\frac{\zeta\rho}{\sqrt{n}r_0}\right). \tag{89}$$

For $j \in [k]$, *there is at least one* $(\gamma - \frac{2\Delta}{r_0-\Delta}, r_0 - \Delta)$*-good vector corresponding to each* Φ_j, *for* $j \in [k]$, *among* $\{u_i\}_{i\in[n]}$ *with probability* $1 - 9\delta$, *when*

$$n = \tilde{\Omega}\left(\alpha_{\min}^{-1} e^{r_0 \widehat{\alpha}_{\max}^{1/2}(\alpha_0 + c_1\sqrt{k\alpha_0})}(2k)^{r_0 c_2}\log(k/\delta)\right), \tag{90}$$

where $c_1 := (1 + \sqrt{8\log 4})$ *and* $c_2 := 4/3(\log 4)$, *when*

$$(1-\gamma)r_0\widehat{\alpha}_{\min}^{1/2}(\alpha_0 + (1+\sqrt{8\log 4})\sqrt{k\alpha_0} + 4/3(\log 4)\widehat{\alpha}_{\min}^{-1/2}\log 2k) > 1. \tag{91}$$

When $\alpha_0 < 1$, *the bound can be improved for* $r_0 \in (0.5, (\alpha_0+1)^{-1})$ *and* $1 - \gamma \geq \frac{1-r_0}{r_0}$ *as*

$$n > \frac{(1+\alpha_0)(1 - r_0\widehat{\alpha}_{\min})}{\widehat{\alpha}_{\min}(\alpha_{\min} + 1 - r_0(\alpha_0+1))}\log(k/\delta). \tag{92}$$

Remark: *(when* $\alpha_0 \geq 1$, $\alpha_0 = \Theta(1)$*) When* r_0 *is chosen as* $r_0 = \alpha_{\max}^{-1/2}(\sqrt{\alpha_0} + c_1\sqrt{k})^{-1}$, *the term* $e^{r_0\widehat{\alpha}_{\max}^{1/2}(\alpha_0 + c_1\sqrt{k\alpha_0})} = e$, *and we require*

$$n = \tilde{\Omega}\left(\alpha_{\min}^{-1} k^{0.43}\log(k/\delta)\right), \quad r_0 = \alpha_{\max}^{-1/2}(\sqrt{\alpha_0} + c_1\sqrt{k})^{-1}, \tag{93}$$

by substituting $c_2/c_1 = 0.43$. *Moreover,* (91) *is satisfied for the above choice of* r_0 *when* $\gamma = \Theta(1)$.

In this case we also need $\Delta < r_0/2$, *which implies*

$$\zeta = O\left(\frac{\sqrt{n}}{\rho k\widehat{\alpha}_{\max}}\right) \tag{94}$$

Remark: *(when* $\alpha_0 < 1$*) In this regime,* (92) *implies that we require* $n = \Omega(\widehat{\alpha}_{\min}^{-1})$. *Also,* r_0 *is a constant, we just need* $\zeta = O(\sqrt{n}/\rho)$.

Proof: Define $\tilde{u}_i := W_A^\top F_A\pi_i / \|W_A^\top F_A\pi_i\|$, when whitening matrix W_A and F_A corresponding to exact statistics are input.

We first observe that if $\tilde{u}_i$ is (γ, r_0) good, then u_i is $(\gamma - \frac{2\Delta}{r_0-\Delta}, r_0 - \Delta)$ good.

When $\tilde{u}_i$ is (γ, r_0) good, note that $W_A^\top F_A\pi_i \geq \widehat{\alpha}_{\max}^{-1/2} r_0$ since $\sigma_{\min}(W_A^\top F_A) = \widehat{\alpha}_{\max}^{-1/2}$ and $\|\pi_i\| \geq r_0$. Now with probability 1/4, conditioned on π_i, we have the event $\mathcal{B}(i)$,

$$\mathcal{B}(i) := \{\|u_i - \tilde{u}_i\| \leq \Delta\},$$

where Δ is given by

$$\Delta = \tilde{O}\left(\frac{\widehat{\alpha}_{\max}^{0.5}\sqrt{(\alpha_0+1)(\max_i(P\widehat{\alpha})_i)}}{r_0 n^{1/2}\widehat{\alpha}_{\min}^{1.5}\sigma_{\min}(P)}\right)$$

from Lemma 19. Thus, we have $\mathbb{P}[\mathcal{B}(i)|\pi_i] \geq 1/4$, i.e., $\mathcal{B}(i)$ occurs with probability $1/4$ for any realization of π_i.

If we perturb a (γ, r_0) good vector by Δ (while maintaining unit norm), then it is still $(\gamma - \frac{2\Delta}{r_0 - \Delta}, r_0 - \Delta)$ good.

We now show that the set $\{\tilde{u}_i\}$ contains good initialization vectors when n is large enough. Consider $Y_i \sim \Gamma(\alpha_i, 1)$, where $\Gamma(\cdot,\cdot)$ denotes the Gamma distribution and we have $Y/\sum_i Y_i \sim \mathrm{Dir}(\alpha)$. We first compute the probability that $\tilde{u}_i := W_A^\top F_A \pi_i / \|W_A^\top F_A \pi_i\|$ is a (r_0, γ)-good vector with respect to $j = 1$ (recall that $\widehat{\alpha}_1 = \widehat{\alpha}_{\min}$). The desired event is

$$\mathcal{A}_1 := (\widehat{\alpha}_1^{-1/2} Y_1 \geq r_0 \sqrt{\sum_j \widehat{\alpha}_j^{-1} Y_j^2}) \cap (\widehat{\alpha}_1^{-1/2} Y_1 \geq \frac{1}{1-\gamma} \max_{j>1} \widehat{\alpha}_j^{-1/2} Y_j) \tag{95}$$

We have

$$\begin{aligned}
\mathbb{P}[\mathcal{A}_1] &\geq \mathbb{P}\left[(\widehat{\alpha}_{\min}^{-1/2} Y_1 \geq r_0 \sqrt{\sum_j \widehat{\alpha}_j^{-1} Y_j^2}) \cap (Y_1 \geq \frac{1}{1-\gamma} \max_{j>1} Y_j) \right] \\
&\geq \mathbb{P}\left[(\widehat{\alpha}_{\min}^{-1/2} Y_1 > r_0 t) \bigcap (\sum_j \widehat{\alpha}_j^{-1} Y_j^2 \leq t^2) \bigcap_{j>1} (Y_1 \leq (1-\gamma) r_0 t \widehat{\alpha}_{\min}^{1/2}) \right], \quad \text{for some } t \\
&\geq \mathbb{P}\left[\widehat{\alpha}_{\min}^{-1/2} Y_1 > r_0 t \right] \mathbb{P}\left[\sum_j \widehat{\alpha}_j^{-1} Y_j^2 \leq t^2 \Big| \widehat{\alpha}_j^{-1/2} Y_j \leq (1-\gamma) r_0 t \widehat{\alpha}_{\min}^{1/2} \right] \\
&\quad \mathbb{P}\left[\max_{j>1} Y_j \leq (1-\gamma) r_0 t \widehat{\alpha}_{\min}^{1/2} \right] \\
&\geq \mathbb{P}\left[\widehat{\alpha}_{\min}^{-1/2} Y_1 > r_0 t \right] \mathbb{P}\left[\sum_j \widehat{\alpha}_j^{-1} Y_j^2 \leq t^2 \right] \mathbb{P}\left[\max_{j>1} Y_j \leq (1-\gamma) r_0 t \widehat{\alpha}_{\min}^{1/2} \right].
\end{aligned}$$

When $\alpha_j \leq 1$, we have

$$\mathbb{P}[\cup_j Y_j \geq \log 2k] \leq 0.5,$$

since $P(Y_j \geq t) \leq t^{\alpha_j - 1} e^{-t} \leq e^{-t}$ when $t > 1$ and $\alpha_j \leq 1$. Applying vector Bernstein's inequality, we have with probability $0.5 - e^{-m}$ that

$$\|\operatorname{Diag}(\widehat{\alpha}_j^{-1/2})(Y - \mathbb{E}(Y))\|_2 \leq (1 + \sqrt{8m})\sqrt{k\alpha_0} + 4/3 m \widehat{\alpha}_{\min}^{-1/2} \log 2k,$$

since $\mathbb{E}[\sum_j \widehat{\alpha}_j^{-1} \operatorname{Var}(Y_j)] = k\alpha_0$ since $\widehat{\alpha}_j = \alpha_j/\alpha_0$ and $\operatorname{Var}(Y_j) = \alpha_j$. Thus, we have

$$\|\operatorname{Diag}(\widehat{\alpha}_j^{-1/2}) Y\|_2 \leq \alpha_0 + (1 + \sqrt{8m})\sqrt{k\alpha_0} + 4/3 m \widehat{\alpha}_{\min}^{-1/2} \log 2k,$$

since $\|\operatorname{Diag}(\widehat{\alpha}_j^{-1/2})\mathbb{E}(Y)\|_2 = \sqrt{\sum_j \widehat{\alpha}_j^{-1} \alpha_j^2} = \alpha_0$. Choosing $m = \log 4$, we have with probability $1/4$ that

$$\begin{aligned}
\|\operatorname{Diag}(\widehat{\alpha}_j^{-1/2}) Y\|_2 \leq t &:= \alpha_0 + (1 + \sqrt{8 \log 4})\sqrt{k\alpha_0} + 4/3 (\log 4) \widehat{\alpha}_{\min}^{-1/2} \log 2k, &(96)\\
&= \alpha_0 + c_1 \sqrt{k\alpha_0} + c_2 \widehat{\alpha}_{\min}^{-1/2} \log 2k. &(97)
\end{aligned}$$

We now have

$$\mathbb{P}\left[\widehat{\alpha}_{\min}^{-1/2} Y_1 > r_0 t\right] \geq \frac{\alpha_{\min}}{4C}\left(r_0 t \widehat{\alpha}_{\min}^{1/2}\right)^{\alpha_{\min}-1} e^{-r_0 t \widehat{\alpha}_{\min}^{1/2}},$$

from Lemma 28.

Similarly,

$$\begin{aligned}\mathbb{P}\left[\max_{j\neq 1} Y_j \leq \widehat{\alpha}_{\min}^{1/2}(1-\gamma) r_0 t\right] &\geq 1 - \sum_j \left((1-\gamma) r_0 t \widehat{\alpha}_{\min}^{1/2}\right)^{\sum_j \alpha_j - 1} e^{-(1-\gamma) r_0 \widehat{\alpha}_{\min}^{1/2} t} \\ &\geq 1 - k e^{-(1-\gamma) r_0 \widehat{\alpha}_{\min}^{1/2} t},\end{aligned}$$

assuming that $(1-\gamma) r_0 \widehat{\alpha}_{\min}^{1/2} t > 1$.

Choosing t as in (96), we have the probability of the event in (95) is greater than

$$\begin{aligned}&\frac{\alpha_{\min}}{16C}\left(1 - \frac{e^{-(1-\gamma) r_0 \widehat{\alpha}_{\min}^{1/2}(\alpha_0 + c_1\sqrt{k\alpha_0})}}{2(2k)^{(1-\gamma) r_0 c_2 - 1}}\right)\frac{e^{-r_0 \widehat{\alpha}_{\min}^{1/2}(\alpha_0 + c_1\sqrt{k\alpha_0})}}{(2k)^{r_0 c_2}} \\ &\cdot \left(r_0 \widehat{\alpha}_{\min}^{1/2}(\alpha_0 + c_1\sqrt{k\alpha_0} + c_2 \widehat{\alpha}_{\min}^{-1/2}\log 2k)\right)^{\alpha_{\min}-1}.\end{aligned}$$

Similarly the (marginal) probability of events $\mathcal{A}_2$ can be bounded from below by replacing $\alpha_{\min}$ with α_2 and so on. Thus, we have

$$\mathbb{P}[\mathcal{A}_m] = \tilde{\Omega}\left(\alpha_{\min}\frac{e^{-r_0 \widehat{\alpha}_{\max}^{1/2}(\alpha_0 + c_1\sqrt{k\alpha_0})}}{(2k)^{r_0 c_2}}\right),$$

for all $m \in [k]$.

Thus, we have each of the events $\mathcal{A}_1(i)\cap\mathcal{B}(i), \mathcal{A}_2(i)\cap\mathcal{B}(i), \ldots, \mathcal{A}_k\cap\mathcal{B}(i)$ occur at least once in $i \in [n]$ i.i.d. tries with probability

$$\begin{aligned}&1 - \mathbb{P}\left[\bigcup_{j\in[k]}\left(\bigcap_{i\in[n]}(\mathcal{A}_j(i)\cap\mathcal{B}(i))^c\right)\right] \\ &\geq 1 - \sum_{j\in[k]}\mathbb{P}\left[\bigcap_{i\in[n]}(\mathcal{A}_j(i) - \mathcal{B}(i))^c\right] \\ &\geq 1 - \sum_{j\in[k]}\exp\left[-n\mathbb{P}(\mathcal{A}_j\cap\mathcal{B})\right], \\ &\geq 1 - k\exp\left[-n\tilde{\Omega}\left(\alpha_{\min}\frac{e^{-r_0 \widehat{\alpha}_{\max}^{1/2}(\alpha_0 + c_1\sqrt{k\alpha_0})}}{(2k)^{r_0 c_2}}\right)\right],\end{aligned}$$

where $\mathcal{A}_j(i)$ denotes the event that $\mathcal{A}_1$ occurs for i^{th} trial and we have that $\mathbb{P}[\mathcal{B}|\mathcal{A}_j] \geq 0.25$ since $\mathcal{B}$ occurs in any trial with probability 0.25 for any realization of π_i and the events $\mathcal{A}_j$ depend only on π_i. We use that $1 - x \leq e^{-x}$ when $x \in [0,1]$. Thus, for the event to occur with probability $1-\delta$, we require

$$n = \tilde{\Omega}\left(\alpha_{\min}^{-1} e^{r_0 \widehat{\alpha}_{\max}^{1/2}(\alpha_0 + c_1\sqrt{k\alpha_0})}(2k)^{r_0 c_2}\log(1/\delta)\right).$$

We can improve the above bound by directly working with the Dirichlet distribution. Let $\pi \sim \mathrm{Dir}(\alpha)$. The desired event corresponding to $j = 1$ is given by

$$\mathcal{A}_1 = \left(\frac{\widehat{\alpha}_1^{-1/2} \pi_1}{\| \mathrm{Diag}(\widehat{\alpha}_i^{-1/2}) \pi \|} \geq r_0 \right) \bigcap_{i>1} \left(\pi_1 \geq \frac{\pi_i}{1-\gamma} \right).$$

Thus, we have

$$\begin{aligned} \mathbb{P}[\mathcal{A}_1] &\geq \mathbb{P}\left[(\pi_1 \geq r_0) \bigcap_{i>1} (\pi_i \leq (1-\gamma) r_0) \right] \\ &\geq \mathbb{P}[\pi_1 \geq r_0] \mathbb{P}\left(\bigcap_{i>1} \pi_i \leq (1-\gamma) r_0 | \pi_1 \geq r_0 \right), \end{aligned}$$

since $\mathbb{P}\left(\bigcap_{i>1} \pi_i \leq (1-\gamma) r_0 | \pi_1 \geq r_0\right) \geq \mathbb{P}\left(\bigcap_{i>1} \pi_i \leq (1-\gamma) r_0\right)$. By properties of Dirichlet distribution, we know $\mathbb{E}[\pi_i] = \widehat{\alpha}_i$ and $\mathbb{E}[\pi_i^2] = \widehat{\alpha}_i \frac{\alpha_i + 1}{\alpha_0 + 1}$. Let $p := \Pr[\pi_1 \geq r_0]$. We have

$$\begin{aligned} \mathbb{E}[\pi_i^2] &= p \mathbb{E}[\pi_i^2 | \pi_i \geq r_0] + (1-p) \mathbb{E}[\pi_i^2 | \pi_i < r_0] \\ &\leq p + (1-p) r_0 \mathbb{E}[\pi_i | \pi_i < r_0] \\ &\leq p + (1-p) r_0 \mathbb{E}[\pi_i]. \end{aligned}$$

Thus, $p \geq \frac{\widehat{\alpha}_{\min}(\alpha_{\min} + 1 - r_0(\alpha_0+1))}{(\alpha_0+1)(1-r_0 \widehat{\alpha}_{\min})}$, which is useful when $r_0(\alpha_0 + 1) < 1$. Also when $\pi_1 \geq r_0$, we have that $\pi_i \leq 1 - r_0$ since $\pi_i \geq 0$ and $\sum_i \pi_i = 1$. Thus, choosing $1 - \gamma = \frac{1-r_0}{r_0}$, we have the other conditions for $\mathcal{A}_1$ are satisfied. Also, verify that we have $\gamma < 1$ when $r_0 > 0.5$ and this is feasible when $\alpha_0 < 1$. ■

We now prove a result that the entries of π_i, which are marginals of the Dirichlet distribution, are likely to be small in the sparse regime of the Dirichlet parameters. Recall that the marginal distribution of π_i is distributed as $B(\alpha_i, \alpha_0 - \alpha_i)$, where $B(a, b)$ is the beta distribution and

$$\mathbb{P}[Z = z] \propto z^{a-1}(1-z)^{b-1}, \quad Z \sim B(a, b).$$

Lemma 26 (Marginal Dirichlet distribution in sparse regime) *For $Z \sim B(a, b)$, the following results hold:*
Case $b \leq 1$**,** $C \in [0, 1/2]$**:**

$$\Pr[Z \geq C] \leq 8 \log(1/C) \cdot \frac{a}{a+b}, \tag{98}$$

$$\mathbb{E}[Z \cdot \delta(Z \leq C)] \leq C \cdot \mathbb{E}[Z] = C \cdot \frac{a}{a+b}. \tag{99}$$

Case $b \geq 1$**,** $C \leq (b+1)^{-1}$**:** *we have*

$$\Pr[Z \geq C] \leq a \log(1/C), \tag{100}$$

$$\mathbb{E}[Z \cdot \delta(Z \leq C)] \leq 6aC. \tag{101}$$

Remark: *The guarantee for $b \geq 1$ is worse and this agrees with the intuition that the Dirichlet vectors are more spread out (or less sparse) when $b = \alpha_0 - \alpha_i$ is large.*

Proof We have

$$\begin{aligned}\mathbb{E}[Z \cdot \delta(Z \leq C)] &= \int_0^C \frac{1}{B(a,b)} x^a (1-x)^{b-1} dx \\ &\leq \frac{(1-C)^{b-1}}{B(a,b)} \int_0^C x^a dx \\ &= \frac{(1-C)^{b-1} C^{a+1}}{(a+1)B(a,b)}.\end{aligned}$$

For $\mathbb{E}[Z \cdot \delta(Z \geq C)]$, we have,

$$\begin{aligned}\mathbb{E}[Z \cdot \delta(Z \geq C)] &= \int_C^1 \frac{1}{B(a,b)} x^a (1-x)^{b-1} dx \\ &\geq \frac{C^a}{B(a,b)} \int_C^1 (1-x)^{b-1} dx \\ &= \frac{(1-C)^b C^a}{bB(a,b)}.\end{aligned}$$

The ratio between these two is at least

$$\frac{\mathbb{E}[Z \cdot \delta(Z \geq C)]}{\mathbb{E}[Z \cdot \delta(Z \leq C)]} \geq \frac{(1-C)(a+1)}{bC} \geq \frac{1}{C}.$$

The last inequality holds when $a, b < 1$ and $C < 1/2$. The sum of the two is exactly $\mathbb{E}[Z]$, so when $C < 1/2$ we know $\mathbb{E}[Z \cdot \delta(Z \leq C)] < C \cdot \mathbb{E}[Z]$.

Next we bound the probability $\Pr[Z \geq C]$. Note that $\Pr[Z \geq 1/2] \leq 2\mathbb{E}[Z] = \frac{2a}{a+b}$ by Markov's inequality. Now we show $\Pr[Z \in [C, 1/2]]$ is not much larger than $\Pr[Z \geq 1/2]$ by bounding the integrals.

$$A = \int_{1/2}^1 x^{a-1}(1-x)^{b-1} dx \geq \int_{1/2}^1 (1-x)^{b-1} dx = (1/2)^b / b.$$

$$\begin{aligned}B = \int_C^{1/2} x^{a-1}(1-x)^{b-1} &\leq (1/2)^{b-1} \int_C^{1/2} x^{a-1} dx \\ &\leq (1/2)^{b-1} \frac{0.5^a - C^a}{a} \\ &\leq (1/2)^{b-1} \frac{1 - (1 - a \log 1/C)}{a} \\ &= (1/2)^{b-1} \log(1/C).\end{aligned}$$

The last inequality uses the fact that $e^x \geq 1 + x$ for all x. Now

$$\Pr[Z \geq C] = (1 + \frac{B}{A}) \Pr[Z \geq 1/2] \leq (1 + 2b \log(1/C)) \frac{2a}{a+b} \leq 8 \log(1/C) \cdot \frac{a}{a+b}$$

and we have the result.

When $b \geq 1$, we have an alternative bound. We use the fact that if $X \sim \Gamma(a,1)$ and $Y \sim \Gamma(b,1)$ then $Z \sim X/(X+Y)$. Since Y is distributed as $\Gamma(b,1)$, its PDF is $\frac{1}{\Gamma(b)}x^{b-1}e^{-x}$. This is proportional to the PDF of $\Gamma(1)$ (e^{-x}) multiplied by a increasing function x^{b-1}.

Therefore we know $\Pr[Y \geq t] \geq \Pr_{Y' \sim \Gamma(1)}[Y' \geq t] = e^{-t}$.

Now we use this bound to compute the probability that $Z \leq 1/R$ for all $R \geq 1$.

This is equivalent to

$$\begin{aligned}\Pr[\frac{X}{X+Y} \leq \frac{1}{R}] &= \int_0^\infty Pr[X=x]Pr[Y \geq (R-1)X]dx \\ &\geq \int_0^\infty \frac{1}{\Gamma(a)}x^{a-1}e^{-Rx}dx \\ &= R^{-a}\int_0^\infty \frac{1}{\Gamma(a)}y^{a-1}e^{-y}dy \\ &= R^{-a}.\end{aligned}$$

In particular, $\Pr[Z \leq C] \geq C^a$, which means $\Pr[Z \geq C] \leq 1 - C^a \leq a\log(1/C)$.

For $\mathbb{E}[Z\delta(Z<C)]$, the proof is similar as before:

$$P = \mathbb{E}[Z\delta(Z<C)] = \int_0^C \frac{1}{B(a,b)}x^a(1-x)^b dx \leq \frac{C^{a+1}}{B(a,b)(a+1)},$$

$$Q = \mathbb{E}[Z\delta(Z \geq C)] = \int_C^1 \frac{1}{B(a,b)}x^a(1-x)^b dx \geq \frac{C^a(1-C)^{b+1}}{B(a,b)(b+1)}.$$

Now $\mathbb{E}[Z\delta(Z \leq C)] \leq \frac{P}{Q}\mathbb{E}[Z] \leq 6aC$ when $C < 1/(b+1)$. ■

C.5.2 Norm Bounds

Lemma 27 (Norm Bounds under Dirichlet distribution) *For $\pi_i \overset{iid}{\sim} \mathrm{Dir}(\alpha)$ for $i \in A$, with probability $1-\delta$, we have*

$$\begin{aligned}\sigma_{\min}(\Pi_A) &\geq \sqrt{\frac{|A|\widehat{\alpha}_{\min}}{\alpha_0+1}} - O((|A|\log k/\delta)^{1/4}), \\ \|\Pi_A\| &\leq \sqrt{|A|\widehat{\alpha}_{\max}} + O((|A|\log k/\delta)^{1/4}), \\ \kappa(\Pi_A) &\leq \sqrt{\frac{(\alpha_0+1)\widehat{\alpha}_{\max}}{\widehat{\alpha}_{\min}}} + O((|A|\log k/\delta)^{1/4}).\end{aligned}$$

This implies that $\|F_A\| \leq \|P\|\sqrt{|A|\widehat{\alpha}_{\max}}$, $\kappa(F_A) \leq O(\kappa(P)\sqrt{(\alpha_0+1)\widehat{\alpha}_{\max}/\widehat{\alpha}_{min}})$. Moreover, with probability $1-\delta$

$$\|F_A\|_1 \leq |A| \cdot \max_i (P\widehat{\alpha})_i + O\left(\|P\|\sqrt{|A|\log\frac{|A|}{\delta}}\right). \tag{102}$$

Remark: *When* $|A| = \Omega\left(\log \frac{k}{\delta}\left(\frac{\alpha_0+1}{\widehat{\alpha}_{\min}}\right)^2\right)$, *we have* $\sigma_{\min}(\Pi_A) = \Omega(\sqrt{\frac{|A|\widehat{\alpha}_{\min}}{\alpha_0+1}})$ *with probability* $1-\delta$ *for any fixed* $\delta \in (0,1)$.
Proof: Consider $\Pi_A \Pi_A^\top = \sum_{i\in A} \pi_i \pi_i^\top$.

$$\begin{aligned}\frac{1}{|A|}\mathbb{E}[\Pi_A \Pi_A^\top] =& \mathbb{E}_{\pi\sim Dir(\alpha)}[\pi\pi^\top] \\ =& \frac{\alpha_0}{\alpha_0+1}\widehat{\alpha}\widehat{\alpha}^\top + \frac{1}{\alpha_0+1}\operatorname{Diag}(\widehat{\alpha}),\end{aligned}$$

from Proposition 29. The first term is positive semi-definite so the eigenvalues of the sum are at least the eigenvalues of the second component. Smallest eigenvalue of second component gives lower bound on $\sigma_{\min}(\mathbb{E}[\Pi_A \Pi_A^\top])$. The spectral norm of the first component is bounded by $\frac{\alpha_0}{\alpha_0+1}\|\hat{\alpha}\| \le \frac{\alpha_0}{\alpha_0+1}\widehat{\alpha}_{\max}$, the spectral norm of second component is $\frac{1}{\alpha_0+1}\alpha_{\max}$. Thus $\left\|\mathbb{E}[\Pi_A \Pi_A^\top]\right\| \le |A| \cdot \widehat{\alpha}_{\max}$.

Now applying Matrix Bernstein's inequality to $\frac{1}{|A|}\sum_i \left(\pi_i\pi_i^\top - \mathbb{E}[\pi\pi^\top]\right)$. We have that the variance is $O(1/|A|)$. Thus with probability $1-\delta$,

$$\left\|\frac{1}{|A|}\left(\Pi_A\Pi_A^\top - \mathbb{E}[\Pi_A\Pi_A^\top]\right)\right\| = O\left(\sqrt{\frac{\log(k/\delta)}{|A|}}\right).$$

For the result on F, we use the property that for any two matrices A, B, $\|AB\| \le \|A\|\,\|B\|$ and $\kappa(AB) \le \kappa(A)\kappa(B)$.

To show bound on $\|F_A\|_1$, note that each column of F_A satisfies $\mathbb{E}[(F_A)_i] = \langle \widehat{\alpha}, (P)_i\rangle\, 1^\top$, and thus $\|\mathbb{E}[F_A]\|_1 \le |A| \max_i (P\widehat{\alpha})_i$. Using Bernstein's inequality, for each column of F_A, we have, with probability $1-\delta$,

$$\left|\, \|(F_A)_i\|_1 - |A|\left\langle \widehat{\alpha}, (P)^i\right\rangle\right| = O\left(\|P\|\sqrt{|A|\log\frac{|A|}{\delta}}\right),$$

by applying Bernstein's inequality, since $|\left\langle \widehat{\alpha}, (P)^i\right\rangle| \le \|P\|$, and thus we have $\sum_{i\in A}\|\mathbb{E}[(P)^j\pi_i\pi_i^\top((P)^j)^\top]\|$, and $\sum_{i\in A}\|\mathbb{E}[\pi_i^\top((P)^j)^\top(P)^j\pi_i]\| \le |A|\cdot\|P\|$. ■

C.5.3 Properties of Gamma and Dirichlet Distributions

Recall Gamma distribution $\Gamma(\alpha,\beta)$ is a distribution on nonnegative real values with density function $\frac{\beta^\alpha}{\Gamma(\alpha)}x^{\alpha-1}e^{-\beta x}$.

Proposition 28 (Dirichlet and Gamma distributions) *The following facts are known for Dirichlet distribution and Gamma distribution.*

1. *Let* $Y_i \sim \Gamma(\alpha_i, 1)$ *be independent random variables, then the vector* $(Y_1, Y_2, ..., Y_k)/\sum_{i=1}^k Y_k$ *is distributed as* $Dir(\alpha)$.
2. *The* Γ *function satisfies Euler's reflection formula:* $\Gamma(1-z)\Gamma(z) \le \pi/\sin\pi_z$.
3. *The* $\Gamma(z) \ge 1$ *when* $0 < z < 1$.

4. *There exists a universal constant C such that $\Gamma(z) \leq C/z$ when $0 < z < 1$.*

5. *For $Y \sim \Gamma(\alpha, 1)$ and $t > 0$ and $\alpha \in (0, 1)$, we have*

$$\frac{\alpha}{4C} t^{\alpha-1} e^{-t} \leq \Pr[Y \geq t] \leq t^{\alpha-1} e^{-t}, \tag{103}$$

and for any $\eta, c > 1$, we have

$$\mathbb{P}[Y > \eta t | Y \geq t] \geq (c\eta)^{\alpha-1} e^{-(\eta-1)t}. \tag{104}$$

Proof: The bounds in (103) is derived using the fact that $1 \leq \Gamma(\alpha) \leq C/\alpha$ when $\alpha \in (0, 1)$ and

$$\int_t^\infty \frac{1}{\Gamma(\alpha_i)} x^{\alpha_i - 1} e^{-x} dx \leq \frac{1}{\Gamma(\alpha_i)} \int_t^\infty t^{\alpha_i - 1} e^{-x} dx \leq t^{\alpha_i - 1} e^{-t},$$

and

$$\int_t^\infty \frac{1}{\Gamma(\alpha_i)} x^{\alpha_i - 1} e^{-x} dx \geq \frac{1}{\Gamma(\alpha_i)} \int_t^{2t} x^{\alpha_i - 1} e^{-x} dx \geq \alpha_i / C \int_t^{2t} (2t)^{\alpha_i - 1} e^{-x} dx \geq \frac{\alpha_i}{4C} t^{\alpha_i - 1} e^{-t}.$$

■

Proposition 29 (Moments under Dirichlet distribution) *Suppose $v \sim Dir(\alpha)$, the moments of v satisfies the following formulas:*

$$\begin{aligned} \mathbb{E}[v_i] &= \frac{\alpha_i}{\alpha_0} \\ \mathbb{E}[v_i^2] &= \frac{\alpha_i(\alpha_i + 1)}{\alpha_0(\alpha_0 + 1)} \\ \mathbb{E}[v_i v_j] &= \frac{\alpha_i \alpha_j}{\alpha_0(\alpha_0 + 1)}, \quad i \neq j. \end{aligned}$$

More generally, if $a^{(t)} = \prod_{i=0}^{t-1}(a + i)$, then we have

$$\mathbb{E}[\prod_{i=1}^k v_i^{(a_i)}] = \frac{\prod_{i=1}^k \alpha_i^{(a_i)}}{\alpha_0^{(\sum_{i=1}^k a_i)}}.$$

C.6 Standard Results

C.6.1 Bernstein's Inequalities

One of the key tools we use is the standard matrix Bernstein inequality (Tropp, 2012, thm. 1.4).

Proposition 30 (Matrix Bernstein Inequality) *Suppose $Z = \sum_j W_j$ where*

1. *W_j are independent random matrices with dimension $d_1 \times d_2$,*
2. *$\mathbb{E}[W_j] = 0$ for all j,*

3. $\|W_j\| \leq R$ *almost surely.*

Let $d = d_1 + d_2$*, and* $\sigma^2 = \max\left\{\left\|\sum_j \mathbb{E}[W_j W_j^\top]\right\|, \left\|\sum_j \mathbb{E}[W_j^\top W_j]\right\|\right\}$*, then we have*

$$\Pr[\|Z\| \geq t] \leq d \cdot exp\left\{\frac{-t^2/2}{\sigma^2 + Rt/3}\right\}.$$

Proposition 31 (Vector Bernstein Inequality) *Let* $z = (z_1, z_2, ..., z_n) \in \mathbb{R}^n$ *be a random vector with independent entries,* $\mathbb{E}[z_i] = 0$*,* $\mathbb{E}[z_i^2] = \sigma_i^2$*, and* $\Pr[|z_i| \leq 1] = 1$*. Let* $A = [a_1|a_2|\cdots|a_n] \in \mathbb{R}^{m\times n}$ *be a matrix, then*

$$\Pr[\|Az\| \leq (1+\sqrt{8t})\sqrt{\sum_{i=1}^n \|a_i\|^2 \sigma_i^2} + (4/3)\max_{i\in[n]}\|a_i\| t] \geq 1 - e^{-t}.$$

C.6.2 Vector Chebyshev Inequality

We will require a vector version of the Chebyshev inequality Ferentios (1982).

Proposition 32 *Let* $z = (z_1, z_2, ..., z_n) \in \mathbb{R}^n$ *be a random vector with independent entries,* $\mathbb{E}[z_i] = \mu$*,* $\sigma := \|\operatorname{Diag}(\mathbb{E}[(z-\mu)^\top(z-\mu)])\|$*. Then we have that*

$$\mathbb{P}[\|z - \mu\| > t\sigma] \leq t^{-2}.$$

C.6.3 Wedin's Theorem

We make use of Wedin's theorem to control subspace perturbations.

Lemma 33 (Wedin's theorem; Theorem 4.4, p. 262 in Stewart and Sun (1990).) *Let* $A, E \in \mathbb{R}^{m\times n}$ *with* $m \geq n$ *be given. Let* A *have the singular value decomposition*

$$\begin{bmatrix} U_1^\top \\ U_2^\top \\ U_3^\top \end{bmatrix} A \begin{bmatrix} V_1 & V_2 \end{bmatrix} = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \\ 0 & 0 \end{bmatrix}.$$

Let $\tilde{A} := A + E$*, with analogous singular value decomposition* $(\tilde{U}_1, \tilde{U}_2, \tilde{U}_3, \tilde{\Sigma}_1, \tilde{\Sigma}_2, \tilde{V}_1 \tilde{V}_2)$*. Let* Φ *be the matrix of canonical angles between* $\operatorname{range}(U_1)$ *and* $\operatorname{range}(\tilde{U}_1)$*, and* Θ *be the matrix of canonical angles between* $\operatorname{range}(V_1)$ *and* $\operatorname{range}(\tilde{V}_1)$*. If there exists* $\delta, \alpha > 0$ *such that* $\min_i \sigma_i(\tilde{\Sigma}_1) \geq \alpha + \delta$ *and* $\max_i \sigma_i(\Sigma_2) \leq \alpha$*, then*

$$\max\{\|\sin\Phi\|_2, \|\sin\Theta\|_2\} \leq \frac{\|E\|_2}{\delta}.$$

References

I. Abraham, S. Chechik, D. Kempe, and A. Slivkins. Low-distortion inference of latent similarities from a multiplex social network. *arXiv preprint arXiv:1202.0922*, 2012.

N. Ailon, Y. Chen, and X. Huan. Breaking the small cluster barrier of graph clustering. *arXiv preprint arXiv:1302.4549*, 2013.

E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9:1981–2014, June 2008.

A. Anandkumar, D. P. Foster, D. Hsu, S. M. Kakade, and Y. Liu. Two svds suffice: Spectral decompositions for probabilistic topic modeling and latent dirichlet allocation. *arXiv preprint arXiv:1202.0922*, 2012a.

A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for latent variable models. *arXiv preprint arXiv:1210.7559*, 2012b.

A. Anandkumar, D. Hsu, and S. M. Kakade. A method of moments for mixture models and hidden markov models. In *Proc. of Conf. on Learning Theory*, June 2012c.

S. Arora, R. Ge, S. Sachdeva, and G. Schoenebeck. Finding overlapping communities in social networks: toward a rigorous approach. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, 2012.

M. F. Balcan, C. Borgs, M. Braverman, J. T. Chayes, and S. Teng. I like her more than you: Self-determined communities. *arXiv preprint arXiv:1201.4899*, 2012.

P. J. Bickel and A. Chen. A nonparametric view of network models and Newman-Girvan and other modularities. *Proceedings of the National Academy of Sciences*, 106(50):21068–21073, 2009.

P. J. Bickel, A. Chen, and E. Levina. The method of moments and degree distributions for network models. *The Annals of Statistics*, 39(5):38–59, 2011.

D. M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003.

B. Bollobás, S. Janson, and O. Riordan. The phase transition in inhomogeneous random graphs. *Random Structures & Algorithms*, 31(1):3–122, 2007.

A. Celisse, J. J. Daudin, and L. Pierre. Consistency of maximum-likelihood and variational estimators in the stochastic block model. *Electronic Journal of Statistics*, 6:1847–1899, 2012.

B. S. Charles and S. S. Vempala. Random tensors and planted cliques. In *RANDOM*, 2009.

S. Chatterjee and P. Diaconis. Estimating and understanding exponential random graph models. *arXiv preprint arXiv:1102.2650*, 2011.

K. Chaudhuri, F. Chung, and A. Tsiatas. Spectral clustering of graphs with general degrees in the extended planted partition model. *Journal of Machine Learning Research*, pages 1–23, 2012.

Y. Chen, S. Sanghavi, and H. Xu. Clustering sparse graphs. In *Advances in Neural Information Processing*, 2012.

A. Condon and R. M. Karp. Algorithms for graph partitioning on the planted partition model. In *Randomization, Approximation, and Combinatorial Optimization. Algorithms and Techniques*, pages 221–232. Springer, 1999.

S. Currarini, M. O. Jackson, and P. Pin. An economic model of friendship: Homophily, minorities, and segregation. *Econometrica*, 77(4):1003–1045, 2009.

V. Feldman, E. Grigorescu, L. Reyzin, S. Vempala, and Y. Xiao. Statistical algorithms and a lower bound for planted clique. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:64, 2012.

K. Ferentios. On Tcebycheff's type inequalities. *Trabajos de Estadística y de Investigación Operativa*, 33(1):125–132, 1982.

S. E. Fienberg, M. M. Meyer, and S. S. Wasserman. Statistical analysis of multiple sociometric relations. *Journal of the American Statistical Association*, 80(389):51–67, 1985.

O. Frank and D. Strauss. Markov graphs. *Journal of the American Statistical Association*, 81(395):832–842, 1986.

A. M. Frieze and R. Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999.

A. M. Frieze and R. Kannan. A new approach to the planted clique problem. In *FSTTCS*, 2008.

M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

A. Gittens and M. W. Mahoney. Revisiting the Nystrom method for improved large-scale machine learning. *arXiv preprint arXiv:1303.1849*, 2013.

P. Gopalan, D. Mimno, S. Gerrish, M. Freedman, and D. Blei. Scalable inference of overlapping communities. In *Advances in Neural Information Processing Systems 25*, pages 2258–2266, 2012.

C. Hillar and L.-H. Lim. Most tensor problems are NP hard. *Journal of the ACM*, 60(6): 45:1–45:39, nov 2013.

M. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *JMLR*, 14:1303–1347, 2012.

P. W. Holland and S. Leinhardt. An exponential family of probability distributions for directed graphs. *Journal of the American Statistical Association*, 76(373):33–50, 1981.

P. W. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: first steps. *Social Networks*, 5(2):109–137, 1983.

F. Huang, U. N. Niranjan, M. Hakeem, and A. Anandkumar. Fast detection of overlapping communities via online tensor methods. *arXiv preprint arXiv:1309.0787*, Sept. 2013.

J. JáJá. *An Introduction to Parallel Algorithms*. Addison Wesley Longman Publishing Co., Inc., 1992.

A. Jalali, Y. Chen, S. Sanghavi, and H. Xu. Clustering partially observed graphs via convex optimization. *arXiv preprint arXiv:1104.4803*, 2011.

M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press., Cambridge, MA, 1994.

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.

P. F. Lazarsfeld, R. K. Merton, et al. Friendship as a social process: A substantive and methodological analysis. *Freedom and Control in Modern Society*, 18(1):18–66, 1954.

Z. Lin, M. Chen, and Y. Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

L. Lovász. Very large graphs. *Current Developments in Mathematics*, 2008:67–128, 2009.

M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, pages 415–444, 2001.

F. McSherry. Spectral partitioning of random graphs. In *FOCS*, 2001.

J. L. Moreno. *Who shall survive?: A new approach to the problem of human interrelations.* Nervous and Mental Disease Publishing Co, 1934.

E. Mossel, J. Neeman, and A. Sly. Stochastic block models and reconstruction. *arXiv preprint arXiv:1202.1499*, 2012.

G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

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

A. Rinaldo, S. E. Fienberg, and Y. Zhou. On the geometry of discrete exponential families with application to exponential random graph models. *Electronic Journal of Statistics*, 3:446–484, 2009.

T. A. B. Snijders and K. Nowicki. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification*, 14(1):75–100, 1997.

G. W. Stewart and J. Sun. *Matrix Perturbation Theory*, volume 175. Academic press New York, 1990.

M. Telgarsky. Dirichlet draws are sparse with high probability. *arXiv preprint arXiv:1301.4917*, 2012.

J. A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2012.

Y. J. Wang and G. Y. Wong. Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association*, 82(397):8–19, 1987.

H. C. White, S. A. Boorman, and R. L. Breiger. Social structure from multiple networks. i. blockmodels of roles and positions. *American Journal of Sociology*, pages 730–780, 1976.

E. P. Xing, W. Fu, and L. Song. A state-space mixed membership blockmodel for dynamic network tomography. *The Annals of Applied Statistics*, 4(2):535–566, 2010.

Journal of Machine Learning Research 15 (2014) 2313-2335 Submitted 12/13; Revised 3/14; Published 6/14

Cover Tree Bayesian Reinforcement Learning

Nikolaos Tziortziotis NTZIORZI@GMAIL.COM
Department of Computer Science and Engineering
University of Ioannina
GR-45110, Greece

Christos Dimitrakakis CHRISTOS.DIMITRAKAKIS@GMAIL.COM
Department of Computer Science and Engineering
Chalmers University of Technology
SE-41296, Sweden

Konstantinos Blekas KBLEKAS@CS.UOI.GR
Department of Computer Science and Engineering
University of Ioannina
GR-45110, Greece

Editor: Laurent Orseau

Abstract

This paper proposes an online tree-based Bayesian approach for reinforcement learning. For inference, we employ a generalised context tree model. This defines a distribution on multivariate Gaussian piecewise-linear models, which can be updated in closed form. The tree structure itself is constructed using the cover tree method, which remains efficient in high dimensional spaces. We combine the model with Thompson sampling and approximate dynamic programming to obtain effective exploration policies in unknown environments. The flexibility and computational simplicity of the model render it suitable for many reinforcement learning problems in continuous state spaces. We demonstrate this in an experimental comparison with a Gaussian process model, a linear model and simple least squares policy iteration.

Keywords: Bayesian inference, non-parametric statistics, reinforcement learning

1. Introduction

In reinforcement learning, an agent must learn how to act in an unknown environment from limited feedback and delayed reinforcement. Efficient learning and planning requires models of the environment that are not only general, but can also be updated online with low computational cost. In addition, probabilistic models allow the use of a number of near-optimal algorithms for decision making under uncertainty. While it is easy to construct such models for small, discrete environments, models for the continuous case have so far been mainly limited to parametric models, which may not have the capacity to represent the environment (such as generalised linear models) and to non-parametric models, which do not scale very well (such as Gaussian processes).

In this paper, we propose a non-parametric family of tree models, with a data-dependent structure constructed through the cover tree algorithm, introduced by Beygelzimer et al. (2006). Cover trees are data structures that cover a metric space with a sequence of data-

dependent partitions. They were initially proposed for the problem of k-nearest neighbour search, but they are in general a good method to generate fine partitions of a state space, due to their low complexity, and can be applied to any state space, with a suitable choice of metric. In addition, it is possible to create a statistical model using the cover tree as a basis. Due to the tree structure, online inference has low (logarithmic) complexity.

In this paper, we specifically investigate the case of a Euclidean state space. For this, we propose a model generalising the context tree weighting algorithm proposed by Willems et al. (1995), combined with Bayesian multivariate linear models. The overall prior can be interpreted as a distribution on piecewise-linear models. We then compare this model with a Gaussian process model, a single linear model, and the model-free method least-squares policy iteration in two well-known benchmark problems in combination with approximate dynamic programming and show that it consistently outperforms other approaches.

The remainder of the paper is organised as follows. Section 1.1 introduces the setting, Section 1.2 discusses related work and Section 1.3 explains our contribution. The model and algorithm are described in Section 2. Finally, comparative experiments are presented in Section 3 and we conclude with a discussion of the advantages of cover-tree Bayesian reinforcement learning and directions of future work in Section 4.

1.1 Setting

We assume that the agent acts within a fully observable discrete-time Markov decision process (MDP), with a metric state space $\mathcal{S}$, for example $\mathcal{S} \subset \mathbb{R}^m$. At time t, the agent observes the current environment state $\boldsymbol{s}_t \in \mathcal{S}$, takes an action a_t from a discrete set $\mathcal{A}$, and receives a reward $r_t \in \mathbb{R}$. The probability over next states is given in terms of a transition kernel $P_\mu(S \mid \boldsymbol{s}, a) \triangleq \mathbb{P}_\mu(\boldsymbol{s}_{t+1} \in S \mid \boldsymbol{s}_t = \boldsymbol{s}, a_t = a)$. The agent selects its actions using a *policy* $\pi \in \Pi$, which in general defines a conditional distribution $\mathbb{P}^\pi(a_t \mid \boldsymbol{s}_1, \ldots, \boldsymbol{s}_t, a_1, \ldots, a_{t-1}, r_1, \ldots, r_{t-1})$ over the actions, given the history of states and actions. This reflects the learning process that the agent undergoes, when the MDP μ is unknown.

The agent's *utility* is $U \triangleq \sum_{t=0}^{\infty} \gamma^t r_t$, the discounted sum of future rewards, with $\gamma \in (0, 1)$ a discount factor such that rewards further into the future are less important than immediate rewards. The goal of the agent is to maximise its expected utility:

$$\max_{\pi \in \Pi} \mathbb{E}_\mu^\pi U = \max_{\pi \in \Pi} \mathbb{E}_\mu^\pi \sum_{t=0}^{\infty} \gamma^t r_t,$$

where the value of the expectation depends on the agent's policy π and the environment μ. If the environment is known, well-known dynamic programming algorithms can be used to find the optimal policy in the discrete-state case (Puterman, 2005), while many approximate algorithms exist for continuous environments (Bertsekas and Tsitsiklis, 1996). In this case, optimal policies are memoryless and we let Π_1 denote the set of memoryless policies. Then MDP and policy define a Markov chain with kernel $P_\mu^\pi(S \mid \boldsymbol{s}, a) = \sum_{a \in A} P_\mu(S \mid \boldsymbol{s}, a)\pi(a \mid \boldsymbol{s})$.

However, since the environment μ is unknown, the above maximisation is ill-posed. In the Bayesian framework for reinforcement learning, this problem is alleviated by performing the maximisation conditioned on the agent's belief about the true environment μ. This converts the problem of reinforcement learning into a concrete, optimisation problem. How-

ever, this is generally extremely complex, as we must optimise over all history-dependent policies.

More specifically, the main assumption in Bayesian reinforcement learning is that the environment μ lies in a given set of environments $\mathcal{M}$. In addition, the agent must select a subjective prior distribution $p(\mu)$ which encodes its belief about which environments are most likely. The Bayes-optimal expected utility for p is:

$$U_p^* \triangleq \max_{\pi \in \Pi^D} \mathbb{E}_p^\pi U = \max_{\pi \in \Pi^D} \int_{\mathcal{M}} \left(\mathbb{E}_\mu^\pi U \right) \mathrm{d}p(\mu). \tag{1}$$

Unlike the known μ case, the optimal policy may not be memoryless, as our belief changes over time. This makes the optimisation over the policies significantly harder (Duff, 2002), as we have to consider the set of all history-dependent deterministic policies, which we denote by $\Pi^D \subset \Pi$. In this paper, we employ the simple, but effective, heuristic of Thompson sampling (Thompson, 1933; Wyatt, 1998; Dearden et al., 1998; Strens, 2000) for finding policies. This strategy is known by various other names, such as probability matching, stochastic dominance, sampling-greedy and posterior sampling. Very recently Osband et al. (2013) showed that it suffers small Bayes-regret relative to the Bayes-optimal policy for finite, discrete MDPs.

The second problem in Bayesian reinforcement learning is the choice of the prior distribution. This can be of critical importance for large or complex problems, for two reasons. Firstly, a well-chosen prior can lead to more efficient learning, especially in the finite-sample regime. Secondly, as reinforcement learning involves potentially unbounded interactions, the computational and space complexity of calculating posterior distributions, estimating marginals and performing sampling become extremely important. The choice of priors is the main focus of this paper. In particular, we introduce a prior over piecewise-linear multivariate Gaussian models. This is based on the construction of a context tree model, using a cover tree structure, which defines a conditional distribution on local linear Bayesian multivariate models. Since inference for the model can be done in closed form, the resulting algorithm is very efficient, in comparison with other non-parametric models such as Gaussian processes. The following section discusses how previous work is related to our model.

1.2 Related Work

One component in our model is the *context tree*. Context trees were introduced by Willems et al. (1995) for sequential prediction (see Begleiter et al., 2004, for an overview). In this model, a distribution of variable order Markov models for binary sequences is constructed, where the tree distribution is defined through context-dependent weights (for probability of a node being part of the tree) and Beta distributions (for predicting the next observation). A recent extension to switching time priors (van Erven et al., 2008) has been proposed by Veness et al. (2012). More related to this paper is an algorithm proposed by Kozat et al. (2007) for prediction. This asymptotically converges to the best univariate piecewise linear model in a class of trees with fixed structure.

Many reinforcement learning approaches based on such trees have been proposed, but have mainly focused on the discrete partially observable case (Daswani et al., 2012; Veness

et al., 2011; Bellemare et al., 2013; Farias et al., 2010).[1] However, tree structures can generally be used to perform Bayesian inference in a number of other domains (Paddock et al., 2003; Meila and Jordan, 2001; Wong and Ma, 2010).

The core of our model is a generalised context tree structure that defines a distribution on multivariate piecewise-linear-Gaussian models. Consequently, a necessary component in our model is a multivariate linear model at each node of the tree. Such models were previously used for Bayesian reinforcement learning in Tziortziotis et al. (2013) and were shown to perform well relatively to least-square policy iteration (LSPI, Lagoudakis and Parr, 2003). Other approaches using linear models include Strehl and Littman (2008), which proves mistake bounds on reinforcement learning algorithms using online linear regression, and Abbeel and Ng (2005) who use separate linear models for each dimension. Another related approach in terms of structure is Brunskill et al. (2009), which partitions the space into *types* and estimates a simple additive model for each type.

Linear-Gaussian models are naturally generalised by Gaussian processes (GP). Some examples of GP in reinforcement learning include those of Rasmussen and Kuss (2004), Deisenroth et al. (2009) and Deisenroth and Rasmussen (2011), which focused on a model-predictive approach, while the work of Engel et al. (2005) employed GPs for expected utility estimation. GPs are computationally demanding, in contrast to our tree-structured prior. Another problem with the cited GP-RL approaches is that they employ the marginal distribution in the dynamic programming step. This heuristic ignores the uncertainty about the model (which is implicitly taken into account in Equations 1, 6). A notable exception to this is the policy gradient approach employed by Ghavamzadeh and Engel (2006) which uses full Bayesian quadrature. Finally, output dimensions are treated independently, which may not make good use of the data. Methods for efficient dependent GPs such as the one introduced by Alvarez et al. (2011) have not yet been applied to reinforcement learning.

For decision making, this paper uses the simple idea of Thompson sampling (Thompson, 1933; Wyatt, 1998; Dearden et al., 1998; Strens, 2000), which has been shown to be near-optimal in certain settings (Kaufmann et al., 2012; Agrawal and Goyal, 2012; Osband et al., 2013). This avoids the computational complexity of building augmented MDP models (Auer et al., 2008; Asmuth et al., 2009; Castro and Precup, 2010; Araya et al., 2012), Monte-Carlo tree search (Veness et al., 2011), sparse sampling (Wang et al., 2005), stochastic branch and bound (Dimitrakakis, 2010b) or creating lower bounds on the Bayes-optimal value function (Poupart et al., 2006; Dimitrakakis, 2011). Thus the approach is reasonable as long as sampling from the model is efficient.

1.3 Our Contribution

Our approach is based upon three ideas. The first idea is to employ a cover tree (Beygelzimer et al., 2006) to create a set of partitions of the state space. This avoids having to prespecify a structure for the tree. The second technical novelty is the introduction of an efficient non-parametric Bayesian conditional density estimator on the cover tree structure. This is a generalised context tree, endowed with a multivariate linear Bayesian model at each node. We use this to estimate the dynamics of the underlying environment. The multivariate

1. We note that another important work in tree-based reinforcement learning, though not directly related to ours, is that of Ernst et al. (2005), which uses trees for expected utility rather than model estimation.

models allow for a sample-efficient estimation by capturing dependencies. Finally, we take a sample from the posterior to obtain a piecewise linear Gaussian model of the dynamics. This can be used to generate policies. In particular, from this, we obtain trajectories of simulated experience, to perform approximate dynamic programming (ADP) in order to select a policy. Although other methods could be used to calculate optimal actions, we leave them for future work.

The main advantage of our approach is its generality and efficiency. The posterior calculation and prediction is fully conjugate and can be performed online. At the t-th time step, inference takes $O(\ln t)$ time. Sampling from the tree, which need only be done infrequently, is $O(t)$. These properties are in contrast to other non-parametric approaches for reinforcement learning such as GPs. The most computationally heavy step of our algorithm is ADP. However, once a policy is calculated, the actions to be taken can be calculated in logarithmic time at each step. The specific ADP algorithm used is not integral to our approach and for some problems it might be more efficient to use an online algorithm.

2. Cover Tree Bayesian RL

The main idea of cover tree Bayesian reinforcement learning (CTBRL) is to construct a cover tree from the observations, simultaneously inferring a conditional probability density on the same structure, and to then use sampling to estimate a policy. We use a cover tree due to its efficiency compared with e.g., a fixed sequence of partitions or other dynamic partitioning methods such as KD-trees. The probabilistic model we use can be seen as a distribution over piecewise linear-Gaussian densities, with one local linear model for each set in each partition. Due to the tree structure, the posterior can be computed efficiently online. By taking a sample from the posterior, we acquire a specific piecewise linear Gaussian model. This is then used to find an approximately optimal policy using approximate dynamic programming.

An overview of CTBRL is given in pseudocode in Alg. 1. As presented, the algorithm works in an episodic manner.[2] When a new episode k starts at time t_k, we calculate a new stationary policy by sampling a tree μ_k from the current posterior $p_{t_k}(\mu)$. This tree corresponds to a piecewise-linear model. We draw a large number of rollout trajectories from μ_k using an arbitrary exploration policy. Since we have the model, we can use an initial state distribution that covers the space well. These trajectories are used to estimate a near-optimal policy π_k using approximate dynamic programming. During the episode, we take new observations using π_k, while growing the cover tree as necessary and updating the posterior parameters of the tree and the local model in each relevant tree node.

We now explain the algorithm in detail. First, we give an overview of the cover tree structure on which the context tree model is built. Then we show how to perform inference on the context tree, while Section 2.3 describes the multivariate model used in each node of the context tree. The sampling approach and the approximate dynamic method are described in Section 2.4, while the overall complexity of the algorithm is discussed in Section 2.5.

2. An online version of the same algorithm (still employing Thompson sampling) would move line 6 to just before line 9. A fully Bayes online version would "simply" take an approximation of the Bayes-optimal action at every step.

Algorithm 1 CTBRL (Episodic, using Thompson sampling)

$k = 0$, $\pi_0 = \mathcal{U}nif(\mathcal{A})$, prior p_0 on $\mathcal{M}$.
for $t = 1, \ldots, T$ **do**
  if `episode-end` **then**
    $k := k + 1$.
    Sample model $\mu_k \sim p_t(\mu)$.
    Calculate policy $\pi_k \approx \arg\max_\pi \mathbb{E}^\pi_{\mu_k} U$.
  end if
  Observe state $\boldsymbol{s}_t$.
  Take action $a_t \sim \pi_k(\cdot \mid \boldsymbol{s}_t)$.
  Observe next state $\boldsymbol{s}_{t+1}$, reward r_{t+1}.
  Add a leaf node to the tree $\mathcal{T}_{a_t}$, containing $\boldsymbol{s}_t$.
  Update posterior: $p_{t+1}(\mu) = p_t(\mu \mid \boldsymbol{s}_{t+1}, \boldsymbol{s}_t, a_t)$ by updating the parameters of all nodes containing $\boldsymbol{s}_t$.
end for

2.1 The Cover Tree Structure

Cover trees are a data structure that can be applied to any metric space and are, among other things, an efficient method to perform nearest-neighbour search in high-dimensional spaces (Beygelzimer et al., 2006). In this paper, we use cover trees to automatically construct a sequence of partitions of the state space. Section 2.1.1 explains the properties of the constructed cover tree. As the formal construction duplicates nodes, in practice we use a reduced tree where every observed point corresponds to one node in the tree. This is explained in Section 2.1.2. An explanation of how nodes are added to the structure is given in Section 2.1.3.

2.1.1 Cover Tree Properties

To construct a cover tree $\mathcal{T}$ on a metric space $(\mathcal{Z}, \psi)$ we require a set of points $D_t = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_t\}$, with $\boldsymbol{z}_i \in \mathcal{Z}$, a metric ψ, and a constant $\zeta > 1$. We introduce a mapping function $[\cdot]$ so that the i-th tree node corresponds to one point $\boldsymbol{z}_{[i]}$ in this set. The nodes are arranged in *levels*, with each point being replicated at nodes in multiple levels, i.e., we may have $[i] = [j]$ for some $i \neq j$. Thus, a point corresponds to multiple nodes in the tree, but to *at most one node* at any one level. Let G_n denote the set of points corresponding to the nodes at level n of the tree and $\mathfrak{C}(i) \subset G_{n-1}$ the corresponding set of children. If $i \in G_n$ then the level of i is $\ell(i) = n$. The tree has the following properties:

1. Refinement: $G_n \subset G_{n-1}$.
2. Siblings separation: $i, j \in G_n$, $\psi(\boldsymbol{z}_{[i]}, \boldsymbol{z}_{[j]}) > \zeta^n$.
3. Parent proximity: If $i \in G_{n-1}$ then $\exists$ a unique $j \in G_n$ such that $\psi(\boldsymbol{z}_{[i]}, \boldsymbol{z}_{[j]}) \leq \zeta^n$ and $i \in \mathfrak{C}(j)$.

These properties can be interpreted as follows. Firstly lower levels always contain more points. Secondly, siblings at a particular level are always well-separated. Finally, a child

must be close to its parent. These properties directly give rise to the theoretical guarantees given by the cover tree structure, as well as methods for searching and adding points to the tree, as explained below.

2.1.2 The Reduced Tree

As formally the cover tree duplicates nodes, in practice we use the *explicit representation* (described in more detail in Section 2 of Beygelzimer et al., 2006). This only stores the top-most tree node i corresponding to a point $\boldsymbol{z}_{[i]}$. We denote this *reduced tree* by $\hat{\mathcal{T}}$. The *depth* $d(i)$ of node $i \in \hat{\mathcal{T}}$ is equal to its number of ancestors, with the root node having a depth of 0. After t observations, the set of nodes containing a point $\boldsymbol{z}$, is:

$$\hat{G}_t(\boldsymbol{z}) \triangleq \left\{ i \in \hat{\mathcal{T}} \;\middle|\; \boldsymbol{z} \in B_i \right\},$$

where $B_i = \left\{ \boldsymbol{z} \in \mathcal{Z} \;\middle|\; \psi(\boldsymbol{z}_{[i]}, \boldsymbol{z}) \leq \zeta^{d(i)} \right\}$ is the neighbourhood of i. Then $\hat{G}_t(\boldsymbol{z})$ forms a path in the tree, as each node only has one parent, and can be discovered in logarithmic time through the `Find-Nearest` function (Beygelzimer et al., 2006, Theorem 5). This fact allows us to efficiently search the tree, insert new nodes, and perform inference.

2.1.3 Inserting Nodes in the Cover Tree

The cover tree insertion we use is only a minor adaptation of the `Insert` algorithm by Beygelzimer et al. (2006). For each action $a \in \mathcal{A}$, we create a different reduced tree $\hat{\mathcal{T}}_a$, over the state space, i.e., $\mathcal{Z} = \mathcal{S}$, and build the tree using the metric $\psi(\boldsymbol{s}, \boldsymbol{s}') = \|\boldsymbol{s} - \boldsymbol{s}'\|_1$.

At each point in time t, we obtain a new observation tuple $\boldsymbol{s}_t, a_t, \boldsymbol{s}_{t+1}$. We select the tree $\hat{\mathcal{T}}_{a_t}$ corresponding to the action. Then, we traverse the tree, decreasing d and keeping a set of nodes $Q_d \subset G_d$ that are ζ^d-close to $\boldsymbol{s}_t$. We stop whenever Q_d contains a node that would satisfy the *parent proximity* property if we insert the new point at $d-1$, while the children of all other nodes in Q_d would satisfy the *sibling separation* property. This means that we can now insert the new datum as a child of that node.[3] Finally, the next state $\boldsymbol{s}_{t+1}$ is only used during the inference process, explained below.

2.2 Generalised Context Tree Inference

In our model, each node $i \in \hat{\mathcal{T}}$ is associated with a particular Bayesian model. The main problem is how to update the individual models and how to combine them. Fortunately, a closed form solution exists due to the tree structure. We use this to define a *generalised context tree*, which can be used for inference.

As with other tree models (Willems et al., 1995; Ferguson, 1974), our model makes predictions by marginalising over a set of simpler models. Each node in the context tree is called a *context*, and each context is associated with a specific local model. At time t, given an observation $\boldsymbol{s}_t = \boldsymbol{s}$ and an action $a_t = a$, we calculate the marginal (predictive) density p_t of the next observation:

$$p_t(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, a_t) = \sum_{c_t} p_t(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, c_t) p_t(c_t \mid \boldsymbol{s}_t, a_t),$$

3. The exact implementation is available in the CoverTree class in Dimitrakakis et al. (2007).

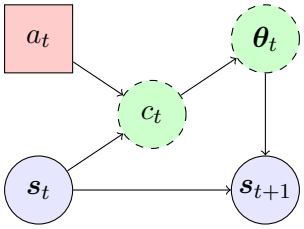


Figure 1: The generalised context tree graphical model. Blue circles indicate observed variables. Green dashed circles indicate latent variables. Red rectangles indicate choice variables. Arrows indicate dependencies. Thus, the context distribution at time t depends on both the state and action, while the parameters depend on the context. The next state depends on the action only indirectly.

where we use the symbol p_t throughout for notational simplicity to denote marginal distributions from our posterior at time t. Here, c_t is such that if $p_t(c_t = i \mid \boldsymbol{s}_t, a_t) > 0$, then the current state is within the neighbourhood of i-th node of the reduced cover tree $\hat{\mathcal{T}}_{a_t}$, i.e., $\boldsymbol{s}_t \in B_i$.

For Euclidean state spaces, the i-th component density $p_t(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, c_t = i)$ employs a linear Bayesian model, which we describe in the next section. The graphical structure of the model is shown in simplified form in Fig. 1. The context at time t depends only on the current state $\boldsymbol{s}_t$ and action a_t. The context corresponds to a particular local model with parameter $\boldsymbol{\theta}_t$, which defines the conditional distribution.

The probability distribution $p_t(c_t \mid \boldsymbol{s}_t, a_t)$ is determined through *stopping probabilities*. More precisely, we set it be equal to the probability of stopping at the i-th context, when performing a walk from the leaf node containing the current observation towards the root, stopping at the j-th node with probability $w_{j,t}$ along the way:

$$p_t(c_t = i \mid \boldsymbol{s}_t, a_t) = w_{i,t} \prod_{j \in \mathfrak{D}_t(i)} (1 - w_{j,t}),$$

where $\mathfrak{D}_t(i)$ are the descendants of i that contain the observation $\boldsymbol{s}_t$. This forms a path from i to the leaf node containing $\boldsymbol{s}_t$. Note that $w_{0,t} = 1$, so we always stop whenever we reach the root. Due to the effectively linear structure of the relevant tree nodes, the stopping probability parameters w can be updated in closed form, as shown in Dimitrakakis (2010a, Theorem 1) via Bayes' theorem as follows:

$$w_{i,t+1} = \frac{p_t(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, c_t = i) w_{i,t}}{p_t(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, c_t \in \{i\} \cup \mathfrak{D}_t(i))}. \tag{2}$$

Since there is a different tree for each action, $c_t = i$ uniquely identifies a tree, the action does not need to enter in the conditional expressions above. Finally, it is easy to see, by marginalisation and the definition of the stopping probabilities, that the denominator in the above equation can be calculated recursively:

$$p_t(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, c_t \in \{i\} \cup \mathfrak{D}_t(i)) = w_{i,t} p_t(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, c_t = i) + (1 - w_{i,t}) p_t(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, c_t \in \mathfrak{D}_t(i)).$$

Consequently, inference can be performed with a simple forward-backward sweep through a single tree path. In the forward stage, we compute the probabilities of the denominator, until we reach the point where we have to insert a new node. Whenever a new node is inserted in the tree, its weight parameter is initialised to $2^{-d(i)}$. We then go backwards to the root node, updating the weight parameters and the posterior of each model. The only remaining question is how to calculate the individual predictive marginal distributions for each context i in the forward sweep and how to calculate their posterior in the backward sweep. In this paper, we associate a linear Bayesian model with each context, which provides this distribution.

2.3 The Linear Bayesian Model

In our model we assume that, given $c_t = i$, the next state $\boldsymbol{s}_{t+1}$ is given by a linear transformation of the current state and additive noise $\boldsymbol{\varepsilon}_{i,t}$:

$$\boldsymbol{s}_{t+1} = \boldsymbol{A}_i \boldsymbol{x}_t + \boldsymbol{\varepsilon}_{i,t}, \qquad \boldsymbol{x}_t \triangleq \begin{pmatrix} \boldsymbol{s}_t \\ 1 \end{pmatrix}, \tag{3}$$

where $\boldsymbol{x}_t$ is the current state vector augmented by a unit basis.[4] In particular, each context models the dynamics via a Bayesian multivariate linear-Gaussian model. For the i-th context, there is a different (unknown) parameter pair $(\boldsymbol{A}_i, \boldsymbol{V}_i)$ where $\boldsymbol{A}_i$ is the *design* matrix and $\boldsymbol{V}_i$ is the *covariance* matrix. Then the next state distribution is:

$$\boldsymbol{s}_{t+1} \mid \boldsymbol{x}_t = \boldsymbol{x}, c_t = i \sim \mathcal{N}(\boldsymbol{A}_i \boldsymbol{x}, \boldsymbol{V}_i).$$

Thus, the parameters $\boldsymbol{\theta}_t$ which are abstractly shown in Fig. 1 correspond to the two matrices $\boldsymbol{A}, \boldsymbol{V}$. We now define the conditional distribution of these matrices given $c_t = i$.

We can model our uncertainty about these parameters with an appropriate prior distribution p_0. In fact, a conjugate prior exists in the form of the *matrix inverse-Wishart normal* distribution. In particular, given $\boldsymbol{V}_i = \boldsymbol{V}$, the distribution for $\boldsymbol{A}_i$ is matrix-normal, while the marginal distribution of $\boldsymbol{V}_i$ is inverse-Wishart:

$$\boldsymbol{A}_i \mid \boldsymbol{V}_i = \boldsymbol{V} \sim \mathcal{N}(\boldsymbol{A}_i \mid \underbrace{\boldsymbol{M}, \boldsymbol{C}}_{\text{prior parameters}}, \boldsymbol{V}) \tag{4}$$

$$\boldsymbol{V}_i \sim \mathcal{W}(\boldsymbol{V}_i \mid \overbrace{\boldsymbol{W}, n}). \tag{5}$$

Here $\mathcal{N}$ is the prior on design matrices, which has a matrix-normal distribution, conditional on the covariance and two prior parameters: $\boldsymbol{M}$, which is the prior mean and $\boldsymbol{C}$ which is the prior covariance of the dependent variable (i.e., the output). Finally, $\mathcal{W}$ is the marginal prior on covariance matrices, which has an inverse-Wishart distribution with $\boldsymbol{W}$ and n. More precisely, the distributions have the following forms:

$$\mathcal{N}(\boldsymbol{A}_i \mid \boldsymbol{M}, \boldsymbol{C}, \boldsymbol{V}) \propto e^{-\frac{1}{2}\operatorname{tr}\left[(\boldsymbol{A}_i - \boldsymbol{M})^\top \boldsymbol{V}^{-1} (\boldsymbol{A}_i - \boldsymbol{M}) \boldsymbol{C}\right]}$$
$$\mathcal{W}(\boldsymbol{V} \mid \boldsymbol{W}, n) \propto |\boldsymbol{V}^{-1}\boldsymbol{W}/2|^{n/2} e^{-\frac{1}{2}\operatorname{tr}(\boldsymbol{V}^{-1}\boldsymbol{W})}.$$

4. While other transformations of $\boldsymbol{s}_t$ are possible, we do not consider them in this paper.

Essentially, the model extends the classic Bayesian linear regression model (e.g., DeGroot, 1970) to the multivariate case via vectorisation of the mean matrix. Since the prior is conjugate, it is relatively simple to calculate the posterior after each observation. For simplicity, and to limit the total number of prior parameters we have to select, we use the same prior parameters $(\boldsymbol{M}_i, \boldsymbol{C}_i, \boldsymbol{W}_i, n_i)$ for all contexts in the tree.

To integrate this with inference in the tree, we must define the marginal distribution used in the nominator of (2). This is a multivariate Student-t distribution, so if the posterior parameters for context i at time t are $(\boldsymbol{M}_i^t, \boldsymbol{C}_i^t, \boldsymbol{W}_i^t, n_i^t)$, then this is:

$$p_t(\boldsymbol{s}_{t+1} \mid \boldsymbol{x}_t = \boldsymbol{x}, c_t = i) = \mathit{Student}(\boldsymbol{M}_i^t, \boldsymbol{W}_i^t / z_i^t, 1 + n_i^t),$$

where $z_i^t = 1 - \boldsymbol{x}^\top (\boldsymbol{C}_i^t + \boldsymbol{x}\boldsymbol{x}^\top)^{-1} \boldsymbol{x}$.

2.3.1 Regression Illustration

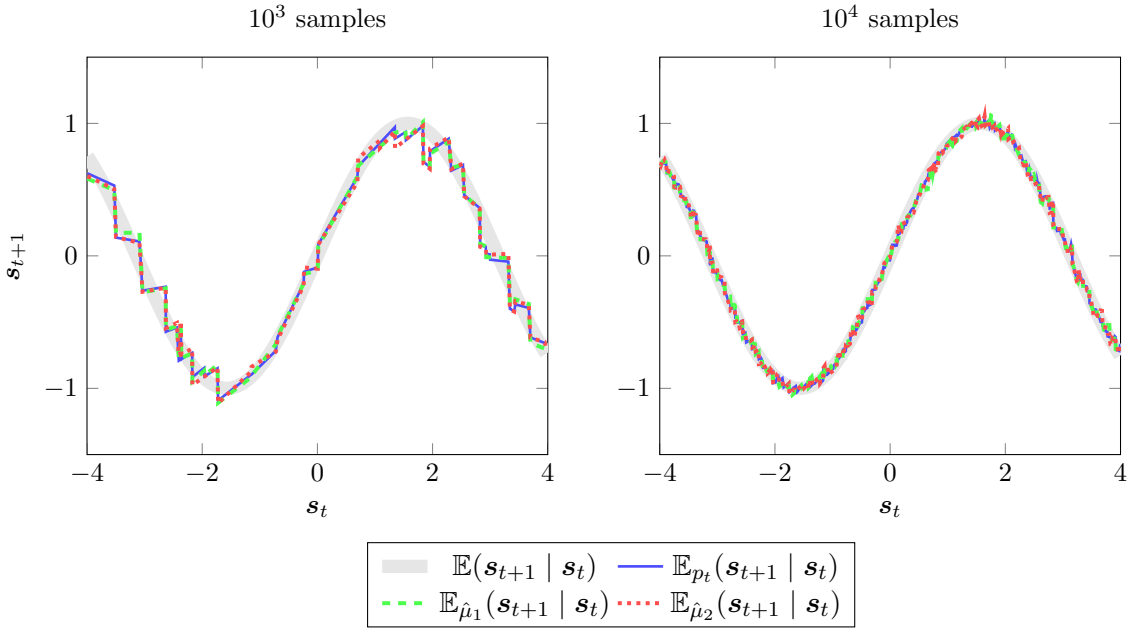


Figure 2: Regression illustration. We plot the expected value for the real distribution, the marginal, as well as two sampled models $\hat{\mu}_1, \hat{\mu}_2 \sim p_t(\mu)$.

An illustration of inference using the generalised context tree is given in Fig. 2, where the piecewise-linear structure is evident. The $\boldsymbol{s}_t$ variates are drawn uniformly in the displayed interval, while $\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t = \boldsymbol{s} \sim \mathcal{N}(\sin(\boldsymbol{s}), 0.1)$, i.e., drawn a normal distribution with mean $\sin(\boldsymbol{s}_t)$ and variance 0.1. The plot shows the marginal expectation $\mathbb{E}_{p_t}$, as well as the expectation from two different models sampled from the posterior $p_t(\mu)$.

2.4 Approximating the Optimal Policy with Thompson Sampling

Many algorithms exist for finding the optimal policy for a specific MDP μ, or for calculating the expected utility of a given policy for that MDP. Consequently, a simple idea is to draw MDP samples μ_i from the current posterior distribution and then calculate the expected utility of each. This can be used to obtain approximate lower and upper bounds on the Bayes-optimal expected utility by maximising over the set of memoryless policies Π_1. Taking K samples, allows us to calculate the upper and lower bounds with accuracy $O(1/\sqrt{K})$.

$$\max_{\pi\in\Pi_1} \mathbb{E}_p^\pi U \approx \max_{\pi\in\Pi_1} \frac{1}{K}\sum_{i=1}^K \mathbb{E}_{\mu_i}^\pi U \le \frac{1}{K}\sum_{i=1}^K \max_{\pi\in\Pi_1} \mathbb{E}_{\mu_i}^\pi U, \qquad \mu_i \sim p_t(\mu). \tag{6}$$

We consider only the special case $K = 1$, i.e., when we only sample a single MDP. Then the two values are identical and we recover Thompson sampling. The main problems we have to solve now is how to sample a model and how to calculate a policy for the sampled model.

2.4.1 Sampling a Model from the Posterior

Each model μ sampled from the posterior corresponds to a particular choice of tree parameters. Sampling is done in two steps. The first generates a partition from the tree distribution and the second step generates a linear model for each context in the partition.

The first step is straightforward. We only need to sample a set of weights $\hat{w}_i \in \{0, 1\}$ such that $\mathbb{P}(\hat{w}_i = 1) = w_{i,t}$, as shown in Dimitrakakis (2010a, Rem. 2). This creates a *partition*, with one Bayesian multivariate linear model responsible for each context in the partition.

The second step is to sample a design and covariance matrix pair $(\hat{\boldsymbol{A}}_i, \hat{\boldsymbol{V}}_i)$ for each context i in the partition. This avoids sampling matrices for contexts not part of the sampled tree. As the model suggests, we can first sample the noise covariance by plugging the posterior parameters in (5) to obtain $\hat{\boldsymbol{V}}_i$. Sampling from this distribution can be done efficiently using the algorithm suggested by Smith and Hocking (1972). We then plug in $\hat{\boldsymbol{V}}_i$ into the conditional design matrix posterior (4) to obtain a design matrix $\hat{\boldsymbol{A}}_i$ by sampling from the resulting matrix-normal distribution.

The final MDP sample μ from the posterior has two elements. Firstly, a set of contexts $\hat{C}^\mu \subset \bigcup_{a\in\mathcal{A}} \hat{\mathcal{T}}_a$, from all action trees. This set is a partition with associated mapping $f^\mu : \mathcal{S}\times\mathcal{A} \to \hat{C}^\mu$. Secondly, a set of associated design and covariance matrices $\left\{ (A_i^\mu, V_i^\mu) \;\middle|\; i \in \hat{C}^\mu \right\}$ for each context. Then the prediction of the sampled MDP is:

$$\mathbb{P}_\mu(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, a_t) = \mathcal{N}(A^\mu_{f(\boldsymbol{s}_t,a_t)}\boldsymbol{x}_t, V^\mu_{f(\boldsymbol{s}_t,a_t)}), \tag{7}$$

where $\boldsymbol{x}_t$ is given in (3).

2.4.2 Finding a Policy for a Sample via ADP

In order to calculate an optimal policy $\pi^*(\mu)$ for μ, we generate a large number of trajectories from μ using a uniform policy. After selecting an appropriate set of basis functions, we then employ a variant of the least-squares policy iteration (LSPI, Lagoudakis and Parr, 2003)

algorithm, using least-squares temporal differences (LSTD, Bradtke and Barto, 1996) rather than LSTDQ. This is possible because since we have μ available, we have access to (7) and it makes LSPI slightly more efficient.

More precisely, consider the *value function* $V_\mu^\pi : \mathcal{S} \to \mathbb{R}$, defined as:

$$V_\mu^\pi(\boldsymbol{s}) \triangleq \mathbb{E}_\mu^\pi \left(U \mid \boldsymbol{s}_t = \boldsymbol{s} \right).$$

Unfortunately, for continuous $\mathcal{S}$ finding an optimal policy requires approximations. A common approach is to make use of the fact that:

$$V_\mu^\pi(\boldsymbol{s}) = \rho(\boldsymbol{s}) + \gamma \int_{\mathcal{S}} V_\mu^\pi(\boldsymbol{s}') \, \mathrm{d}P_\mu^\pi(\boldsymbol{s}' \mid \boldsymbol{s}),$$

where we assume for simplicity that $\rho(\boldsymbol{s})$ is the reward obtained at state $\boldsymbol{s}$. The conditional measure P_μ^π is the transition kernel on $\mathcal{S}$ induced by μ, π, introduced in Section 1.1. We then select a parametric family $v_\omega : \mathcal{S} \to \mathbb{R}$ with parameter $\omega \in \Omega$ and minimise:

$$h(\omega) + \int_{\mathcal{S}} \left\| v_\omega(\boldsymbol{s}) - \rho(\boldsymbol{s}) - \gamma \int_{\mathcal{S}} v_\omega(\boldsymbol{s}') \, \mathrm{d}\hat{P}_\mu^\pi(\boldsymbol{s}'|\boldsymbol{s}) \right\| \mathrm{d}\chi(\boldsymbol{s}), \tag{8}$$

where h is a regularisation term, χ is an appropriate measure on $\mathcal{S}$ and $\hat{P}_\mu^\pi$ is an empirical estimate of the transition kernel, used to approximate the respective integral that uses P_μ^π. As we can take an arbitrary number of trajectories from μ, π, this can be as accurate as our computational capacity allows.

In practice, we minimise (8) with a generalised linear model (defined on an appropriate basis) for v_ω while χ need only be positive on a set of representative states. Specifically, we employ a variant of the least-squares policy iteration (LSPI, Lagoudakis and Parr, 2003) algorithm, using the least-squares temporal differences (LSTD, Bradtke and Barto, 1996) for the minimisation of (8). Then the norm is the euclidean norm and the regularisation term is $h(\omega) = \lambda \|\omega\|$. In order to estimate the inner integral, we take $K_{\mathrm{L}} \geq 1$ samples from the model so that

$$\hat{P}_\mu^\pi(\boldsymbol{s}' \mid \boldsymbol{s}) \triangleq \frac{1}{K_{\mathrm{L}}} \sum_{i=1}^{K_{\mathrm{L}}} \mathbb{I}\left\{ \boldsymbol{s}_{t+1}^i = \boldsymbol{s}' \mid \boldsymbol{s}_t^i = \boldsymbol{s} \right\}, \tag{9}$$

$$\boldsymbol{s}_{t+1}^i \mid \boldsymbol{s}_t^i = \boldsymbol{s} \sim P_\mu^\pi(\cdot \mid \boldsymbol{s}),$$

where $\mathbb{I}\{\cdot\}$ is an indicator function and P_μ^π is decomposable in known terms. Equation (9) is also used for action selection in order to calculate an approximate expected utility $q_\omega(\boldsymbol{s}, a)$ for each state-action pair $(\boldsymbol{s}, a)$:

$$q_\omega(\boldsymbol{s}, a) \triangleq \rho(s) + \gamma \int_{\mathcal{S}} v_\omega(\boldsymbol{s}') \, \mathrm{d}\hat{P}_\mu^\pi(\boldsymbol{s}'|\boldsymbol{s})$$

Effectively, this approximates the integral via sampling. This may add a small amount[5] of additional stochasticity to action selection, which can be reduced[6] by increasing K_{L}.

Finally, we optimise the policy by approximate policy iteration. At the j-th iteration we obtain an improved policy $\hat{\pi}_j(a \mid \boldsymbol{s}) \propto \mathbb{P}[a \in \arg\max_{a' \in \mathcal{A}} q_{\omega_{j-1}}(\boldsymbol{s}, a')]$ from ω_{j-1} and then estimate ω_j for the new policy.

5. Generally, this error is bounded by $O(K_{\mathrm{L}}^{-1/2})$.
6. We remind the reader that Thompson sampling itself results in considerable exploration by sampling an MDP from the posterior. Thus, additional randomness may be detrimental.

2.5 Complexity

We now analyse the computational complexity of our approach, including the online complexity of inference and decision making, and of the sampling and ADP taking place every episode. It is worthwhile to note two facts. Firstly, that the complexity bounds related to the cover tree depend on a constant c, which however depends on the distribution of samples in the state space. In the worst case (i.e., a uniform distribution), this is bounded exponentially in the dimensionality of the actual state space. While we do not expect this to be the case in practice, it is easy to construct a counterexample where this is the case. Secondly, that the complexity of the ADP step is largely independent of the model used, and mostly depends on the number of trajectories we take in the sampled model and the dimensionality of the feature space.

First, we examine the total computation time that is required to construct the tree.

Corollary 1 *Cover tree construction from t observations takes $O(t \ln t)$ operations.*

Proof In the cover tree, node insertion and query are $O(\ln t)$(Beygelzimer et al., 2006, Theorems 5, 6). Then note that $\sum_{k=1}^{t} \ln k \leq \sum_{k=1}^{t} \ln t = t \ln t$. ■

At every step of the process, we must update our posterior parameters. Fortunately, this also takes logarithmic time as we only need to perform calculations for a single path from the root to a leaf node.

Lemma 2 *If $\mathcal{S} \subset \mathbb{R}^m$, then inference at time step t has complexity $O(m^3 \ln t)$.*

Proof At every step, we must perform inference on a number of nodes equal to the length of the path containing the current observation. This is bounded by the depth of the tree, which is in turn bounded by $O(\ln t)$ from Beygelzimer et al. (2006, Lemma 4.3). Calculating (2) is linear in the depth. For each node, however, we must update the linear-Bayesian model, and calculate the marginal distribution. Each requires inverting an $m \times m$ matrix, which has complexity $O(m^3)$. ■

Finally, at every step we must choose an action through value function look-up. This again takes logarithmic time, but there is a scaling depending on the complexity of the value function representation.

Lemma 3 *If the LSTD basis has dimensionality m_L, then taking a decision at time t has complexity $O(K_L m_L \ln t)$.*

Proof To take a decision we merely need to search in each action tree to find a corresponding path. This takes $O(\ln t)$ time for each tree. After Thompson sampling, there will only be one linear model for each action tree. LSTD takes K_L operations, and requires the inner product of two m_L-dimensional vectors. ■

The above lemmas give the following result:

Theorem 4 *At time t, the online complexity of CTBRL is $O((m^3 + K_L m_L) \ln t)$.*

We now examine the complexity of finding a policy. Although this is the most computationally demanding part, its complexity is not dependent on the cover tree structure or the probabilistic inference method used. However, we include it here for completeness.

Lemma 5 *Thompson sampling at time t is $O(tm^3)$.*

Proof In the worst case, our sampled tree will contain all the leaf nodes of the reduced tree, which are $O(t)$. For each sampled node, the most complex operation is Wishart generation, which is $O(m^3)$ (Smith and Hocking, 1972). ■

Lemma 6 *If we use n_s samples for LSTD estimation and the basis dimensionality is m_L, this step has complexity $O(m_L^3 + n_s(m_L^2 + K_L m_L \ln t))$.*

Proof For each sample we must take a decision according to the last policy, which requires $O(K_{\mathrm{L}} m_L \ln t)$ as shown previously. We also need to update two matrices (see Boyan, 2002), which is $O(m_L^2)$. So, $O(n_s(m_L^2 + K_{\mathrm{L}} m_L \ln t))$ computations must be performed for the total number of the selected samples. Since LSTD requires an $m_L \times m_L$ matrix inversion, with complexity $O(m_L^3)$, we obtain the final result. ■

From Lemmas 3 and 6 it follows that:

Theorem 7 *If we employ API with K_A iterations, the total complexity of calculating a new policy is $O(tm^3 + K_A(m_L^3 + n_s(m_L^2 + K_L m_L \ln t)))$.*

Thus, while the online complexity of CTBRL is only logarithmic in t, there is a substantial cost when calculating a new policy. This is only partially due to the complexity of sampling a model, which is manageable when the state space has small dimensionality. Most of the computational effort is taken by the API procedure, at least as long as $t < (m_L/m)^3$. However, we think this is unavoidable no matter what the model used is.

The complexity of Gaussian process (GP) models is substantially higher. In the simplest model, where each output dimension is modelled independently, inference is $O(mt^3)$, while the fully multivariate tree model has complexity $O(m^3 t \ln t)$. Since there is no closed form method for sampling a function from the process, one must resort to iterative sampling of points. For n points, the cost is approximately $O(nmt^3)$, which makes sampling long trajectories prohibitive. For that reason, in our experiments we only use the mean of the GP.

3. Experiments

We conducted two sets of experiments to analyse the offline and the online performance. We compared CTBRL with the well-known LSPI algorithm (Lagoudakis and Parr, 2003) for the offline case, as well as an online variant (Buşoniu et al., 2010) for the online case. We also compared CTBRL with linear Bayesian reinforcement learning (LBRL, Tziortziotis et al., 2013) and finally GP-RL, where we simply replaced the tree model with a Gaussian process. For CTBRL and LBRL we use Thompson sampling. However, since Thompson sampling cannot be performed on GP models, we use the mean GP instead. In order to compute policies given a model, all model-based methods use the variant of LSPI explained in Section 2.4.2. Hence, the only significant difference between each approach is the model used, and whether or not they employ Thompson sampling.

A significant limitation of Gaussian processes is that their computational complexity becomes prohibitive as the number of samples becomes extremely large. In order to make

the GP model computationally practical, the greedy approximation approach introduced by Engel et al. (2002) has been adopted. This is a kernel sparsification methodology which incrementally constructs a dictionary of the most representative states. More specifically, an *approximate linear dependence* analysis is performed in order to examine whether a state can be approximated sufficiently as a linear combination of current dictionary members or not.

We used one preliminary run and guidance from the literature to make an initial selection of possible hyper-parameters, such as the number of samples and the features used for LSTD and LSTD-Q. We subsequently used 10 runs to select a single hyper-parameter combination for each algorithm-domain pair. The final evaluation was done over an independent set of 100 runs.

For CTBRL and the GP model, we had the liberty to draw an arbitrary number of trajectories for the value function estimation. We drew 1-step transitions from a set of 3000 uniformly drawn states from the *sampled* model (the mean model in the GP case). We used 25 API iterations on this data.

For the offline performance evaluation, we first drew rollouts from $k = \{10, 20, \ldots, 50, 100, \ldots, 1000\}$ states drawn from the *true environment's* starting distribution, using a uniformly random policy. The maximum horizon of each rollout was set equal to 40. The collected data was then fed to each algorithm in order to produce a policy. This policy was evaluated over 1000 rollouts on the environment.

In the online case, we simply use the last policy calculated by each algorithm at the end of the last episode, so there is no separate learning and evaluation phase. This means that efficient exploration must be performed. For CTBRL, this is done using Thompson sampling. For online-LSPI, we followed the approach of Buşoniu et al. (2010), who adopts an ϵ-greedy exploration scheme with an exponentially decaying schedule $\epsilon_t = \epsilon_d^t$, with $\epsilon_0 = 1$. In preliminary experiments, we found $\epsilon_d = 0.997$ to be a reasonable compromise. We compared the algorithms online for 1000 episodes.

3.1 Domains

We consider two well-known continuous state, discrete-action, episodic domains. The first is the inverted pendulum domain and the second is the mountain car domain.

3.1.1 Inverted Pendulum

The goal in this domain, is to balance a pendulum by applying forces of a mixed magnitude (50 Newtons). The state space consists of two continuous variables, the vertical angle and the angular velocity of the pendulum. There are three actions: no force, left force or right force. A zero reward is received at each time step except in the case where the pendulum falls. In this case, a negative (-1) reward is given and a new episode begins. An episode also ends with 0 reward after 3000 steps, after which we consider that the pendulum is successfully balanced. Each episode starts by setting the pendulum in a perturbed state close to the equilibrium point. More information about the specific dynamics can be found at Lagoudakis and Parr (2003). The discount factor γ was 0.95. The basis we used for LSTD/LSPI, was equidistant 3×3 grid of RBFs over the state space following the sugges-

tions of Lagoudakis and Parr (2003). This was replicated for each action for the LSTD-Q algorithm used in LSPI.

3.1.2 Mountain Car

The aim in this domain is to drive an underpowered car to the top of a hill. Two continuous variables characterise the vehicle state in the domain, its position and its velocity. The objective is to drive an underpowered vehicle up a steep valley from a randomly selected position to the right hilltop (at position > 0.5) within 1000 steps. There are three actions: forward, reverse and zero throttle. The received reward is -1 except in the case where the target is reached (zero reward). At the beginning of each rollout, the vehicle is positioned to a new state, with the position and the velocity uniformly randomly selected. The discount factor is set to $\gamma = 0.999$. An equidistant 4×4 grid of RBFs over the state space plus a constant term is selected for LSTD and LSPI.

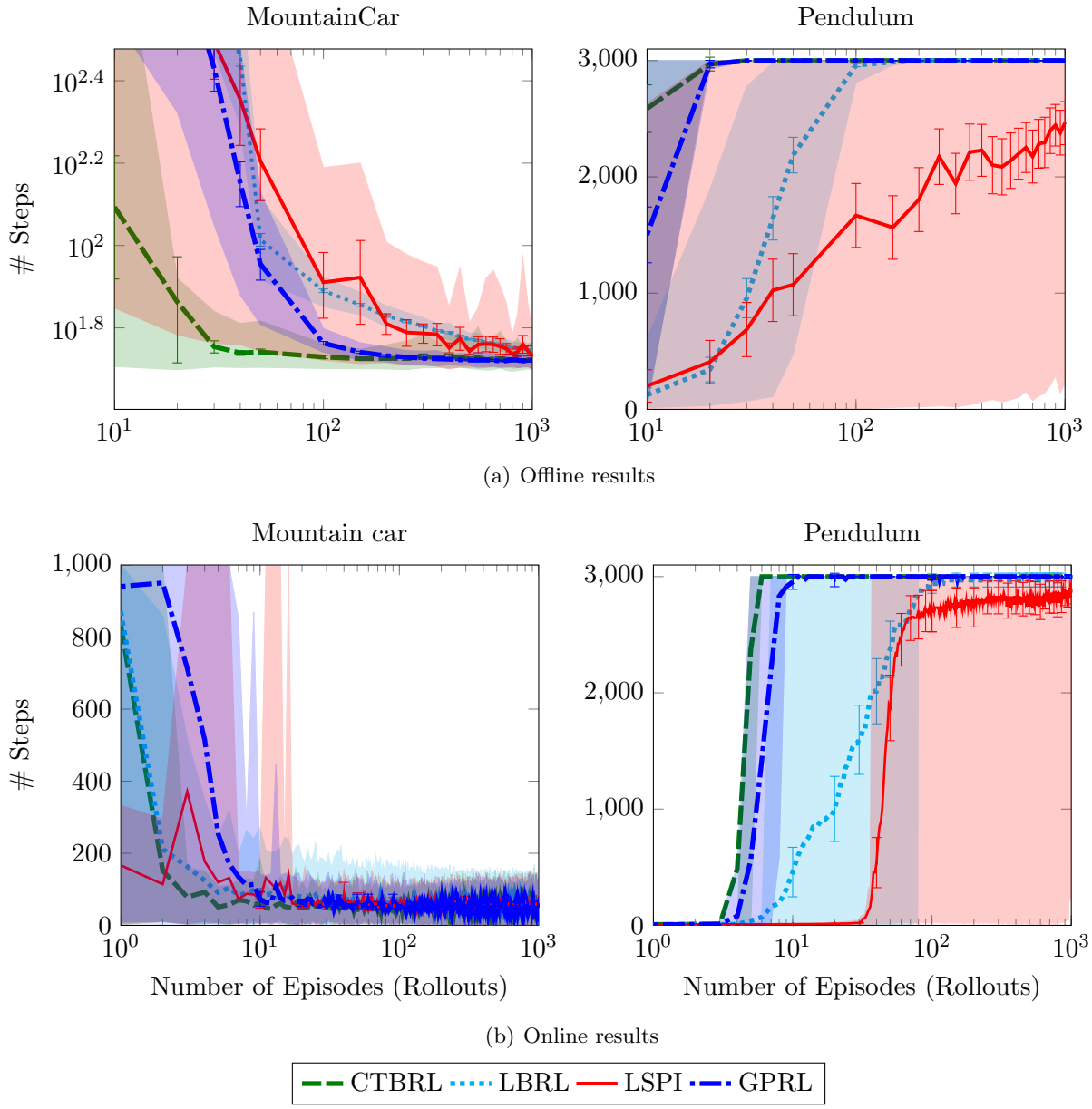
3.2 Results

In our results, we show the average performance in terms of number of steps of each method, averaged over 100 runs. For each average, we also plot the 95% confidence interval for the accuracy of the mean estimate with error bars. In addition, we show the 90% percentile region of the runs, in order to indicate inter-run variability in performance.

Figure 3(a) shows the results of the experiments in the *offline* case. For the *mountain car*, it is clear that CTBRL is significantly more stable compared to GPRL and LSPI. In contrast to the other two approaches, CTBRL needs only a small number of rollouts in order to discover the optimal policy. For the *pendulum* domain, the performance of both CTBRL and GPRL is almost perfect, as they need only about twenty rollouts in order to discover the optimal policy. On the other hand, LSPI despite the fact that manages to find the optimal policy frequently, around 5% of its runs fail.

Figure 3(b) shows the results of the experiments in the *online* case. For the *mountain car*, CTBRL managed to find an excellent policy in the vast majority of runs, while converging earlier than GPRL and LSPI. Moreover, CTBRL presents a more stable behaviour in contrast to the other two. In the *pendulum* domain, the performance difference relative to LSPI is even more impressive. It becomes apparent that both CTBRL and GPRL reach near optimal performances with an order of magnitude fewer episodes than LSPI, while the latter remains unstable. In this experiment, we see that CTBRL reaches an optimal policy slightly before GPRL. Although the difference is small, it is very consistent.

The success of CTBRL over the other approaches can be attributed to a number of reasons. Firstly, it could be a better model. Indeed, in the offline results for the mountain car domain, where the starting state distribution is uniform, and all methods have the same data, we can see that CTBRL has a far better performance than everything else. The second could be the more efficient exploration afforded by Thompson sampling. Indeed, in the mountain car online experiments we see that the LBRL performs quite well (Fig. 3(b)), even though its offline performance is not very good (Fig. 3(a)). However, Thompson sampling is not sufficient for obtaining a good performance, as seen by both the offline results and the performance in the pendulum domain.



(a) Offline results

(b) Online results

Figure 3: Experimental evaluation. The dashed line shows CTBRL, the dotted line shows LBRL, the solid line shows LSPI, while the dash-dotted line shows GPRL. The error bars denote 95% confidence intervals for the mean (i.e., statistical significance). The shaded regions denote 90% percentile performance (i.e., robustness) across runs. In all cases, CTBRL converges significantly quicker than the other approaches. In addition, as the percentile regions show, it is also much more stable than LBRL, GPRL and LSPI.

4. Conclusion

We proposed a computationally efficient, fully Bayesian approach for the exact inference of unknown dynamics in continuous state spaces. The total computation for inference after t steps is $O(t \ln t)$, in stark contrast to other non-parametric models such as Gaussian processes, which scale $O(t^3)$. In addition, inference is naturally performed online, with the computational cost at time t being $O(\ln t)$.

In practice, the computational complexity is orders of magnitude lower for cover trees than GP, even for these problems. We had to use a dictionary and a lot of tuning to make GP methods work, while cover trees worked out of the box. Another disadvantage of GP methods is that it is infeasible to implement Thompson sampling with them. This is because it is not possible to directly sample a function from the GP posterior. Although Thompson sampling confers no advantage in the offline experiments (as the data there were the same for all methods), we still see that the performance of CTBRL is significantly better on average and that it is much more stable.

Experimentally, we showed that cover trees are more efficient both in terms of computation and in terms of reward, relative to GP models that used the same ADP method to optimise the policy and to a linear Bayesian model which used both the same ADP method and the same exploration strategy. We can see that overall the linear model performs significantly worse than both GP-RL and CTBRL, though better than ϵ-greedy LSPI. This shows that the main reason for the success of CTBRL is the cover tree inference and not the linear model itself, or Thompson sampling.

CTBRL is particularly good in online settings, where the exact inference, combined with the efficient exploration provided by Thompson sampling give it an additional advantage. We thus believe that CTBRL is a method that is well-suited for exploration in unknown continuous state problems. Unfortunately, it is not possible to implement Thompson sampling in practice using GPs, as there is no reasonable way to sample a function from the GP posterior. Nevertheless, we found that in both online and offline experiments (where Thompson sampling should be at a disadvantage) the cover tree method achieved superior performance to Gaussian processes.

Although we have demonstrated the method in low dimensional problems, higher dimensions are not a problem for the cover tree inference itself. The bottleneck is the value function estimation and ADP. This is independent of the model used, however. For example, GP methods for estimating the value function (c.f. Deisenroth et al., 2009) typically have a large number of hyper-parameters for value function estimation, such as choice of representative states and trajectories, kernel parameters and method for updating the dictionary, to avoid problems with many observations.

While in practice ADP can be performed in the background while inference is taking place, and although we seed the ADP with the previous solution, one would ideally like to use a more incremental approach for that purpose. One interesting idea would be to employ a gradient approach in a similar vein to Deisenroth and Rasmussen (2011). An alternative approach would be to employ an online method, in order to avoid estimating a policy for the complete space.[7] Promising such approaches include running bandit-based tree search methods such as UCT (Kocsis and Szepesvári, 2006) on the sampled models.

7. A suggestion made by the anonymous reviewers.

Another direction of future work is to consider more sophisticated exploration policies, particularly for larger problems. Due to the efficiency of the model, it should be possible to compute near-Bayes-optimal policies by applying the tree search method used by Veness et al. (2011). Finally, it would be interesting to examine continuous actions. These can be handled efficiently both by the cover tree and the local linear models by making the next state directly dependent on the action through an augmented linear model. While optimising over a continuous action space is challenging, more recent efficient tree search methods such as metric bandits (Bubeck et al., 2011) may alleviate that problem.

An interesting theoretical direction would be to obtain regret bounds for the problem. This could perhaps be done building upon the analyses of Kozat et al. (2007) for context tree prediction, and of Ortner and Ryabko (2012) for continuous MDPs. The statistical efficiency of the method could be improved by considering edge-based (rather than node-based) distributions on trees, as was suggested by Pereira and Singer (1999).

Finally, as the cover tree method only requires specifying an appropriate metric, the method could be applicable to many other problems. This includes both large discrete problems, and partially observable problems. It would be interesting to see if the approach also gives good results in those cases.

Acknowledgements

We would like to thank the anonymous reviewers, for their careful and detailed comments and suggestions, for this and previous versions of the paper, which have significantly improved the manuscript. We also want to thank Mikael Kågebäck for additional proofreading. This work was partially supported by the Marie Curie Project ESDEMUU "Efficient Sequential Decision Making Under Uncertainty" , Grant Number 237816 and by an ERASMUS exchange grant.

References

P. Abbeel and A. Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *International Conference on Machine learning (ICML)*, pages 1–8, 2005.

S. Agrawal and N. Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory (COLT)*, pages 39.1–39.26, 2012.

M. Alvarez, D. Luengo-Garcia, M. Titsias, and N. Lawrence. Efficient multioutput Gaussian processes through variational inducing kernels. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 25–32, 2011.

M. Araya, V. Thomas, and O. Buffet. Near-optimal BRL using optimistic local transitions. In *International Conference on Machine Learning (ICML)*, pages 97–104, 2012.

J. Asmuth, L. Li, M. L. Littman, A. Nouri, and D. Wingate. A Bayesian sampling approach to exploration in reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 19–26, 2009.

P. Auer, T. Jaksch, and R. Ortner. Near-optimal regret bounds for reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 89–96, 2008.

R. Begleiter, R. El-Yaniv, and G. Yona. On prediction using variable order Markov models. *Journal of Artificial Intelligence Research*, 22:385–421, 2004.

M. Bellemare, J. Veness, and M. Bowling. Bayesian learning of recursively factored environments. In *International Conference on Machine Learning (ICML)*, pages 1211–1219, 2013.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *International Conference on Machine Learning (ICML)*, pages 97–104, 2006.

J. A. Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49(2):233–246, 2002.

S. J. Bradtke and A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1):33–57, 1996.

E. Brunskill, B. R. Leffler, L. Li, M. L. Littman, and N. Roy. Provably efficient learning with type parametric models. *Journal of Machine Learning Research*, 10:1955–1988, 2009.

S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári. X-armed bandits. *Journal of Machine Learning Research*, 12:1655–1695, 2011.

L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška. Online least-squares policy iteration for reinforcement learning control. In *American Control Conference (ACC)*, pages 486–491, 2010.

P. Castro and D. Precup. Smarter sampling in model-based Bayesian reinforcement learning. *Machine Learning and Knowledge Discovery in Databases*, pages 200–214, 2010.

M. Daswani, P. Sunehag, and M. Hutter. Feature reinforcement learning using looping suffix trees. In *European Workshop on Reinforcement Learning (EWRL)*, pages 11–24, 2012.

R. Dearden, N. Friedman, and S. J. Russell. Bayesian Q-learning. In *AAAI/IAAI*, pages 761–768, 1998. URL `citeseer.ist.psu.edu/dearden98bayesian.html`.

M. H. DeGroot. *Optimal Statistical Decisions*. John Wiley & Sons, 1970.

M. P. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning (ICML)*, pages 465–472, 2011.

M. P. Deisenroth, C. E. Rasmussen, and J. Peters. Gaussian process dynamic programming. *Neurocomputing*, 72(7-9):1508–1524, 2009.

C. Dimitrakakis. Bayesian variable order Markov models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 161–168, 2010a.

C. Dimitrakakis. Complexity of stochastic branch and bound methods for belief tree search in Bayesian reinforcement learning. In *International Conference on Agents and Artificial Intelligence (ICAART)*, pages 259–264, 2010b.

C. Dimitrakakis. Robust bayesian reinforcement learning through tight lower bounds. In *European Workshop on Reinforcement Learning (EWRL)*, pages 177–188, 2011.

C. Dimitrakakis, N. Tziortziotis, and A. Tossou. Beliefbox: A framework for statistical methods in sequential decision making. `http://code.google.com/p/beliefbox/`, 2007.

M. Duff. *Optimal Learning Computational Procedures for Bayes-adaptive Markov Decision Processes*. PhD thesis, University of Massachusetts at Amherst, 2002.

Y. Engel, S. Mannor, and R. Meir. Sparse online greedy support vector regression. In *European Conference on Machine Learning (ECML)*, pages 84–96, 2002.

Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian process. In *International Conference on Machine Learning (ICML)*, pages 201–208, 2005.

D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

V. F. Farias, C. C. Moallemi, B. Van Roy, and T. Weissman. Universal reinforcement learning. *IEEE Transactions on Information Theory*, 56(5):2441–2454, 2010.

T. S. Ferguson. Prior distributions on spaces of probability measures. *The Annals of Statistics*, 2(4):615–629, 1974.

M. Ghavamzadeh and Y. Engel. Bayesian policy gradient algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, pages 457–464, 2006.

E. Kaufmann, N. Korda, and R. Munos. Thompson sampling: An optimal finite time analysis. In *Algorithmic Learning Theory (ALT)*, pages 199–213, 2012.

L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning (ECML)*, pages 282–293, 2006.

S. S. Kozat, A. C. Singer, and G. C. Zeitler. Universal piecewise linear prediction via context trees. *IEEE Transactions on Signal Processing*, 55(7):3730–3745, 2007.

M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

M. Meila and M. I. Jordan. Learning with mixtures of trees. *The Journal of Machine Learning Research*, 1:1–48, 2001.

R. Ortner and D. Ryabko. Online regret bounds for undiscounted continuous reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1772–1780, 2012.

I. Osband, D. Russo, and B. Van Roy. (More) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3003–3011, 2013.

S. M. Paddock, F. Ruggeri, M. Lavine, and M. West. Randomized Polya tree models for nonparametric Bayesian inference. *Statistica Sinica*, 13(2):443–460, 2003.

F. C. Pereira and Y. Singer. An efficient extension to mixture techniques for prediction and decision trees. *Machine Learning*, 36(3):183–199, 1999.

P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete Bayesian reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 697–704, 2006.

M. L. Puterman. *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New Jersey, US, 2005.

C. E. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 751–759, 2004.

W. B. Smith and R. R. Hocking. Wishart variates generator, algorithm AS 53. *Applied Statistics*, 21:341–345, 1972.

A. L. Strehl and M. L. Littman. Online linear regression and its application to model-based reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1417–1424, 2008.

M. Strens. A Bayesian framework for reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 943–950, 2000.

W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

N. Tziortziotis, C. Dimitrakakis, and K. Blekas. Linear Bayesian reinforcement learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1721–1728, 2013.

T. van Erven, P. D. Grünwald, and S. de Rooij. Catching up faster by switching sooner: a prequential solution to the AIC-BIC dilemma. *arXiv*, 2008. A preliminary version appeared in NIPS 2007.

J. Veness, K. S. Ng, M. Hutter, W. Uther, and D. Silver. A Monte-Carlo AIXI approximation. *Journal of Artificial Intelligence Research*, 40:95–142, 2011.

J. Veness, K. S. Ng, M. Hutter, and M. Bowling. Context tree switching. In *Data Compression Conference (DCC)*, pages 327–336, 2012.

T. Wang, D. Lizotte, M. Bowling, and D. Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *International Conference on Machine Learning (ICML)*, pages 956–963, 2005.

F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens. The context tree weighting method: basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.

W. H. Wong and L. Ma. Optional Pólya tree and Bayesian inference. *The Annals of Statistics*, 38(3):1433–1459, 2010.

J. Wyatt. *Exploration and inference in learning from reinforcement*. PhD thesis, University of Edinburgh. College of Science and Engineering. School of Informatics, 1998.

Journal of Machine Learning Research 15 (2014) 2337-2397 Submitted 10/12; Revised 2/14; Published 7/14

Efficient State-Space Inference of Periodic Latent Force Models

Steven Reece REECE@ROBOTS.OX.AC.UK
Department of Engineering Science
University of Oxford
Parks Road
Oxford OX1 3PJ, UK

Siddhartha Ghosh SG2@ECS.SOTON.AC.UK
Alex Rogers ACR@ECS.SOTON.AC.UK
Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK

Stephen Roberts SJROB@ROBOTS.OX.AC.UK
Department of Engineering Science
University of Oxford
Parks Road
Oxford OX1 3PJ, UK

Nicholas R. Jennings NRJ@ECS.SOTON.AC.UK
Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK
and
Department of Computing and Information Technology
King Abdulaziz University
Saudi Arabia

Editor: Neil Lawrence

Abstract

Latent force models (LFM) are principled approaches to incorporating solutions to differential equations within non-parametric inference methods. Unfortunately, the development and application of LFMs can be inhibited by their computational cost, especially when closed-form solutions for the LFM are unavailable, as is the case in many real world problems where these latent forces exhibit periodic behaviour. Given this, we develop a new sparse representation of LFMs which considerably improves their computational efficiency, as well as broadening their applicability, in a principled way, to domains with periodic or near periodic latent forces. Our approach uses a linear basis model to approximate one generative model for each periodic force. We assume that the latent forces are generated from Gaussian process priors and develop a linear basis model which fully expresses these priors. We apply our approach to model the thermal dynamics of domestic buildings and show that it is effective at predicting day-ahead temperatures within the homes. We also apply our approach within queueing theory in which quasi-periodic arrival rates are modelled as latent forces. In both cases, we demonstrate that our approach can be implemented

efficiently using state-space methods which encode the linear dynamic systems via LFMs. Further, we show that state estimates obtained using periodic latent force models can reduce the root mean squared error to 17% of that from non-periodic models and 27% of the nearest rival approach which is the resonator model (Särkkä et al., 2012; Hartikainen et al., 2012).

Keywords: latent force models, Gaussian processes, Kalman filter, kernel principle component analysis, queueing theory

1. Introduction

Latent force models (LFMs) have received considerable interest in the machine learning community as they combine underlying physical knowledge of a system with data driven models expressed as Bayesian non-parametric Gaussian process (GP) priors (see, for example, Alvarez et al., 2009; Hartikainen and Särkkä, 2010). In more detail, the physical process that generates the data is typically represented by one or more differential equations. These differential equations can then be accommodated within covariance functions along with the data driven priors. Doing so allows inferences to be drawn in regimes where data may be sparse or absent, where a purely data driven model will typically perform poorly. To date, such models have been applied in areas such as computational biology and understanding motion patterns (Alvarez et al., 2009, 2010).

Despite growing interest in LFMs, their real world applicability has been limited as inference using LFMs expressed directly through covariance functions can be computationally prohibitive on large data sets. It is well known that regression with GPs imposes high computational cost which scales as $\mathcal{O}(N^3T^3)$ during training, where N is the dimension of the data observed at each time point and T is the number of time points. However, it has also been shown that training LFMs using state-space methods can be considerably less computationally demanding (Rasmussen and Williams, 2006; Hartikainen and Särkkä, 2010) as state-space methods scale as $\mathcal{O}(N^3T)$. It is this computational saving that motivates the state-space approach to LFM inference in this paper.

The state-space approach to LFM inference advocated by Hartikainen and Särkkä (2010, 2011) augments the state vector so that Matérn and squared-exponential priors can be accommodated (although only approximately in the case of the squared-exponential). All the information encoded within the GP prior (that is, process smoothness, stationarity etc) is fully captured within their state-space representation. However, their approach assumes that the LFM kernel's inverse power spectrum can be represented by a power series in the frequency domain. Unfortunately, this requirement severely inhibits the applicability of their approach and, consequently, only a small repertoire of GP priors have been investigated within LFMs to date, namely, squared-exponential and Matérn kernels. Specifically, the state-space approach advocated by Hartikainen and Särkkä (2010) does not accommodate periodic kernels as we shall demonstrate in this paper. This is a key limitation as periodicity is common in many physical processes as we shall demonstrate in our empirical evaluation. Expressing our prior knowledge of the periodicity, as a GP prior, within the state-space approach is the key challenge problem addressed in this paper.

Thus, against this background, we describe a principled method for incorporating stationary periodic, non-stationary periodic and quasi-periodic Gaussian process priors within state-space approaches to LFM inference. Within our approach all LFM parameters can be

inferred using Bayesian methods or maximum likelihood and thus we circumvent the need to set any of these parameters by hand. Further, to accommodate periodic and quasi-periodic models within LFMs we develop a novel state-space approach to inference. In particular, we propose to represent periodic and quasi-periodic driving forces, which are assumed smooth, by linear basis models (LBMs) with eigenfunction basis functions derived using kernel principal component analysis (KPCA) in the temporal domain. These basis models, although parametric in form, are optimized so that their generative properties accurately approximate the driving force kernel. We will show that efficient inference can then be performed using a state-space approach by augmenting the state with additional variables which sparsely represent the periodic latent forces.

Our LBM approach to accommodating periodic kernels is inspired by the *resonator model* (Särkkä et al., 2012; Hartikainen et al., 2012) in which the periodic process is modelled as a superposition of *resonators*, each of which can be represented within the state-vector. Unfortunately, the resonator model, in its current form, does not encode all the underlying GP prior information of the periodic process as the resonator is not tailored to accommodate all the prior information encoded via the covariance function (see Section 4 for more detail). An alternative approach to modelling stationary kernels, including periodic kernels, is sparse spectrum Gaussian process regression (SSGPR) of Lázaro-Gredilla et al. (2010). This approach is similar in spirit to the resonator model in that it encodes stationary GP priors via basis functions (sinusoidal functions, in this case). However, unlike the resonator model, the SSGPR is able to encode the GP prior by reinterpreting the spectral density of a stationary GP kernel as a probability density function over frequency space. This pdf is then sampled using Monte Carlo to yield the frequencies of the sinusoidal basis functions. Unfortunately, this stochastic approach can often provide a poor approximation to the covariance function (see Section 5 for more detail).

We shall develop a LBM which captures all the information encoded within the GP prior and demonstrate its superior accuracy over the resonator model and the SSGPR. We shall also establish the close link between the resonator basis and the eigenfunction basis used in our approach and consequently, derive a novel method for tailoring the resonator basis to accommodate all the information encoded within the covariance function.

Our research is driven by two specific applications although the methods that we propose are of general applicability. Specifically, we apply our approach to the estimation and prediction of the behaviour of customer queues in call centres, based on flow models of queue dynamics represented as LFMs. The behaviour of queues is of general importance in several applications including communication networks (Wang et al., 1996), weather monitoring (Sims et al., 2005) and truck coordination at ports (Ji and Zhou, 2010). Accurate predictions of the customer queue arrival rates based on an underlying LFM is a key requirement for determining the number of call centre agents required at various times throughout the day. We also apply our approach to the estimation and prediction of the internal temperature within a home, based on thermal models of home heating systems represented as LFMs. Accurate predictions of the internal temperature based on an underlying LFM is a key component for predicting energy used in heating a home and, consequently, an integral part of many home energy saving systems (Bacher and Madsen, 2011). These applications demonstrate our approach under two different modelling conditions, the queue LFM is nonlinear whereas the thermal LFM is linear, while the queue application is a tracking

application and regular measurements are available whereas the thermal application requires long term predictions (a day ahead) during which no measurements are available.

In more detail, telephone call centre managers are concerned with staffing and specifically, assigning the appropriate number of agents to guarantee that the customers' queueing time does not prohibit sales (Feigin et al., 2006). Although there is significant literature on attempts to accurately model the dynamics of queues, it has failed to offer a method for inferring the highly quasi-periodic arrival rates from sparse measurements of the queue lengths (Wang et al., 1996). Determining such arrival rates is key to predicting queue lengths, and hence customer waiting times. These predictions help the call centre manager to plan staffing throughout the day to ensure an acceptable customer waiting time. We will demonstrate that our approach to modelling LFMs is capable of inferring these unknown arrival rates. Furthermore, although the dynamic system in this application is nonlinear and the arrival rate is quasi-periodic, it is still Markovian and, consequently, a state-space approach to inference is ideally suited to this application.

Energy saving in homes is a key issue as governments aim to reduce the carbon footprint of their countries. A significant amount of energy is expended in heating homes and home owners need to be encouraged to reduce their energy consumption and carbon emissions incurred through home heating (MacKay, 2009; DECC, 2009). Consequently, we apply our approach to the estimation and prediction of internal temperatures using thermal models of home heating systems. Our approach allows us to make day ahead predictions of the energy usage, which can then be fed back to the householder in real-time so that they can take appropriate mitigating actions to reduce their energy consumption. Home heating systems typically consist of a thermostat with a set-point that controls the activations of a gas or electrical boiler to ensure that the internal temperature follows the set-point. Although there is significant literature on attempts to accurately model the thermal dynamics of buildings, it has failed to take into account the daily human behaviours within their homes, which can have a significant impact on the energy signatures obtained from similar homes (Bacher and Madsen, 2011). For instance, during cold periods, a householder may deploy an additional heater or, in hot periods, open a window. Furthermore, the thermal dynamics of real homes are more complex in reality than existing thermal models suggest; sunlight through windows contributes to extra heat while open windows cause heat loss. Residual heat can also be retained by thermal blocks such as walls and ceilings that then re-radiate heat. Crucially, many of these heat sources are periodic in nature. For instance, an additional heater may be switched on every night during cold periods, whilst the diurnal sun cycle will contribute additional heat during the day. We will demonstrate that our approach is capable of inferring these unknown periodic heat sources. Again, the dynamic system in this application is linear and Markovian and, consequently, a state-space approach to inference is again ideally suited to this problem.

In undertaking this work, we advance the state of the art in the following ways:

- We offer the only principled approach to incorporating all Gaussian process prior models within a state-space approach to inference with LFMs.[1]

1. What this paper does not aim to establish is the value of GP models per se over other models. The paper thus focuses on developing efficient, scalable representations and tools for performing GP inference.

- We are the first to demonstrate that the eigenfunction model of Gaussian process priors out-performs an alternative approach to modelling periodic Gaussian process priors; namely, the sparse spectrum Gaussian process regression (SSGPR) approach developed by Lázaro-Gredilla et al. (2010).

- We demonstrate, for the first time, the close link between the eigenfunction model and the resonator model (Särkkä et al., 2012; Hartikainen et al., 2012; Solin and Särkkä, 2013). Consequently, we offer a novel mechanism for incorporating all information encoded within the latent force covariance function into the resonator model.

- We propose the only approach that is able to incorporate all types of periodic Gaussian process priors within a state-space approach to LFM inference. These priors include stationary periodic, non-stationary periodic and quasi-periodic covariance functions.

- We are the first to apply LFMs to queueing theory, specifically to the modelling of queue arrival rates. Through empirical evaluation, we show that for tracking the customer queue lengths in the call centre application, the RMSE of our approach using a quasi-periodic kernel model of the arrival rate can be 17% of that using the same approach with a non-periodic kernel model.

- We are the first to apply LFMs to the modelling of thermal dynamics within real homes, specifically to unknown physical thermal processes. We show that for day ahead predictions of temperature in homes, the RMSE of our approach is 45% of that obtained using the resonator model (Solin and Särkkä, 2013) when the latent forces exhibit quasi-periodic behaviour.

The structure of our paper is as follows: in Section 2 we review approaches to regression and time-series analysis using Gaussian processes and the Kalman filter. In Section 3 we review LFMs with a particular focus on periodic latent forces and then in Section 4 we present a critique of the existing state-space approaches to inference with LFMs. In Section 5 we present a novel approach to representing periodic LFMs by linear basis models. We critique the existing spectral models for representing periodic, stationary Gaussian process priors and argue that kernel principal component analysis is the most effective approach to inferring the Fourier basis for the corresponding LBMs. In Section 6 we extend our approach to representing quasi-periodic latent forces by linear basis models. Then, in Section 7 (with further details in Appendix A), we derive a state-space approach to inference with LFMs which accommodates both periodic and quasi-periodic forces via LBMs. In Section 8, we empirically demonstrate the utility of our approach in tracking the length of call centre customer queues in the presence of, initially, unknown arrival rates which are modelled as latent forces. In Section 9 we also apply our approach to predicting the internal temperature of homes in the presence of, a priori, unknown residual heat periodic latent forces. Furthermore, we demonstrate our approach on both single output and multi-output Gaussian process thermal models. We conclude in Section 10. Finally, in Appendix B we demonstrate the theoretical link between the eigenfunction basis used in our approach and the basis used within the resonator model. Consequently, we offer a novel method for encoding periodic latent force covariance functions within the resonator model.

2. A Review of Gaussian Process Priors and Inference

A Gaussian process (GP) is often thought of as a Gaussian distribution over functions (Rasmussen and Williams, 2006). A GP is fully described by its *mean function*, μ, and *covariance function*, K. A draw, f, from a GP is traditionally written as

$$f \sim \mathcal{GP}(\mu, K).$$

The value of the function, f, at inputs X is denoted $f(X)$. Similarly, the value of the mean function and covariance function at these inputs are denoted $\mu(X)$ and $K(X, X)$, respectively. The meaning of a GP becomes clear when we consider that, for any finite set of inputs, X, $f(X)$ is a draw from a multi-variate Gaussian, $f(X) \sim \mathcal{N}(\mu(X), K(X, X))$.

Suppose we have a set of training data

$$D = \{(x_1, y_1), \ldots, (x_n, y_n)\}, \tag{1}$$

drawn from a function, f with

$$y_i = f(x_i) + \epsilon_i,$$

where ϵ_i is a zero-mean Gaussian random variable with variance σ^2. For convenience both inputs and outputs are aggregated into sets $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$, respectively. The GP estimates the value of the function f at test inputs $X_* = \{x_{*1}, \ldots, x_{*m}\}$. The basic GP regression equations are given by

$$\bar{f}_* = \mu(X_*) + K(X_*, X)[K(X, X) + \sigma^2 I]^{-1}(Y - \mu(X)), \tag{2}$$

$$\mathrm{Var}(f_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma^2 I]^{-1} K(X_*, X)^T, \tag{3}$$

where I is the identity matrix, $\bar{f}_*$ is the posterior mean function at X_* and $\mathrm{Var}(f_*)$ is the posterior covariance (Rasmussen and Williams, 2006). The inversion operation present in Equations (2) and (3) is the source of the cubic computational complexity reported in the previous section.

The matrix $K(X, X)$ is the covariance of the Gaussian prior distribution over $f(X)$. The covariance matrix has elements

$$K(x_i, x_j) = \mathrm{Cov}(f(x_i), f(x_j)),$$

where the term $K(X_*, X)$ is the covariance between the function, f, evaluated at the test inputs X_* and the training inputs X. The function K is alternatively called the *kernel* or the *covariance function*. There are many off-the-shelf kernels available (see, for example, Rasmussen and Williams, 2006) and appropriate kernels are chosen to model functions with requisite qualitative properties such as smoothness and stationarity. Further, basic kernels can be combined together to form more sophisticated kernels tailored to particular modelling needs. The mean function encodes our prior knowledge of the function mean. For ease of exposition we will assume that the mean function is zero a priori although the approaches to GP inference presented in later sections are not limited to this case.

The GP parameters θ (which includes σ and hyperparameters associated with the covariance function) can be inferred from the data through Bayes' rule

$$p(\theta \mid Y, X) = \frac{p(Y \mid X, \theta)}{p(Y \mid X)} p(\theta).$$

The parameters are usually given a vague prior distribution $p(\theta)$. In this paper, since our applications in Sections 8 and 9 exploit large data sets, we use maximum likelihood to infer the parameters and identify the assumed unique value for θ which maximizes $p(Y \mid X, \theta)$. This approach is preferred over full Bayesian marginalisation (Bishop, 1999) as the preponderance of data in the applications we consider produces very tight posterior distributions over the parameters.

When the Gaussian process models a time series then the input variables, X, are values of time. We shall assume that increasing input indices correspond to sequential time stamps, $x_1 \leq x_2 \leq \ldots \leq x_{n-1} \leq x_n$. We are at liberty to deploy GP inference using Equations (2) and (3) to either *interpolate* the function $f(x_*)$ at x_* when $x_1 < x_* < x_n$ or *extrapolate* $f(x_*)$ when either $x_* < x_1$ or $x_* > x_n$. When measurements are obtained sequentially, extrapolation forward in time is termed *prediction* and the inference of $f(x_*)$ is termed *filtering*. Interpolation with sequential measurements is termed *smoothing*. Although both smoothing and filtering approaches have been developed for Gaussian process regression (Hartikainen and Särkkä, 2010), we shall be concerned with filtering only. However, the eigenfunction models for periodic Gaussian processes developed in this paper can also be used for smoothing.

In the next section we review the latent force model (LFM) which is a principled approach to incorporating solutions to differential equations within Gaussian process inference methods.

3. Latent Force Models

In this section we present a brief introduction to latent force models and describe their practical limitations. Specifically, we consider dynamic processes which can be described by a set of E coupled, stochastic, linear, first order differential equations

$$\frac{dz_q(t)}{dt} = \sum_{e=1}^{E} F_{e,q} z_e(t) + \sum_{r=1}^{R} L_{r,q} u_r(t),$$

where q and e index each variable z, R is the number of latent forces and r indexes each latent force u, and L and F are coefficients of the system. For example, in our home heating application (described in detail in Section 9), $z_1(t)$ models the internal temperature of a home, $z_2(t)$ the ambient temperature immediately outside the home, $u_1(t)$ is the heater output from a known proportional controller and $u_2(t)$ is an unknown residual force. In this application, we assume $u_2(t)$ is periodic as it is used to model solar warming, some habitual human behaviour and the thermal lags in the heating system. The resulting differential equations can be written as

$$\frac{d\mathbf{z}(t)}{dt} = \mathbf{F}\ \mathbf{z}(t) + \mathbf{L}\mathbf{u}(t), \tag{4}$$

where $\mathbf{u}(t)$ is a vector of R independent *driving forces* (also called the *latent forces*). We distinguish non-periodic latent forces, `np`, and periodic latent forces, `p`, as they will be modelled differently in our approach. Non-periodic forces will be modelled using the existing approach advocated in Hartikainen and Särkkä (2010), which is reviewed in Section 4, and periodic forces will be modelled using our novel linear basis approach presented in Section 5. In Equation (4) the $E \times E$ matrix $\mathbf{F}$ and the $E \times R$ matrix $\mathbf{L}$ are non-random coefficients that link the latent forces to the dynamic processes. Although we deal with first order differential equations only, all higher order differential equations can be converted to a set of coupled first order equations (Hartikainen and Särkkä, 2010).

Following Alvarez et al. (2009) and Hartikainen and Särkkä (2010, 2011) we assume that the latent forces, $\mathbf{u}$, are independent draws from Gaussian processes, $u_i \sim \mathcal{GP}(0, K_i)$ where K_i is the GP covariance function (Rasmussen and Williams, 2006) for force u_i. Consequently, the covariance for $\mathbf{z}$ at any times t and t' can be evaluated as

$$E[(\mathbf{z}(t) - \bar{\mathbf{z}}(t))(\mathbf{z}(t') - \bar{\mathbf{z}}(t'))^T] = \mathbf{\Phi}(t_0, t)\mathbf{P}_z^0\mathbf{\Phi}(t_0, t')^T + \Gamma(t_0, t, t'), \tag{5}$$

where $\mathbf{\Phi}(t_0, t)$ denotes the matrix exponential, $\mathbf{\Phi}(t_0, t) = \exp(\mathbf{F}(t - t_0))$ expressed in Alvarez et al. (2009), $\bar{\mathbf{z}}(t) = E[\mathbf{z}(t)]$ and[2]

$$\Gamma(t_0, t, t') = \int_{t_0}^{t}\int_{t_0}^{t'} \mathbf{\Phi}(s, t)\mathbf{L}\mathbf{K}(s, s')\mathbf{L}^T\mathbf{\Phi}(s', t')^T \; ds \; ds'.$$

$\mathbf{P}_z^0$ is the state covariance at time t_0, $\mathbf{P}_z^0 = E[(\mathbf{z}(t_0) - \bar{\mathbf{z}}(t_0))(\mathbf{z}(t_0) - \bar{\mathbf{z}}(t_0))^T]$ and $\mathbf{K}(s, s')$ is the diagonal matrix $\mathbf{K}(s, s') = \mathrm{diag}(K_1(s, s'), \ldots, K_R(s, s'))$. Since $\mathbf{z}(t)$ is a vector and defined for any times t and t' then $E[(\mathbf{z}(t) - \bar{\mathbf{z}}(t))(\mathbf{z}(t') - \bar{\mathbf{z}}(t'))^T]$ is a multi-output Gaussian process covariance function. A kernel for covariances between the target, $\mathbf{z}$, and the latent forces, $\mathbf{u}$, can also be derived. Inference with these kernels is then undertaken directly using Equations (2) and (3).

Unfortunately, a naïve implementation of LFM inference using Equations (2) and (3) and covariance functions derived using Equation (5) can be computationally prohibitive. As we have already pointed out, this approach can be computationally expensive due to the need to invert prohibitively large covariance matrices. To mitigate computational intensive matrix inversion in the GP equations, various sparse solutions have been proposed (see, for example, Williams and Seeger, 2001; Snelson and Ghahramani, 2006; Lázaro-Gredilla et al., 2010) and an early review of some of these methods is presented in Quiñonero Candela and Rasmussen (2005). Unfortunately, the spectral decomposition approach of Lázaro-Gredilla et al. (2010) is sub-optimal in that it randomly assigns the components of a sparse spectral representation and this limitation is explored in detail in Section 5. The Nyström method for approximating eigenfunctions is used in Williams and Seeger (2001) to derive a sparse approximation for the kernel which can then be used to improve the computational efficiency of the GP inference Equations (2) and (3). Unfortunately, this approximate kernel is not used consistently throughout the GP equations and this can lead incorrectly to negative predicted variances.

The pseudo-input approach (also called *inducing inputs,* Snelson and Ghahramani, 2006; Quiñonero Candela and Rasmussen, 2005) is a successful method for reducing the number

2. All integrals in this paper should be interpreted as Itô integrals.

of input samples used within GP inference without significantly losing information encoded within the full data set. In essence, densely packed samples are summarized around sparsely distributed inducing points. Pseudo-inputs have been successfully deployed within sparse approximations of dependent output Gaussian processes (Alvarez and Lawrence, 2008, 2011). Pseudo-inputs have recently been introduced to GP time-series inference and applied to problems which exploit differential equations of the physical process via the latent force model (Alvarez et al., 2011). In Alvarez et al. (2011) the latent force is expressed at pseudo-inputs and then convolved with a smooth function to interpolate between the pseudo-inputs. However, although inducing inputs can reduce the sampling rate and summarize local information, they still have to be liberally distributed over the entire time sequence. We may assume, for simplicity, the pseudo-inputs are evenly spread over time and, therefore, the number of pseudo-inputs, P, would have to increase linearly with the number of observations (although with a rate considerably lower than the observation sampling rate). Unfortunately, the computational complexity of GP inference with pseudo-inputs is $\mathcal{O}(TP^2)$ where T is the number of observations (Alvarez and Lawrence, 2008). Thus, although pseudo-inputs are able to improve the efficiency of GP inference to some extent, for time series analysis their computational cost is still cubic in the number of measurements and this can be computationally prohibitive.

In the next section we describe a state-space reformulation of the LFM. The state-space approach has the advantage that it has a computational complexity for inferring the target process, $\mathbf{z}$, which is $\mathcal{O}(T)$ but at the expense of representing the target process with extra *state* variables.

4. State-Space Approaches to Latent Force Models

In this section we review the current state-space approach to inference with LFMs (Hartikainen and Särkkä, 2010) and show how some covariance functions can be represented exactly in state-space. Unfortunately, we shall also demonstrate that periodic kernels cannot be incorporated into LFMs using the approach advocated by Hartikainen and Särkkä (2010). To address this key issue, we will then propose to approximate a periodic covariance function with a sparse linear basis model. This will allow us to represent periodic behaviour within a LFM efficiently and also incorporate information encoded within the periodic kernel prior. Our work is inspired by, and can be seen as, an extension of the resonator model (Särkkä et al., 2012; Hartikainen et al., 2012), which is an alternative linear basis model that allows periodic processes to be modelled within the state-space approach. Our LBM approach, described in Section 5, builds on the resonator model and extends it by incorporating the prior information encoded within the latent force covariance function.

When the target processes, $\mathbf{z}$ as per Equation (4), can be expressed in Markov form, we can avoid the need to invert large covariance matrices and also avoid the need to evaluate Equation (5) over long time intervals, $[t_0,\ t]$, by using the more efficient state-space inference approach advocated by Hartikainen and Särkkä (2010) and in this paper. The temporal computational complexity of the state-space approach is $\mathcal{O}(T)$ as we integrate over short time intervals, $[t_0,\ t]$, and then reconstruct long term integrations by conflating the local integrations via the Kalman filter. This is an alternative approach to that advocated by Alvarez et al. (2009) in which we integrate the differential equations, as per Equation (5),

over long intervals, $[t_0,\ t]$, and then regress using Equations (2) and (3). Both approaches are mathematically equivalent in that they produce identical inferences when they are applied to the same differential model, latent force covariance functions and data.

The Kalman filter is a state-space tool for time series estimation with Gaussian processes (Kalman et al., 1960). The *Kalman smoother* is also available for interpolation with sequential data. The Kalman filter is a state-space inference tool which summarizes all information about the process, f, at time x via a *state* description. The advantage of the Kalman filter is that any process f_* at any future time x_* can be inferred from the current state without any need to refer to the process history. The state at any time x is captured by a finite set of Gaussian distributed *state variables*, $\mathbf{U}$, and we assume that f is a linear function of the state variables. In Hartikainen and Särkkä (2010) the state variables corresponding to each latent force f are the function f and its derivatives. In our approach the state variables corresponding to each periodic latent force will be the eigenfunctions of the periodic covariance function. The key advantage of the Kalman filter is that its computational complexity is linear in the amount of data from a single output time-series. Contrast this with the standard Gaussian process approach, as per Equations (2) and (3), which require the inversion of a covariance matrix and thus, have a computational complexity which is cubic in the amount of data.[3]

To illustrate the state-space approach consider a single non-periodic latent force, $u_r(t)$, indexed by r, in Equation (4). We assume that this force is drawn from a Gaussian process thus

$$u_r \sim \mathcal{GP}(0, K_r),$$

where K_r is a stationary kernel. In Hartikainen and Särkkä (2010) the authors demonstrate that a large range of stationary Gaussian process kernels, K_r, representing the latent force prior can be transformed into multivariate linear time-invariant (LTI) stochastic differential equations of the form

$$\frac{d\mathbf{U}_r(t)}{dt} = \mathbf{F}_r\ \mathbf{U}_r(t) + \mathbf{W}_r\ \omega_r(t), \tag{6}$$

where $\mathbf{U}_r(t) = (u_r(t),\ \frac{du_r(t)}{dt},\ \cdots, \frac{d^{p_r-1}u_r(t)}{dt^{p_r-1}})^T$ and

$$\mathbf{F}_r = \begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ -c_r^0 & \cdots & -c_r^{p_r-2} & -c_r^{p_r-1} \end{pmatrix}, \qquad \mathbf{W}_r = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \tag{7}$$

where c are coefficients which can be set using spectral analysis of the kernel as per Hartikainen and Särkkä (2010). The force, $u_r(t)$, can be recovered from $\mathbf{U}_r(t)$ using the indicator vector $\mathbf{\Delta}_r = (1, 0, \ldots, 0)$ where

$$u_r(t) = \mathbf{\Delta}_r \mathbf{U}_r(t).$$

3. The Kalman filter has a cubic computational complexity in the number of measured processes for multi-output Gaussian processes. We shall clarify the computational complexity of Kalman filter models for multi-output GPs in Section 7 and investigate an application of multi-output GPs in Section 9.

By choosing the coefficients $c_r^0, \ldots, c_r^{p_r-1}$ in Equation (7), the spectral density of the white noise process $\omega_r(t)$ in Equation (6) and the dimensionality p_r of $\mathbf{U}_r(t)$ appropriately, the covariance of $u_r(t)$, corresponding to the dynamic model, can be chosen to correspond to the GP prior K_r. The differential equations expressed in Equation (6) can then be integrated into the LFM to form the augmented dynamic model expressed later in Equation (12). The coefficients $c_r^0, \ldots, c_r^{p_r-1}$ are found by initially taking the Fourier transform of both sides of Equation (6). The coefficients can then be expressed in terms of the spectral density of the latent force kernel, K_r, provided that its spectral density, $S_r(\varpi)$, can be written as a rational function of ϖ^2 thus

$$S_r(\varpi) = \frac{(\text{constant})}{(\text{polynomial in } \varpi^2)} \ . \tag{8}$$

The *inverse* power spectrum is then approximated by a polynomial series from which the transfer function of an equivalent stable Markov process for the kernel can be inferred along with the corresponding spectral density of the white noise process. The stochastic differential equation coefficients are then calculated from the transfer function. For example, for the first-order Matérn kernel given by

$$K_r(t,t') = \sigma_r^2 \exp\left(-\frac{|t-t'|}{l_r}\right), \tag{9}$$

with output scale σ_r and input scale l_r, $u_r \sim \mathcal{GP}(0, K_r)$ can be represented by Equation (6) with $\mathbf{U}_r(t) = u_r$, $\mathbf{W}_r = 1$ and

$$\mathbf{F}_r = -1/l_r. \tag{10}$$

The spectral density, λ_r, of the white noise process, ω_r, is

$$\lambda_r = \frac{2\sigma_r^2\sqrt{\pi}}{l_r\ \Gamma(0.5)}, \tag{11}$$

and Γ is the Gamma function (Hartikainen and Särkkä, 2010).

Now, by augmenting the state vector, $\mathbf{z}$ in Equation (4), with the non-periodic forces $\mathbf{U}_r(t)$ and their derivatives, Hartikainen and Särkkä (2011) demonstrate that the dynamic equation can be rewritten as a joint stochastic differential model thus

$$\frac{d\mathbf{z}_a(t)}{dt} = \mathbf{F}_a\ \mathbf{z}_a(t) + \mathbf{L}_a\omega_a(t), \tag{12}$$

where

$$\begin{aligned} \mathbf{z}_a(t) &= (\mathbf{z}(t),\ \mathbf{U}_1(t),\ \ldots,\ \mathbf{U}_R(t))^T, \\ \mathbf{F}_a &= \begin{pmatrix} \mathbf{F} & \mathbf{L}\mathbf{S}_1\mathbf{\Delta}_1 & \ldots & \mathbf{L}\mathbf{S}_R\mathbf{\Delta}_R \\ \mathbf{0} & \mathbf{F}_1 & \ldots & \mathbf{0} \\ & & \vdots & \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{F}_R \end{pmatrix}, \end{aligned}$$

R is the number of latent forces, $\mathbf{S}_r = (0, \ldots, 1, \ldots, 0)$ is the indicator vector which extracts the rth column of $\mathbf{L}$ corresponding to the rth force, u_r, and $\omega_a(t)$ is the appropriate scalar process noise

$$\begin{aligned} \omega_a(t) &= (0,\ \omega_1(t),\ \ldots,\ \omega_R)^T, \quad (13) \\ \mathbf{L}_a &= \text{blockdiag}(\mathbf{0},\ \mathbf{W}_1,\ \ldots, \mathbf{W}_R). \quad (14) \end{aligned}$$

These differential equations have the solution

$$\mathbf{z}_a(t) = \mathbf{\Phi}(t_0, t)\mathbf{z}_a(t_0) + \mathbf{q}_a(t_0, t),$$

where, again, $\mathbf{\Phi}(t_0, t)$ denotes the matrix exponential, $\mathbf{\Phi}(t_0, t) = \exp(\mathbf{F}_a(t - t_0))$ expressed in Alvarez et al. (2009). The process noise vector, $\mathbf{q}_a(t_0, t)$, is required to accommodate the Matérn or SE latent forces within the discrete time dynamic model, $\mathbf{q}_a(t_0, t) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_a(t_0, t))$ where

$$\mathbf{Q}_a(t_0, t) = \int_{t_0}^{t} \mathbf{\Phi}(s, t)\mathbf{L}_a\Lambda_a\mathbf{L}_a^T\mathbf{\Phi}(s, t)^T ds,$$

and Λ_a is a diagonal matrix

$$\Lambda_a = \text{diag}(0, \lambda_1, \ldots, \lambda_R), \quad (15)$$

where λ_r is the spectral density of the white noise process corresponding to the Matérn or SE process, K_r (Hartikainen and Särkkä, 2010).

We now briefly describe the reasons why this spectral analysis approach advocated by Hartikainen and Särkkä (2010, 2011) cannot be immediately applied to periodic kernels. For illustrative purposes we shall investigate the commonly used squared-exponential periodic kernel expressed as

$$K_{\text{SE}}(t, t') = \exp\left(-\frac{\sin\left(\frac{\pi(t-t')}{D}\right)^2}{l^2}\right), \quad (16)$$

with input scale $l = 3$, an implicit output scale of unity and period $D = 0.7$, although our analysis and conclusions apply to all periodic kernels, in general. Unfortunately, as is shown in the left panel of Figure 1, the power spectrum for this periodic kernel is a weighted sum of Dirac delta functions, each delta function identifying a sinusoidal mode. The inverse of the power spectrum is highly nonlinear and not amenable to the polynomial series approximations expressed in Equation (8). The left panel also shows the best (in a least squares sense) polynomial fit to the inverse spectrum. The polynomial coefficients are shown in the central panel and very little weight is assigned to higher order frequencies. Now, using the approach advocated in Hartikainen and Särkkä (2010) we can infer the covariance function corresponding to this polynomial approximation of the inverse spectrum. This covariance function and the true periodic covariance function are shown in the right panel of Figure 1. It is clear that the covariance function obtained using Hartikainen and Särkkä (2010) is a poor representation of the true periodic kernel.

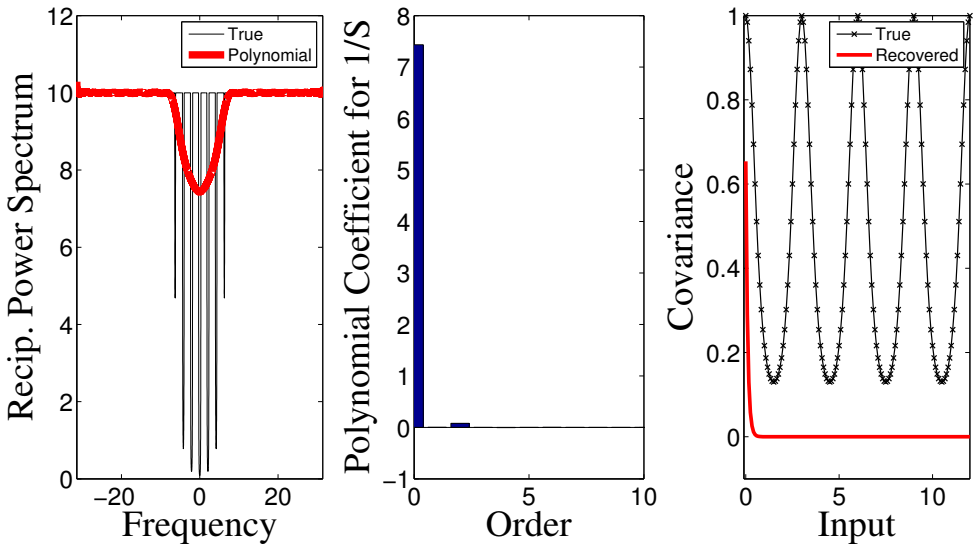


Figure 1: Spectral analysis of a periodic covariance function. The left panel shows inverse power spectrum for a periodic squared-exponential kernel (thin line) and its polynomial approximation (thick line). The central panel shows the coefficients of the polynomial approximation. The right panel shows the true covariance function (crossed line) and its approximation (solid line) recovered from the polynomial representation of the inverse power spectrum.

So, it is not possible to formulate all periodic latent forces via Equation (6). However, by approximating the latent force as a linear sum of basis functions, such that each basis function, ϕ, can be formulated via Equation (6) as

$$u_r(t) = \sum_j a_{rj}\phi_j(t), \tag{17}$$

then it is possible to represent the periodic latent force within the KF. In essence, the latent force, u_r, is decomposed into a weighted sum of basis latent forces, $\{\phi_j\}$, such that each ϕ satisfies Equation (6). This is the approach of Särkkä et al. (2012) for representing both stationary and quasi-periodic latent forces via their *resonator model*. In Hartikainen et al. (2012), the *resonator*, ϕ_r, is chosen to be a Fourier basis, $\phi_r(t) = \cos(f_r t)$ or $\phi_r(t) = \sin(f_r t)$. The resonator can be represented by Equation (6) as a state comprising the instantaneous resonator value, $\phi_r(t)$, and its derivative, $\dot{\phi}_r(t)$ thus, $\mathbf{U}_r(t) = (\phi_r(t),\ \dot{\phi}_r(t))^T$. The corresponding SDE has $\mathbf{F}_r = [0 \quad 1\ ;\ -f_r^2 \quad 0]$ and $\mathbf{W}_r = 0$. The Fourier basis is particularly useful for modelling stationary covariance functions.

In Hartikainen et al. (2012) quasi-periodic latent forces were implemented as a superposition

$$u(t) = \sum_j \psi_j(t), \tag{18}$$

of resonators, ψ, of the form

$$\frac{d^2\psi_j(t)}{dt^2} = -(2\pi f_j(t))^2\psi_j(t) + \omega_j(t), \tag{19}$$

where ω is a white noise component. Crucially, the resonator *frequencies*, f, are time variant and this supports non-stationary and quasi-periodic forces. This model is very flexible and both periodic and quasi-periodic processes can be expressed using the resonator model (as detailed in Appendix B). However, currently no mechanism has been proposed to incorporate prior information encoded in periodic GP kernels within the resonator model. Further, inferring the parameters and the frequency profiles, $f(t)$, for each resonator can be prohibitively computationally expensive (as we demonstrate in Appendix B). Despite these shortcomings there is a very close connection between the resonator model for periodic latent forces and the eigenfunction approach proposed in this paper. This connection is explored in detail in Appendix B in which we assert that the eigenfunction basis is an instance of the resonator basis for perfectly periodic covariance functions. We subsequently demonstrate how the eigenfunction approach can both inform the resonator model of the GP prior and also simplify the inference of the resonator model parameters including the frequency profile. Further, we show that the optimal minimum mean-square resonator model is an alternative way of representing the corresponding eigenfunction basis within the Kalman filter.

In the original implementation of the resonator model (Särkkä et al., 2012) the model parameters were set by hand. Recently, a new variation of the resonator model has been proposed in which the most likely model parameters are learned from the data (Solin and Särkkä, 2013). In this version the resonator is the solution of the time invariant second order differential equation

$$\frac{d^2\psi_j(t)}{dt^2} = A_j\psi_j(t) + B_j\frac{d\psi_j(t)}{dt} + \omega_j(t), \tag{20}$$

where A and B are constant coefficients. We note that this variation of the resonator model is a special case of the original resonator model with a frequency profile $f_j(t) = \frac{i}{2\pi}\sqrt{A_j + B_j\frac{1}{\psi_j(t)}\frac{d\psi_j(t)}{dt}}$ in Equation (19). To model quasi-periodic processes Equation (20) comprises a *decay* term via the first derivative of the resonator function. This new model is computationally efficient as it imposes constant coefficients unlike the original resonator model in Särkkä et al. (2012). However, the computational efficiency of the model in Equation (20), gained by losing the requirement to infer a frequency profile for each resonator, is at the expense of the model's flexibility. We compare the resonator model in Equation (20) with our eigenfunction approach on a real world application in Section 9.

In preparation for the approach advocated in this paper, in which we also represent the periodic kernel via a linear basis model, the following section compares the two key alternative approaches to directly inferring linear basis models from Gaussian Process kernels, namely the sparse spectrum Gaussian process regression (SSGPR, Lázaro-Gredilla et al., 2010) and kernel principal component analysis (Schölkopf and Müller, 1998).

5. Representing Periodic Latent Forces with Linear Basis Models

In this section, we exploit linear basis models and propose a novel approach to representing periodic latent force GP kernels. Our aim is to derive a sparse representation for periodic

kernels so that they can be accommodated within a state-space formulation of the LFM. Linear basis models (LBMs) have a long history in machine learning. In particular, special cases of them include kernel density estimators (Parzen, 1962) and the Relevance Vector Machine (Tipping, 2001). There are two key advantages to representing periodic kernels using a sparse basis model: firstly, they can approximate the kernel using a weighted sum over a finite set of functions. As we will see, for relatively smooth kernels the number of basis functions can be small. The second advantage, as we will show in Section 7, is that the LBM representation is amenable to inference using computationally efficient state-space methods. We exploit the Nyström approximation as opposed to other sparse approximations (such as the sparse spectrum Gaussian process regression method of Lázaro-Gredilla et al., 2010) as, we will see, the eigenfunctions of the kernel form the most efficient basis for the corresponding driving forces. This approximation will accommodate both the prior information about the driving forces (encoded in the kernel) within a state-space approach and also provide a means to learn these driving forces from data using iterative state-space methods. Approximating Gaussian process priors via the Nyström method is not new (see, for example, Williams and Seeger, 2001). However, using this to accommodate periodic and quasi-periodic latent forces within LFMs is novel.

In order to develop our LBM for latent forces we shall first investigate current approaches to sparse representations of stationary covariance functions and then demonstrate that one of these approaches, namely the eigenfunction approach, generalizes to non-stationary covariance functions. Bochner's theorem asserts that all stationary covariance functions can be expressed as the Fourier transform of their corresponding spectral densities (where the spectral density exists. See, for example, Rasmussen and Williams, 2006). Furthermore, in the stationary case, the Fourier basis is the eigenfunctions of the covariance function. There has been a long history of research into the spectral analysis of stationary Gaussian process kernels (see, for example, Bengio et al., 2004). However, only recently has the Fourier basis been investigated in the context of latent force models. To date, two approaches have been proposed to incorporate knowledge of all stationary kernels, including periodic kernels, within the linear basis representation via spectral analysis: the SSGPR (Lázaro-Gredilla et al., 2010) and the KPCA (Drineas and Mahoney, 2005) method. The key advantage of these approaches is that the basis frequencies can be calculated from the prior latent force kernel. These approaches are described and compared next.

The SSGPR (Lázaro-Gredilla et al., 2010) approach reinterprets the spectral density of a stationary GP kernel as the probability density function over frequency space. This pdf is then sampled using Monte Carlo to yield the frequencies of the sinusoidal basis functions of the LBM.[4] The advantage of this approach is that a sparse set of sinusoidal basis functions is identified such that the most significant frequencies of these sinusoidal basis functions have the greatest probability of being chosen. The phase of each basis function is then inferred from the data. The disadvantage of this approach is it can often provide a poor approximation to the covariance function as we will demonstrate shortly in Figure 3.

An alternative approach to the SSGPR is KPCA which effectively intelligently samples the most informative frequencies within the spectral density. Mercer's theorem (Mercer, 1909) allows us to represent each periodic latent force, $u(t)$, at arbitrary inputs, t, via an

4. In their code, available at `http://www.tsc.uc3m.es/~miguel/downloads.php`, the authors try several frequency initializations and use the best one.

infinite set of basis functions, ϕ_j, as

$$u(t) = \sum_{j=1}^{\infty} a_j \phi_j(t), \tag{21}$$

where $\{a_j\}$ are the model *weights* which are independently drawn from a Gaussian thus

$$a_j \sim \mathcal{N}(0, \mu_j^{\phi}), \tag{22}$$

where μ_j^{ϕ} is the variance of a_j. For any choice of probability density function, p, there exists an orthonormal basis, $\{\phi\}$, such that

$$\int \phi_i(t)\phi_j(t)p(t)dt = \begin{cases} 1 & \text{if } \ i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, the latent force prior, $K(t, t') = E[u(t)u(t')]$, can be expressed as

$$K(t, t') = \sum_{j=1}^{\infty} \mu_j^{\phi} \phi_j(t)\phi_j(t'), \tag{23}$$

where, ϕ_j are the *eigenfunctions* of the kernel, K, under p such that

$$\int K(t, t')\phi_j(t')p(t')dt' = \mu_j^{\phi}\phi_j(t), \tag{24}$$

and the variance, μ_j^{ϕ}, is also an *eigenvalue* of the kernel.

Of course, it is not feasible to actually use an infinite basis. Thus, we approximate the infinite sum in Equation (21) by a finite sum over a subset of significant eigenfunctions which have the J most significant eigenvalues, μ^{ϕ}, as

$$u(t) \approx \sum_{j=1}^{J} a_j \phi_j(t). \tag{25}$$

Fortunately, kernel principal component analysis (KPCA) allows us to identify the most significant J eigenfunctions a priori as well as compute their form approximately (Schölkopf and Müller, 1998).

The role of p, in Equation (24), is to weight the values of time t. We are free to choose the probability density function, $p(t)$, as we wish. For stationary covariance functions, a uniform pdf is appropriate as it weights each time instance, t, equally. To evaluate the integral in Equation (24) we use a quadrature-based method and N equally spaced quadrature points, S, of t, where $S = \{s_1, \ldots, s_N\}$ (see, for example, Shawe-Taylor et al., 2005). Thus

$$\int K(t, t')\phi_j(t')p(t')dt' \approx \frac{1}{N} \sum_{i=1}^{N} K(t, s_i)\phi_j(s_i). \tag{26}$$

The points, S, are also used to construct an $N \times N$ covariance matrix, G, called the *Gram matrix*, where

$$G_{ij} = K(s_i, s_j). \tag{27}$$

The Nyström approach is then used to derive approximate eigenfunctions of K using the eigenvectors, $\mathbf{v}$, and eigenvalues, μ, of the Gram matrix (Drineas and Mahoney, 2005). We denote the Nyström approximation for ϕ_j with uniform pdf p as $\tilde{\phi}_j$. For each eigenvector, $\mathbf{v}_j$, we have

$$\tilde{\phi}_j(t) = \frac{\sqrt{N}}{\mu_j} K(t, S)\mathbf{v}_j. \tag{28}$$

Since $\{\mathbf{v}_j\}$ are orthonormal then $\{\tilde{\phi}_j\}$ are orthogonal. Now, substituting the approximation for ϕ into Equation (25) we get

$$u(t) \approx \sum_{j=1}^{J} a_j \tilde{\phi}_j(t). \tag{29}$$

By forming the covariance between $u(t)$ and $u(t')$ we can derive a relationship between the latent force prior, the approximate eigenfunctions and the variances μ_j^{ϕ} of the model weights, a_j, as

$$K(t, t') \approx \sum_{j=1}^{J} \mu_j^{\phi} \tilde{\phi}_j(t) \tilde{\phi}_j(t'), \tag{30}$$

where $\mu_j^{\phi} \approx \mu_j / N$, is the scaled Gram matrix eigenvalue (Williams and Seeger, 2001).

As we can compare the covariance function, K, with the corresponding Nyström covariance function approximation, as per Equation (30), then the sample set, S, can be chosen a priori to provide a comprehensive representation of the kernel K. Furthermore, as $N \to \infty$ then $\tilde{\phi}_j \to \phi_j$. Finally, although the eigenfunction LBM is a parametric model, the eigenfunctions accurately reproduce the periodic GP prior across an entire period and undesirable extrapolation errors often associated with spatially degenerate LBMs are alleviated here (Rasmussen and Williams, 2006).

Throughout this paper the LBMs will comprise the most significant eigenfunctions according to the following definition,

Definition 1 *An eigenfunction is* significant *if its eigenvalue is more than a pre-defined fraction γ of the maximum eigenvalue.*

We have found that $\gamma = 1/100$ is a robust choice for the applications in Sections 8 and 9 in which fewer than 30 basis functions are required to model the latent forces.

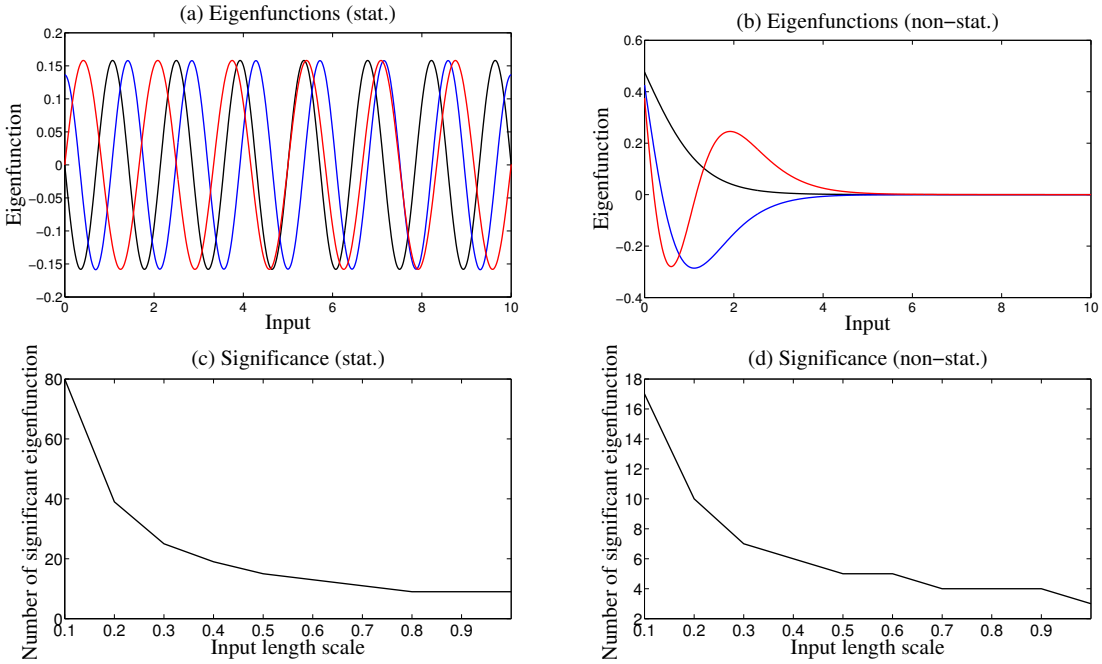


Figure 2: Example eigenfunctions for (a) stationary periodic and (b) non-stationary covariance functions, both with period 10 units. Also, the number of significant eigenfunctions for input length scales, l, for the (c) stationary periodic and (d) non-stationary covariance function.

To demonstrate the eigenfunction approach to representing Gaussian process priors via a finite basis, Figure 2(a) shows example eigenfunctions for a stationary periodic Matérn process. The Matérn kernel is defined (Rasmussen and Williams, 2006) as

$$\text{Matérn}(\tau, \nu, \sigma, l) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{l} \tau \right)^{\nu} \check{K}_{\nu} \left(\frac{\sqrt{2\nu}}{l} \tau \right), \tag{31}$$

where $\tau \geq 0$, Γ and $\check{K}_{\nu}$ are the gamma and modified Bessel functions, respectively, ν indicates the order, σ is the output scale which governs the amplitude of the kernel and l is the input length scale which governs the smoothness of the kernel. When the target function is periodic it is a direct function of the period *phase*, $\kappa(\tau) = |\sin(\pi\tau/D)|$ where D is the function period. Consequently, the periodic Matérn is given by $\text{Matérn}(\kappa(\tau), \nu, \sigma, 1)$. The periodic Matérn is of particular interest to us as it is used in Section 8 to model customer call centre arrival rates and in Section 9 to model the residual dynamics within home heating.

We observe that the eigenfunctions of the periodic kernel are the sinusoidal basis functions as shown in Figure 2. This basis corresponds to the Fourier basis functions for the power spectrum that can be obtained by Fourier analysis of the kernel. However, although through Fourier analysis we would be able to determine the power spectrum of the covari-

ance function, and consequently the magnitude of the basis function, we would be unable to determine the phase of the basis function. KPCA, in contrast, is able to determine both the magnitude, and consequently phase, of the Fourier basis functions.

A key property of the KPCA approach is that the eigenfunctions are not limited to the Fourier basis and, consequently, KPCA is also able to model non-stationary periodic covariance functions efficiently, in which case the eigenfunctions, which are inferred using KPCA from the non-stationary covariance function, are anharmonic as we will now demonstrate. Figure 2(b) shows the first three most significant eigenfunctions for an exponentially moderated periodic kernel

$$K(t,t') = \text{Matérn}(\kappa(t-t'),\nu,\sigma,l)\exp\left(-|t|-|t'|\right). \tag{32}$$

Figure 2, panels (c) and (d) show how the number of significant eigenfunctions decreases with increasing kernel smoothness for both the harmonic and anharmonic kernels above. The smoothness of the kernel is parameterized by the phase length scale, l. As above, we choose to declare an eigenfunction as significant if its eigenvalue is more than one hundredth of the maximum eigenvalue. Although this is a conservative definition of significance we can see that only a small number of basis functions are required to model these kernels.

We now compare the SSGPR, described above, and the eigenfunction approaches to modelling stationary kernels. For stationary kernels both the SSGPR and eigenfunction methods use a linear basis model with sinusoidal basis functions. The only difference between the approaches is that SSGPR assigns basis function frequencies (called *spectral points*) by sampling the kernel power spectrum. Both sine and cosine functions are used for each frequency. The KPCA infers its frequencies deterministically from the kernel and uses the basis functions with the most significant eigenvalues. Each spectral point corresponds to a Fourier basis function with known frequency with indeterminate phase. So, S spectral points produce S Fourier basis functions which has the same complexity as S Fourier basis functions in the eigenfunction approach. We compare the efficacy of both linear basis approaches when representing the squared-exponential kernel. The SSGPR was specifically developed with this kernel in mind and thus we present the fairest comparison. In order to investigate this difference and isolate the inference procedure by which the GP hyperparameters are learned from the data, the SSGPR algorithm is changed only slightly so that the actual kernel hyperparameters used correspond to the actual hyperparameters of the model which generated the training data. We also use the known generative GP hyperparameters within the eigenfunction model.

To demonstrate the superiority of the eigenfunction approach over the SSGPR approach, Figures 3 and 4 compare the SSGPR and eigenfunction representations of a squared-exponential kernel with an input scale of 10 units and output scale of 1 unit. The significant twenty two eigenfunctions were used and, equivalently, twenty two SSGPR spectral points were randomly chosen from the SE spectral density as proposed by Lázaro-Gredilla et al. (2010). Further, the eigenfunction approach used 20 evenly spaced points to construct the Gram matrix. In the case of the KPCA the corresponding covariance functions differed by no more than 9.6×10^{-5} from the actual covariance function. The SSGPR, using the same number of Fourier basis functions, deviated by as much as 0.36 (that is 36% of the prior function variance) when 22 spectral points were used. Figures 3 and 4 also show the covariance function for the SSGPR when 88 spectral points were used. In this case, the

SSGPR covariance function approximation differed by as much as 0.23 (that is, 23% of the prior function variance). Clearly, the eigenfunction model is a much more accurate representation of the actual generative kernel even when using only a quarter of the number of basis functions as the SSGPR.

The error in the SSGPR representation of the covariance function can have a significant impact on the accuracy of GP inference as the SSGPR can significantly underestimate the posterior variance of the target function. To illustrate the extent of this problem, Figure 4 shows the posterior distributions of a sparsely measured function inferred using Equations (2) and (3) and the SSGPR and eigenfunction approximations of the covariance functions. Clearly, the SSGPR variances in the top two panes are less than those calculated using the squared-exponential model (bottom right pane) and the approximate eigenfunction model (lower left pane). Furthermore, Table 1 compares the RMSE and expected log likelihood for the SSGPR and KPCA approaches over 100 functions drawn from the GP. Each function is measured every 10 units, as above, with no measurement noise. The SSGPR propensity to underestimate the posterior variance is demonstrated by a very low expected log likelihood of -6.9×10^4 compared to 75 for the KPCA eigenfunction method. Even when the number of spectral points is increased four fold the KPCA approach is still more accurate.

In summary, the eigenfunction model is a more efficient representation than the SSGPR in that it identifies an orthogonal basis and consequently requires fewer basis functions to capture the significant features of the generative kernel. Further, as we saw earlier, the eigenfunction approach generalizes to non-stationary kernels which can be represented efficiently by non-sinusoidal basis functions. Consequently, we advocate the eigenfunction approach over the SSGPR approach for generating the basis for use with LFMs.

Method	RMSE	ELL
SSGPR	3.03 ± 0.04	$-6.9 \times 10^4 \pm 0.6 \times 10^4$
SSGPR (x4)	2.74 ± 0.03	$-1.2 \times 10^4 \pm 0.1 \times 10^4$
KPCA	$\mathbf{2.49 \pm 0.03}$	$\mathbf{75.0 \pm 0.9}$

Table 1: RMSE and expected log likelihood for KPCA and SSGPR with the same number of basis functions and also SSGPR with four fold increase in the number basis functions.

In the next section we extend our eigenfunction approach to quasi-periodic latent force models. This is a key contribution of our paper.

6. Representing Quasi-Periodic Latent Forces with Linear Basis Models

The eigenfunction basis model presented in the previous section assumes that the latent force is perfectly periodic. However, the force may change gradually from cycle to cycle despite the latent force kernel parameters remaining fixed. For example, the force's phase may

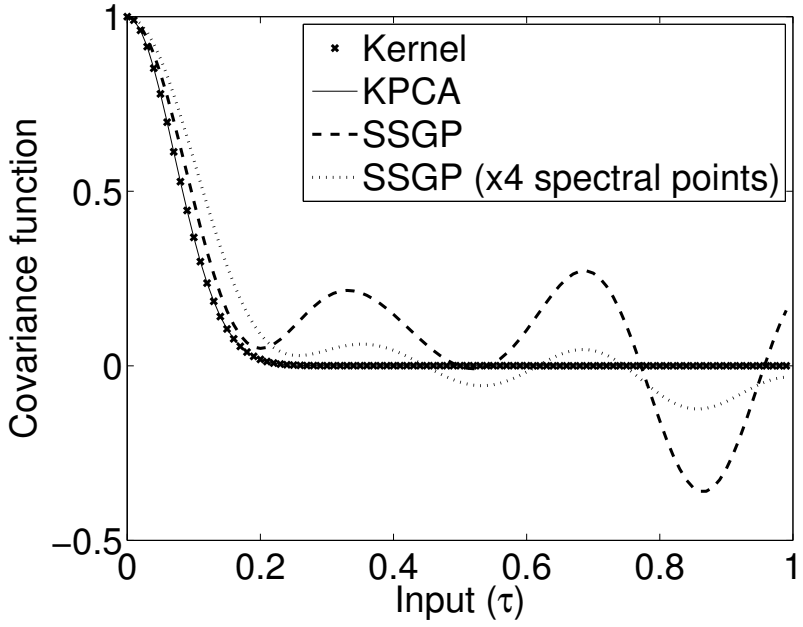


Figure 3: A comparison of SSGPR and eigenfunction approaches to modelling GP kernels via basis functions. The plots show the covariance functions corresponding to each of the eigenfunction and SSGPR models.

change between cycles. In the home heating application (described in detail in Section 9), where the residual heat within a home is modelled as a latent force, a phase shift in the residual heat profile may arise from cooking dinner at slightly different times from day to day.

When the latent force process, $u(t)$, is not perfectly periodic but exhibits some regularity from cycle to cycle it is called *quasi-periodic* and is often modelled as the product of two kernels (Rasmussen and Williams, 2006) thus

$$K_{\text{quasi-periodic}}(t,t') = K_{\text{quasi}}(t,t')K_{\text{periodic}}(t,t'), \tag{33}$$

where $K_{\text{periodic}}(t,t')$ is a periodic kernel (stationary or non-stationary) and $K_{\text{quasi}}(t,t')$ is a non-periodic kernel which reduces the inter-cycle correlations. For example, in Roberts et al. (2013), their quasi-periodic kernel is the product of a squared-exponential kernel and a periodic squared-exponential kernel and takes the form

$$K_{\text{quasi-periodic}}(t,t') = \sigma^2 \exp\left(-\frac{(t-t')^2}{l^2_{\text{quasi}}}\right) \exp\left(-\frac{\sin\left(\frac{\pi(t-t')}{D_{\text{periodic}}}\right)^2}{l^2_{\text{periodic}}}\right). \tag{34}$$

We note that Equation (32) is also a quasi-periodic covariance function.

We will now demonstrate that $K_{\text{quasi-periodic}}(t,t')$ can be modelled within the state-space approach by LBMs by letting the eigenfunction weights, a as per Equation (21), change

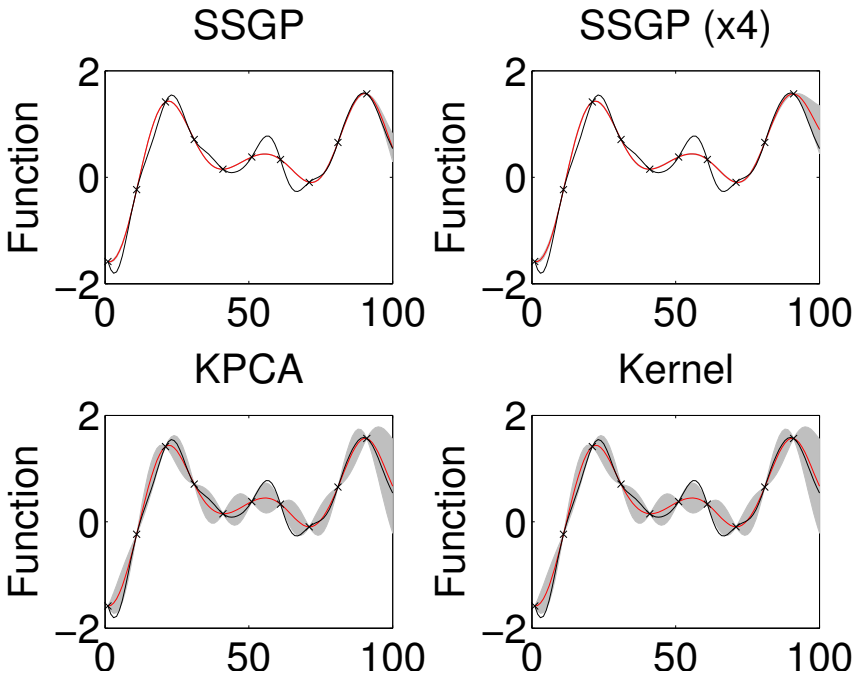


Figure 4: A comparison of SSGPR and eigenfunction approaches to modelling GP kernels via basis functions. The plots show typical example function estimates drawn using both approaches. The KPCA uses 22 basis functions and the SSGPR uses 22 spectral points and 88 spectral points respectively. The grey regions are the first standard deviation confidence regions.

dynamically. Equation (21) can be extended to include time varying process weights, $a(t)$ (O'Hagan, 1978) thus

$$u(t) = \sum_j a_j(t)\phi_j(t). \tag{35}$$

Consequently, when $u(t)$ is generated by a quasi-periodic kernel then

$$K_{\text{quasi-periodic}}(t,t') = E[u(t)\ u(t')] = \sum_{ij} \phi_i(t) E[a_i(t)a_j(t')]\phi_j(t').$$

We assume that $a_i(t)$ is drawn from a Gaussian process, so that

$$a_i \sim \mathcal{GP}(0, \mu_i^{\phi} K_{\text{quasi}}), \tag{36}$$

where μ_i^{ϕ} is the eigenvalue for the eigenfunction, ϕ_i, of K_{periodic} as per Equation (23). We also assume that each weight process is independent. Thus

$$E[a_i(t)a_j(t')] = \begin{cases} \mu_i^{\phi} K_{\text{quasi}}(t,t') & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

Consequently

$$\begin{aligned} K_{\text{quasi-periodic}}(t,t') &= \sum_i \phi_i(t)\mu_i^{\phi} K_{\text{quasi}}(t,t')\phi_i(t') \\ &= K_{\text{quasi}}(t,t') \sum_i \phi_i(t)\mu_i^{\phi}\phi_i(t') \\ &= K_{\text{quasi}}(t,t')K_{\text{periodic}}(t,t'). \end{aligned}$$

We see that the periodic component of the model, K_{periodic}, is represented by the basis function, ϕ, in the LBM whereas the non-periodic component, K_{quasi}, is represented via the time varying LBM coefficients, a. Note that, whereas for the resonator model, as per Equations (18) and (19), the Fourier basis functions, ϕ, are stochastic functions of time, in the eigenfunction approach, the coefficients, a, are stochastic functions of time and they reassign weight to fixed basis functions, $\phi(t)$.

In order to accommodate variant LBM coefficients in the Kalman filter we assume that each LBM coefficient is drawn from a stationary Gaussian process with covariance function, K_{quasi}, as per Equation (36). In which case, we can express the eigenfunction weight Gaussian process, $a_r(t)$, as a stochastic differential equation, as per Equation (6), thus

$$\frac{d\mathbf{A}_r(t)}{dt} = \mathbf{F}_r\, \mathbf{A}_r(t) + \mathbf{W}_r\, \omega_r(t), \tag{37}$$

where the state vector, $\mathbf{A}_r(t)$, comprises the coefficient time series and its derivatives, $\mathbf{A}_r(t) = (a_r(t)\ \frac{da_r(t)}{dt}, \cdots, \frac{d^{p_r-1}a_r(t)}{dt^{p_r-1}})^T$. Thus, as $\mathbf{A}_r$ can be expressed as a stochastic differential equation then it can be inferred using the Kalman filter as demonstrated in Hartikainen and Särkkä (2010). We can weaken the stationarity assumption and thus permit a greater choice for K_{quasi} by allowing changes in K_{quasi}'s output scale at discrete time instances called *change points*.

We propose three forms for K_{quasi} which are the *Continuous Quasi model* (CQM), the *Step Quasi model* (SQM) and the *Wiener-step Quasi model* (WQM). Although many other quasi-periodic forms are possible these models are chosen as they can each be represented efficiently within the Kalman filter state vector, as we will see in Section 7, whilst capturing the key qualitative properties of the data we wish to model. Specifically, the CQM models smooth, continuous deviations from cyclic behaviour over time, and, consequently, closely resembles the quasi-periodic model in Roberts et al. (2013). Alternatively, the SQM and WQM impose stationarity within a cycle but allow for function variation between cycles. We demonstrate that each can be represented in the Kalman filter via a single variable in the state-vector.

6.1 Continuous Quasi Model (CQM)

This stationary model imposes changes in the cycle continuously over time t. It is equivalent to the Matérn kernel with order $\nu = 1/2$ expressed as

$$K_{\text{quasi}}^{\text{CQM}}(t,t') = \sigma_r^2 \exp\left(-\frac{|t-t'|}{l_r}\right). \tag{38}$$

The input hyperparameter, l_r, is positive. As the CQM covariance function, K_{CQM}, is a first order Matérn, as per Equation (9), it can be represented as a Markov process, as per Equation (37). The process model, $\mathbf{F}_r$, and white noise spectral density, q_r, for the first order Matérn are presented in Equations (10) and (11). Reproducing this model here for completeness, if a is drawn from a GP with the quasi-periodic kernel in Equation (38), $a_r \sim \mathcal{GP}(0, K_{\text{quasi}}^{\text{CQM}})$, then

$$\frac{da_r(t)}{dt} = F_r a_r(t) + \omega_r(t),$$

where, $\omega_r(t)$ is a white noise process with spectral density q_r and

$$F_r = -\frac{1}{l_r}, \qquad\qquad q_r = \frac{2\sigma_r^2\sqrt{\pi}}{l_r\ \Gamma(0.5)},$$

and l_r and σ_r are the input and output scales, respectively, as per Equation (38). We note that, by using the CQM kernel as part of the quasi-periodic latent force covariance function, each LBM coefficient, $a_r(t)$, can be represented by a single variable in the Kalman filter state vector. In Section 7 we will demonstrate how this continuous time LTI model can be incorporated into a discrete time LFM model.

6.2 Step Quasi Model (SQM)

This model can be used to decorrelate cycles at change points between cycles. This non-stationary model preserves the variance of the periodic function each side of the change point. However, the function's correlation across the change point is diminished. For times, t and t', with t and t' in the same cycle $K_{\text{quasi-periodic}}(t, t') = K_{\text{periodic}}(t, t')$. When times t and t' correspond to different cycles then $K_{\text{quasi-periodic}}(t, t') < K_{\text{periodic}}(t, t')$. If N consecutive cycles are labelled $C = 1, 2, \ldots, N$ and $C(t)$ denotes the cycle index for time t then

$$K_{\text{quasi}}^{\text{SQM}}(t, t') = \sigma_r^2 \exp\left(-\frac{|C(t) - C(t')|}{l_r}\right). \tag{39}$$

Again, the kernel input hyperparameter, l_r, is positive.

6.3 Wiener-step Quasi Model (WQM)

Again, we assume the presence of change points between cycles. This non-stationary model increases the variance of the function at the change point. If N consecutive cycles are labelled $C = 1, \ldots, N$ then

$$K_{\text{quasi}}^{\text{WQM}}(t, t') = \xi_0 + \min(C(t'), C(t))\xi_r, \tag{40}$$

where ξ_0 and ξ_r are positive.

Example covariance functions for the three forms for K_{quasi} are shown in Figure 5. Also, sample quasi-periodic function draws are shown for each kernel. The functions are drawn from a quasi-periodic squared-exponential kernel $K_{\text{quasi-periodic}}(t,t')$ with $K_{\text{periodic}}(t,t')$ the periodic squared-exponential K_{SE}, as per Equation (16), with period $D = 10$ units, various input scales l (specified within each subfigure) and $K_{\text{quasi}}(t,t')$ set to either $K^{\text{CQM}}_{\text{quasi}}(t,t')$, $K^{\text{SQM}}_{\text{quasi}}(t,t')$ or $K^{\text{WQM}}_{\text{quasi}}(t,t')$. In the case of SQM and WQM a new cycle begins every 10 time units.

The SQM and WQM kernels can be incorporated into the discrete time Kalman filter by firstly expressing them as continuous time differential equations as per Equation (6). Suppose that either $a_r \sim \mathcal{GP}(0, K^{\text{SQM}})$ or $a_r \sim \mathcal{GP}(0, K^{\text{WQM}})$ then

$$\frac{da_r(t)}{dt} = 0,$$

everywhere, except at change points. Thus, in the case of SQM and WQM the corresponding process $a_r(t)$ can be represented via a first order differential equation as per Equation (37) with $\mathbf{A}_r(t) = a_r(t)$, $\mathbf{\Delta}_r = 1$, $\mathbf{F}_r = 0$ and $\mathbf{W}_r = 0$. However, at a change point, τ, the SQM and WQM covariance functions jump in value as can be seen in Figure 5 at input distances 20 and 40, for example. The value of the process, $a_r(\tau)$, immediately after the change point is related to the process, $a_r(\tau_-)$, immediately before the jump thus

$$a_r(\tau) = G^*_r a_r(\tau_-) + \chi^*_r(\tau), \tag{41}$$

where G^*_r is the process model and $\chi^*_r(\tau)$ is a Gaussian random variable, $\chi^*_r(\tau) \sim \mathcal{N}(0, Q^*_r)$. In Appendix A we demonstrate that the process model, G^*_r, and process noise variance, Q^*_r, for the SQM at the change point are

$$G^*_{\text{r,SQM}} = \exp\left(-\frac{1}{l_r}\right) \tag{42}$$

and

$$Q^*_{\text{r,SQM}} = \sigma_r^2\left(1 - \exp\left(-\frac{2}{l_r}\right)\right), \tag{43}$$

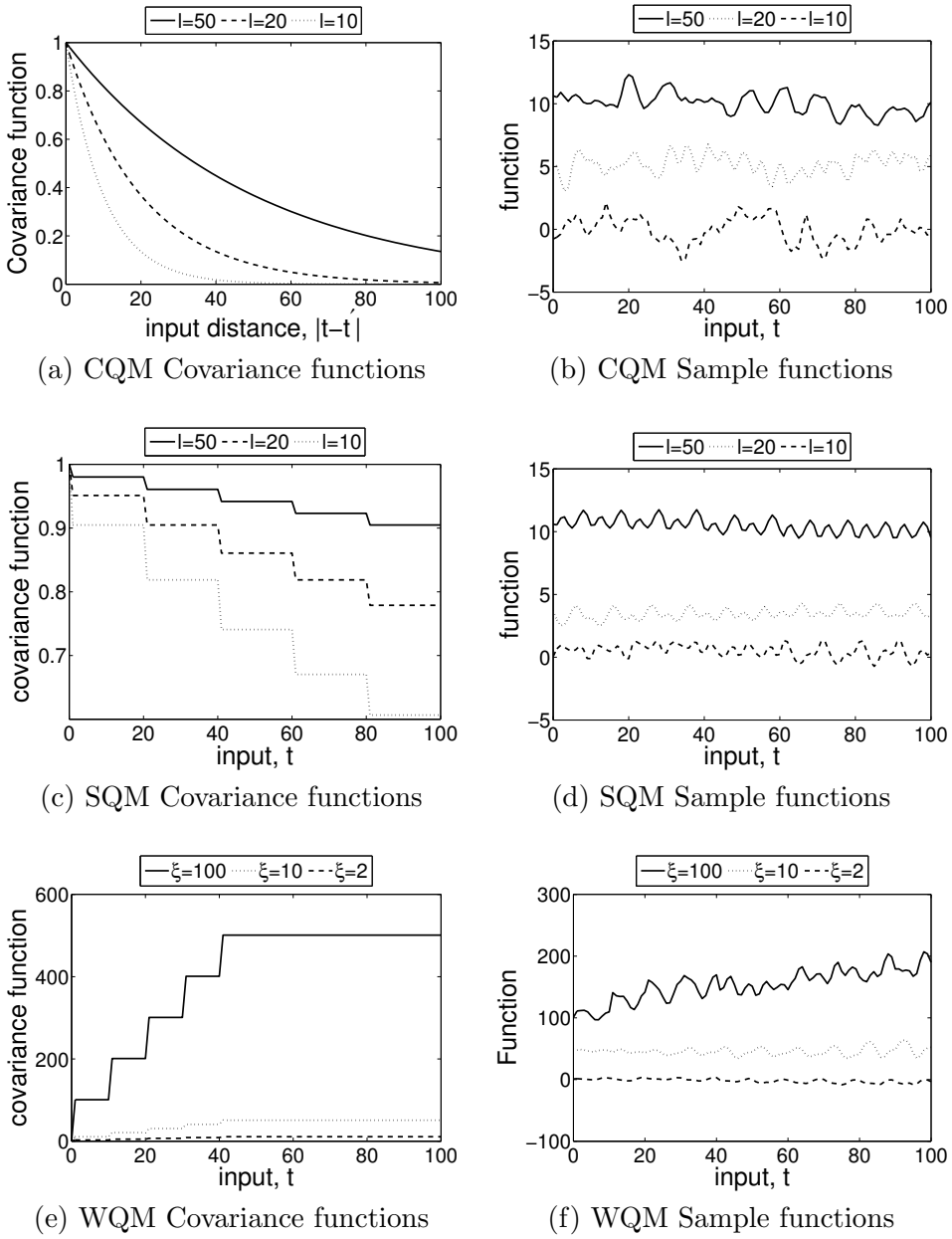
respectively. Similarly, Appendix A also shows that the process model, G^*_r, and process noise variance, Q^*_r, for the WQM at a change point are

$$G^*_{\text{r,WQM}} = 1 \tag{44}$$

and

$$Q^*_{\text{r,WQM}} = \xi_r, \tag{45}$$

respectively. The latent force variance increases at the change point under the WQM whereas the variance remains unchanged for the SQM. In Section 7, we demonstrate how these expressions for G^* and Q^* are incorporated within the discrete form of the Kalman filter.



(a) CQM Covariance functions (b) CQM Sample functions

(c) SQM Covariance functions (d) SQM Sample functions

(e) WQM Covariance functions (f) WQM Sample functions

Figure 5: Covariance functions (left column) for CQM, SQM and WQM. Also, sample quasi-periodic functions (right column) for CQM, SQM and WQM quasi kernels and a squared-exponential periodic kernel.

The three forms for K_{quasi} will be applied to both the call centre customer queue tracking and home temperature prediction problem domains in Sections 8 and 9. In the next section we describe how we perform inference with a LFM using a state-space approach, where the

state vector is augmented with periodic or quasi-periodic latent forces that are approximated using the latent force eigenfunctions.

7. Recursive Estimation with Periodic and Quasi-Periodic Latent Force Models

This section describes a state-space approach to inference with LFMs in some detail. We shall treat the periodic and non-periodic latent forces differently when performing inference with them. Following Hartikainen and Särkkä (2010, 2011), non-periodic forces will be modelled using the power spectrum of their corresponding covariance functions. Alternatively, the periodic latent forces will be modelled using the eigenfunctions of the corresponding periodic covariance function. The key idea in this section is to infer the LFM unknowns via the Kalman filter. The unknowns include the non-periodic forces and their derivatives, as per Equation (12), along with the coefficients of the periodic forces, as per Equation (37). The remainder of this section describes in detail how the KF state is predicted forward in time and how measurements of the system are folded into the state estimate.

We examine periodic and quasi-periodic cases separately as state-space inference with periodic latent forces uses a more compact model. For the periodic case, we assume that the latent forces, $\mathbf{u}$, as per Equation (4), can be separated into two distinct sets, periodic forces, $\mathbf{u}_\mathrm{p}$, and non-periodic forces, $\mathbf{u}_\mathrm{np}$, so that $\mathbf{L}\mathbf{u}(t) = \mathbf{L}_\mathrm{np}\mathbf{u}_\mathrm{np}(t) + \mathbf{L}_\mathrm{p}\mathbf{u}_\mathrm{p}(t)$ as described in Section 3. Then, Equation (4) becomes

$$\frac{d\mathbf{z}(t)}{dt} = \mathbf{F}\ \mathbf{z}(t) + \mathbf{L}_\mathrm{np}\mathbf{u}_\mathrm{np}(t) + \mathbf{L}_\mathrm{p}\mathbf{u}_\mathrm{p}(t).$$

We model non-periodic latent forces and their derivatives, as per Equation (6), and periodic forces using eigenfunctions as per Equation (29). We define the augmented state vector, $\mathbf{z}_a$, as per Equation (12), and also the corresponding periodic force coefficients, $\mathbf{L}_\mathrm{p}^a = [\mathbf{L}_\mathrm{p}^T,\ \mathbf{0}^T]^T$ so that the forces $\mathbf{u}_\mathrm{p}$ still act on $\mathbf{z}$ within $\mathbf{z}_a$ thus

$$\frac{d\mathbf{z}_a(t)}{dt} = \mathbf{F}_a\ \mathbf{z}_a(t) + \mathbf{L}_a\omega_a(t) + \mathbf{L}_\mathrm{p}^a\mathbf{u}_\mathrm{p}(t), \tag{46}$$

where ω_a and $\mathbf{L}_a$ are as per Equations (13) and (14).

We now introduce our eigenfunction model for the periodic latent forces into Equation (46). First, we consider periodic latent forces, introduced in Section 5, for which the corresponding LBM coefficients, $\{a\}$ in Equation (29), are constant over time. Substituting our Nyström approximation basis model for the periodic forces, as per Equation (28), into the dynamic differential model, as per Equation (46), we get

$$\frac{d\mathbf{z}_a(t)}{dt} = \mathbf{F}_a\ \mathbf{z}_a(t) + \mathbf{L}_a\omega_a(t) + \sum_{r=1}^{R}\sum_{j=1}^{J_r}\mathbf{L}_p^a(\cdot, r)\ \tilde{\phi}_{rj}(t)\ a_{rj}, \tag{47}$$

where R is the number of latent forces, J_r is the number of eigenfunctions for latent force r, a_{rj} are the eigenfunction weights and the vector $\mathbf{L}_\mathrm{p}^a(\cdot, r)$ is the r^{th} column of the matrix $\mathbf{L}_\mathrm{p}^a$ in Equation (46). The Nyström basis function, $\tilde{\phi}_{rj}$, is

$$\tilde{\phi}_{rj}(t) = \frac{\sqrt{N_r}}{\mu_{rj}}K_r(t, S_r)\mathbf{v}_{rj}, \tag{48}$$

where K_r, S_r and N_r are the covariance function, the quadrature points at which the kernel is sampled for force r, as per Equation (26), and the cardinality of S_r. The μ_{rj} and $\mathbf{v}_{rj}$ are the Gram matrix eigenvalues and eigenvectors, respectively.

The differential equations (47) have the solution

$$\mathbf{z}_a(t) = \mathbf{\Phi}(t_0, t)\mathbf{z}_a(t_0) + \mathbf{q}_a(t_0, t) + \sum_{r=1}^{R}\sum_{j=1}^{J_r} a_{rj}\mathbf{M}_{rj}(t_0, t), \tag{49}$$

where, again, $\mathbf{\Phi}(t_0, t)$ denotes the matrix exponential, $\mathbf{\Phi}(t_0, t) = \exp(\mathbf{F}_a(t - t_0))$, and $\mathbf{q}_a(t_0, t) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_a(t_0, t))$ where

$$\mathbf{Q}_a(t_0, t) = \int_{t_0}^{t} \mathbf{\Phi}(s, t)\mathbf{L}_a\Lambda_a\mathbf{L}_a^T\mathbf{\Phi}(s, t)^T ds,$$

and Λ_a, as per Equation (15), is the spectral density of the white noise processes corresponding to the non-periodic latent forces. The matrix $\mathbf{M}_{rj}(t_0, t)$ is the convolution of the state transition model, $\mathbf{\Phi}$, with each of the periodic latent force eigenfunctions expressed as

$$\mathbf{M}_{rj}(t_0, t) = \frac{\sqrt{N_r}}{\mu_{rj}} \left[\int_{t_0}^{t} ds\ \mathbf{\Phi}(s, t)\mathbf{L}_{\mathrm{p}}^{a}(\cdot, r)K_r(s, S_r)\right] \mathbf{v}_{rj}.$$

For small time intervals $[t_0,\ t]$, which is the case for our applications in Sections 8 and 9, $\mathbf{M}_{rj}$ can be calculated using numerical matrix exponential integration methods. Further, we note $\mathbf{\Phi}(\mathbf{t_0}, \mathbf{t})$ is stationary and this can mitigate the need to recalculate this matrix exponential at each instance of the time series.

To accommodate the latent forces within the Kalman filter we must ensure that our discrete time dynamic model, as per Equation (49), has the appropriate form. Specifically

$$\mathbf{X}(t) = \mathbf{G}(t_0, t)\mathbf{X}(t_0) + \omega(t_0, t),$$

where the noise process, ω, is i.i.d Gaussian and zero-mean. In order to rewrite Equation (49) into the appropriate form for Kalman filter inference we define a vector, $\mathbf{a}$, as per Equation (21), which collects together the eigenfunction weights thus

$$\mathbf{a} = (a_{11}, \ldots, a_{1J_1}, a_{21}, \ldots, a_{2J_2} \ldots)^T,$$

and, similarly, a matrix, $\mathbf{M}$, which collects together the convolutions, $\mathbf{M}_{rj}$, thus

$$\mathbf{M}(t_0, t) = (\mathbf{M}_{11}(t_0, t), \ldots, \mathbf{M}_{1J_1}(t_0, t), \mathbf{M}_{21}(t_0, t), \ldots, \mathbf{M}_{2J_2}(t_0, t) \ldots).$$

We further augment the state vector to accommodate the model weights, $\mathbf{a}$, corresponding to the periodic latent forces. Let

$$\mathbf{X}(t) = (\mathbf{z}_a^T(t), \mathbf{a}^T)^T, \tag{50}$$

be our augmented state vector which now accommodates the derivative auxiliary variables in $\mathbf{z}_a$ required by the non-periodic forces as per Hartikainen and Särkkä (2011) and the

eigenfunction weights, $\mathbf{a}$, required by the periodic forces as per our approach. When the eigenfunction weights are constant we can rewrite Equation (49) thus

$$\mathbf{X}(t) = \mathbf{G}(t_0, t)\mathbf{X}(t_0) + \omega(t_0, t), \tag{51}$$

where

$$\mathbf{G}(t_0, t) = \begin{pmatrix} \mathbf{\Phi}(t_0, t) & \mathbf{M}(t_0, t) \\ \mathbf{0} & \mathbf{I}, \end{pmatrix} \tag{52}$$

and

$$\omega(t_0, t) = \begin{pmatrix} \mathbf{q}_a(t_0, t) \\ \mathbf{0} \end{pmatrix}.$$

Thus, predictions of the Gaussian process, $\mathbf{X}$, can be inferred using the Kalman filter. Of course, the model in Equation (51) can also be incorporated within the Kalman Smoother to perform full (that is, forward and backward) regression over $\mathbf{X}(t)$ for all time t if required (Hartikainen and Särkkä, 2010). The prediction equations for the state mean, $\bar{\mathbf{X}}(t \mid t_0)$, and covariance, $\mathbf{P}(t \mid t_0)$, at time t conditioned on measurements obtained up to time t_0, are

$$\bar{\mathbf{X}}(t \mid t_0) = \mathbf{G}(t_0, t)\bar{\mathbf{X}}(t_0 \mid t_0), \tag{53}$$

$$\mathbf{P}(t \mid t_0) = \mathbf{G}(t_0, t)\mathbf{P}(t_0 \mid t_0)\mathbf{G}(t_0, t)^T + \mathbf{Q}(t_0, t), \tag{54}$$

where $\mathbf{Q}(t_0, t) \triangleq \begin{pmatrix} \mathbf{Q}_a(t_0, t) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$.

We assume that measurements, $\mathbf{y}$, are Gaussian distributed thus

$$\mathbf{y}(t) = \mathbf{H}\ \mathbf{X}(t) + \eta(t), \tag{55}$$

where η is zero-mean multivariate Gaussian, $\eta \sim \mathcal{N}(\mathbf{0}, \mathbf{Z})$, where $\mathbf{Z}$ is the observation noise covariance matrix and the *measurement model*, $\mathbf{H}$, extracts the appropriate elements of the state vector. These measurements can be folded into the Kalman filter in the usual way. The update equations given measurement, $\mathbf{y}(t)$, as per equation (55), are

$$\bar{\mathbf{X}}(t \mid t) = \bar{\mathbf{X}}(t \mid t_0) + \mathbf{K}(\mathbf{y}(t) - \mathbf{H}\bar{\mathbf{X}}(t \mid t_0)), \tag{56}$$

$$\mathbf{P}(t \mid t) = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}(t \mid t_0), \tag{57}$$

where $\mathbf{K}$ is the Kalman gain given by

$$\mathbf{K} = \mathbf{P}(t \mid t_0)\mathbf{H}^T(\mathbf{H}\mathbf{P}(t \mid t_0)\mathbf{H}^T + \mathbf{Z})^{-1}. \tag{58}$$

The computational complexity of the Kalman gain is cubic in the cardinality of the measurement vector, $\mathbf{y}$ (that is, not necessarily a function of the cardinality of the state). The cubic cost arises from the need to invert a covariance matrix in Equation (58). For a single output Gaussian process this covariance will be a scalar. However, for multi-output Gaussian processes, when each physical process is measured, $\mathbf{y}(t)$ will be a vector of (noisy) measurements of each process at time t. In which case, the computational complexity of the

Kalman gain will be cubic in the number of measured physical processes. So, although the state vector may be augmented in order to model both physical processes and latent forces, as described above, these additions will not impact on the cost of the matrix inversion in Equation (58).

We next extend our state-space approach to accommodate quasi-periodic latent forces. For the quasi-periodic latent forces the corresponding kernel LBM coefficients, $\mathbf{a}$, are functions of time, as per Equation (35). We assume that each LBM coefficient is drawn from a Gaussian process with covariance function, K_{quasi}, as per Equation (33), and we now demonstrate how these dynamic weight processes, $\mathbf{a}(t)$, are incorporated into the Kalman filter, Equations (53) to (57).

As above, a_{rj}, corresponds to the jth eigenfunction for latent force r. However, for quasi-periodic latent forces each eigenfunction weight is variant and we assume $a_{rj}(t)$ can be written as a stochastic differential equation, as proposed in Section 6, thus

$$\frac{d\mathbf{A}_{rj}(t)}{dt} = \mathbf{F}_{rj}\mathbf{A}_{rj}(t) + \mathbf{W}_{rj}\ \omega_{rj}(t), \tag{59}$$

where the state vector, $\mathbf{A}_{rj}(t)$, comprises derivatives of the coefficient time series, $\mathbf{A}_{rj}(t) = (a_{rj}(t), \frac{da_{rj}(t)}{dt}, \cdots, \frac{d^{p_{rj}-1}a_{rj}(t)}{dt^{p_{rj}-1}})^T$. We can recover the eigenfunction coefficient from $\mathbf{A}_{rj}$ thus

$$a_{rj}(t) = \mathbf{\Delta}_{rj}\mathbf{A}_{rj}(t),$$

where the vector $\mathbf{\Delta}_{rj} = (1, 0, \ldots, 0)$ is an indicator vector which extracts the LBM coefficient a_{rj} from $\mathbf{A}_{rj}$. Thus, the latent force, $u_r(t)$, as per Equation (35), is

$$u_r(t) = \sum_j a_{rj}(t)\phi_{rj}(t) = \sum_j \phi_{rj}(t)\mathbf{\Delta}_{rj}\mathbf{A}_{rj}(t), \tag{60}$$

where ϕ_{rj} is the jth eigenfunction for the latent force r. Substituting our Nyström approximation for the eigenfunction, $\phi(t)$ as per Equation (28), into Equation (60) we get

$$u_r(t) = \sum_j \frac{\sqrt{N_r}}{\mu_{rj}} \left[K_r(t, S_r)\right] \mathbf{v}_{rj}\mathbf{\Delta}_{rj}\mathbf{A}_{rj}(t).$$

Then, substituting u_r into the differential latent force model, Equation (47), we get

$$\frac{d\mathbf{z}_a(t)}{dt} = \mathbf{F}_a\ \mathbf{z}_a(t) + \mathbf{L}_a\omega_a(t) + \sum_{r=1}^{R}\sum_{j=1}^{J_r} \mathbf{m}_{rj}(t)\mathbf{A}_{rj}(t), \tag{61}$$

where R is the number of latent forces, J_r is the number of eigenfunctions for latent force r, $\mathbf{L}_a$ and $\omega_a(t)$ are as per Equations (13) and (14) and the vector $\mathbf{m}_{rj}$ is

$$\mathbf{m}_{rj}(t) = \frac{\sqrt{N_r}}{\mu_{rj}} \left[\mathbf{L}_{\text{p}}^a(\cdot, r)K_r(t, S_r)\right] \mathbf{v}_{rj}\mathbf{\Delta}_{rj}, \tag{62}$$

where K_r, S_r and N_r are the covariance function, the quadrature points at which the kernel is sampled for force r, as per Equation (26), and the cardinality of S_r. The μ_{rj} and $\mathbf{v}_{rj}$ are

the Gram matrix eigenvalues and eigenvectors, respectively. The vector $\mathbf{L}_{\mathrm{p}}^{a}(\cdot, r)$ is the r^{th} column of the matrix $\mathbf{L}_{\mathrm{p}}^{a}$ in Equation (46).

Now, as for the constant eigenfunction coefficient case, to exploit the Kalman filter for LFM inference with quasi-periodic latent forces we gather together all the LFM Gaussian variables, $\mathbf{z}_a$ and $\{\mathbf{A}_{rj}\}$, into a single state-vector. In so doing, we define a vector $\mathbf{A}(t)$ which collects together the eigenfunction coefficients and their derivatives thus

$$\mathbf{A}(t) \triangleq (\mathbf{A}_{11}(t)^T, \mathbf{A}_{12}(t)^T, \ldots, \mathbf{A}_{21}(t)^T, \mathbf{A}_{22}(t)^T, \ldots,)^T, \tag{63}$$

a matrix $\mathbf{m}(t)$ which collects together the vectors $\{\mathbf{m}_{rj}\}$ thus

$$\mathbf{m}(t) \triangleq (\mathbf{m}_{11}(t), \mathbf{m}_{12}(t), \ldots, \mathbf{m}_{21}(t), \mathbf{m}_{22}(t), \ldots,),$$

a matrix $\mathbf{F}_{\mathbf{A}}$ which collects together the process models for all eigenfunction coefficients for all latent forces, as per Equation (59) thus

$$\mathbf{F}_{\mathbf{A}} \triangleq \text{blockdiag}(\mathbf{F}_{11}, \mathbf{F}_{12} \ldots, \mathbf{F}_{21}, \mathbf{F}_{22}, \ldots),$$

a vector $\omega_{\mathbf{A}}$ which collects together the noise processes for the non-periodic forces, ω_a, as per Equation (61), and noise processes for all the eigenfunction coefficients as per Equation (59) thus

$$\omega_{\mathbf{A}} \triangleq (\omega_a^T, \omega_{11}, \omega_{12}, \ldots, \omega_{21}, \omega_{22}, \ldots)^T, \tag{64}$$

and the block diagonal matrix $\mathbf{L}_{\mathbf{A}}$ which collects together the corresponding process noise coefficients, $\mathbf{L}_a$, as per Equation (61) and $\mathbf{W}_{rj}$ as per Equation (59) thus

$$\mathbf{L}_{\mathbf{A}} \triangleq \text{blockdiag}(\mathbf{L}_a, \mathbf{W}_{11}, \mathbf{W}_{12} \ldots, \mathbf{W}_{21}, \mathbf{W}_{22}, \ldots).$$

As per Equation (50) let

$$\mathbf{X}(t) \triangleq (\mathbf{z}_a^T(t), \mathbf{A}^T(t))^T, \tag{65}$$

be our augmented state vector which now accommodates the derivative auxiliary variables required by the non-periodic forces as per Hartikainen and Särkkä (2011) and the eigenfunction weights required by the quasi-periodic forces as per our approach. Combining Equations (59) and (61) we get

$$\frac{d\mathbf{X}(t)}{dt} = \begin{pmatrix} \mathbf{F}_a & \mathbf{m}(t) \\ \mathbf{0} & \mathbf{F}_{\mathbf{A}} \end{pmatrix} \mathbf{X}(t) + \mathbf{L}_{\mathbf{A}} \omega_{\mathbf{A}}(t), \tag{66}$$

where $\omega_A(t)$ is a vector of independent white noise processes. The spectral density of the ith white noise process in this vector is $[\Lambda_A]_i$ where

$$\Lambda_A = (\Lambda_a, q_{11}, q_{12}, \ldots, q_{21,}, q_{22}, \ldots).$$

The Λ_a, as per Equation (15), is the spectral density of the white noise processes corresponding to the non-periodic latent forces and q_{rj}, as per Equation (59), is the spectral density of the white noise process for the eigenfunction weight, a_{rj}.

Unfortunately, Equation (66) is an inhomogeneous SDE as $\mathbf{m}$ is a function of time. Consequently, in this form, $\mathbf{X}(t)$ cannot be folded into the Kalman filter. However, by assuming $\mathbf{m}(t)$ is approximately constant over the short time interval, $[t_0, t]$, and asserting $\mathbf{m}(t) \approx \mathbf{m}(t_0)$ then $\frac{d\mathbf{X}(t)}{dt}$ can be integrated into the appropriate form

$$\mathbf{X}(t) = \mathbf{\Phi}(t_0, t)\mathbf{X}(t_0) + \mathbf{q}(t_0, t), \tag{67}$$

where

$$\mathbf{\Phi}(t_0, t) = \exp\left[\begin{pmatrix} \mathbf{F}_a & \mathbf{m}(t_0) \\ \mathbf{0} & \mathbf{F_A} \end{pmatrix}(t - t_0)\right]. \tag{68}$$

The process noise, $\mathbf{q}(t_0, t) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}(t_0, t))$, where

$$\mathbf{Q}(t_0, t) = \int_{t_0}^{t} \mathbf{\Phi}(s, t)\ \mathbf{L}_A \Lambda_A \mathbf{L}_A^T \mathbf{\Phi}(s, t)^T ds. \tag{69}$$

Thus, the state $\mathbf{X}(t)$ can be predicted using the Kalman filter, as per Equations (53) and (54), by defining the process model, as per Equation (68), thus

$$\mathbf{G}(t_0, t) \quad = \quad \mathbf{\Phi}(t_0, t),$$

and the process noise covariance, $\mathbf{Q}(t_0, t)$, as per Equation (69). We note the quasi-periodic covariance functions *Step Quasi* (SQM) and *Wiener-step Quasi* (WQM), eigenfunction coefficients are perturbed, as per Equation (41), at change points. The discrete form of the Kalman filter can readily predict the value of each coefficient across a change point using the process model, G_j^*, and process noise variance, Q_j^*, for each coefficient, a_{rj}, as per Equation (41).

In general, our Kalman filter approach to LFM inference requires a process model, $\mathbf{A}_{rj}$, for each eigenfunction coefficient. Thus, the computational complexity of the prediction step of the Kalman filter which employs quasi-periodic models increases quadratically with the number of latent forces R, the number of derivatives used to represent each non-periodic latent force (N in Equation (64)) and the number of derivatives used to represent each time variant eigenfunction coefficient. Although, our approach supports any quasi-periodic covariance function, for practical applications, we recommend using the quasi-periodic covariance functions developed in Section 6 which are readily convertible to the Markovian form as per Equation (59) and for which only one variable is required to represent each time varying eigenfunction coefficient. These quasi-periodic covariance functions are the *Continuous Quasi model* (CQM), the *Step Quasi model* (SQM) and the *Wiener-step Quasi model* (WQM).

In Sections 8 and 9 we determine the efficacy of our state-space approach to LFM inference on two real world applications: i) the inference of call centre customer arrival rates and the tracking of customer queue lengths and ii) the inference of periodic residual heat dynamics within real homes and the prediction of internal temperature. We compare our approaches for the different periodic and quasi-periodic kernels developed in Section 6 on the call centre application and demonstrate the utility of incorporating periodic latent force models over non-periodic models. Then we compare our approaches to periodic and quasi-periodic LFMs to the resonator model in the thermal application.

8. Modelling Queues with Quasi-Periodic Arrival Rates

In this section we apply our approach to LFM inference to the dynamics of telephone queues in call centres as outlined in Section 1 with the aim of tracking the diurnal customer queue length when different agent deployment strategies are used. We use real customer arrival rate data, provided by Feigin et al. (2006), in which the customer telephone arrival rates for a loan company sales line have been collected every 5 minutes over a three month period starting from October 2001. The arrival rates during eleven consecutive Thursdays over this period are shown in Figure 6.

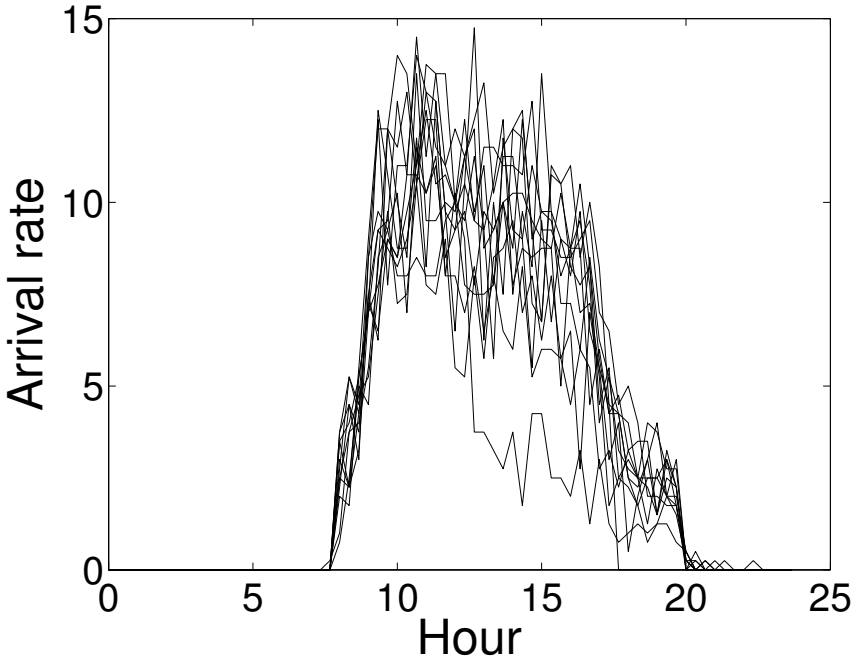


Figure 6: Customer arrival rates (per minute) for the same week day (Thursday) over a 11 week period showing the quasi-periodic nature of the data.

To model the queue dynamics as a latent force model we use the *Pointwise Stationary Fluid Flow Approximation* (PSFFA) for queues (Wang et al., 1996). The PSFFA models the *mean* queue length, L, in terms of arrival processes, ζ, using a first order differential equation given by

$$\frac{dL(t)}{dt} = g(L) + \zeta(t) \tag{70}$$

where, g, is a non-negative, non-linear function of the queue length, L. This model is often used to model queues in call centres where $L(t)$ is the *average* length of the queue at time t and $\zeta(t)$ is the mean *arrival rate*, that is the average rate at which customers join the queue (Wang et al., 1996). The PSFFA is a first order, non-linear differential equation. Ignoring

the non-linearity of g for now, we see that Equation (70) is of the form of Equation (4). Thus, Equation (70) is an example of a LFM in which the queue length, L in Equation (70) is the target process, $\mathbf{z}$ in Equation (4) and the customer arrival rate, $\zeta(t)$ in Equation (70), is the sole latent force, $\mathbf{u}$, in Equation (4). Consequently, we apply our approach to LFM inference to the tracking of queue lengths using the PSFFA.

We consider the M/M/1 queue as it corresponds exactly to the customer arrival process, ζ, which is Poisson and the service time is arbitrarily distributed with successive service times being independent and identically distributed.[5] Service times have an exponential distribution with parameter Ω in the M/M/1 queue. A single server serves customers one at a time from the front of the queue, according to a first-come, first-served basis. When the service is complete the customer leaves the queue and the number of customers in the system reduces by one. The queue buffer is of infinite size, so there is no limit on the number of customers it can contain.

The PSFFA allows us to represent this M/M/1 system via the following differential equation for the mean queue length, L (Wang et al., 1996)

$$\frac{dL(t)}{dt} = -\Omega(t)\left(\frac{L(t)}{1+L(t)}\right) + \zeta(t), \tag{71}$$

where $\Omega(t)$ is the mean queue *service rate*. We use this model to simulate the true queue length, L, using the real customer arrival rate, ζ, from data provided by Feigin et al. (2006) and realistic service rate profiles, Ω. Measurements of the instantaneous customer queue length, $y(t^*)$, are generated for times, t^*, during the day and are given by

$$y(t^*) = L(t^*) + \epsilon(t^*),$$

where, t^*, are sufficiently spaced so that $\epsilon(t^*)$ is zero-mean, i.i.d. Gaussian.

To recover the customer arrival rate from the measured queue length we assume that the mean arrival rate, ζ, is drawn from a Gaussian process, $\zeta \sim \mathcal{GP}(0, K_{\text{arrival rate}})$, where $K_{\text{arrival rate}}$ is the arrival rate process covariance function. Note that the arrival rate can be positive or negative. Negative arrival rates correspond to customers who leave the system without being served. We choose $K_{\text{arrival rate}}$ to be either the first order Matérn kernel, a periodic first order Matérn kernel as per Equation (31) or a quasi-periodic first order Matérn kernel utilizing a CQM, SQM or WQM kernel, as per Section 6.

As per Equation (65), the augmented state-vector, $\mathbf{X}(t)$, is $\mathbf{X}(t) = \left(L(t), \mathbf{A}^T(t)\right)^T$ where $\mathbf{A}(t)$ are the eigenfunction weights corresponding to the periodic latent force covariance functions, as per Equation (63). Unfortunately, the transition dynamics in Equation (71) are nonlinear. However, if we assume that the mean, $\bar{L}(t_0)$, of L, conditioned on the measurements up to time t_0 is a good approximation for L over the entire, yet small, interval $[t_0,\ t]$ then we can rewrite Equation (71) as a locally linear model thus

$$\frac{dL(t)}{dt} \approx -\frac{\Omega(t)}{1+\bar{L}(t_0)}\ L(t) + \zeta(t), \tag{72}$$

5. The term 'M/M/1' is Kendall's queue classification notation (Kendall, 1953) corresponding to a stochastic process whose state-space is the set $\{0, 1, 2, 3, ...\}$ where, in our case, the value corresponds to the number of customers in the system.

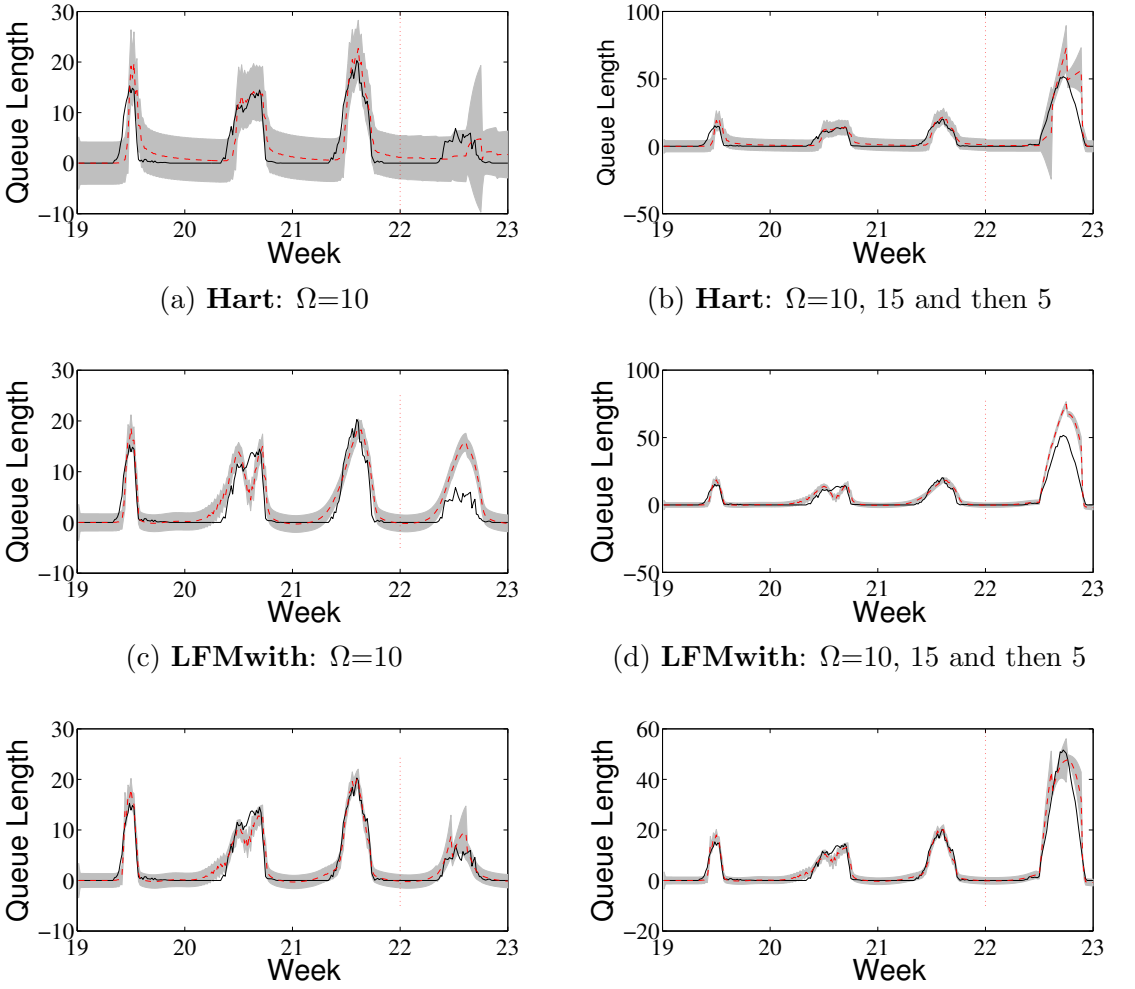
over $[t_0,\ t]$. This model then has the appropriate form for inclusion within the Kalman filter. In our experiments predictions are made over 2 minute time intervals. This time interval is chosen so that Equation (72) is a stable local approximation to the queue dynamics. We shall call this KF algorithm **LFMwith** as it contains a GP model of the arrival rate process. We use maximum likelihood to obtain the GP hyperparameters and the model parameters which are the Matérn output and input scales and the observation noise variance. The cycle period is fixed at 24 hours.

The efficacy of our customer queue model is evaluated by training the model using data over three full consecutive Thursdays and then tracking the mean queue length over the following Thursday. The queue length observations are made every 40 minutes during the training period and every three hours during the fourth day tracking phase. The longer tracking interval is specifically chosen to test the predictive power of our model with sparse observations. The efficacy of our approach to LFM inference for even longer term predictions (that is, day ahead predictions) is explored in Section 9.

We also introduce four further algorithms to empirically demonstrate the importance of using periodic and quasi-periodic latent force models in our domain and to demonstrate the efficacy of our algorithm at tracking customer queue lengths. Three of these algorithms use quasi-periodic latent force models. **LFMquasi (CQM)**, **LFMquasi (SQM)** and **LFMquasi (WQM)** algorithms use the Continuous Quasi-periodic model, the Step Quasi-periodic model and the Wiener-step periodic model respectively, described in Section 6, to model the arrival rates. These models are identical to the periodic model used in **LFMwith** except that the correlation between cycles is reduced by the quasi-periodic kernel. The most likely hyperparameters are used for the SQM, CQM and WQM kernels. The change points required by the SQM and WQM quasi-periodic latent force models are set to midnight for all cycles. We also implement Hartikainen's algorithm (Hartikainen and Särkkä, 2010) for sequential inference which uses the M/M/1 model described above and a non-periodic first order Matérn kernel for the customer arrival rate process (**Hart**) to demonstrate the performance of a non-periodic model.

An example run of our algorithms is shown in Figure 7 and this shows the ground truth queue lengths (in black) and first standard deviation estimates of the queue lengths using **Hart**, **LFMwith** and **LFMquasi (SQM)** latent force models. This figure shows the tracked queue length over four days. The LFM parameters are learned using the first three days of data only. The fourth day is tracked without any further learning of the LFM model parameters. The left column of plots shows the queue length estimates for a fixed service rate, $\Omega = 10$, applying to both training and tracking phases. The right column of plots shows the estimates for a fixed training service rate, $\Omega = 10$, and a variable test service rate of $\Omega = 15$ for the first half of the fourth day and $\Omega = 5$ for the second half. By testing the algorithm with variable service rates, we are able to test the efficacy of the algorithms at both reproducing the training data and at making predictions in regimes not encountered during the training period. Clearly the quasi-periodic model is the most accurate, in this case, with a RMSE of 1.9 compared to 2.3 and 4.6 for **Hart** and **LFMwith**, respectively.

To fully test the accuracy of the inferred residual model we evaluated the RMSE and expected log likelihood of the predicted average queue length for each day and for each algorithm over 11 days. Firstly, the service rate was held constant throughout at an arbitrary



(a) **Hart**: Ω=10

(b) **Hart**: Ω=10, 15 and then 5

(c) **LFMwith**: Ω=10

(d) **LFMwith**: Ω=10, 15 and then 5

(e) **LFMquasi (SQM)**: Ω=10

(f) **LFMquasi (SQM)**: Ω=10, 15 and then 5

Figure 7: Call centre customer queue length over four consecutive Thursdays. The first three days of data are used to train the model. The fourth day is tracked. The 1st standard deviation confidence interval is shown (grey). The solid black line shows the ground truth. The left column of plots shows the results for a fixed service rate, $\Omega = 10$, for both training and test phases. The right column of plots shows the results for a fixed training service rate, $\Omega = 10$, and a variable test service rate of $\Omega = 15$ for the first half of the fourth day and $\Omega = 5$ for the second half.

value of $\Omega = 10$. The results are summarized in Table 2. Clearly, the RMSE is lower for the quasi-periodic models but their expected log likelihoods are larger than the periodic model indicating the superiority of the quasi-periodic models.

Method	RMSE	ELL
LFMquasi (SQM)	4.6 ± 2.2	-146 ± 22
LFMquasi (CQM)	4.4 ± 1.3	-142 ± 15
LFMquasi (WQM)	$\mathbf{1.8 \pm 0.2}$	$\mathbf{-139 \pm 20}$
LFMwith	2.2 ± 0.4	-276 ± 60
Hart	10.6 ± 5.9	-209 ± 29

Table 2: Day ahead tracking: Queue length RMSE and expected log likelihood (ELL) for periodic, quasi-periodic and non-periodic arrival rate models. Both training and test epochs have the same fixed service rate $\Omega = 10$.

In the final experiment in this section we demonstrate the ability of our approach to make inferences in regimes where data is absent. This is a powerful and useful property of Gaussian process models. Specifically, to plan future staffing requirements the call centre manager needs to predict the impact that a novel service rate will have on future queue lengths given predicted customer arrival rates. However, the service centre may not have utilized this staffing profile to date. In this case, for illustrative purposes, we assume that the staff profile to date has been constant during working hours with a fixed service rate, $\Omega = 10$. However, the service manager has noticed a significant queue of customers forming in the morning and then relatively few customers arriving in the afternoon. Consequently, the service manager contemplates employing a variable staffing profile and hiring more staff during the first half of the day, so that the service rate increases to $\Omega = 15$, and then retaining fewer staff in the afternoon, so that the service rate drops to $\Omega = 5$.

To determine the efficacy of our approach at predicting the impact of variable staffing profiles given only data from constant staffing profiles we repeated the experiment above with a fixed service rate, $\Omega = 10$ during training and a variable service rate during the test period. In this case, we chose a high service rate of $\Omega = 15$ for the first half of the test day and then a low rate, $\Omega = 5$ over the second half. The RMSE and expected log likelihood are shown in Table 3. Again, the quasi-periodic models have similar efficacy and produce the most accurate estimates of the customer queue length in this case with an RMSE of 3.3 compared to 11.6 and 5.7 for **Hart** and **LFMwith**, respectively. We note that, for both experiments, the **LFMquasi (SQM)**, **LFMquasi (CQM)**, **LFMquasi (WQM)** and **LFMwith** used fewer than 28, 28, 20, 28 basis functions, respectively, to represent the arrival rate process.[6] Consequently, our LBM Kalman filter approach to LFM tracking is computationally efficient.

In the next section we evaluate our approach to LFM inference for longer term predictions than those considered in the call centre application. We shall demonstrate that our approach can exploit the quasi-periodic nature of the latent force to project far forward in time an accurate estimate of the force. Consequently, we shall see that our approach is

6. The actual number of basis functions used varied between runs.

Method	RMSE	ELL
LFMquasi (SQM)	7.2 ± 1.8	-183 ± 21
LFMquasi (CQM)	15.2 ± 4.0	-205 ± 24
LFMquasi (WQM)	$\mathbf{3.3 \pm 0.6}$	$\mathbf{-152 \pm 15}$
LFMwith	5.7 ± 1.2	-725 ± 301
Hart	11.6 ± 1.1	-305 ± 49

Table 3: Queue length RMSE and expected log likelihood (ELL) for periodic and quasi-periodic arrival rate models. Training over three days with a fixed service rate $\Omega = 10$. The test day had a service rate of $\Omega = 15$ for the first half of the day followed by $\Omega = 5$ for the remainder.

effective at performing day ahead predictions of temperatures in the home using differential thermal models and non-parametric models of the residual heat within the home.

9. Modelling the Thermal Dynamics of Home Heating

In this section we apply our approach to LFM to the thermal modelling problem outlined in Section 1. We assume that the differential equation governing the thermal dynamics within a home is given by

$$\frac{dT_{\text{int}}(t)}{dt} = \alpha(T_{\text{ext}}(t) - T_{\text{int}}(t)) + \beta E(t) + R(t), \tag{73}$$

where T_{int} and T_{ext} are the internal temperature within the home and the onsite ambient external temperature respectively in °C (Bacher and Madsen, 2011; Rogers et al., 2011; Ramchurn et al., 2012). $E(t)$ represents the thermostat control output at time t ($E(t) \in \{0, 1\}$), β represents the thermal output of the heater and α is the leakage coefficient to the ambient environment. In this model $T_{\text{ext}}(t)$ and $E(t)$ are known latent forces for the LFM in Equation (73). $R(t)$ is the residual generated by latent forces which are not captured by the differential thermal model, such as heat generated by solar warming and lags in the heating system. These are completely unknown a priori but, since they are expected to exhibit periodic behaviour, a periodic Matérn kernel prior is used to model them.

We assume that $T_{\text{ext}}(t)$ at times t and t_0 can be modelled by a non-periodic GP prior Matérn$(|\mathtt{t} - \mathtt{t_0}|, \mathtt{0.5}, \sigma_{\text{ext}}, \mathtt{l}_{\text{ext}})$. We choose the Matérn kernel as this imposes continuity in the function but imposes no strong assumptions about higher order derivatives. However, our approach can be applied to Matérn functions of higher smoothness if required. The state vector, $\mathbf{X}$, as per Equation (51), comprises the internal temperature, the external temperature and its derivative and eigenfunction coefficient weights, $\mathbf{A}$, for our sparse basis model of the residual as per Equation (63). We model the residual process by a periodic Matérn kernel, Matérn$(|\sin(\pi\tau/\mathtt{D})|, \nu, \sigma, \mathtt{l})$ with order $\nu = 1/2$, smoothness, l, and D set to correspond to a daily period. Again, we choose the Matérn for the same reasons as above. The residual is represented via J basis functions $(\phi_1(\theta), \ldots, \phi_J(\theta))$ corresponding to Equation (28), where θ is the periodic phase as described in Section 7. The augmented state-

vector, $\mathbf{X}(\mathbf{t})$, is $\mathbf{X}(t) = \left(T_{\text{int}}(t), T_{\text{ext}}(t), \frac{T_{\text{ext}}(t)}{dt}, \mathbf{A}(t)^T\right)^T$ and the continuous time dynamic model corresponding to Equation (66) is

$$\frac{d\mathbf{X}(t)}{dt} = \begin{pmatrix} \mathbf{F}_a & \mathbf{m}(t) \\ \mathbf{0} & \mathbf{F}_{\mathbf{A}} \end{pmatrix} \mathbf{X}(t) + \mathbf{L}_{\mathbf{A}}\omega_{\mathbf{A}}(t) + \beta\mathbf{E}(t),$$

where $\mathbf{E}(t) = (E(t), 0, \ldots, 0)^T$ is the same size as $\mathbf{X}(t)$.

We will now describe the role of each term in the dynamic model. Within the transition model, $\mathbf{F}_a$ captures the temperature gradient components of Equation (73) and the Matérn driving forces for the external temperature thus

$$\mathbf{F}_a = \begin{pmatrix} -\alpha & \alpha & 0 \\ 0 & 0 & 1 \\ 0 & -\rho_{\text{ext}}^2 & -2\rho_{\text{ext}} \end{pmatrix},$$

with $\rho_{\text{ext}} = 2/l_{\text{ext}}$. The derivative of the external temperature is represented in the state vector so that the Matérn latent force kernel can be encoded within the Kalman filter as summarized in Section 4 and described in detailed in Hartikainen and Särkkä (2010). We set the order of this Matérn covariance function to $\nu = 3/2$ as the external temperature process is relatively smooth. The matrix $\mathbf{m}$ captures the residual heat contribution to the internal temperate and depends on the choice of the residual model covariance function, K, in Equation (62). Further, the matrix $\mathbf{F}_A$ models the dynamics of each periodic residual heat process and this also depends on the choice of residual model covariance function, as per Equation (59). The corresponding discrete form of the Kalman filter, as per Equation (67), is evaluated over 10 minute time intervals, $[t_0,\ t]$. This time interval is chosen to coincide with the heater on/off control cycle.

The Kalman filter is initialised with known current temperature values. The initial covariance is block diagonal with a diagonal matrix over the temperature components (including the solution to the appropriate Riccati equation for the external temperature Matérn model presented in Hartikainen and Särkkä, 2010) and a diagonal covariance over the residual model weights corresponding to the periodic Matérn residual process. The covariance for the model weights is obtained using the periodic Matérn prior and corresponding eigenfunctions as described in Section 4.

When tracking the internal room temperature we know the state of the heater, that is, whether it is "on" (that is, $E(t) = 1$) or "off" (that is, $E(t) = 0$), at each point in time. However, when predicting the internal room temperature a full day ahead the times at which the heater will switch on or off will not be known in advance. Uncertainty in the future controller behaviour arises because the heater behaviour depends on the internal room temperature and the predicted internal room temperature will be uncertain. The heater will be on if the room temperature is below the set-point or off if above the set-point. In order to accommodate the uncertainty in the heater switching process and, as the control output is binary, then prediction is performed using the Rao-Blackwellized Particle filter (RBPF, Doucet et al., 2000). The RBPF uses a set of particles to represent the many possible states of the system (the internal temperature and residual). The corresponding on/off control output is determined for each particle from the value of the internal temperature

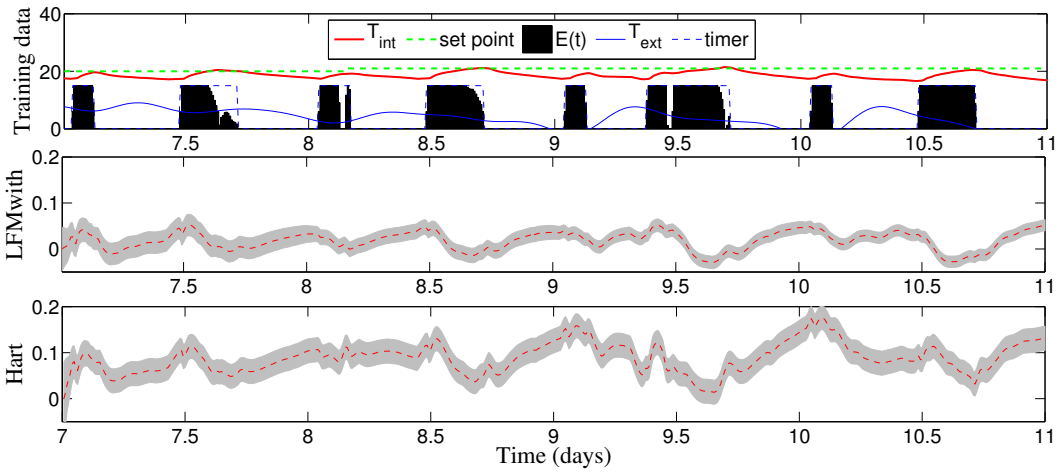


Figure 8: Internal and external temperature, thermostat set-point and heater activation for a four day training period (upper). Also shown is the residual sequentially inferred using **LFMwith** (central) and the **Hart** (lower) algorithms. The 1st standard deviation confidence interval is shown (grey).

associated with that particle and the set point. For each particle the Kalman filter is used to predict the room temperature conditioned on the control output for that particle which is held constant for each 10 minute interval. For each 10 minute interval there are RBPF particles corresponding to the control output being "on" or "off" over that interval. Each particle also has a prior Gaussian distribution over $\mathbf{X}$ and the Kalman filter is used to predict the state $\mathbf{X}$ at the end of the current interval conditioned on the binary value of the heater for that particle. A new set of particles is then generated by taking each particle in turn, sampling the posterior of the internal temperature, conditioning the posterior on that internal temperature sample and then assigning the heater state according to the controller (set point minus the internal temperature when the heater is primed). This procedure is iterated to predict over the entire day ahead. With P particles and cardinality C of $\mathbf{X}$, the complexity of the prediction phase scales as $\mathcal{O}(C^2TP)$ over T time steps. Following nomenclature in the call centre theory application in Section 8, we shall call this RBPF algorithm **LFMwith**$^+$ as it contains a GP model of the residual heat process. However, we have added the superscript '+' to denote that LFM inference is performed by the RBPF. We use maximum likelihood to obtain the model parameters for the thermal model, $\{\alpha,\ \beta\}$, the GP hyperparameters, $\{\sigma,\ l,\ \sigma_{\text{ext}},\ l_{\text{ext}}\}$ and the observation noise variance. We note that, if the set-point process is also uncertain but a distribution over the future set-point process is known then the RBPF particles can be drawn from the on/off control distribution and the set-point distribution. We do not examine the case of uncertain set-point values in this paper.

We also implement four further algorithms to empirically demonstrate the importance of using a periodic residual model in our domain and to demonstrate the efficacy of our algorithm at predicting internal temperatures. Three of these algorithms use quasi-periodic latent force models. **LFMquasi (SQM)**$^+$, **LFMquasi (CQM)**$^+$ and **LFMquasi (WQM)**$^+$ algorithms use the step quasi-periodic model, the continuous quasi-periodic model and the Wiener quasi-periodic model, respectively, described in Section 6, to model the residual driving forces. These models are identical to the periodic model used in **LFMwith**$^+$ except that the correlation between cycles is reduced by the quasi-periodic kernel. Again, these models use the same quasi-periodic covariance functions as their counterparts in the call centre application in Section 8 and, again, we have added the superscript '+' to denote that LFM inference is performed by the RBPF. A fifth algorithm, **LFMwithout**$^+$, assumes that there is no residual heat within the home. This algorithm is an instance of **LFMwith**$^+$ with no periodic latent force basis functions in the state vector. We also implement Hartikainen's algorithm (Hartikainen and Särkkä, 2010) for sequential inference which uses the thermal model described above and a non-periodic Matérn kernel for the residual (**Hart**$^+$). To accommodate the binary thermostat controller within **Hart**$^+$ we use the RBPF, as described above, but with Hartikainen's Kalman filter formalism.

We also implement a recent version of the resonator model (Solin and Särkkä, 2013) which we call the **Resonator**$^+$. The resonators are defined via the second order differential equation, as per Equation (20), which includes a decay term with fixed frequency and decay coefficients. We chose to implement this version of the resonator model as opposed to the time varying frequency version (Särkkä et al., 2012) as this version of the resonator model is completely developed in the literature and inferring the coefficients using maximum likelihood techniques has been thoroughly tested (Solin and Särkkä, 2013). In order to undertake a fair comparison between the performance of the resonator model and our eigenfunction approaches we choose the number of resonators and eigenfunctions to be the same. Further, we infer the most likely resonator frequencies and decay coefficients using the same Nelder-Mead optimisation algorithm implemented in our eigenfunction approach. As the residual process is quasi-periodic with period, D (corresponding to one day), we initialize the resonator frequencies to be distinct and contiguous multiples of $1/D$. As with all the LFM algorithms above, day ahead predictions with the resonator model are performed by the RBPF.

We collected two data sets from two different homes in January 2012 recording the internal temperature, T_{int}, the external temperature, T_{ext}, the thermostat set point and the heater activity, E, at one minute intervals. Each data set comprises fourteen consecutive days of data. We label these data sets `data1` and `data2`. For each home four complete consecutive days of the data set are chosen to train each algorithm. We then predict the internal temperature, $T_{\text{int}}(t)$, over the next full day. With 14 days of data for each data set, 10 full day predictions can be made for each data set with each algorithm. Note that both data sets have thermostat set point changes that require predictions to be made in regimes in which the algorithms have not been trained.

We shall first illustrate the efficacy of the three algorithms on a single example of the training and prediction process before presenting a statistical comparison of the algorithms over the full data set. Figure 8 shows four days of training data from `data1`. The central

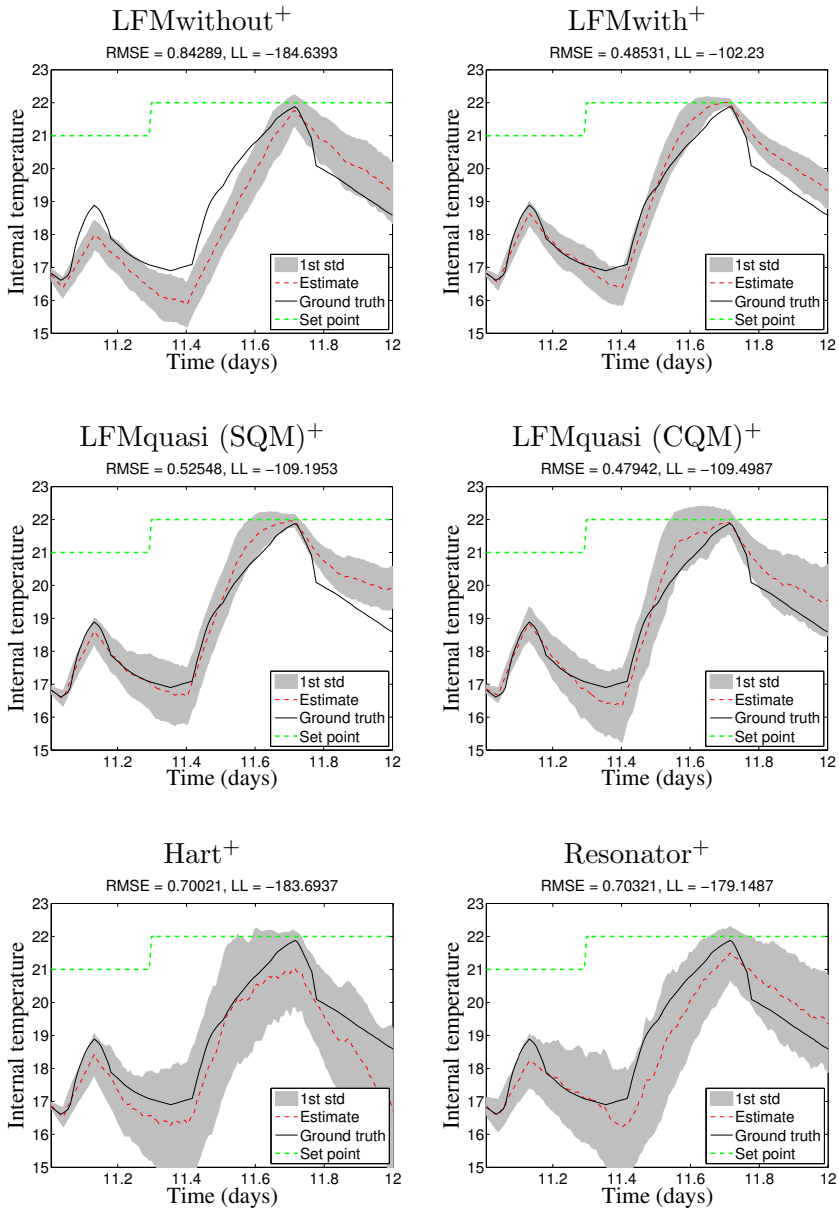


Figure 9: Internal temperature prediction compared to actual measured value using the **LFMwithout**$^+$ (top left), the **LFMwith**$^+$ (top right), the **LFMquasi (SQM)**$^+$ (middle left), the **LFMquasi (CQM)**$^+$ (middle right), the **Hart**$^+$ (bottom left) and the **Resonator**$^+$ (bottom right) algorithms. The 1st standard deviation confidence interval is shown (grey). Also shown is the thermostat set point (green).

and bottom plots show the residual over the two day period inferred by **LFMwith**$^+$ and by **Hart**$^+$. The **Hart**$^+$ plot shows that, although the residual exhibits some daily periodicity,

the cycle is imperfect. However, the inferred residual for **LFMwith**$^+$ is more certain than that for **Hart**$^+$ as the periodic residual model in **LFMwith**$^+$ shares information between cycles. Consequently, the predictions drawn using **LFMwith**$^+$ are more accurate than those from **Hart**$^+$ as we will see subsequently. In addition in Figure 8 in the plot of the residual for **LFMwith**$^+$, we observe that this residual tries to compensate for the errors in the daily fall in temperature at the start of each day between times 7.0 and 7.2, 8.0 and 8.2, 9.0 and 9.2 and again between 10.0 and 10.2. We will show that these effects can have a significant impact on the day ahead prediction of the internal temperature. Although the residual model for **LFMwith**$^+$ is less certain than that for **Hart**$^+$, it captures the residual errors that arise due to using the simple thermal model in Equation (73). For instance, at the start of each day, when the heating comes on, the residual for **LFMwith**$^+$ shows a sharp spike, which represents a thermal lag in the physical process: in our homes a boiler heats up water, which, as it flows through radiators, indirectly heats up the air inside. The residual for **Hart**$^+$ however, is unable to accurately model this lag.

Figure 9 shows the day-ahead prediction of the temperature for the day immediately following the training days in Figure 8 using all seven algorithms. The **LFMquasi (CQM)**$^+$ prediction of the internal temperature has the smallest RMSE and one of the largest log likelihoods. This indicates that the underlying model is much more accurate than those of the other approaches. The RMSE for the example in Figure 9 is shown in Table 4.

Method	RMSE
LFMquasi (SQM)$^+$	0.53
LFMquasi (CQM)$^+$	**0.48**
LFMwith$^+$	0.49
LFMwithout$^+$	0.84
Hart$^+$	0.70
Resonator$^+$	0.70

Table 4: Internal temperature prediction RMSE of real home data `data1` over day 11 for non-periodic, quasi-periodic, periodic and no residual models.

To fully test the accuracy of the inferred residual model we evaluated the RMSE and the expected log likelihood of the predicted temperature for each day and for each algorithm over the 10 days for both homes for which predictions were generated. The example in Figures 8 and 9 corresponds to data set `data1`, day 11. Table 5 presents the expected RMSE and the expected log likelihood of the predicted internal temperatures for each home. LFMs with periodic and quasi-periodic eigenfunction residual models have both the best RMSE and expected log likelihoods for `data1` and `data2`. The best periodic model overall with a mean RMSE of 0.52 ± 0.05 across both data sets and an expected log likelihood of -106 ± 13 is the **LFMwith**$^+$. The **Resonator**$^+$ model has a lower consistency with an expected log likelihood of -1373 ± 1027 and it also has a higher overall RMSE at 1.15 ± 0.22.[7]

7. The relative performance of the eigenfunction and resonator models depends on how well the model parameters are learned. Of course, changing the parameter inference mechanism could effect the per-

The **LFMwithout**$^+$ model is weak as it is unable to accurately explain the training data without a residual model. Furthermore, although the **Hart**$^+$ approach has a very precise residual model, as shown in Figure 8, its predictions are weak since it is unaware that the residual is periodic. The non-periodic Matérn kernel, that is used by **Hart**$^+$, is unable to make accurate long term predictions of the residual since the learned input length scale for the residual is short. We note that the **LFMquasi (WQM)**$^+$ performs relatively badly on these data sets whereas the same algorithm performs well in the call centre application in Section 8. The reason for this is that the **LFMquasi (WQM)**$^+$ best models quasi-periodicity when the output scale of the residual changes between periods. Since the output scale for the heat residual does not vary from day to day then this model gives a poor fit. However, the scale of the queue length varies significantly from day to day within the call centre application and this is best modelled via the **LFMquasi (WQM)**$^+$. We note that the **LFMwith**$^+$, **LFMquasi (WQM)**$^+$, **LFMquasi (SQM)**$^+$ and **LFMquasi (CQM)**$^+$ each used fewer than 24 significant basis functions to represent the residual process. Consequently, our LBM Kalman filter approach to LFM prediction is computationally efficient.

Method	data1		data2		Overall	
	RMSE	ELL	RMSE	ELL	RMSE	ELL
LFMquasi (SQM)$^+$	0.73 ± 0.22	-133 ± 24	0.59 ± 0.07	$\mathbf{-90 \pm 8}$	0.67 ± 0.13	-116 ± 15
LFMquasi (CQM)$^+$	0.85 ± 0.15	-166 ± 17	0.89 ± 0.19	-144 ± 20	0.87 ± 0.11	-157 ± 13
LFMquasi (WQM)$^+$	1.15 ± 0.14	-260 ± 17	0.94 ± 0.17	-179 ± 27	1.06 ± 0.11	-227 ± 18
LFMwith$^+$	$\mathbf{0.51 \pm 0.06}$	$\mathbf{-104 \pm 19}$	$\mathbf{0.55 \pm 0.09}$	-108 ± 19	$\mathbf{0.52 \pm 0.05}$	$\mathbf{-106 \pm 13}$
LFMwithout$^+$	0.59 ± 0.08	-156 ± 40	0.65 ± 0.11	-121 ± 21	0.61 ± 0.06	-142 ± 25
Hart$^+$	1.03 ± 0.17	-183 ± 21	0.75 ± 0.17	-130 ± 20	0.92 ± 0.12	-162 ± 16
Resonator$^+$	1.39 ± 0.32	-2122 ± 1702	0.79 ± 0.22	-250 ± 110	1.15 ± 0.22	-1373 ± 1027

Table 5: Day ahead prediction (real home data): RMSE and expected log likelihood (ELL) for non-periodic, quasi-periodic, periodic and no residual models.

We also evaluated the algorithms on the data when tracking the internal temperature over a day. We note, when tracking, the heater output is known at each time instant and, thus, it is not necessary to use the RBPF whose sole purpose is to accommodate uncertainty in the binary heater output. Thus, each LFM is now implemented through a standard Kalman filter. The LFM models were trained over four consecutive days as described above, but, in this case, the day ahead internal temperatures were filtered using measurements obtained every 100 minutes. Table 6 presents the expected RMSE and the expected log likelihood of the internal temperatures for each home.

To determine the efficacy of the algorithms under more pronounced residual forces we simulated the heater output and, consequently, the internal temperature for residual heat drawn from a crisp quasi-periodic Matérn Gaussian process. We drew the residual process from the step-quasi model (SQM) as this model was a good representation of the real

formance measures reported in this paper. Given this, we endeavoured to extract the best performance from each model.

Method	data1		data2		Overall	
	RMSE	ELL	RMSE	ELL	RMSE	ELL
LFMquasi (SQM)	0.19 ± 0.01	85 ± 13	0.28 ± 0.04	12 ± 25	$\mathbf{0.22 \pm 0.02}$	$\mathbf{56 \pm 15}$
LFMquasi (CQM)	0.19 ± 0.02	63 ± 12	$\mathbf{0.27 \pm 0.04}$	-9 ± 26	$\mathbf{0.22 \pm 0.02}$	34 ± 15
LFMquasi (WQM)	0.26 ± 0.06	51 ± 30	0.29 ± 0.06	-23 ± 44	0.27 ± 0.04	21 ± 26
LFMwith	$\mathbf{0.18 \pm 0.02}$	$\mathbf{87 \pm 11}$	0.32 ± 0.04	-41 ± 29	0.24 ± 0.02	35 ± 21
LFMwithout	0.22 ± 0.03	48 ± 16	0.29 ± 0.04	6 ± 25	0.25 ± 0.02	31 ± 14
Hart	0.21 ± 0.02	78 ± 15	$\mathbf{0.27 \pm 0.05}$	$\mathbf{26 \pm 24}$	0.23 ± 0.02	55 ± 14
Resonator	0.82 ± 0.34	-190 ± 87	0.81 ± 0.35	-343 ± 225	0.82 ± 0.24	-251 ± 101

Table 6: Tracking a day ahead: RMSE and expected log likelihood (ELL) for non-periodic, quasi-periodic, periodic and no residual models.

data as demonstrated in Table 5. We then inferred the internal temperature process using Equation (73). Although we found that all three quasi-periodic models exhibited similar RMSE performance for day ahead tracking, the SQM model showed significant performance improvement over all other models when predicting a day ahead.

Example estimates for each prediction algorithm are shown in Figure 10. We re-evaluated the filter algorithms on this simulated data and the results are presented in Table 7. The **LFMquasi (SQM)$^+$** exhibits the lowest RMSE and highest log likelihood overall with values 1.00 ± 0.16 and -190 ± 28, respectively, which isn't surprising as the alternative approaches all use incorrect models for the residual. However, specifically the **LFMquasi(SQM)$^+$** is significantly more accurate and consistent than the **Resonator$^+$** model, which has an RMSE and expected likelihood of 1.43 ± 0.19 and -279 ± 54, respectively. Consequently, despite the flexibility of the resonator model, it is unable to capture the dynamics of the SQM generated residual as it has not been informed of the prior nature of the residual and further, is unable to recover this information from the data. Clearly, encoding the appropriate prior model for the residual is critical for tracking the internal temperature accurately.

Method	data1		data2		Overall	
	RMSE	ELL	RMSE	ELL	RMSE	ELL
LFMquasi (SQM)$^+$	$\mathbf{0.75 \pm 0.24}$	$\mathbf{-161 \pm 30}$	$\mathbf{1.12 \pm 0.20}$	$\mathbf{-204 \pm 40}$	$\mathbf{1.00 \pm 0.16}$	$\mathbf{-190 \pm 28}$
LFMquasi (CQM)$^+$	1.37 ± 0.26	-288 ± 59	1.37 ± 0.23	-258 ± 48	1.37 ± 0.17	-268 ± 37
LFMquasi (WQM)$^+$	1.43 ± 0.40	-236 ± 39	1.46 ± 0.27	-241 ± 27	1.45 ± 0.21	-239 ± 21
LFMwith$^+$	1.02 ± 0.34	-338 ± 198	1.23 ± 0.25	-377 ± 116	1.16 ± 0.19	-364 ± 97
LFMwithout$^+$	1.57 ± 0.38	-376 ± 128	1.70 ± 0.36	-362 ± 101	1.66 ± 0.26	-367 ± 77
Hart$^+$	1.48 ± 0.37	-247 ± 36	1.40 ± 0.22	-294 ± 80	1.43 ± 0.19	-279 ± 54
Resonator$^+$	1.48 ± 0.37	-247 ± 36	1.40 ± 0.22	-294 ± 80	1.43 ± 0.19	-279 ± 54

Table 7: Day ahead prediction (partially simulated home data): RMSE and expected log likelihood (ELL) for simulated non-periodic, quasi-periodic, periodic, Resonator and no residual models.

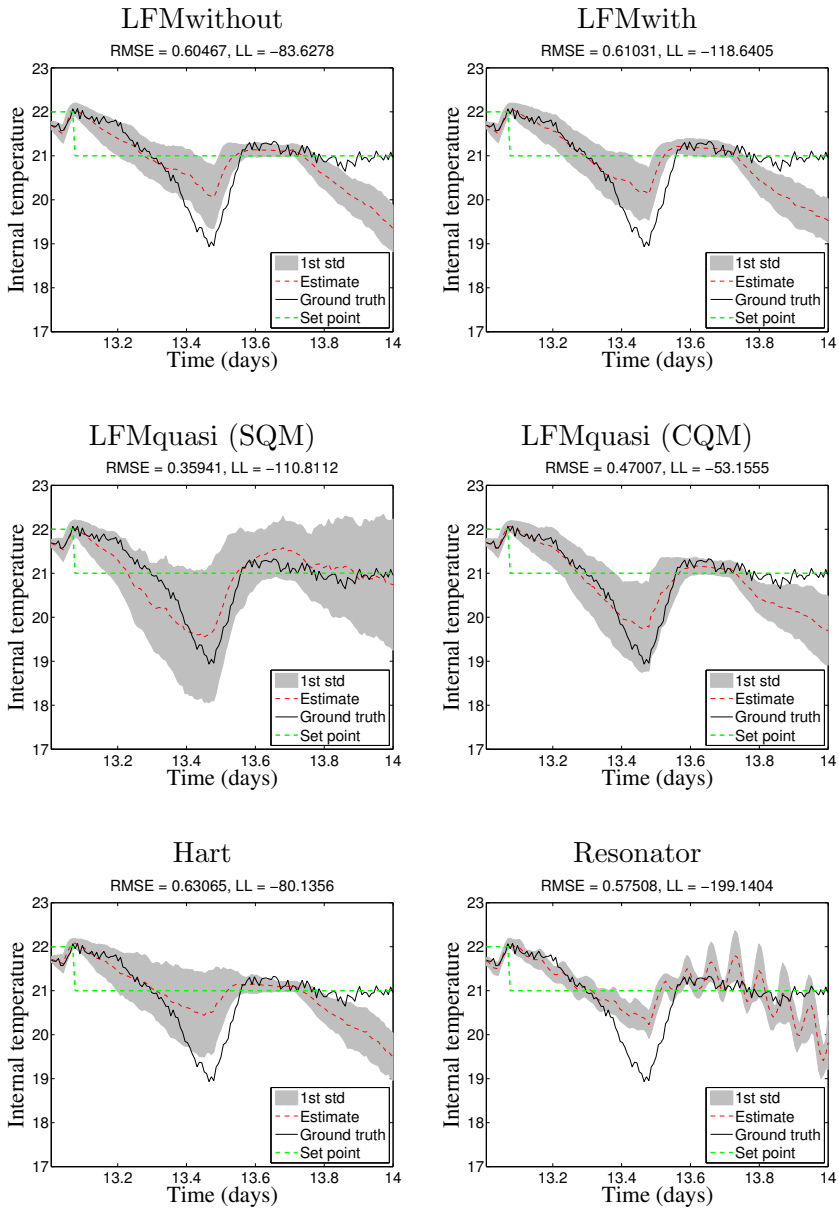


Figure 10: Predicted internal temperature compared to simulated value using the **LFMwithout** (top left), the **LFMwith** (top right), the **LFMquasi (SQM)** (middle left), the **LFMquasi (CQM)** (middle right), the **Hart** (bottom left) and the **Resonator** (bottom right) algorithms. The 1st standard deviation confidence interval is shown (grey). Also shown is the thermostat set point (green).

We also compared the run times for each algorithm.[8] We collected the time it took to train each model on four days of data, predict an entire day ahead and then track

8. The run times were determined using a Macbook Pro with a 2.4 GHz Intel Core i7 processor and 8 GB of memory.

the internal temperature over that day. For each run the resonator model and **LFMquasi (SQM)**$^+$ used exactly the same number of resonators and eigenfunctions, respectively. The resonator model used between 19 and 21 resonators during the experiment. Further, the resonator model was provided with a bias term to accommodate non-zero mean residuals. Figure 11(a) shows a box plot of the single output algorithm run times. The resonator algorithm is clearly the slowest as the model inference for the resonator requires a search over a space of frequency and decay coefficients. A detailed breakdown and comparison of the computational costs of the eigenfunction model and resonator model is presented in Section B.2.

Finally, we demonstrate the efficacy of our approach at modelling a multi-output system and consider an extension to the thermal model that incorporates the effect of a building's *envelope* as proposed in Bacher and Madsen (2011). The building's envelope comprises mainly the walls which act as a thermal reservoir and delay the heat transfer between the inside and the outside of the building. The multi-output model is represented by a system of coupled differential equations as

$$\frac{dT_{int}(t)}{dt} = \alpha\left(T_{env}(t) - T_{int}(t)\right) + \beta E(t) + R(t), \tag{74}$$

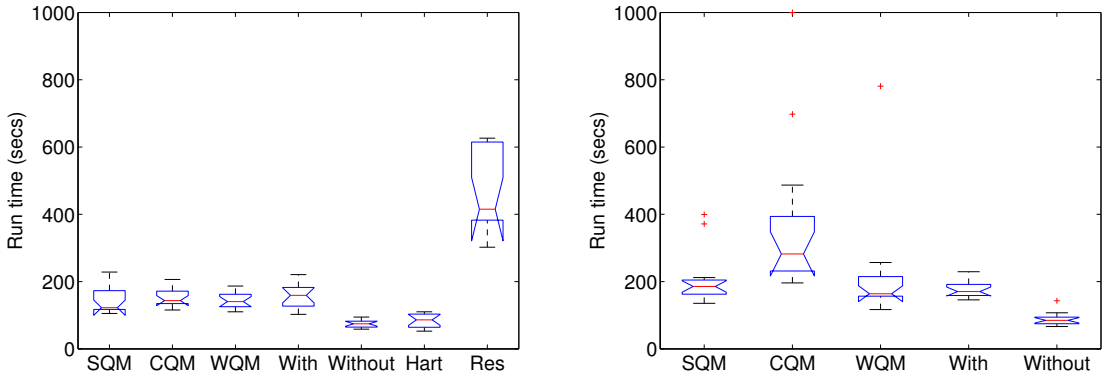
$$\frac{dT_{env}(t)}{dt} = \Gamma\left(T_{int}(t) - T_{env}(t)\right) + \Psi\left(T_{ext}(t) - T_{env}(t)\right). \tag{75}$$

Here, T_{int} and T_{env} are the internal temperature within the home and the temperature of a building's envelope, respectively. T_{env} is not directly observed, and has to be inferred from the data. The parameters in this model include: i) β, which represents the thermal output of the heater, ii) α, which regulates the convective heat transfer from the internal ambient air to the envelope, iii) Γ, which weights the convective heat transfer from the envelope to the ambient air and iv) Ψ, which represents the leakage coefficient to the ambient environment. In this model $T_{ext}(t)$ and $E(t)$ are the latent forces.

To infer the internal temperature of the building using the envelope model we introduce T_{env} to the Kalman filter state vector and two further parameters, Γ and Ψ, whose most likely values are inferred from the training data. We repeated the experiments on the real data above but this time using the envelope model. Figure 12 shows example day-ahead predictions of the internal and envelope temperatures for day 10 in data set `data1`.

Table 8 presents the expected RMSE and the expected log likelihood of the predicted internal temperatures for each home. The best algorithm is the **LFMquasi (SQM)**$^+$ which uses a quasi-periodic residual model. Referring to the performance of the single-output model in Table 5 it is interesting to note that the addition of the envelope, as proposed in Bacher and Madsen (2011), improves the overall performance of all the algorithms.

However, of key importance for the application of our approach to multi-output latent force models in general is the RBPF run times for this model and how they compare with the single output case. Figure 11(b) shows a box plot of the run times (the total time to train the RBPF, predict a day ahead and also track a day ahead) for the multi-output case. The run times compare favourably with the run times for the single output case despite the fact that the multi-output model requires two extra parameters. The increased



(a) Single output thermal model (b) Multi-output thermal model

Figure 11: Empirical distribution of run times (in seconds) for each LFM algorithm for both the single output and multi-output latent force models. The time for a sample run includes the time to train the model, predict a day ahead and also track a day ahead.

Method	data1		data2		Overall	
	RMSE	ELL	RMSE	ELL	RMSE	ELL
LFMquasi $(SQM)^+$	$\mathbf{0.19 \pm 0.02}$	$\mathbf{80 \pm 12}$	$\mathbf{0.27 \pm 0.04}$	7 ± 19	$\mathbf{0.22 \pm 0.02}$	$\mathbf{51 \pm 14}$
LFMquasi $(CQM)^+$	0.29 ± 0.07	12 ± 30	0.28 ± 0.04	-8 ± 18	0.29 ± 0.05	4 ± 19
LFMquasi $(WQM)^+$	0.20 ± 0.02	48 ± 27	$\mathbf{0.27 \pm 0.05}$	-1 ± 32	0.23 ± 0.02	29 ± 21
$LFMwith^+$	0.21 ± 0.02	56 ± 19	0.32 ± 0.03	-26 ± 23	0.25 ± 0.02	23 ± 18
$LFMwithout^+$	0.27 ± 0.05	24 ± 26	0.28 ± 0.05	$\mathbf{10 \pm 23}$	0.27 ± 0.04	19 ± 18

Table 8: Predicting internal temperature a day ahead using the multi-output model: RMSE and expected log likelihood (ELL) for quasi-periodic and periodic models on real thermal data from two homes.

computational cost for the multi-output model is due to the extra parameters in the multi-output model and this cost would be present if the standard Gaussian process inference equations, as Equations (2) and (3), were used in place of the Kalman filter. In general, the computational complexity of the Kalman filter scales quadratically with the size of the state vector and so multiple output processes can be accommodated efficiently. We note that our approach has a linear cost when conditionally independent measurements of multiple processes are incorporated. This contrasts with the standard Gaussian process inference equations which have a cubic cost in the number of processes and measurements from each process due to the need to invert a covariance matrix over all the processes.

In both the home heating application and the call centre application the eigenfunction-based models demonstrated the best performance, with improved RMSE and expected log likelihood over non-residual models, non-periodic models and the resonator model. Further, the quasi-periodic residual models were shown to outperform perfectly periodic models on

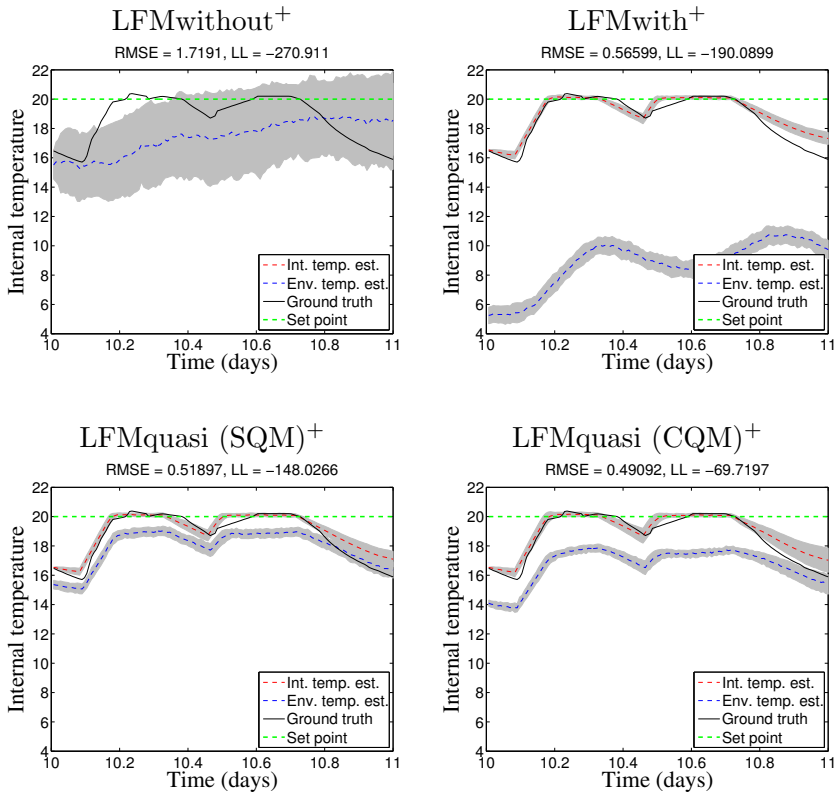


Figure 12: Internal temperature predictions compared to actual real value using the **LFMwithout**$^+$ (top left), the **LFMwith**$^+$ (top right), the **LFMquasi (SQM)**$^+$ (bottom left), the **LFMquasi (CQM)**$^+$ (bottom right) algorithms. The 1st standard deviation confidence interval is shown (grey). Also shown is the thermostat set point (green).

problems for which regular human behaviours, such as queuing as customers or heating homes through cooking or switching on the heating, have some influence. We noted that the WQM model had the best performance on the call centre application but the SQM exhibited the best predictive performance on the thermal modelling application. The WQM performed well on the call centre application because that application included residual forces, in this case arrival rates, which varied in amplitude from day to day. The SQM succeeded in the thermal modelling application because the residual heat profile varied slightly from day to day whilst maintaining a constant overall amplitude.

10. Conclusions

We have derived a novel and principled Bayesian approach to latent force modelling which accommodates both periodic and non-periodic forces. This approach can be incorporated within computationally efficient, iterative state-space approaches to inference. We are the

first to demonstrate that eigenfunctions can be used to model periodic forces within state-space approaches to LFM inference and we offer the only principled approach to incorporating periodic covariance functions within a state-space approach to inference with LFMs. We use the approach in Hartikainen and Särkkä (2010) for modelling non-periodic kernels and eigenfunction basis functions for modelling periodic kernels within a state-space approach to inference. We demonstrated that our eigenfunction approach out-performs the sparse spectrum Gaussian process regression (SSGPR) approach developed by Lázaro-Gredilla et al. (2010). Further, we demonstrated the close link between the eigenfunction model and the resonator model proposed by Särkkä et al. (2012). Consequently, we are the first to demonstrate how any periodic covariance function can be encoded within the resonator model using the covariance function's eigenfunctions. We are also the first to demonstrate that eigenfunctions can be represented via the resonator model within Kalman filters if required. Thus, we have proposed, in this paper, the only two approaches to date that are able to incorporate all types of Gaussian periodic model priors within a state-space approach to LFM inference. These priors include stationary periodic, non-stationary periodic and quasi-periodic covariance functions.

We have applied our approach to two applications: call centre customer queues and thermal modelling of homes. In detail, within the call centre application, customer arrival rates were modelled as driving forces through a differential model approximation of the Poisson arrival process. Both periodic and quasi-periodic models were developed to model the arrival rates of customers. The periodic models improve on the non-periodic model by as much as 83% in the root-mean-squared error. In the home heating application we modelled the thermal dynamics of homes where the physics of the energy exchange process is known but some of the heat generating processes are not known in advance. Our approach can learn the unknown heat dynamics from data and is able to accurately predict internal temperatures 24 hours ahead. Again, both periodic and quasi-periodic models were developed but, in this case, to model residual heat within the home. In this case the periodic models improve on the non-periodic model by reducing the RMSE by as much as 28%. Overall, the quasi-periodic models produced the lowest mean-squared-error and the highest expected log likelihood. Further, the eigenfunction model demonstrated improved performance over the resonator model. In the thermal application the eigenfunction models improve on the resonator by reducing the RMSE by as much as 74%.

In both the thermal modelling application and the call centre application the periodic residual models demonstrated the best performance, with improved RMSE and expected log likelihood over non-residual models, non-periodic models and the resonator model. Further, the quasi-periodic residual models were shown to outperform perfectly periodic models in the presence of regular human behaviours, such as customer queues and heating homes through cooking or switching on the heating. We noted that the WQM model had the best performance for the call centre application but the SQM exhibited the best performance on the thermal application. The WQM performed well on the call centre application because that application included residual forces, in this case arrival rates, which varied in amplitude from day to day. The SQM succeeded in the thermal application because the residual heat profile varied slightly from day to day whilst maintaining a constant overall amplitude.

Both applications deployed state-space approaches to LFMs and both applications utilized the eigenfunction representation of the periodic latent forces acting on the system.

These applications demonstrated the efficacy of our approach on both long term predictions and tracking problems. The applications demonstrated LFM inference on both linear (home heating) and non-linear (call centre) problems; on latent forces with constant output scale (home heating) and variable output scale (call centre) and on purely Gaussian models (call centre) and models involving both Gaussian and binomial variables (thermal). We also demonstrated both single output Gaussian process and multi-output Gaussian process regression in the home heating application.

As we noted in the Appendix, the eigenfunction is the optimal RMSE basis model for any covariance function. However, our approach uses only the eigenfunctions derived from the covariance function prior. Consequently, an optimal J-dimensional model should adapt its basis functions to the data set and the eigenfunctions of the posterior covariance function should be used. We believe it is possible to extend our approach to accommodate adaptable eigenfunctions and this will be the focus of further work. Further, the Kalman formalism expressed in this paper lends itself immediately to control problems and we intend to investigate our approach to LFM inference within model-based predictive control. This research will be of particular value to domains in which some physical knowledge of the process is known (and expressible via differential equations) and nonparametric models can be used to express the latent forces. We will explore the relative merits of expressing control problems directly via the Gaussian process prior as in, for example, Ažman and Kocijan (2008), and via the Markovian formalism advocated in this paper.

Acknowledgments

This work was funded in the UK by the EPSRC ORCHID programme grant (EP/I011587/1) and the EPSRC 'Intelligent Agents for Home Energy Management' project (EP/I000143/1), and in Kingdom of Saudi Arabia by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah (9-15-1432-HiCi).

Appendix A. Discrete Jump Markov Processes for Non-Stationary Covariance Functions

We describe how the Step Quasi model (SQM) and the Wiener-step Quasi model (WQM) can be incorporated within the discrete time Kalman filter.

Suppose that either $a \sim \mathcal{GP}(0, K^{\mathrm{SQM}})$ or $a \sim \mathcal{GP}(0, K^{\mathrm{WQM}})$, where K^{SQM} and K^{WQM} are the covariance functions for the SQM and WQM, respectively. Consider a change point, τ and some earlier time τ_- close to τ such that $\tau > \tau_-$. We assume that

$$a(\tau) = Ga(\tau_-) + \chi(\tau), \tag{76}$$

where G is the Kalman filter process model and $\chi(\tau) \sim \mathcal{N}(0, Q)$. We will now see that G and Q can be expressed in terms of the kernels, K^{SQM} and K^{WQM} at the change point τ. Recall $E[a(t)] = E[\chi(t)] = 0$, $E[a(t)\chi(t)] = 0$ and $K(t, t') = E[a(t)a(t')]$ for all t and t'. Thus, by squaring both sides of Equation (76) and then taking the expectation we can express the variance, $K(\tau, \tau)$, of $a(\tau)$ as

$$K(\tau, \tau) = GK(\tau_-, \tau_-)G + Q. \tag{77}$$

Also, multiplying Equation (76) throughout by $a(\tau_-)$ before taking the expectation gives the covariance between $a(\tau)$ and $a(\tau_-)$ as

$$K(\tau, \tau_-) = GK(\tau_-, \tau_-). \tag{78}$$

Specifically, from Equation (39), the SQM variance, $K^{\text{SQM}}(\tau, \tau) = \sigma^2$ and $K^{\text{SQM}}(\tau, \tau_-) = \sigma^2 \exp(-1/l)$ as $C(\tau) - C(\tau_-) = 1$ in Equation (39) across a single change point. Thus, using Equations (77) and (78), the process model, G, and process noise variance, Q, for the SQM are

$$G_{\text{SQM}} = \exp\left(-\frac{1}{l}\right) \qquad \text{and} \qquad Q_{\text{SQM}} = \sigma^2\left(1 - \exp\left(-\frac{2}{l}\right)\right).$$

Also, by Equation (40) the WQM variances, $K^{\text{WQM}}(\tau, \tau) = \xi_0 + C(\tau)\xi$, $K^{\text{WQM}}(\tau_-, \tau_-) = \xi_0 + C(\tau_-)\xi$ and covariance, $K^{\text{WQM}}(\tau, \tau_-) = \xi_0 + C(\tau_-)\xi$. Thus, by Equations (77) and (78), the process model, G, and process noise variance, Q, for the WQM are

$$G_{\text{WQM}} = 1 \qquad \text{and} \qquad Q_{\text{WQM}} = \xi$$

as $C(\tau) - C(\tau_-) = 1$.

Appendix B. Comparison of Eigenfunction and Resonator Models

In this section we assert that the eigenfunction basis model advocated in this paper is optimal in that it minimizes the mean squared error for all possible J-dimensional basis models and thus establish the eigenfunction approach as the preferred approach. We shall then develop the theoretical link between the eigenfunction basis model and the *resonator model* (Särkkä et al., 2012; Hartikainen et al., 2012; Solin and Särkkä, 2013) which is the most significant alternative approach to modelling periodic forces in LFMs. Consequently, we will demonstrate that the resonator model parameters can be chosen so that the resonator basis is equivalent to the eigenfunction basis. As a corollary we propose a novel mechanism for encoding periodic covariance function priors in the resonator model.

B.1 Establishing the Link Between the Resonator Basis and Eigenfunctions

A *J-dimensional linear model* is a linear combination of J basis functions. Both the eigenfunction model, as per Equation (25), and resonator model, as per Equation (18), are J-dimensional linear models. The eigenfunction model, as per Equation (25), is a linear combination of orthonormal basis functions, ϕ_j, whereas the resonator model is a linear combination of resonators, ψ_j, which are not necessarily orthogonal.

Let g be some function drawn from a Gaussian process with covariance function K. Then the Karhunen-Loéve expansion theorem (Loéve, 1955) states that the eigenfunction basis is the orthonormal basis that minimizes the total mean squared error between the J-dimensional model and g. Further, any non-orthonormal basis with cardinality, α, can be converted to an orthonormal basis with cardinality, υ, such that $\upsilon \leq \alpha$, by Gram-Schmidt orthogonalisation (Arfken et al., 2005) and renormalisation. Thus, we can establish imme-

diately that the eigenfunction basis is the optimal mean squared basis for all J-dimensional linear models.[9]

In the remainder of this section we determine the conditions under which each version of the resonator model, as per Equations (19) and (20), is equivalent to the optimal eigenfunction model.

B.1.1 Perfectly Periodic and Stationary Covariance Functions

A *perfectly periodic stationary process* $g \sim \mathcal{GP}(b, K)$ with period D satisfies, $g(t + nD) = g(t)$ for all $t \in \mathbb{R}$ and $n \in \mathcal{N}$. Such functions (for example, the squared-exponential in Equation (16)) are generated from Gaussian processes with covariance functions of the form $K(t, t') = h(t - t')$ for some function h.

Bochner's theorem (see, for example, Rasmussen and Williams, 2006) states that the eigenfunctions of a stationary kernel are the Fourier basis functions. Thus, the optimal J-dimensional linear model for a stationary Gaussian process is a linear combination of Fourier basis functions. Both resonator models, in Equations (19) and (20), can model Fourier basis functions exactly by asserting $\omega_j(t) = 0$ for all time t and all resonators, j, in Equation (19), and assigning a constant resonator frequency, f, in the original resonator model, as per Equation (19), or removing the decay term by setting $B_j = 0$ in the later model, as per Equation (20), and assigning $A_j = -(2\pi f_j)^2$ so that

$$\frac{d^2\psi_j(t)}{dt^2} = -(2\pi f_j)^2\psi_j(t).$$

Thus, the optimal J-dimensional linear model for the stationary kernel case is an instance of both resonator models.

B.1.2 Perfectly Periodic and Non-stationary Covariance Functions

A *perfectly periodic non-stationary process* $g \sim \mathcal{GP}(b, K)$ with period D satisfies, $g(t+nD) = g(t)$ for all $t \in \mathbb{R}$ and $n \in \mathcal{N}$. Such processes (for example, Equation (32)) are Gaussian processes with covariance functions of the form $K(t, t') = h(t, t')$ where $h(t, t') \neq h(t - t')$. Note that since the latent force g is perfectly periodic then the resonator cannot be stochastic (that is, $\omega_j(t) = 0$ for all time t and resonator, j, in Equation (19)).

We demonstrate that the eigenfunctions for non-stationary covariance functions can be represented by the resonator model using the time varying frequency model, as per Equation (20), provided that the eigenfunction is second order differentiable. We note that the eigenfunction linear basis model, as per Equation (29), and the resonator model, as per Equation (18), are equivalent if

$$\psi_j(t) = a_j\phi_j(t), \tag{79}$$

9. In this paper, we use a static basis chosen from the prior covariance function. However, the eigenfunctions are dependent on the covariance function and consequently, an optimal J-dimensional model should adapt its basis when evidence is integrated with the prior. We believe it is possible to extend our approach to accommodate adaptable eigenfunctions and this will be the focus of a further paper. We note that the resonator model can also adapt to the evidence provided the frequency process in Equation (19) adapts with the data.

for eigenfunction, ϕ_j, resonator, ψ_j, times, t, and some positive coefficient, a_j, as per Equation (18). Substituting Equation (79) into Equation (19), asserting $\omega_j(t) = 0$ (as above) and rearranging we get

$$(2\pi f_j(t))^2 = -\frac{1}{\phi_j(t)}\frac{d^2\phi_j(t)}{dt^2}. \tag{80}$$

Thus, any perfectly periodic covariance function can be encoded within the resonator model by defining the frequency process, $f_j(t)$, in terms of the covariance function eigenfunctions, $\phi_j(t)$. Furthermore, we can also represent eigenfunctions via the resonator model within Kalman filters if required. In practise, the Nyström approximation, $\tilde{\phi}$, for the eigenfunction basis is used in place of ϕ in Equation (80) to calculate the frequency process for the resonator model.[10]

To illustrate the link between the eigenfunction and corresponding resonator models for perfectly periodic covariance functions we derive the frequency process, $f(t)$, for a variation of the non-stationary covariance function in Equation (32) with a low smoothness, $\nu = 3/2$, as

$$K(t,t') = \text{Matérn}(\kappa(t-t'),\nu,\sigma,l)\exp(-\alpha(\kappa(t)^2 + \kappa(t')^2)), \tag{81}$$

where $\kappa(\tau) = |\sin(\pi\tau/D)|$, D is the covariance function period and $\alpha > 0$ is the decay rate. This covariance function differs from Equation (32) in two crucial respects. Firstly, it is now perfectly periodic with period D and secondly, it is second order differentiable everywhere, as required by Equation (80). For $\nu = 3/2$ the Matérn simplifies

$$\text{Matérn}(\kappa(\tau),3/2,\sigma,l) = \sigma^2(1+\sqrt{3}\kappa(\tau)/l)\exp(-\sqrt{3}\kappa(\tau)/l).$$

Using the Nyström approximation, as per Equation (28), we get

$$\frac{d^2\tilde{\phi}_i(t)}{dt^2} = \frac{\sqrt{N}}{\mu_i}\frac{d^2K(t,S)}{dt^2}\mathbf{v}_i. \tag{82}$$

After some algebra we obtain

$$\begin{aligned}\frac{d^2K(t,t')}{dt^2} &= \frac{d^2\text{Matérn}(\kappa(\tau),3/2,\sigma,l)}{d\tau^2}\exp(-\alpha\kappa(t)^2)\exp(-\alpha\kappa(t')^2)\\ &\quad+\text{Matérn}(\kappa(\tau),3/2,\sigma,l)\frac{d^2\exp(-\alpha\kappa(t)^2)}{dt^2}\exp(-\alpha\kappa(t')^2)\\ &\quad+2\frac{d\text{Matérn}(\kappa(\tau),3/2,\sigma,l)}{d\tau}\frac{d\exp(-\alpha\kappa(t)^2)}{dt}\exp(-\alpha\kappa(t')^2),\end{aligned}$$

10. We note by Equation (80) the resonator can become unstable close to $\tilde{\phi} = 0$. This problem is easily solved by initially adding some offset, Δ, to $\tilde{\phi}$ for some suitably large Δ before calculating the frequency process $f(t)$. Consequently, when $f(t)$ is used in the resonator model, as per Equation (19) the corresponding resonator, $\psi(t)$, represents the eigenfunction basis plus the bias Δ. This bias can be removed by subtracting Δ from $\psi(t)$.

where $\tau = t - t'$ and

$$\begin{aligned}
\frac{d\text{Matérn}(\kappa(\tau), 3/2, \sigma, l)}{d\tau} &= -\frac{3\pi\sigma^2}{2Dl^2}\sin\left(\frac{2\pi}{D}\tau\right)\exp\left(-\frac{\sqrt{3}}{l}\kappa(\tau)\right), \\
\frac{d\exp(-\alpha\kappa(t)^2)}{dt} &= -\frac{\pi\alpha}{D}\sin\left(\frac{2\pi}{D}t\right)\exp(-\alpha\kappa(t)^2), \\
\frac{d^2\text{Matérn}(\kappa(\tau), 3/2, \sigma, l)}{d\tau^2} &= \frac{3\pi^2\sigma^2}{D^2l^3}\exp\left(-\frac{\sqrt{3}}{l}\kappa(\tau)\right)\left(\sqrt{3}\kappa(\tau)(1-\kappa(\tau)^2) - l(1-2\kappa(\tau)^2)\right), \\
\frac{d^2\exp(-\alpha\kappa(t)^2)}{dt^2} &= -\frac{2\pi^2\alpha}{D^2}\left(\frac{\alpha}{2}\left([1-2\kappa(t)^2]^2 - 1\right) + 1 - 2\kappa(t)^2\right)\exp(-\alpha\kappa(t)^2).
\end{aligned}$$

Subsequently, using Equations (80) and (82), the resonator model frequency process, f, for each resonator model can be chosen so that the resonator is equivalent to the eigenfunction thus

$$(2\pi f_j(t))^2 = -\frac{1}{\tilde{\phi}_j(t)}\frac{d^2\tilde{\phi}_j(t)}{dt^2}. \tag{83}$$

Figure 13 compares the eigenfunction and corresponding resonator whose frequency profiles are calculated using Equation (83). These basis functions are the four most significant eigenfunctions for the non-stationary periodic covariance function in Equation (81) with $D = 10$, $\alpha = 0.8$ and $l = 20$. The top panes show the eigenfunction and corresponding resonator and the bottom panes show the resonator coefficient, $(2\pi f(t))^2$, as per Equation (83), required by the resonator model to equate the resonator with the eigenfunction. We note the presence of negative resonator coefficient values $(2\pi f(t))^2$. These correspond to complex valued frequencies which model basis decay in a manner similar to the basis decay term in the alternative resonator model as per Equation (20).

We next examine the properties of the alternative resonator model, as per Equation (20), when representing perfectly periodic, non-stationary Gaussian processes. The alternative resonator model uses time invariant coefficients, A and B, thus

$$\frac{d^2\psi_j(t)}{dt^2} + A_j\frac{d\psi_j(t)}{dt} + B_j\psi_j(t) = \omega_j(t), \tag{84}$$

where ω_j is a white noise component. Modelling the non-stationary process via Equation (84) avoids the need to compute frequency processes using the interacting multiple model (IMM) in the original formalization of the resonator model (Särkkä et al., 2012). For perfectly periodic covariance functions (with period D) then $\psi_j(t + D) = \psi_j(t)$ for all t and consequently, as ω_j is i.i.d., then $\omega_j(t) = 0$ for all t. Thus, the solution of the previous equation is

$$\psi_j(t) = G_j\exp\left[(\pm i\sqrt{B_j - 0.25A_j^2} - 0.5A_j)t\right].$$

So that $\psi_j(t + D) = \psi_j(t)$ for all t then $A_j = 0$ and therefore

$$\psi_j(t) = G_j\exp\left[i\sqrt{B_j}t\right].$$

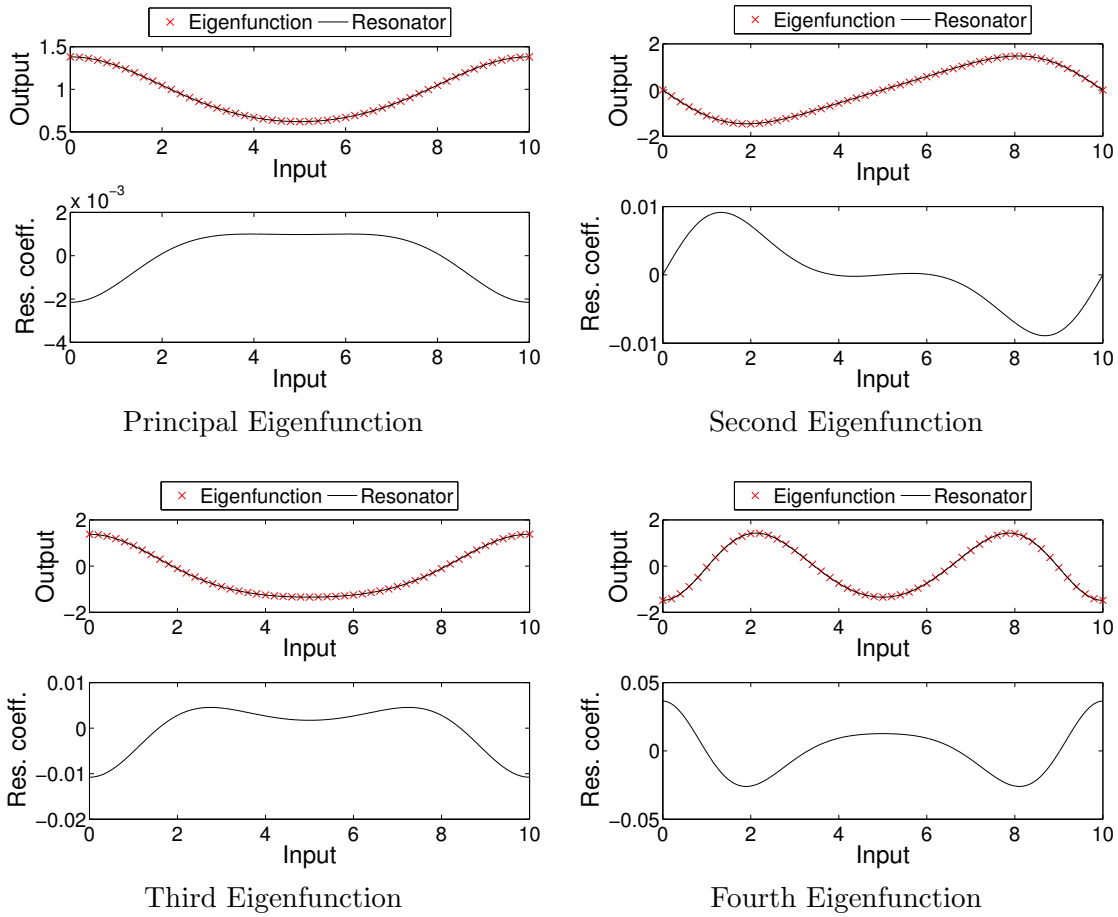


Figure 13: The four most significant Eigenfunctions and coincident resonators for the non-stationary periodic covariance function in Equation (81). In each pane the top graph shows the eigenfunction and the lower graph shows the resonator coefficient $(2\pi f(t))^2$ profile required by the resonator model to equate the resonator with the eigenfunction.

Consequently, expanding the exponential in terms of cosine and sine functions we see that ψ_j must be the Fourier basis functions. The Fourier basis is a sub-optimal basis for non-stationary covariance functions as, in general, the optimal eigenfunction basis is not Fourier (see, for example, Figure 13). Thus, the resonator model, as per Equation (84), is a sub-optimal representation for non-stationary periodic covariance functions.

B.1.3 Quasi-Periodic Covariance Functions

A *quasi-periodic process* $g \sim \mathcal{GP}(b, K)$ is generated from a Gaussian process with covariance function of the form $K(t,t') = K_{\text{quasi}}(t,t')K_{\text{periodic}}(t,t')$ where K_{quasi} is non-periodic and K_{periodic} is perfectly periodic (either stationary or non-stationary). Equation (34) is an

example of a quasi-periodic process covariance function. In general, when the periodic kernel, K_{periodic}, has period D then, with high probability, $g(t+D) \neq g(t)$ for all $t \in \mathbb{R}$ unlike the perfectly periodic case presented above.

Quasi-periodic eigenfunction models use a time varying weight coefficient, $a_j(t)$, as per Equation (35). Thus, extending Equation (79), in this case the resonator and eigenfunction linear basis models are equivalent if

$$\psi_j(t) = a_j(t)\phi_j(t).$$

Consequently, by substituting $\psi_j(t)$ into Equation (19), setting $\omega_j(t) = 0$ and rearranging we get the frequency process, $f_j(t)$, for each resonator for the quasi-periodic process we have

$$(2\pi f_j(t))^2 = -\frac{\ddot{\phi}_j(t)a_j(t) + \phi_j(t)\ddot{a}_j(t) + 2\dot{\phi}_j(t)\dot{a}_j(t)}{a_j(t)\phi_j(t)}.$$

Since the coefficient process $a_j(t)$ is stochastic then so too is $f_j(t)$. We note that both $\psi_j(t)$ and $a_j(t)$ must be inferred when using the resonator model and the Kalman filter. This places significant computational cost on the Kalman filter prediction equations. Thus, we do not recommend implementing quasi-periodic GP priors with the resonator model as per Equation (19).

The alternative resonator model, as per Equation (20) encodes a decay term, via the first order derivative of the basis, appropriate for modelling quasi-periodic covariance functions. This model is investigated empirically in Section 9 on a home heating prediction problem which exploits quasi-periodic latent forces.

B.2 Computational Complexity of Eigenfunction and Resonator Models

When the Gaussian process covariance function for each latent force is known, so that we can generate the appropriate eigenfunction basis for any choice of covariance function hyperparameters, then searching over the hyperparameter values of the covariance function can be significantly less computationally demanding than searching over the frequency space for a potentially large number of resonators.

When constructing the eigenfunction model the greatest computational cost arises from calculating the Nyström approximation. However, the significant eigenfunctions can be found iteratively and efficiently using Von Mises iteration. At each iteration the next largest eigenvalue and corresponding eigenfunction are found. This approach continues until all the significant eigenvalues are found. If J eigenfunctions with the largest eigenvalues are found using Von Mises iteration then the complexity of our approach is $\mathcal{O}(JN^2)$ where $N \times N$ is the size of the Gram matrix in Equation (27) obtained by sampling the periodic covariance function. Inferring the eigenfunction model also involves searching over a relatively small set of p hyperparameters, often of the order of about $p = 3$ parameters comprising the input scale, output scale and the period of the covariance function. If the set of admissible values along each hyperparameter dimension has cardinality Υ then the computational complexity of searching the parameter space is $\mathcal{O}(\Upsilon^p)$. The overall computational complexity of inferring the eigenfunction model is therefore $\mathcal{O}(\Upsilon^p JN^2)$.

The parameters of the J-dimensional resonator model can be found by solving a non-convex optimisation problem over a $3J$ dimension parameter space where the parameters are J Fourier basis function frequencies, J basis function phases and J magnitudes for the basis power spectrum. If the set of admissible values along each dimension has cardinality Υ then the computational complexity of searching the parameter space is $\mathcal{O}(\Upsilon^{3J})$. To identify the optimal choice of parameter values each parameter vector constructed during the search over the parameter space requires the comparison of $\mathcal{O}(N^2)$ entries between the sampled kernel and the covariance matrix of the target function induced by the resonator model. Thus the resonator model is inferred with computational complexity $\mathcal{O}(\Upsilon^{3J}N^2)$. We note that, whereas the resonator model training phase is exponentially complex in the number of basis functions, J, the eigenfunction model is linear in J.[11]

We compare the run times for the eigenfunction and resonator approaches empirically in Section 9.

B.3 Summary

We have demonstrated the link between the resonator model and the eigenfunction approach. Through this link we have been able to identify that,

1. the eigenfunction basis is optimal in that it minimizes the mean squared error between the J-dimensional model and the target function.

2. the variant frequency term in the resonator second order differential equation provides sufficient flexibility to yield basis functions which are equivalent to the eigenfunctions.

3. we have developed an algorithm for deriving optimal resonator models for all perfectly periodic covariance functions from the eigenfunctions of the covariance function. Thus, we are able to offer an efficient mechanism for encoding the GP prior in the resonator model.

References

M. Alvarez, J. Peters, B. Schoelkopf, and N. Lawrence. Switched latent force models for movement segmentation. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 55–63, 2010.

M. A. Alvarez and N. D. Lawrence. Sparse convolved Gaussian processes for multi-output regression. In *Advances in Neural Information Processing Systems 21*, pages 57–64, 2008.

M. A. Alvarez and N. D. Lawrence. Computationally efficient convolved multiple output Gaussian processes. *Journal of Machine Learning Research*, 12:1459–1500, 2011.

M. A. Alvarez, D. Luengo, and N. D. Lawrence. Latent force models. *Journal of Machine Learning Research - Proceedings Track*, 5:9–16, 2009.

11. Note that, when using the Nyström approximation of the eigenfunctions it is necessary to store the eigenvectors from which the eigenfunctions can be calculated. Although, for the stationary case, they can be calculated when required from the cosine and sine functions.

M. A. Alvarez, D. Luengo, and N. D. Lawrence. Linear latent force models using Gaussian processes. Technical report, Department of Computer Science, University of Manchester, UK, 2011. http://arxiv.org/abs/1107.2699.

G. B. Arfken, H. J. Weber, and F. E. Harris. *Mathematical Methods for Physicists*. Academic press, 2005.

K. Ažman and J. Kocijan. Non-linear model predictive control for models with local information and uncertainties. *Transactions of the Institute of Measurement and Control*, 30 (5):371–396, 2008.

P. Bacher and H. Madsen. Identifying suitable models for the heat dynamics of buildings. *Energy and Buildings*, 43(7):1511–1522, 2011.

Y. Bengio, O. Delalleau, N. Le Roux, J.-F. Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions link spectral embedding and kernel PCA. *Neural Computation*, 16:2197–2219, 2004.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1999. ISBN 0198538642.

DECC. Smarter grids: the opportunity. Technical report, Department of Energy and Climate Change (DECC), 2009. http://www.decc.gov.uk.

A. Doucet, J. F. G. de Freitas, K. Murphy, and S. Russel. Rao-Blackwellized particle filtering for dynamic Bayesian networks. In *Proc of the Conference on Uncertainty in Artificial Intelligence (UAI '00)*, pages 176—183, 2000.

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.

P. D. Feigin, A. Mandelbaum, S. Zeltyn, V. Trofimov, E. Ishay, P. Khudiakov, and E. Nadjharov. Datamocca: Data model for call center analysis, the call center of "US bank". http://ie.technion.ac.il/Labs/Serveng/files/The_Call_Center_of_US_Bank.pdf, 2006.

J. Hartikainen and S Särkkä. Kalman filtering and smoothing solutions to temporal Gaussian process regression models. In *Proc of IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 379–384, 2010.

J. Hartikainen and S. Särkkä. Sequential inference for latent force models. In *Proc of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11)*, pages 311–318, 2011.

J. Hartikainen, M. Seppanen, and S. Särkkä. State-space inference for non-linear latent force models with application to satellite orbit prediction. In *Proceedings of the 29th International Conference on Machine Learning (ICML '12), Edinburgh, Scotland*, 2012.

S. Ji and R. Zhou. Simulation and bottleneck analysis of container port handling resources based on bounding theory. In *International Conference on Logistics Systems and Intelligent Management (ICLSIM '10)*, volume 2, pages 705–708, 2010.

R. E. Kalman et al. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

D. G. Kendall. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, pages 338–354, 1953.

M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal. Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, 11: 1865–1881, 2010.

M. Loéve. *Probability Theory; Foundations, Random Sequences*. New York: D. Van Nostrand Company, 1955.

D. MacKay. *Sustainable Energy: Without the Hot Air*. UIT, Cambridge, 2009.

J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A*, 209:415–446, 1909.

A. O'Hagan. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society*, 40:1–42, 1978.

E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065—1076, 1962.

J. Quiñonero Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

S. Ramchurn, P. Vytelingum, A. Rogers, and N. R. Jennings. Putting the "smarts" into the smart grid: A grand challenge for artificial intelligence. *Communications of the ACM*, 55 (4):86–97, 2012.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain. Gaussian processes for time series modelling. *Phil. Trans. R. Soc. A*, 371(1984 20110550), 13 February 2013.

A. Rogers, S. Maleki, S. Ghosh, and N. R. Jennings. Adaptive home heating control through Gaussian process prediction and mathematical programming. In *Second International Workshop on Agent Technology for Energy Systems (ATES 2011)*, pages 71–78, 2011.

S. Särkkä, A. Solin, A. Nummenmaa, A. Vehtari, T. Auranen, S. Vanni, and F.-H. Lin. Dynamic retrospective filtering of physiological noise in BOLD FMRI: DRIFTER. *NeuroImage*, 2012.

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

J. Shawe-Taylor, C. K. I. Williams, N. Cristianini, and J. Kandola. On the eigenspectrum of the Gram matrix and the generalization error of kernel-PCA. *Information Theory, IEEE Transactions on*, 51(7):2510–2522, 2005.

M. Sims, J. Kurose, and V. Lesser. Streaming versus batch processing of sensor data in a hazardous weather detection system. In *Proceedings of Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2005)*, pages 185–196, September 2005.

E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1257–1264, 2006.

A. Solin and S. Särkkä. Infinite-dimensional Bayesian filtering for detection of quasi-periodic phenomena in spatio-temporal data. Technical report, Department of Biochemical Engineering and Computational Science, Aalto University, 2013.

M. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

W.-P. Wang, D. Tipper, and S. Banerjee. A simple approximation for modeling nonstationary queues. In *Proceedings of the Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies Conference on The Conference on Computer Communications - Volume 1*, pages 255–262, 1996.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

Journal of Machine Learning Research 15 (2014) 2399-2449 Submitted 1/14; Published 7/14

Spectral Learning of Latent-Variable PCFGs: Algorithms and Sample Complexity

Shay B. Cohen SCOHEN@INF.ED.AC.UK
School of Informatics
University of Edinburgh
Edinburgh, EH8 9LE, UK

Karl Stratos STRATOS@CS.COLUMBIA.EDU
Michael Collins MCOLLINS@CS.COLUMBIA.EDU
Department of Computer Science
Columbia University
New York, NY 10027, USA

Dean P. Foster DEAN@FOSTER.NET
Yahoo! Labs
New York, NY 10018, USA

Lyle Ungar UNGAR@CIS.UPENN.EDU
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA

Editor: Alexander Clark

Abstract

We introduce a spectral learning algorithm for latent-variable PCFGs (Matsuzaki et al., 2005; Petrov et al., 2006). Under a separability (singular value) condition, we prove that the method provides statistically consistent parameter estimates. Our result rests on three theorems: the first gives a tensor form of the inside-outside algorithm for PCFGs; the second shows that the required tensors can be estimated directly from training examples where hidden-variable values are missing; the third gives a PAC-style convergence bound for the estimation method.

Keywords: latent-variable PCFGs, spectral learning algorithms

1. Introduction

Statistical models with hidden or latent variables are of great importance in natural language processing, speech, and many other fields. The EM algorithm is a remarkably successful method for parameter estimation within these models: it is simple, it is often relatively efficient, and it has well understood formal properties. It does, however, have a major limitation: it has no guarantee of finding the global optimum of the likelihood function. From a theoretical perspective, this means that the EM algorithm is not guaranteed to give statistically consistent parameter estimates. From a practical perspective, problems with local optima can be difficult to deal with.

Recent work has introduced a polynomial-time learning algorithm for an important case of hidden-variable models: hidden Markov models (Hsu et al., 2009). This algorithm uses a spectral method: that is, an algorithm based on eigenvector decompositions of linear systems, in particular singular value decomposition (SVD). In the general case, learning of HMMs is intractable (e.g., see Terwijn, 2002). The spectral method finesses the problem of intractability by assuming separability conditions. More precisely, the algorithm of Hsu et al. (2009) has a sample complexity that is polynomial in $1/\sigma$, where σ is the minimum singular value of an underlying decomposition. The HMM learning algorithm is not susceptible to problems with local maxima.

In this paper we derive a spectral algorithm for learning of latent-variable PCFGs (L-PCFGs) (Petrov et al., 2006; Matsuzaki et al., 2005). L-PCFGs have been shown to be a very effective model for natural language parsing. Under a condition on singular values in the underlying model, our algorithm provides consistent parameter estimates; this is in contrast with previous work, which has used the EM algorithm for parameter estimation, with the usual problems of local optima.

The parameter estimation algorithm (see Figure 7) is simple and efficient. The first step is to take an SVD of the training examples, followed by a projection of the training examples down to a low-dimensional space. In a second step, empirical averages are calculated on the training examples, followed by standard matrix operations. On test examples, tensor-based variants of the inside-outside algorithm (Figures 4 and 5) can be used to calculate probabilities and marginals of interest.

Our method depends on the following results:

- *Tensor form of the inside-outside algorithm.* Section 6.1 shows that the inside-outside algorithm for L-PCFGs can be written using tensors and tensor products. Theorem 3 gives conditions under which the tensor form calculates inside and outside terms correctly.

- *Observable representations.* Section 7.2 shows that under a singular-value condition, there is an *observable form* for the tensors required by the inside-outside algorithm. By an observable form, we follow the terminology of Hsu et al. (2009) in referring to quantities that can be estimated directly from data where values for latent variables are unobserved. Theorem 6 shows that tensors derived from the observable form satisfy the conditions of Theorem 3.

- *Estimating the model.* Section 8 gives an algorithm for estimating parameters of the observable representation from training data. Theorem 8 gives a sample complexity result, showing that the estimates converge to the true distribution at a rate of $1/\sqrt{M}$ where M is the number of training examples.

The algorithm is strikingly different from the EM algorithm for L-PCFGs, both in its basic form, and in its consistency guarantees. The techniques developed in this paper are quite general, and should be relevant to the development of spectral methods for estimation in other models in NLP, for example alignment models for translation, synchronous PCFGs, and so on. The tensor form of the inside-outside algorithm gives a new view of basic calculations in PCFGs, and may itself lead to new models.

In this paper we derive the basic algorithm, and the theory underlying the algorithm. In a companion paper (Cohen et al., 2013), we describe experiments using the algorithm to learn an L-PCFG for natural language parsing. In these experiments the spectral algorithm gives models that are as accurate as the EM algorithm for learning in L-PCFGs. It is significantly more efficient than the EM algorithm on this problem (9h52m of training time vs. 187h12m), because after an SVD operation it requires a single pass over the data, whereas EM requires around 20-30 passes before converging to a good solution.

2. Related Work

The most common approach for statistical learning of models with latent variables is the expectation-maximization (EM) algorithm (Dempster et al., 1977). Under mild conditions, the EM algorithm is guaranteed to converge to a local maximum of the log-likelihood function. This is, however, a relatively weak guarantee; there are in general no guarantees of consistency for the EM algorithm, and no guarantees of sample complexity, for example within the PAC framework (Valiant, 1984). This has led a number of researchers to consider alternatives to the EM algorithm, which do have PAC-style guarantees.

One focus of this work has been on the problem of learning Gaussian mixture models. In early work, Dasgupta (1999) showed that under separation conditions for the underlying Gaussians, an algorithm with PAC guarantees can be derived. For more recent work in this area, see for example Vempala and Wang (2004), and Moitra and Valiant (2010). These algorithms avoid the issues of local maxima posed by the EM algorithm.

Another focus has been on spectral learning algorithms for hidden Markov models (HMMs) and related models. This work forms the basis for the L-PCFG learning algorithms described in this paper. This line of work started with the work of Hsu et al. (2009), who developed a spectral learning algorithm for HMMs which recovers an HMM's parameters, up to a linear transformation, using singular value decomposition and other simple matrix operations. The algorithm builds on the idea of observable operator models for HMMs due to Jaeger (2000). Following the work of Hsu et al. (2009), spectral learning algorithms have been derived for a number of other models, including finite state transducers (Balle et al., 2011); split-head automaton grammars (Luque et al., 2012); reduced rank HMMs in linear dynamical systems (Siddiqi et al., 2010); kernel-based methods for HMMs (Song et al., 2010); and tree graphical models (Parikh et al., 2011; Song et al., 2011). There are also spectral learning algorithms for learning PCFGs in the unsupervised setting (Bailly et al., 2013).

Foster et al. (2012) describe an alternative algorithm to that of Hsu et al. (2009) for learning of HMMs, which makes use of tensors. Our work also makes use of tensors, and is closely related to the work of Foster et al. (2012); it is also related to the tensor-based approaches for learning of tree graphical models described by Parikh et al. (2011) and Song et al. (2011). In related work, Dhillon et al. (2012) describe a tensor-based method for dependency parsing.

Bailly et al. (2010) describe a learning algorithm for weighted (probabilistic) tree automata that is closely related to our own work. Our approach leverages functions ϕ and ψ that map inside and outside trees respectively to feature vectors (see Section 7.2): for example, $\phi(t)$ might track the context-free rule at the root of the inside tree t, or features

corresponding to larger tree fragments. Cohen et al. (2013) give definitions of ϕ and ψ used in parsing experiments with L-PCFGs. In the special case where ϕ and ψ are identity functions, specifying the entire inside or outside tree, the learning algorithm of Bailly et al. (2010) is the same as our algorithm. However, our work differs from that of Bailly et al. (2010) in several important respects. The generalization to allow arbitrary functions ϕ and ψ is important for the success of the learning algorithm, in both a practical and theoretical sense. The inside-outside algorithm, derived in Figure 5, is not presented by Bailly et al. (2010), and is critical in deriving marginals used in parsing. Perhaps most importantly, the analysis of sample complexity, given in Theorem 8 of this paper, is much tighter than the sample complexity bound given by Bailly et al. (2010). The sample complexity bound in theorem 4 of Bailly et al. (2010) suggests that the number of samples required to obtain $|\hat{p}(t) - p(t)| \leqslant \epsilon$ for some tree t of size N, and for some value ϵ, is *exponential* in N. In contrast, we show that the number of samples required to obtain $\sum_t |\hat{p}(t) - p(t)| \leqslant \epsilon$ where the sum is over *all* trees of size N is polynomial in N. Thus our bound is an improvement in a couple of ways: first, it applies to a sum over all trees of size N, a set of exponential size; second, it is polynomial in N.

Spectral algorithms are inspired by the method of moments, and there are latent-variable learning algorithms that use the method of moments, without necessarily resorting to spectral decompositions. Most relevant to this paper is the work in Cohen and Collins (2014) for estimating L-PCFGs, inspired by the work by Arora et al. (2013).

3. Notation

Given a matrix A or a vector v, we write $A^\top$ or $v^\top$ for the associated transpose. For any integer $n \geqslant 1$, we use $[n]$ to denote the set $\{1, 2, \ldots n\}$.

We use $\mathbb{R}^{m \times 1}$ to denote the space of m-dimensional column vectors, and $\mathbb{R}^{1 \times m}$ to denote the space of m-dimensional row vectors. We use $\mathbb{R}^m$ to denote the space of m-dimensional vectors, where the vector in question can be either a row or column vector. For any row or column vector $y \in \mathbb{R}^m$, we use $\text{diag}(y)$ to refer to the $(m \times m)$ matrix with diagonal elements equal to y_h for $h = 1 \ldots m$, and off-diagonal elements equal to 0. For any statement Γ, we use $[\![\Gamma]\!]$ to refer to the indicator function that is 1 if Γ is true, and 0 if Γ is false. For a random variable X, we use $\mathbf{E}[X]$ to denote its expected value.

We will make use of tensors of rank 3:

Definition 1 *A tensor $C \in \mathbb{R}^{(m \times m \times m)}$ is a set of m^3 parameters $C_{i,j,k}$ for $i, j, k \in [m]$. Given a tensor C, and vectors $y^1 \in \mathbb{R}^m$ and $y^2 \in \mathbb{R}^m$, we define $C(y^1, y^2)$ to be the m-dimensional row vector with components*

$$[C(y^1, y^2)]_i = \sum_{j \in [m], k \in [m]} C_{i,j,k} y^1_j y^2_k.$$

Hence C can be interpreted as a function $C : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^{1 \times m}$ that maps vectors y^1 and y^2 to a row vector $C(y^1, y^2) \in \mathbb{R}^{1 \times m}$.

In addition, we define the tensor $C_{(1,2)} \in \mathbb{R}^{(m \times m \times m)}$ for any tensor $C \in \mathbb{R}^{(m \times m \times m)}$ to be the function $C_{(1,2)} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^{m \times 1}$ defined as

$$[C_{(1,2)}(y^1, y^2)]_k = \sum_{i \in [m], j \in [m]} C_{i,j,k} y^1_i y^2_j.$$

Similarly, for any tensor C we define $C_{(1,3)} : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}^{m \times 1}$ as

$$[C_{(1,3)}(y^1, y^2)]_j = \sum_{i \in [m], k \in [m]} C_{i,j,k} y^1_i y^2_k.$$

Note that $C_{(1,2)}(y^1, y^2)$ and $C_{(1,3)}(y^1, y^2)$ are both **column** *vectors.*

For vectors $x, y, z \in \mathbb{R}^m$, $xy^\top z^\top$ is the tensor $D \in \mathbb{R}^{m \times m \times m}$ where $D_{i,j,k} = x_i y_j z_k$ (this is analogous to the outer product: $[xy^\top]_{i,j} = x_i y_j$).

We use $||\ldots||_F$ to refer to the Frobenius norm for matrices or tensors: for a matrix A, $||A||_F = \sqrt{\sum_{i,j}(A_{i,j})^2}$, for a tensor C, $||C||_F = \sqrt{\sum_{i,j,k}(C_{i,j,k})^2}$. For a matrix A we use $||A||_{2,o}$ to refer to the operator (spectral) norm, $||A||_{2,o} = \max_{x \neq 0} ||Ax||_2 / ||x||_2$.

4. L-PCFGs

In this section we describe latent-variable PCFGs (L-PCFGs), as used for example by Matsuzaki et al. (2005) and Petrov et al. (2006). We first give the basic definitions for L-PCFGs, and then describe the underlying motivation for them.

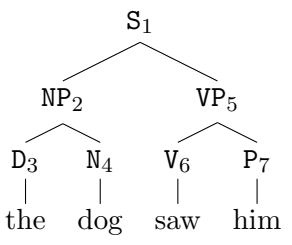
4.1 Basic Definitions

An L-PCFG is an 8-tuple $(\mathcal{N}, \mathcal{I}, \mathcal{P}, m, n, t, q, \pi)$ where:

- $\mathcal{N}$ is the set of non-terminal symbols in the grammar. $\mathcal{I} \subset \mathcal{N}$ is a finite set of *in-terminals*. $\mathcal{P} \subset \mathcal{N}$ is a finite set of *pre-terminals*. We assume that $\mathcal{N} = \mathcal{I} \cup \mathcal{P}$, and $\mathcal{I} \cap \mathcal{P} = \varnothing$. Hence we have partitioned the set of non-terminals into two subsets.
- $[m]$ is the set of possible hidden states.
- $[n]$ is the set of possible words.
- For all $a \in \mathcal{I}$, $b \in \mathcal{N}$, $c \in \mathcal{N}$, $h_1, h_2, h_3 \in [m]$, we have a context-free rule $a(h_1) \rightarrow b(h_2)\ c(h_3)$.
- For all $a \in \mathcal{P}$, $h \in [m]$, $x \in [n]$, we have a context-free rule $a(h) \rightarrow x$.
- For all $a \in \mathcal{I}, b, c \in \mathcal{N}$, and $h_1, h_2, h_3 \in [m]$, we have a parameter $t(a \rightarrow b\ c, h_2, h_3 | h_1, a)$.
- For all $a \in \mathcal{P}$, $x \in [n]$, and $h \in [m]$, we have a parameter $q(a \rightarrow x | h, a)$.
- For all $a \in \mathcal{I}$ and $h \in [m]$, we have a parameter $\pi(a, h)$ which is the probability of non-terminal a paired with hidden variable h being at the root of the tree.

Note that each in-terminal $a \in \mathcal{I}$ is always the left-hand-side of a binary rule $a \rightarrow b\ c$; and each pre-terminal $a \in \mathcal{P}$ is always the left-hand-side of a rule $a \rightarrow x$. Assuming that the non-terminals in the grammar can be partitioned this way is relatively benign, and makes the estimation problem cleaner.

For convenience we define the set of possible "skeletal rules" as $\mathcal{R} = \{a \rightarrow b\ c : a \in \mathcal{I}, b \in \mathcal{N}, c \in \mathcal{N}\}$.



$r_1 = \texttt{S} \rightarrow \texttt{NP VP}$
$r_2 = \texttt{NP} \rightarrow \texttt{D N}$
$r_3 = \texttt{D} \rightarrow \texttt{the}$
$r_4 = \texttt{N} \rightarrow \texttt{dog}$
$r_5 = \texttt{VP} \rightarrow \texttt{V P}$
$r_6 = \texttt{V} \rightarrow \texttt{saw}$
$r_7 = \texttt{P} \rightarrow \texttt{him}$

Figure 1: s-tree, and its sequence of rules. (For convenience we have numbered the nodes in the tree.)

These definitions give a PCFG, with rule probabilities

$$p(a(h_1) \rightarrow b(h_2)\ c(h_3)|a(h_1)) = t(a \rightarrow b\ c, h_2, h_3|h_1, a),$$

and

$$p(a(h) \rightarrow x|a(h)) = q(a \rightarrow x|h, a).$$

Remark 2 *In the previous paper on this work (Cohen et al., 2012), we considered an L-PCFG model where*

$$p(a(h_1) \rightarrow b(h_2)\ c(h_3)|a(h_1)) = p(a \rightarrow b\ c|h_1, a) \times p(h_2|h_1, a \rightarrow b\ c) \times p(h_3|h_1, a \rightarrow b\ c)$$

In this model the random variables h_2 and h_3 are assumed to be conditionally independent given h_1 and $a \rightarrow b\ c$.

In this paper we consider a model where

$$p(a(h_1) \rightarrow b(h_2)\ c(h_3)|a(h_1)) = t(a \rightarrow b\ c, h_2, h_3, |h_1, a). \tag{1}$$

That is, we do **not** *assume that the random variables h_2 and h_3 are independent when conditioning on h_1 and $a \rightarrow b\ c$. This is also the model considered by Matsuzaki et al. (2005) and Petrov et al. (2006).*

Note however that the algorithms in this paper are the same as those in Cohen et al. (2012): we have simply proved that the algorithms give consistent estimators for the model form in Eq. 1.

As in usual PCFGs, the probability of an entire tree is calculated as the product of its rule probabilities. We now give more detail for these calculations.

An L-PCFG defines a distribution over parse trees as follows. A *skeletal tree* (s-tree) is a sequence of rules $r_1 \ldots r_N$ where each r_i is either of the form $a \rightarrow b\ c$ or $a \rightarrow x$. The rule sequence forms a top-down, left-most derivation under a CFG with skeletal rules. See Figure 1 for an example.

A *full tree* consists of an s-tree $r_1 \ldots r_N$, together with values $h_1 \ldots h_N$. Each h_i is the value for the hidden variable for the left-hand-side of rule r_i. Each h_i can take any value in $[m]$.

Define a_i to be the non-terminal on the left-hand-side of rule r_i. For any $i \in [N]$ such that $a_i \in \mathcal{I}$ (i.e., a_i is an in-terminal, and rule r_i is of the form $a \rightarrow b\ c$) define $h_i^{(2)}$ to be the hidden variable value associated with the left child of the rule r_i, and $h_i^{(3)}$ to be the hidden variable value associated with the right child. The probability mass function (PMF) over full trees is then

$$p(r_1 \ldots r_N, h_1 \ldots h_N) = \pi(a_1, h_1) \times \prod_{i:a_i \in \mathcal{I}} t(r_i, h_i^{(2)}, h_i^{(3)} | h_i, a_i) \times \prod_{i:a_i \in \mathcal{P}} q(r_i | h_i, a_i). \quad (2)$$

The PMF over s-trees is $p(r_1 \ldots r_N) = \sum_{h_1 \ldots h_N} p(r_1 \ldots r_N, h_1 \ldots h_N)$.

In the remainder of this paper, we make use of a matrix form of parameters of an L-PCFG, as follows:

- For each $a \rightarrow b\ c \in \mathcal{R}$, we define $T^{a \rightarrow b\ c} \in \mathbb{R}^{m \times m \times m}$ to be the tensor with values

$$T^{a \rightarrow b\ c}_{h_1, h_2, h_3} = t(a \rightarrow b\ c, h_2, h_3 | a, h_1).$$

- For each $a \in \mathcal{P}$, $x \in [n]$, we define $q_{a \rightarrow x} \in \mathbb{R}^{1 \times m}$ to be the row vector with values

$$[q_{a \rightarrow x}]_h = q(a \rightarrow x | h, a)$$

  for $h = 1, 2, \ldots m$.

- For each $a \in \mathcal{I}$, we define the column vector $\pi^a \in \mathbb{R}^{m \times 1}$ where $[\pi^a]_h = \pi(a, h)$.

4.2 Application of L-PCFGs to Natural Language Parsing

L-PCFGs have been shown to be a very useful model for natural language parsing (Matsuzaki et al., 2005; Petrov et al., 2006). In this section we describe the basic approach.

We assume a training set consisting of sentences paired with parse trees, which are similar to the skeletal tree shown in Figure 1. A naive approach to parsing would simply read off a PCFG from the training set: the resulting grammar would have rules such as

```
S → NP VP
NP → D N
VP → V NP
D → the
N → dog
```

and so on. Given a test sentence, the most likely parse under the PCFG can be found using dynamic programming algorithms.

Unfortunately, simple "vanilla" PCFGs induced from treebanks such as the Penn treebank (Marcus et al., 1993) typically give very poor parsing performance. A critical issue is that the set of non-terminals in the resulting grammar (`S, NP, VP, PP, D, N`, etc.) is often quite small. The resulting PCFG therefore makes very strong independence assumptions, failing to capture important statistical properties of parse trees.

In response to this issue, a number of PCFG-based models have been developed which make use of grammars with *refined* non-terminals. For example, in lexicalized models

(Collins, 1997; Charniak, 1997), non-terminals such as S are replaced with non-terminals such as S-sleeps: the non-terminals track some lexical item (in this case *sleeps*), in addition to the syntactic category. For example, the parse tree in Figure 1 would include rules

$$\begin{aligned}
\texttt{S-saw} &\rightarrow \texttt{NP-dog VP-saw} \\
\texttt{NP-dog} &\rightarrow \texttt{D-the N-dog} \\
\texttt{VP-saw} &\rightarrow \texttt{V-saw P-him} \\
\texttt{D-the} &\rightarrow \texttt{the} \\
\texttt{N-dog} &\rightarrow \texttt{dog} \\
\texttt{V-saw} &\rightarrow \texttt{saw} \\
\texttt{P-him} &\rightarrow \texttt{him}
\end{aligned}$$

In this case the number of non-terminals in the grammar increases dramatically, but with appropriate smoothing of parameter estimates lexicalized models perform at much higher accuracy than vanilla PCFGs.

As another example, Johnson describes an approach where non-terminals are refined to also include the non-terminal one level up in the tree; for example rules such as

$$\texttt{S} \rightarrow \texttt{NP VP}$$

are replaced by rules such as

$$\texttt{S-ROOT} \rightarrow \texttt{NP-S VP-S}$$

Here NP-S corresponds to an NP non-terminal whose parent is S; VP-S corresponds to a VP whose parent is S; S-ROOT corresponds to an S which is at the root of the tree. This simple modification leads to significant improvements over a vanilla PCFG.

Klein and Manning (2003) develop this approach further, introducing annotations corresponding to parents and siblings in the tree, together with other information, resulting in a parser whose performance is just below the lexicalized models of Collins (1997) and Charniak (1997).

The approaches of Collins (1997), Charniak (1997), Johnson, and Klein and Manning (2003) all use hand-constructed rules to enrich the set of non-terminals in the PCFG. A natural question is whether refinements to non-terminals can be learned automatically. Matsuzaki et al. (2005) and Petrov et al. (2006) addressed this question through the use of L-PCFGs in conjunction with the EM algorithm. The basic idea is to allow each non-terminal in the grammar to have m possible *latent* values. For example, with $m = 8$ we would replace the non-terminal S with non-terminals S-1, S-2, ..., S-8, and we would replace rules such as

$$\texttt{S} \rightarrow \texttt{NP VP}$$

with rules such as

$$\texttt{S-4} \rightarrow \texttt{NP-3 VP-2}$$

The latent values are of course unobserved in the training data (the treebank), but they can be treated as latent variables in a PCFG-based model, and the parameters of the model can

be estimated using the EM algorithm. More specifically, given training examples consisting of skeletal trees of the form $t^{(i)} = (r_1^{(i)}, r_2^{(i)}, \ldots, r_{N_i}^{(i)})$, for $i = 1 \ldots M$, where N_i is the number of rules in the i'th tree, the log-likelihood of the training data is

$$\sum_{i=1}^{M} \log p(r_1^{(i)} \ldots r_{N_i}^{(i)}) = \sum_{i=1}^{M} \log \sum_{h_1 \ldots h_{N_i}} p(r_1^{(i)} \ldots r_{N_i}^{(i)}, h_1 \ldots h_{N_i})$$

where $p(r_1^{(i)} \ldots r_{N_i}^{(i)}, h_1 \ldots h_{N_i})$ is as defined in Eq. 2. The EM algorithm is guaranteed to converge to a local maximum of the log-likelihood function. Once the parameters of the L-PCFG have been estimated, the algorithm of Goodman (1996) can be used to parse test-data sentences using the L-PCFG: see Section 4.3 for more details. Matsuzaki et al. (2005) and Petrov et al. (2006) show very good performance for these methods.

4.3 Basic Algorithms for L-PCFGs: Variants of the Inside-Outside Algorithm

Variants of the inside-outside algorithm (Baker, 1979) can be used for basic calculations in L-PCFGs, in particular for calculations that involve marginalization over the values for the hidden variables.

To be more specific, given an L-PCFG, two calculations are central:

1. For a given s-tree $r_1 \ldots r_N$, calculate $p(r_1 \ldots r_N) = \sum_{h_1 \ldots h_N} p(r_1 \ldots r_N, h_1 \ldots h_N)$.
2. For a given input sentence $x = x_1 \ldots x_N$, calculate the marginal probabilities

$$\mu(a, i, j) = \sum_{\tau \in \mathcal{T}(x) : (a,i,j) \in \tau} p(\tau)$$

for each non-terminal $a \in \mathcal{N}$, for each (i, j) such that $1 \leqslant i \leqslant j \leqslant N$. Here $\mathcal{T}(x)$ denotes the set of all possible s-trees for the sentence x, and we write $(a, i, j) \in \tau$ if non-terminal a spans words $x_i \ldots x_j$ in the parse tree τ.

The marginal probabilities have a number of uses. Perhaps most importantly, for a given sentence $x = x_1 \ldots x_N$, the parsing algorithm of Goodman (1996) can be used to find

$$\arg \max_{\tau \in \mathcal{T}(x)} \sum_{(a,i,j) \in \tau} \mu(a, i, j).$$

This is the parsing algorithm used by Petrov et al. (2006), for example.[1] In addition, we can calculate the probability for an input sentence, $p(x) = \sum_{\tau \in \mathcal{T}(x)} p(\tau)$, as $p(x) = \sum_{a \in \mathcal{I}} \mu(a, 1, N)$.

Figures 2 and 3 give the conventional (as opposed to tensor) form of inside-outside algorithms for these two problems. In the next section we describe the tensor form. The algorithm in Figure 2 uses dynamic programming to compute

$$p(r_1 \ldots r_N) = \sum_{h_1 \ldots h_N} p(r_1 \ldots r_N, h_1 \ldots h_N)$$

1. Note that finding $\arg\max_{\tau \in \mathcal{T}(x)} p(\tau)$, where $p(\tau) = \sum_{h_1 \ldots h_N} p(\tau, h_1 \ldots h_N)$, is NP hard, hence the use of Goodman's algorithm. Goodman's algorithm minimizes a different loss function when parsing: it minimizes the expected number of spans which are incorrect in the parse tree according to the underlying L-PCFG. We use it while restricting the output tree to be valid under the PCFG grammar extracted from the treebank. There are variants of Goodman's algorithm that do not follow this restriction.

Inputs: s-tree $r_1 \ldots r_N$, L-PCFG $(\mathcal{N}, \mathcal{I}, \mathcal{P}, m, n, t, q, \pi)$, with parameters

- $t(a \rightarrow b\ c, h_2, h_3 | h_1, a)$ for all $a \rightarrow b\ c \in \mathcal{R}$, $h_1, h_2, h_3 \in [m]$.
- $q(a \rightarrow x | h, a)$ for all $a \in \mathcal{P}, x \in [n], h \in [m]$
- $\pi(a, h)$ for all $a \in \mathcal{I}$, $h \in [m]$.

Algorithm: (calculate the b^i terms bottom-up in the tree)

- For all $i \in [N]$ such that $a_i \in \mathcal{P}$, for all $h \in [m]$, $b^i_h = q(r_i | h, a_i)$
- For all $i \in [N]$ such that $a_i \in \mathcal{I}$, for all $h \in [m]$, $b^i_h = \sum_{h_2, h_3} t(r_i, h_2, h_3 | h, a_i) b^\beta_{h_2} b^\gamma_{h_3}$ where β is the index of the left child of node i in the tree, and γ is the index of the right child.

Return: $\sum_h b^1_h \pi(a_1, h) = p(r_1 \ldots r_N)$

Figure 2: The conventional inside-outside algorithm for calculation of $p(r_1 \ldots r_N)$.

for a given parse tree $r_1 \ldots r_N$. The algorithm in Figure 3 uses dynamic programming to compute marginal terms.

5. Roadmap

The next three sections of the paper derive the spectral algorithm for learning of L-PCFGs. The structure of these sections is as follows:

- Section 6 introduces a *tensor form* of the inside-outside algorithms for L-PCFGs. This is analogous to the matrix form for hidden Markov models (see Jaeger 2000, and in particular Lemma 1 of Hsu et al. 2009), and is also related to the use of tensors in spectral algorithms for directed graphical models (Parikh et al., 2011).
- Section 7.2 derives an *observable form* for the tensors required by algorithms of Section 6. The implication of this result is that the required tensors can be estimated directly from training data consisting of skeletal trees.
- Section 8 gives the algorithm for estimation of the tensors from a training sample, and gives a PAC-style generalization bound for the approach.

6. Tensor Form of the Inside-Outside Algorithm

This section first gives a tensor form of the inside-outside algorithms for L-PCFGs, then give an illustrative example.

6.1 The Tensor-Form Algorithms

Recall the two calculations for L-PCFGs introduced in Section 4.3:

Inputs: Sentence $x_1 \dots x_N$, L-PCFG $(\mathcal{N}, \mathcal{I}, \mathcal{P}, m, n, t, q, \pi)$, with parameters

- $t(a \to b\ c, h_2, h_3 | h_1, a)$ for all $a \to b\ c \in \mathcal{R}$, $h_1, h_2, h_3 \in [m]$.
- $q(a \to x | h, a)$ for all $a \in \mathcal{P}, x \in [n], h \in [m]$
- $\pi(a, h)$ for all $a \in \mathcal{I}$, $h \in [m]$.

Data structures:

- Each $\bar{\alpha}^{a,i,j} \in \mathbb{R}^{1 \times m}$ for $a \in \mathcal{N}$, $1 \leqslant i \leqslant j \leqslant N$ is a row vector of inside terms.
- Each $\bar{\beta}^{a,i,j} \in \mathbb{R}^{m \times 1}$ for $a \in \mathcal{N}$, $1 \leqslant i \leqslant j \leqslant N$ is a column vector of outside terms.
- Each $\bar{\mu}(a, i, j) \in \mathbb{R}$ for $a \in \mathcal{N}$, $1 \leqslant i \leqslant j \leqslant N$ is a marginal probability.

Algorithm:

(Inside base case) $\forall a \in \mathcal{P}, i \in [N], h \in [m]$ $\bar{\alpha}_h^{a,i,i} = q(a \to x_i | h, a)$

(Inside recursion) $\forall a \in \mathcal{I}, 1 \leqslant i < j \leqslant N, h \in [m]$

$$\bar{\alpha}_h^{a,i,j} = \sum_{k=i}^{j-1} \sum_{a \to b\ c} \sum_{h_2 \in [m]} \sum_{h_3 \in [m]} t(a \to b\ c, h_2, h_3 | h, a) \times \bar{\alpha}_{h_2}^{b,i,k} \times \bar{\alpha}_{h_3}^{c,k+1,j}$$

(Outside base case) $\forall a \in \mathcal{I}, h \in [m]$ $\bar{\beta}_h^{a,1,n} = \pi(a, h)$

(Outside recursion) $\forall a \in \mathcal{N}, 1 \leqslant i \leqslant j \leqslant N, h \in [m]$

$$\bar{\beta}_h^{a,i,j} = \sum_{k=1}^{i-1} \sum_{b \to c\ a} \sum_{h_2 \in [m]} \sum_{h_3 \in [m]} t(b \to c\ a, h_3, h | h_2, b) \times \bar{\beta}_{h_2}^{b,k,j} \times \bar{\alpha}_{h_3}^{c,k,i-1}$$

$$+ \sum_{k=j+1}^{N} \sum_{b \to a\ c} \sum_{h_2 \in [m]} \sum_{h_3 \in [m]} t(b \to a\ c, h, h_3 | h_2, b) \times \bar{\beta}_{h_2}^{b,i,k} \times \bar{\alpha}_{h_3}^{c,j+1,k}$$

(Marginals) $\forall a \in \mathcal{N}, 1 \leqslant i \leqslant j \leqslant N$,

$$\bar{\mu}(a, i, j) = \bar{\alpha}^{a,i,j} \bar{\beta}^{a,i,j} = \sum_{h \in [m]} \bar{\alpha}_h^{a,i,j} \bar{\beta}_h^{a,i,j}$$

Figure 3: The conventional form of the inside-outside algorithm, for calculation of marginal terms $\bar{\mu}(a, i, j)$.

Inputs: s-tree $r_1 \dots r_N$, L-PCFG $(\mathcal{N}, \mathcal{I}, \mathcal{P}, m, n)$, parameters

- $C^{a \to b\ c} \in \mathbb{R}^{(m \times m \times m)}$ for all $a \to b\ c \in \mathcal{R}$
- $c^{\infty}_{a \to x} \in \mathbb{R}^{(1 \times m)}$ for all $a \in \mathcal{P}, x \in [n]$
- $c^{1}_{a} \in \mathbb{R}^{(m \times 1)}$ for all $a \in \mathcal{I}$.

Algorithm: (calculate the f^i terms bottom-up in the tree)

- For all $i \in [N]$ such that $a_i \in \mathcal{P}$, $f^i = c^{\infty}_{r_i}$
- For all $i \in [N]$ such that $a_i \in \mathcal{I}$, $f^i = C^{r_i}(f^{\beta}, f^{\gamma})$ where β is the index of the left child of node i in the tree, and γ is the index of the right child.

Return: $f^1 c^1_{a_1} = p(r_1 \dots r_N)$

Figure 4: The tensor form for calculation of $p(r_1 \dots r_N)$.

1. For a given s-tree $r_1 \dots r_N$, calculate $p(r_1 \dots r_N)$.
2. For a given input sentence $x = x_1 \dots x_N$, calculate the marginal probabilities

$$\mu(a, i, j) = \sum_{\tau \in \mathcal{T}(x) : (a,i,j) \in \tau} p(\tau)$$

for each non-terminal $a \in \mathcal{N}$, for each (i, j) such that $1 \leqslant i \leqslant j \leqslant N$, where $\mathcal{T}(x)$ denotes the set of all possible s-trees for the sentence x, and we write $(a, i, j) \in \tau$ if non-terminal a spans words $x_i \dots x_j$ in the parse tree τ.

The tensor form of the inside-outside algorithms for these two problems are shown in Figures 4 and 5. Each algorithm takes the following inputs:

1. A tensor $C^{a \to b\ c} \in \mathbb{R}^{(m \times m \times m)}$ for each rule $a \to b\ c$.
2. A vector $c^{\infty}_{a \to x} \in \mathbb{R}^{(1 \times m)}$ for each rule $a \to x$.
3. A vector $c^{1}_{a} \in \mathbb{R}^{(m \times 1)}$ for each $a \in \mathcal{I}$.

The following theorem gives conditions under which the algorithms are correct:

Theorem 3 *Assume that we have an L-PCFG with parameters $q_{a \to x}$, $T^{a \to b\ c}$, π^a, and that there exist matrices $G^a \in \mathbb{R}^{(m \times m)}$ for all $a \in \mathcal{N}$ such that each G^a is invertible, and such that:*

1. *For all rules $a \to b\ c$, $C^{a \to b\ c}(y^1, y^2) = \left(T^{a \to b\ c}(y^1 G^b, y^2 G^c)\right)(G^a)^{-1}$.*
2. *For all rules $a \to x$, $c^{\infty}_{a \to x} = q_{a \to x}(G^a)^{-1}$.*
3. *For all $a \in \mathcal{I}$, $c^{1}_{a} = G^a \pi^a$.*

Then: 1) The algorithm in Figure 4 correctly computes $p(r_1 \dots r_N)$ under the L-PCFG. 2) The algorithm in Figure 5 correctly computes the marginals $\mu(a, i, j)$ under the L-PCFG.

Proof: see Section A.1. The next section (Section 6.2) gives an example that illustrates the basic intuition behind the proof. ■

Remark 4 *It is easily verified (see also the example in Section 6.2), that if the inputs to the tensor-form algorithms are of the following form (equivalently, the matrices G^a for all a are equal to the identity matrix):*

1. *For all rules $a \rightarrow b\ c$, $C^{a \rightarrow b\ c}(y^1, y^2) = T^{a \rightarrow b\ c}(y^1, y^2)$.*
2. *For all rules $a \rightarrow x$, $c^{\infty}_{a \rightarrow x} = q_{a \rightarrow x}$.*
3. *For all $a \in \mathcal{I}$, $c^1_a = \pi^a$.*

then the algorithms in Figures 4 and 5 are identical to the algorithms in Figures 2 and 3 respectively. More precisely, we have the identities

$$b^i_h = f^i_h$$

for the quantities in Figures 2 and 4, and

$$\bar{\alpha}^{a,i,j}_h = \alpha^{a,i,j}_h$$
$$\bar{\beta}^{a,i,j}_h = \beta^{a,i,j}_h$$

for the quantities in Figures 3 and 5.

The theorem shows, however, that it is sufficient[2] to have parameters that are equal to $T^{a \rightarrow b\ c}$, $q_{a \rightarrow x}$ and π^a up to linear transforms defined by the matrices G^a for all non-terminals a. The linear transformations add an extra degree of freedom that is crucial in what follows in this paper: in the next section, on observable representations, we show that it is possible to directly estimate values for $C^{a \rightarrow b\ c}$, $c^{\infty}_{a \rightarrow x}$ and c^1_a that satisfy the conditions of the theorem, but where the matrices G^a are not the identity matrix.

The key step in the proof of the theorem (see Section A.1) is to show that under the assumptions of the theorem we have the identities

$$f^i = b^i (G^a)^{-1}$$

for Figures 2 and 4, and

$$\alpha^{a,i,j} = \bar{\alpha}^{a,i,j} (G^a)^{-1}$$
$$\beta^{a,i,j} = G^a \bar{\beta}^{a,i,j}$$

for Figures 3 and 5. Thus the quantities calculated by the tensor-form algorithms are equivalent to the quantities calculated by the conventional algorithms, up to linear transforms. The linear transforms and their inverses cancel in useful ways: for example in the output from Figure 4 we have

$$\mu(a, i, j) = \alpha^{a,i,j} \beta^{a,i,j} = \bar{\alpha}^{a,i,j} (G^a)^{-1} G^a \bar{\beta}^{a,i,j} = \sum_h \bar{\alpha}^{a,i,j}_h \bar{\beta}^{a,i,j}_h,$$

showing that the marginals calculated by the conventional and tensor-form algorithms are identical.

2. Assuming that the goal is to calculate $p(r_1 \dots r_N)$ for any skeletal tree, or marginal terms $\mu(a, i, j)$.

Inputs: Sentence $x_1 \dots x_N$, L-PCFG $(\mathcal{N}, \mathcal{I}, \mathcal{P}, m, n)$, parameters $C^{a \to b\ c} \in \mathbb{R}^{(m \times m \times m)}$ for all $a \to b\ c \in \mathcal{R}$, $c^{\infty}_{a \to x} \in \mathbb{R}^{(1 \times m)}$ for all $a \in \mathcal{P}, x \in [n]$, $c^{1}_{a} \in \mathbb{R}^{(m \times 1)}$ for all $a \in \mathcal{I}$.
Data structures:

- Each $\alpha^{a,i,j} \in \mathbb{R}^{1 \times m}$ for $a \in \mathcal{N}$, $1 \leqslant i \leqslant j \leqslant N$ is a row vector of inside terms.
- Each $\beta^{a,i,j} \in \mathbb{R}^{m \times 1}$ for $a \in \mathcal{N}$, $1 \leqslant i \leqslant j \leqslant N$ is a column vector of outside terms.
- Each $\mu(a,i,j) \in \mathbb{R}$ for $a \in \mathcal{N}$, $1 \leqslant i \leqslant j \leqslant N$ is a marginal probability.

Algorithm:
(Inside base case) $\forall a \in \mathcal{P}, i \in [N]$, $\alpha^{a,i,i} = c^{\infty}_{a \to x_i}$
(Inside recursion) $\forall a \in \mathcal{I}, 1 \leqslant i < j \leqslant N$,

$$\alpha^{a,i,j} = \sum_{k=i}^{j-1} \sum_{a \to b\ c} C^{a \to b\ c}(\alpha^{b,i,k}, \alpha^{c,k+1,j})$$

(Outside base case) $\forall a \in \mathcal{I}$, $\beta^{a,1,n} = c^{1}_{a}$
(Outside recursion) $\forall a \in \mathcal{N}, 1 \leqslant i \leqslant j \leqslant N$,

$$\beta^{a,i,j} = \sum_{k=1}^{i-1} \sum_{b \to c\ a} C^{b \to c\ a}_{(1,2)}(\beta^{b,k,j}, \alpha^{c,k,i-1}) + \sum_{k=j+1}^{N} \sum_{b \to a\ c} C^{b \to a\ c}_{(1,3)}(\beta^{b,i,k}, \alpha^{c,j+1,k})$$

(Marginals) $\forall a \in \mathcal{N}, 1 \leqslant i \leqslant j \leqslant N$,

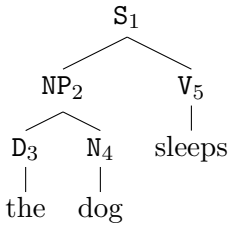
$$\mu(a,i,j) = \alpha^{a,i,j}\beta^{a,i,j} = \sum_{h \in [m]} \alpha^{a,i,j}_{h} \beta^{a,i,j}_{h}$$

Figure 5: The tensor form of the inside-outside algorithm, for calculation of marginal terms $\mu(a,i,j)$.

6.2 An Example

In the remainder of this section we give an example that illustrates how the algorithm in Figure 4 is correct, and gives the basic intuition behind the proof in Section A.1. While we concentrate on the algorithm in Figure 4, the intuition behind the algorithm in Figure 5 is very similar.

Consider the skeletal tree in Figure 6. We will demonstrate how the algorithm in Figure 4, under the assumptions in the theorem, correctly calculates the probability of this tree. In brief, the argument involves the following steps:



$r_1 = \texttt{S} \rightarrow \texttt{NP V}$
$r_2 = \texttt{NP} \rightarrow \texttt{D N}$
$r_3 = \texttt{D} \rightarrow \texttt{the}$
$r_4 = \texttt{N} \rightarrow \texttt{dog}$
$r_5 = \texttt{V} \rightarrow \texttt{sleeps}$

Figure 6: An s-tree, and its sequence of rules. (For convenience we have numbered the nodes in the tree.)

1. We first show that the algorithm in Figure 4, when run on the tree in Figure 6, calculates the probability of the tree as

$$C^{S \rightarrow NP\ V}(C^{NP \rightarrow D\ N}(c^{\infty}_{D \rightarrow the}, c^{\infty}_{N \rightarrow dog}), c^{\infty}_{V \rightarrow sleeps}) c^{1}_{S}.$$

   Note that this expression mirrors the structure of the tree, with $c^{\infty}_{a \rightarrow x}$ terms for the leaves, $C^{a \rightarrow b\ c}$ terms for each rule production $a \rightarrow b\ c$ in the tree, and a c^{1}_{S} term for the root.

2. We then show that under the assumptions in the theorem, the following identity holds:

$$\begin{aligned} & C^{S \rightarrow NP\ V}(C^{NP \rightarrow D\ N}(c^{\infty}_{D \rightarrow the}, c^{\infty}_{N \rightarrow dog}), c^{\infty}_{V \rightarrow sleeps}) c^{1}_{S}. \\ = \quad & T^{S \rightarrow NP\ V}(T^{NP \rightarrow D\ N}(q_{D \rightarrow the}, q_{N \rightarrow dog}), q_{V \rightarrow sleeps}) \pi^{S} \end{aligned} \tag{3}$$

   This follows because the G^a and $(G^a)^{-1}$ terms for the various non-terminals in the tree cancel. Note that the expression in Eq. 3 again follows the structure of the tree, but with $q_{a \rightarrow x}$ terms for the leaves, $T^{a \rightarrow b\ c}$ terms for each rule production $a \rightarrow b\ c$ in the tree, and a π^{S} term for the root.

3. Finally, we show that the expression in Eq. 3 implements the conventional dynamic-programming method for calculation of the tree probability, as described in Eqs. 11–13 below.

We now go over these three points in detail. The algorithm in Figure 4 calculates the following terms (each f^i is an m-dimensional row vector):

$$\begin{aligned} f^3 &= c^{\infty}_{D \rightarrow the} \\ f^4 &= c^{\infty}_{N \rightarrow dog} \\ f^5 &= c^{\infty}_{V \rightarrow sleeps} \\ f^2 &= C^{NP \rightarrow D\ N}(f^3, f^4) \\ f^1 &= C^{S \rightarrow NP\ V}(f^2, f^5) \end{aligned}$$

The final quantity returned by the algorithm is

$$f^1 c_S^1 = \sum_h f_h^1 [c_S^1]_h.$$

Combining the definitions above, it can be seen that

$$f^1 c_S^1 = C^{S \to NP\ V}(C^{NP \to D\ N}(c^\infty_{D \to the}, c^\infty_{N \to dog}), c^\infty_{V \to sleeps}) c_S^1,$$

demonstrating that point 1 above holds.

Next, given the assumptions in the theorem, we show point 2, that is, that

$$\begin{aligned} & C^{S \to NP\ V}(C^{NP \to D\ N}(c^\infty_{D \to the}, c^\infty_{N \to dog}), c^\infty_{V \to sleeps}) c_S^1 \\ = \ & T^{S \to NP\ V}(T^{NP \to D\ N}(q_{D \to the}, q_{N \to dog}), q_{V \to sleeps}) \pi^S. \end{aligned} \tag{4}$$

This follows because the G^a and $(G^a)^{-1}$ terms in the theorem cancel. More specifically, we have

$$\begin{aligned} f^3 &= c^\infty_{D \to the} = q_{D \to the} (G^D)^{-1} && (5) \\ f^4 &= c^\infty_{N \to dog} = q_{N \to dog} (G^N)^{-1} && (6) \\ f^5 &= c^\infty_{V \to sleeps} = q_{V \to sleeps} (G^V)^{-1} && (7) \\ f^2 &= C^{NP \to D\ N}(f^3, f^4) = T^{NP \to D\ N}(q_{D \to the}, q_{D \to dog})(G^{NP})^{-1} && (8) \\ f^1 &= C^{S \to NP\ V}(f^2, f^5) = T^{S \to NP\ V}(T^{NP \to D\ N}(q_{D \to the}, q_{N \to dog}), q_{V \to sleeps})(G^S)^{-1} && (9) \end{aligned}$$

Eqs. 5, 6, 7 follow by the assumptions in the theorem. Eq. 8 follows because by the assumptions in the theorem

$$C^{NP \to D\ N}(f^3, f^4) = T^{NP \to D\ N}(f^3 G^D, f^4 G^N)(G^{NP})^{-1}$$

hence

$$\begin{aligned} C^{NP \to D\ N}(f^3, f^4) &= T^{NP \to D\ N}(q_{D \to the}(G^D)^{-1} G^D, q_{N \to dog}(G^N)^{-1} G^N)(G^{NP})^{-1} \\ &= T^{NP \to D\ N}(q_{D \to the}, q_{N \to dog})(G^{NP})^{-1} \end{aligned}$$

Eq. 9 follows in a similar manner.

It follows by the assumption that $c_S^1 = G^S \pi^S$ that

$$\begin{aligned} & C^{S \to NP\ V}(C^{NP \to D\ N}(c^\infty_{D \to the}, c^\infty_{N \to dog}), c^\infty_{V \to sleeps}) c_S^1 \\ = \ & T^{S \to NP\ V}(T^{NP \to D\ N}(q_{D \to the}, q_{N \to dog}), q_{V \to sleeps})(G^S)^{-1} G^S \pi^S \\ = \ & T^{S \to NP\ V}(T^{NP \to D\ N}(q_{D \to the}, q_{N \to dog}), q_{V \to sleeps}) \pi^S \end{aligned} \tag{10}$$

The final step (point 3) is to show that the expression in Eq. 10 correctly calculates the probability of the example tree. First consider the term $T^{NP \to D\ N}(q_{D \to the}, q_{N \to dog})$—this

is an m-dimensional row vector, call this b^2. By the definition of the tensor $T^{NP \to D\ N}$, we have

$$\begin{aligned} b^2_h &= \left[T^{NP \to D\ N}(q_{D \to the}, q_{N \to dog})\right]_h \\ &= \sum_{h_2, h_3} t(NP \to D\ N, h_2, h_3 | h, NP) \times q(D \to the | h_2, D) \times q(N \to dog | h_3, N) \end{aligned} \quad (11)$$

By a similar calculation, $T^{S \to NP\ V}(T^{NP \to D\ N}(q_{D \to the}, q_{N \to dog}), q_{V \to sleeps})$—call this vector b^1—is

$$b^1_h = \sum_{h_2, h_3} t(S \to NP\ V, h_2, h_3 | h, S) \times b^2_{h_2} \times q(V \to sleeps | h_3, V) \quad (12)$$

Finally, the probability of the full tree is calculated as

$$\sum_h b^1_h \pi^S_h. \quad (13)$$

It can be seen that the expression in Eq. 4 implements the calculations in Eqs. 11, 12 and 13, which are precisely the calculations used in the conventional dynamic programming algorithm for calculation of the probability of the tree.

7. Estimating the Tensor Model

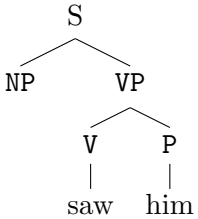
A crucial result is that it is possible to directly estimate parameters $C^{a \to b\ c}$, $c^{\infty}_{a \to x}$ and c^1_a that satisfy the conditions in Theorem 3, from a training sample consisting of s-trees (i.e., trees where hidden variables are unobserved). We first describe random variables underlying the approach, then describe observable representations based on these random variables.

7.1 Random Variables Underlying the Approach

Each s-tree with N rules $r_1 \ldots r_N$ has N nodes. We will use the s-tree in Figure 1 as a running example.

Each node has an associated rule: for example, node 2 in the tree in Figure 1 has the rule `NP` $\to$ `D N`. If the rule at a node is of the form $a \to b\ c$, then there are left and right *inside trees* below the left child and right child of the rule. For example, for node 2 we have a left inside tree rooted at node 3, and a right inside tree rooted at node 4 (in this case the left and right inside trees both contain only a single rule production, of the form $a \to x$; however in the general case they might be arbitrary subtrees).

In addition, each node has an *outside* tree. For node 2, the outside tree is



The outside tree contains everything in the s-tree $r_1 \ldots r_N$, excluding the subtree below node i.

Our random variables are defined as follows. First, we select a random internal node, from a random tree, as follows:

- Sample a full tree $r_1 \ldots r_N, h_1 \ldots h_N$ from the PMF $p(r_1 \ldots r_N, h_1 \ldots h_N)$.
- Choose a node i uniformly at random from $[N]$.

If the rule r_i for the node i is of the form $a \rightarrow b\ c$, we define random variables as follows:

- R_1 is equal to the rule r_i (e.g., $\texttt{NP} \rightarrow \texttt{D}\ \texttt{N}$).
- T_1 is the inside tree rooted at node i. T_2 is the inside tree rooted at the left child of node i, and T_3 is the inside tree rooted at the right child of node i.
- H_1, H_2, H_3 are the hidden variables associated with node i, the left child of node i, and the right child of node i respectively.
- A_1, A_2, A_3 are the labels for node i, the left child of node i, and the right child of node i respectively. (e.g., $A_1 = \texttt{NP}$, $A_2 = \texttt{D}$, $A_3 = \texttt{N}$.)
- O is the outside tree at node i.
- B is equal to 1 if node i is at the root of the tree (i.e., $i = 1$), 0 otherwise.

If the rule r_i for the selected node i is of the form $a \rightarrow x$, we have random variables R_1, T_1, H_1,
A_1, O, B as defined above, but H_2, H_3, T_2, T_3, A_2, and A_3 are not defined.

We assume a function ψ that maps outside trees o to feature vectors $\psi(o) \in \mathbb{R}^{d'}$. For example, the feature vector might track the rule directly above the node in question, the word following the node in question, and so on. We also assume a function ϕ that maps inside trees t to feature vectors $\phi(t) \in \mathbb{R}^d$. As one example, the function ϕ might be an indicator function tracking the rule production at the root of the inside tree. Later we give formal criteria for what makes good definitions of $\psi(o)$ and $\phi(t)$. One requirement is that $d' \geqslant m$ and $d \geqslant m$.

In tandem with these definitions, we assume projection matrices $U^a \in \mathbb{R}^{(d \times m)}$ and $V^a \in \mathbb{R}^{(d' \times m)}$ for all $a \in \mathcal{N}$. We then define additional random variables Y_1, Y_2, Y_3, Z as

$$Y_1 = (U^{a_1})^\top \phi(T_1) \quad Z = (V^{a_1})^\top \psi(O)$$
$$Y_2 = (U^{a_2})^\top \phi(T_2) \quad Y_3 = (U^{a_3})^\top \phi(T_3)$$

where a_i is the value of the random variable A_i. Note that Y_1, Y_2, Y_3, Z are all in $\mathbb{R}^m$.

7.2 Observable Representations

Given the definitions in the previous section, our representation is based on the following matrix, tensor and vector quantities, defined for all $a \in \mathcal{N}$, for all rules of the form $a \rightarrow b\ c$, and for all rules of the form $a \rightarrow x$ respectively:

$$\begin{aligned}
\Sigma^a &= \mathbf{E}[Y_1 Z^\top | A_1 = a], \\
D^{a \rightarrow b\ c} &= \mathbf{E}\left[[\![R_1 = a \rightarrow b\ c]\!] Z Y_2^\top Y_3^\top | A_1 = a\right], \\
d^\infty_{a \rightarrow x} &= \mathbf{E}\left[[\![R_1 = a \rightarrow x]\!] Z^\top | A_1 = a\right].
\end{aligned}$$

Assuming access to functions ϕ and ψ, and projection matrices U^a and V^a, these quantities can be estimated directly from training data consisting of a set of s-trees (see Section 8).

Our observable representation then consists of:

$$C^{a\to b\ c}(y^1, y^2) = D^{a\to b\ c}(y^1, y^2)(\Sigma^a)^{-1}, \quad (14)$$

$$c^{\infty}_{a\to x} = d^{\infty}_{a\to x}(\Sigma^a)^{-1}, \quad (15)$$

$$c^1_a = \mathbf{E}\left[[\![A_1 = a]\!] Y_1 | B = 1\right]. \quad (16)$$

We next introduce conditions under which these quantities satisfy the conditions in Theorem 3.

The following definition will be important:

Definition 5 *For all $a \in \mathcal{N}$, we define the matrices $I^a \in \mathbb{R}^{(d\times m)}$ and $J^a \in \mathbb{R}^{(d'\times m)}$ as*

$$[I^a]_{i,h} = \mathbf{E}[\phi_i(T_1) \mid H_1 = h, A_1 = a],$$

$$[J^a]_{i,h} = \mathbf{E}[\psi_i(O) \mid H_1 = h, A_1 = a].$$

In addition, for any $a \in \mathcal{N}$, we use $\gamma^a \in \mathbb{R}^m$ to denote the vector with $\gamma^a_h = P(H_1 = h | A_1 = a)$.

The correctness of the representation will rely on the following conditions being satisfied (these are parallel to conditions 1 and 2 in Hsu et al. (2009)):

Condition 1 *$\forall a \in \mathcal{N}$, the matrices I^a and J^a are of full rank (i.e., they have rank m). For all $a \in \mathcal{N}$, for all $h \in [m]$, $\gamma^a_h > 0$.*

Condition 2 *$\forall a \in \mathcal{N}$, the matrices $U^a \in \mathbb{R}^{(d\times m)}$ and $V^a \in \mathbb{R}^{(d'\times m)}$ are such that the matrices $G^a = (U^a)^\top I^a$ and $K^a = (V^a)^\top J^a$ are invertible.*

We can now state the following theorem:

Theorem 6 *Assume conditions 1 and 2 are satisfied. For all $a \in \mathcal{N}$, define $G^a = (U^a)^\top I^a$. Then under the definitions in Eqs. 14-16:*

1. *For all rules $a \to b\ c$, $C^{a\to b\ c}(y^1, y^2) = \left(T^{a\to b\ c}(y^1 G^b, y^2 G^c)\right)(G^a)^{-1}$*
2. *For all rules $a \to x$, $c^{\infty}_{a\to x} = q_{a\to x}(G^a)^{-1}$.*
3. *For all $a \in \mathcal{N}$, $c^1_a = G^a \pi^a$*

Proof: The following identities hold (see Section A.2):

$$D^{a\to b\ c}(y^1, y^2) = \left(T^{a\to b\ c}(y^1 G^b, y^2 G^c)\right) \mathrm{diag}(\gamma^a)(K^a)^\top \quad (17)$$

$$d^{\infty}_{a\to x} = q_{a\to x}\mathrm{diag}(\gamma^a)(K^a)^\top \quad (18)$$

$$\Sigma^a = G^a \mathrm{diag}(\gamma^a)(K^a)^\top \quad (19)$$

$$c^1_a = G^a \pi^a \quad (20)$$

Under conditions 1 and 2, Σ^a is invertible, and $(\Sigma^a)^{-1} = ((K^a)^\top)^{-1}(\text{diag}(\gamma^a))^{-1}(G^a)^{-1}$. The identities in the theorem follow immediately. ■

This theorem leads directly to the spectral learning algorithm, which we describe in the next section. We give a sketch of the approach here. Assume that we have a training set consisting of skeletal trees (no latent variables are observed) generated from some underlying L-PCFG. Assume in addition that we have definitions of ϕ, ψ, U^a and V^a such that conditions 1 and 2 are satisfied for the L-PCFG. Then it is straightforward to use the training examples to derive i.i.d. samples from the joint distribution over the random variables $(A_1, R_1, Y_1, Y_2, Y_3, Z, B)$ used in the definitions in Eqs. 14–16. These samples can be used to estimate the quantities in Eqs. 14–16; the estimated quantities $\hat{C}^{a\to b\,c}$, $\hat{c}^{\infty}_{a\to x}$ and $\hat{c}^1_a$ can then be used as inputs to the algorithms in Figures 4 and 5. By standard arguments, the estimates $\hat{C}^{a\to b\,c}$, $\hat{c}^{\infty}_{a\to x}$ and $\hat{c}^1_a$ will converge to the values in Eqs. 14–16.

The following lemma justifies the use of an SVD calculation as one method for finding values for U^a and V^a that satisfy condition 2, assuming that condition 1 holds:

Lemma 7 *Assume that condition 1 holds, and for all $a \in \mathcal{N}$ define*

$$\Omega^a = \mathbf{E}[\phi(T_1)\,(\psi(O))^\top \,|A_1 = a] \tag{21}$$

Then if U^a is a matrix of the m left singular vectors of Ω^a corresponding to non-zero singular values, and V^a is a matrix of the m right singular vectors of Ω^a corresponding to non-zero singular values, then condition 2 is satisfied.

Proof sketch: It can be shown that $\Omega^a = I^a \text{diag}(\gamma^a)(J^a)^\top$. The remainder is similar to the proof of lemma 2 in Hsu et al. (2009). ■

The matrices Ω^a can be estimated directly from a training set consisting of s-trees, assuming that we have access to the functions ϕ and ψ. Similar arguments to those of Hsu et al. (2009) can be used to show that with a sufficient number of samples, the resulting estimates of U^a and V^a satisfy condition 2 with high probability.

8. Deriving Empirical Estimates

Figure 7 shows an algorithm that derives estimates of the quantities in Eqs. 14, 15, and 16. As input, the algorithm takes a sequence of tuples $(r^{(i,1)}, t^{(i,1)}, t^{(i,2)}, t^{(i,3)}, o^{(i)}, b^{(i)})$ for $i \in [M]$.

These tuples can be derived from a training set consisting of s-trees $\tau_1 \ldots \tau_M$ as follows:

• $\forall i \in [M]$, choose a single node j_i uniformly at random from the nodes in τ_i. Define $r^{(i,1)}$ to be the rule at node j_i. $t^{(i,1)}$ is the inside tree rooted at node j_i. If $r^{(i,1)}$ is of the form $a \to b\ c$, then $t^{(i,2)}$ is the inside tree under the left child of node j_i, and $t^{(i,3)}$ is the inside tree under the right child of node j_i. If $r^{(i,1)}$ is of the form $a \to x$, then $t^{(i,2)} = t^{(i,3)} = \texttt{NULL}$. $o^{(i)}$ is the outside tree at node j_i. $b^{(i)}$ is 1 if node j_i is at the root of the tree, 0 otherwise.

Under this process, assuming that the s-trees $\tau_1 \ldots \tau_M$ are i.i.d. draws from the distribution $p(\tau)$ over s-trees under an L-PCFG, the tuples $(r^{(i,1)}, t^{(i,1)}, t^{(i,2)}, t^{(i,3)}, o^{(i)}, b^{(i)})$ are i.i.d. draws from the joint distribution over the random variables R_1, T_1, T_2, T_3, O, B defined in the previous section.

The algorithm first computes estimates of the projection matrices U^a and V^a: following Lemma 7, this is done by first deriving estimates of Ω^a, and then taking SVDs of each Ω^a.

The matrices are then used to project inside and outside trees $t^{(i,1)}, t^{(i,2)}, t^{(i,3)}, o^{(i)}$ down to m-dimensional vectors $y^{(i,1)}, y^{(i,2)}, y^{(i,3)}, z^{(i)}$; these vectors are used to derive the estimates of $C^{a \to b\ c}$, $c^{\infty}_{a \to x}$, and c^{1}_{a}. For example, the quantities

$$\begin{aligned} D^{a \to b\ c} &= \mathbf{E}\left[[\![R_1 = a \to b\ c]\!] Z Y_2^\top Y_3^\top | A_1 = a\right] \\ d^{\infty}_{a \to x} &= \mathbf{E}\left[[\![R_1 = a \to x]\!] Z^\top | A_1 = a\right] \end{aligned}$$

can be estimated as

$$\hat{D}^{a \to b\ c} = \delta_a \times \sum_{i=1}^{M} [\![r^{(i,1)} = a \to b\ c]\!] z^{(i)} (y^{(i,2)})^\top (y^{(i,3)})^\top$$

$$\hat{d}^{\infty}_{a \to x} = \delta_a \times \sum_{i=1}^{M} [\![r^{(i,1)} = a \to x]\!] (z^{(i)})^\top$$

where $\delta_a = 1 / \sum_{i=1}^{M} [\![a_i = a]\!]$, and we can then set

$$\hat{C}^{a \to b\ c}(y^1, y^2) = \hat{D}^{a \to b\ c}(y^1, y^2)(\hat{\Sigma}^a)^{-1}$$

$$\hat{c}^{\infty}_{a \to x} = \hat{d}^{\infty}_{a \to x}(\hat{\Sigma}^a)^{-1}.$$

We now state a PAC-style theorem for the learning algorithm. First, we give the following assumptions and definitions:

- We have an L-PCFG $(\mathcal{N}, \mathcal{I}, \mathcal{P}, m, n, t, q, \pi)$. The samples used in Figures 7 and 8 are i.i.d. samples from the L-PCFG (for simplicity of analysis we assume that the two algorithms use independent sets of M samples each: see above for how to draw i.i.d. samples from the L-PCFG).
- We have functions $\phi(t) \in \mathbb{R}^d$ and $\psi(o) \in \mathbb{R}^{d'}$ that map inside and outside trees respectively to feature vectors. We will assume without loss of generality that for all inside trees $||\phi(t)||_2 \leqslant 1$, and for all outside trees $||\psi(o)||_2 \leqslant 1$.
- See Section 7.2 for a definition of the random variables

  $$(R_1, T_1, T_2, T_3, A_1, A_2, A_3, H_1, H_2, H_3, O, B),$$

  and the joint distribution over them.
- For all $a \in \mathcal{N}$ define

  $$\Omega^a = \mathbf{E}[\phi(T_1)(\psi(O))^\top | A_1 = a]$$

  and define $I^a \in \mathbb{R}^{d \times m}$ to be the matrix with entries

  $$[I^a]_{i,h} = \mathbf{E}[\phi_i(T_1) | A_1 = a, H_1 = h]$$

- Define

  $$\sigma = \min_a \sigma_m(\Omega^a)$$

  and

  $$\xi = \min_a \sigma_m(I^a)$$

  where $\sigma_m(A)$ is the m'th largest singular value of the matrix A.

- Define
$$\gamma = \min_{a,b,c \in \mathcal{N}, h_1, h_2, h_3 \in [m]} t(a \rightarrow b\ c, h_2, h_3 | a, h_1)$$
- Define $\mathcal{T}(a, N)$ to be the set of of all skeletal trees with N binary rules (hence $2N+1$ rules in total), with non-terminal a at the root of the tree.

The following theorem gives a bound on the sample complexity of the algorithm:

Theorem 8 *There exist constants C_1, C_2, C_3, C_4, C_5 such that the following holds. Pick any $\epsilon > 0$, any value for δ such that $0 < \delta < 1$, and any integer N such that $N \geqslant 1$. Define $L = \log \frac{2|\mathcal{N}|+1}{\delta}$. Assume that the parameters $\hat{C}^{a \rightarrow b\ c}$, $\hat{c}^{\infty}_{a \rightarrow x}$ and $\hat{c}^{1}_{a}$ are output from the algorithm in Figure 7, with values for N_a, M_a and R such that*

$$\forall a \in \mathcal{I}, N_a \geqslant \frac{C_1 L N^2 m^2}{\gamma^2 \epsilon^2 \xi^4 \sigma^4} \quad \forall a \in \mathcal{P}, N_a \geqslant \frac{C_2 L N^2 m^2 n}{\epsilon^2 \sigma^4}$$

$$\forall a \in \mathcal{I}, M_a \geqslant \frac{C_3 L N^2 m^2}{\gamma^2 \epsilon^2 \xi^4 \sigma^2} \quad \forall a \in \mathcal{P}, M_a \geqslant \frac{C_4 L N^2 m^2}{\epsilon^2 \sigma^2}$$

$$R \geqslant \frac{C_5 L N^2 m^3}{\epsilon^2 \sigma^2}$$

It follows that with probability at least $1 - \delta$, for all $a \in \mathcal{N}$,

$$\sum_{t \in \mathcal{T}(a,N)} |\hat{p}(t) - p(t)| \leqslant \epsilon$$

, where $\hat{p}(t)$ is the output from the algorithm in Figure 4 with parameters $\hat{C}^{a \rightarrow b\ c}$, $\hat{c}^{\infty}_{a \rightarrow x}$ and $\hat{c}^{1}_{a}$, and $p(t)$ is the probability of the skeletal tree under the L-PCFG.

See Appendix B for a proof.

The method described of selecting a single tuple $(r^{(i,1)}, t^{(i,1)}, t^{(i,2)}, t^{(i,3)}, o^{(i)}, b^{(i)})$ for each s-tree ensures that the samples are i.i.d., and simplifies the analysis underlying Theorem 8. In practice, an implementation should use all nodes in all trees in training data; by Rao-Blackwellization we know such an algorithm would be better than the one presented, but the analysis of how much better would be challenging (Bickel and Doksum, 2006; section 3.4.2). It would almost certainly lead to a faster rate of convergence of $\hat{p}$ to p.

9. Discussion

There are several applications of the method. The most obvious is parsing with L-PCFGs (Cohen et al., 2013).[3] The approach should be applicable in other cases where EM has traditionally been used, for example in semi-supervised learning. Latent-variable HMMs for sequence labeling can be derived as special case of our approach, by converting tagged sequences to right-branching skeletal trees (Stratos et al., 2013).

3. Parameters can be estimated using the algorithm in Figure 7; for a test sentence $x_1 \ldots x_N$ we can first use the algorithm in Figure 5 to calculate marginals $\mu(a, i, j)$, then use the algorithm of Goodman (1996) to find $\arg\max_{\tau \in \mathcal{T}(x)} \sum_{(a,i,j) \in \tau} \mu(a, i, j)$.

Inputs: Training examples $(r^{(i,1)}, t^{(i,1)}, t^{(i,2)}, t^{(i,3)}, o^{(i)}, b^{(i)})$ for $i \in \{1 \dots M\}$, where $r^{(i,1)}$ is a context free rule; $t^{(i,1)}$, $t^{(i,2)}$ and $t^{(i,3)}$ are inside trees; $o^{(i)}$ is an outside tree; and $b^{(i)} = 1$ if the rule is at the root of tree, 0 otherwise. A function ϕ that maps inside trees t to feature-vectors $\phi(t) \in \mathbb{R}^d$. A function ψ that maps outside trees o to feature-vectors $\psi(o) \in \mathbb{R}^{d'}$.

Definitions: For each $a \in \mathcal{N}$, define $N_a = \sum_{i=1}^{M} [\![a_i = a]\!]$. Define $R = \sum_{i=1}^{M} [\![b^{(i)} = 1]\!]$. (These definitions will be used in Theorem 8.)

Algorithm:
Define a_i to be the non-terminal on the left-hand side of rule $r^{(i,1)}$. If $r^{(i,1)}$ is of the form $a \to b\ c$, define b_i to be the non-terminal for the left-child of $r^{(i,1)}$, and c_i to be the non-terminal for the right-child.
(Step 0: Singular Value Decompositions)

- Use the algorithm in Figure 8 to calculate matrices $\hat{U}^a \in \mathbb{R}^{(d \times m)}$, $\hat{V}^a \in \mathbb{R}^{(d' \times m)}$ and $\hat{\Sigma}^a \in \mathbb{R}^{(m \times m)}$ for each $a \in \mathcal{N}$.

(Step 1: Projection)

- For all $i \in [M]$, compute $y^{(i,1)} = (\hat{U}^{a_i})^\top \phi(t^{(i,1)})$.
- For all $i \in [M]$ such that $r^{(i,1)}$ is of the form $a \to b\ c$, compute $y^{(i,2)} = (\hat{U}^{b_i})^\top \phi(t^{(i,2)})$ and $y^{(i,3)} = (\hat{U}^{c_i})^\top \phi(t^{(i,3)})$.
- For all $i \in [M]$, compute $z^{(i)} = (\hat{V}^{a_i})^\top \psi(o^{(i)})$.

(Step 2: Calculate Correlations)

- For each $a \in \mathcal{N}$, define $\delta_a = 1 / \sum_{i=1}^{M} [\![a_i = a]\!]$.
- For each rule $a \to b\ c$, compute

$$\hat{D}^{a \to b\ c} = \delta_a \times \sum_{i=1}^{M} [\![r^{(i,1)} = a \to b\ c]\!] z^{(i)} (y^{(i,2)})^\top (y^{(i,3)})^\top.$$

- For each rule $a \to x$, compute $\hat{d}^{\infty}_{a \to x} = \delta_a \times \sum_{i=1}^{M} [\![r^{(i,1)} = a \to x]\!] (z^{(i)})^\top$.

(Step 3: Compute Final Parameters)

- For all $a \to b\ c$, $\hat{C}^{a \to b\ c}(y^1, y^2) = \hat{D}^{a \to b\ c}(y^1, y^2)(\hat{\Sigma}^a)^{-1}$.
- For all $a \to x$, $\hat{c}^{\infty}_{a \to x} = \hat{d}^{\infty}_{a \to x}(\hat{\Sigma}^a)^{-1}$.
- For all $a \in \mathcal{I}$, $\hat{c}^1_a = \frac{\sum_{i=1}^{M} [\![a_i = a \text{ and } b^{(i)} = 1]\!] y^{(i,1)}}{\sum_{i=1}^{M} [\![b^{(i)} = 1]\!]}$.

Figure 7: The spectral learning algorithm.

Inputs: Identical to algorithm in Figure 7.
Definition: For each $a \in \mathcal{N}$, define $M_a = \sum_{i=1}^{M} [\![a_i = a]\!]$ (this definition will be used in Theorem 8).
Algorithm:
• For each $a \in \mathcal{N}$, compute $\hat{\Omega}^a \in \mathbb{R}^{(d \times d')}$ as

$$\hat{\Omega}^a = \frac{\sum_{i=1}^{M} [\![a_i = a]\!] \phi(t^{(i,1)})(\psi(o^{(i)}))^\top}{\sum_{i=1}^{M} [\![a_i = a]\!]}$$

and calculate a singular value decomposition of $\hat{\Omega}^a$.
• For each $a \in \mathcal{N}$, define $\hat{U}^a \in \mathbb{R}^{m \times d}$ to be a matrix of the left singular vectors of $\hat{\Omega}^a$ corresponding to the m largest singular values. Define $\hat{V}^a \in \mathbb{R}^{m \times d'}$ to be a matrix of the right singular vectors of $\hat{\Omega}^a$ corresponding to the m largest singular values. Define $\hat{\Sigma}^a = (\hat{U}^a)^\top \hat{\Omega}^a \hat{V}^a$.

Figure 8: Singular value decompositions.

In terms of efficiency, the first step of the algorithm in Figure 7 requires an SVD calculation: modern methods for calculating SVDs are very efficient (e.g., see Dhillon et al., 2011 and Tropp et al., 2009). The remaining steps of the algorithm require manipulation of tensors or vectors, and require $O(Mm^3)$ time.

The sample complexity of the method depends on the minimum singular values of Ω^a; these singular values are a measure of how well correlated ψ and ϕ are with the unobserved hidden variable H_1. Experimental work is required to find a good choice of values for ψ and ϕ for parsing.

For simplicity we have considered the case where each non-terminal has the same number, m, of possible hidden values. It is simple to generalize the algorithms to the case where the number of hidden values varies depending on the non-terminal; this is important in applications such as parsing.

Acknowledgements

The authors gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of DARPA, AFRL, or the US government. Shay Cohen was supported by the National Science Foundation under Grant #1136996 to the Computing Research Association for the CIFellows Project. Dean Foster was supported by National Science Foundation grant 1106743. This work also used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575.

Appendix A. Proofs of Theorems 1 and 2

This section gives proofs of Theorems 3 and 6.

A.1 Proof of Theorem 3

The key idea behind the proof of Theorem 3 is to show that the algorithms in Figures 4 and 5 compute the same quantities as the conventional version of the inside outside algorithms, as shown in Figures 2 and 3.

First, the following lemma leads directly to the correctness of the algorithm in Figure 4:

Lemma 9 *Assume that conditions 1-3 of Theorem 3 are satisfied, and that the input to the algorithm in Figure 4 is an s-tree $r_1 \dots r_N$. Define a_i for $i \in [N]$ to be the non-terminal on the left-hand-side of rule r_i. For all $i \in [N]$, define the row vector $b^i \in \mathbb{R}^{(1\times m)}$ to be the vector computed by the conventional inside-outside algorithm, as shown in Figure 2, on the s-tree $r_1 \dots r_N$. Define $f^i \in \mathbb{R}^{(1\times m)}$ to be the vector computed by the tensor-based inside-outside algorithm, as shown in Figure 4, on the s-tree $r_1 \dots r_N$.*

Then for all $i \in [N]$, $f^i = b^i (G^{(a_i)})^{-1}$. It follows immediately that

$$f^1 c^1_{a_1} = b^1 (G^{(a_1)})^{-1} G^{a_1} \pi_{a_1} = b^1 \pi_{a_1} = \sum_h b^1_h \pi(a, h).$$

Hence the output from the algorithms in Figures 2 and 4 is the same, and it follows that the tensor-based algorithm in Figure 4 is correct.

This lemma shows a direct link between the vectors f^i calculated in the algorithm, and the terms b^i_h, which are terms calculated by the conventional inside algorithm: each f^i is a linear transformation (through G^{a_i}) of the corresponding vector b^i.

Proof: The proof is by induction.

First consider the base case. For any leaf—i.e., for any i such that $a_i \in \mathcal{P}$—we have $b^i_h = q(r_i|h, a_i)$, and it is easily verified that $f^i = b^i (G^{(a_i)})^{-1}$.

The inductive case is as follows. For all $i \in [N]$ such that $a_i \in \mathcal{I}$, by the definition in the algorithm,

$$\begin{aligned} f^i &= C^{r_i}(f^\beta, f^\gamma) \\ &= \left(T^{r_i}(f^\beta G^{a_\beta}, f^\gamma G^{a_\gamma})\right)(G^{a_i})^{-1} \end{aligned}$$

Assuming by induction that $f^\beta = b^\beta (G^{(a_\beta)})^{-1}$ and $f^\gamma = b^\gamma (G^{(a_\gamma)})^{-1}$, this simplifies to

$$f^i = \left(T^{r_i}(b^\beta, b^\gamma)\right)(G^{a_i})^{-1}. \tag{22}$$

By the definition of the tensor T^{r_i},

$$\left[T^{r_i}(b^\beta, b^\gamma)\right]_h = \sum_{h_2 \in [m], h_3 \in [m]} t(r_i, h_2, h_3 | a_i, h) b^\beta_{h_2} b^\gamma_{h_3}$$

But by definition (see the algorithm in Figure 2),

$$b^i_h = \sum_{h_2 \in [m], h_3 \in [m]} t(r_i, h_2, h_3 | a_i, h) b^\beta_{h_2} b^\gamma_{h_3},$$

hence $b^i = T^{r_i}(b^\beta, b^\gamma)$ and the inductive case follows immediately from Eq. 22. ■

Next, we give a similar lemma, which implies the correctness of the algorithm in Figure 5:

Lemma 10 *Assume that conditions 1-3 of Theorem 3 are satisfied, and that the input to the algorithm in Figure 5 is a sentence $x_1 \dots x_N$. For any $a \in \mathcal{N}$, for any $1 \leqslant i \leqslant j \leqslant N$, define $\bar{\alpha}^{a,i,j} \in \mathbb{R}^{(1\times m)}$, $\bar{\beta}^{a,i,j} \in \mathbb{R}^{(m\times 1)}$ and $\bar{\mu}(a,i,j) \in \mathbb{R}$ to be the quantities computed by the conventional inside-outside algorithm in Figure 3 on the input $x_1 \dots x_N$. Define $\alpha^{a,i,j} \in \mathbb{R}^{(1\times m)}$, $\beta^{a,i,j} \in \mathbb{R}^{(m\times 1)}$ and $\mu(a,i,j) \in \mathbb{R}$ to be the quantities computed by the algorithm in Figure 3.*

Then for all $i \in [N]$, $\alpha^{a,i,j} = \bar{\alpha}^{a,i,j}(G^a)^{-1}$ and $\beta^{a,i,j} = G^a \bar{\beta}^{a,i,j}$. It follows that for all (a,i,j),

$$\mu(a,i,j) = \alpha^{a,i,j}\beta^{a,i,j} = \bar{\alpha}^{a,i,j}(G^a)^{-1}G^a\bar{\beta}^{a,i,j} = \bar{\alpha}^{a,i,j}\bar{\beta}^{a,i,j} = \bar{\mu}(a,i,j).$$

Hence the outputs from the algorithms in Figures 3 and 5 are the same, and it follows that the tensor-based algorithm in Figure 5 is correct.

Thus the vectors $\alpha^{a,i,j}$ and $\beta^{a,i,j}$ are linearly related to the vectors $\bar{\alpha}^{a,i,j}$ and $\bar{\beta}^{a,i,j}$, which are the inside and outside terms calculated by the conventional form of the inside-outside algorithm.

Proof: The proof is by induction, and is similar to the proof of Lemma 9.

First, we prove that the inside terms satisfy the relation $\alpha^{a,i,j} = \bar{\alpha}^{a,i,j}(G^a)^{-1}$.

The base case of the induction is as follows. By definition, for any $a \in \mathcal{P}, i \in [N], h \in [m]$, we have $\bar{\alpha}_h^{a,i,i} = q(a \to x_i | h, a)$. We also have for any $a \in \mathcal{P}, i \in [N]$, $\alpha^{a,i,i} = c^\infty_{a\to x_i} = q_{a\to x_i}(G^a)^{-1}$. It follows directly that $\alpha^{a,i,i} = \bar{\alpha}^{a,i,i}(G^a)^{-1}$ for any $a \in \mathcal{P}, i \in [N]$.

The inductive case is as follows. By definition, we have $\forall a \in \mathcal{I}, 1 \leqslant i < j \leqslant N, h \in [m]$

$$\bar{\alpha}_h^{a,i,j} = \sum_{k=i}^{j-1} \sum_{b,c} \sum_{h_2 \in [m]} \sum_{h_3 \in [m]} t(a \to b\ c, h_2, h_3 | h, a) \times \bar{\alpha}_{h_2}^{b,i,k} \times \bar{\alpha}_{h_3}^{c,k+1,j}.$$

We also have $\forall a \in \mathcal{I}, 1 \leqslant i < j \leqslant N$,

$$\begin{aligned}
\alpha^{a,i,j} &= \sum_{k=i}^{j-1} \sum_{b,c} C^{a\to b\ c}(\alpha^{b,i,k}, \alpha^{c,k+1,j}) && (23)\\
&= \sum_{k=i}^{j-1} \sum_{b,c} \left(T^{a\to b\ c}(\alpha^{b,i,k}G^b, \alpha^{c,k+1,j}G^c) \right)(G^a)^{-1} && (24)\\
&= \sum_{k=i}^{j-1} \sum_{b,c} \left(T^{a\to b\ c}(\bar{\alpha}^{b,i,k}, \bar{\alpha}^{c,k+1,j}) \right)(G^a)^{-1} && (25)\\
&= \bar{\alpha}^{a,i,j}(G^a)^{-1}. && (26)
\end{aligned}$$

Eq. 23 follows by the definitions in algorithm 5. Eq. 24 follows by the assumption in the theorem that

$$C^{a\to b\ c}(y^1, y^2) = \left(T^{a\to b\ c}(y^1 G^b, y^2 G^c) \right)(G^a)^{-1}$$

Eq. 25 follows because by the inductive hypothesis,

$$\alpha^{b,i,k} = \bar{\alpha}^{b,i,k}(G^b)^{-1}$$

and

$$\alpha^{c,k+1,j} = \bar{\alpha}^{c,k+1,j}(G^c)^{-1}.$$

Eq. 26 follows because

$$\left[T^{a\to b\ c}(\bar{\alpha}^{b,i,k}, \bar{\alpha}^{c,k+1,j})\right]_h = \sum_{h_2,h_3} t(a \to b\ c, h_2, h_3|h, a)\bar{\alpha}^{b,i,k}_{h_2}\bar{\alpha}^{c,k+1,j}_{h_3}$$

hence

$$\sum_{k=i}^{j-1}\sum_{b,c} T^{a\to b\ c}(\bar{\alpha}^{b,i,k}, \bar{\alpha}^{c,k+1,j}) = \bar{\alpha}^{a,i,j}.$$

We now turn the outside terms, proving that $\beta^{a,i,j} = G^a\bar{\beta}^{a,i,j}$. The proof is again by induction.

The base case is as follows. By the definitions in the algorithms, for all $a \in \mathcal{I}$, $\beta^{a,1,n} = c^1_a = G^a\pi^a$, and for all $a \in \mathcal{I}, h \in [m]$, $\bar{\beta}^{a,1,n}_h = \pi(a,h)$. It follows directly that for all $a \in \mathcal{I}$, $\beta^{a,1,n} = G^a\bar{\beta}^{a,1,n}$.

The inductive case is as follows. By the definitions in the algorithms, we have $\forall a \in \mathcal{N}, 1 \leqslant i \leqslant j \leqslant N, h \in [m]$

$$\bar{\beta}^{a,i,j}_h = \gamma^{1,a,i,j}_h + \gamma^{2,a,i,j}_h$$

where

$$\gamma^{1,a,i,j}_h = \sum_{k=1}^{i-1}\sum_{b\to c\ a}\sum_{h_2\in[m]}\sum_{h_3\in[m]} t(b \to c\ a, h_3, h|h_2, b) \times \bar{\beta}^{b,k,j}_{h_2} \times \bar{\alpha}^{c,k,i-1}_{h_3}$$

$$\gamma^{2,a,i,j}_h = \sum_{k=j+1}^{N}\sum_{b\to a\ c}\sum_{h_2\in[m]}\sum_{h_3\in[m]} t(b \to a\ c, h, h_3|h_2, b) \times \bar{\beta}^{b,i,k}_{h_2} \times \bar{\alpha}^{c,j+1,k}_{h_3}$$

and $\forall a \in \mathcal{N}, 1 \leqslant i \leqslant j \leqslant N$,

$$\beta^{a,i,j} = \sum_{k=1}^{i-1}\sum_{b\to c\ a} C^{b\to c\ a}_{(1,2)}(\beta^{b,k,j}, \alpha^{c,k,i-1}) + \sum_{k=j+1}^{N}\sum_{b\to a\ c} C^{b\to a\ c}_{(1,3)}(\beta^{b,i,k}, \alpha^{c,j+1,k}).$$

Critical identities are

$$\sum_{k=1}^{i-1}\sum_{b\to c\ a} C^{b\to c\ a}_{(1,2)}(\beta^{b,k,j}, \alpha^{c,k,i-1}) = G^a\gamma^{1,a,i,j} \quad (27)$$

$$\sum_{k=j+1}^{N}\sum_{b\to a\ c} C^{b\to a\ c}_{(1,3)}(\beta^{b,i,k}, \alpha^{c,j+1,k}) = G^a\gamma^{2,a,i,j} \quad (28)$$

from which $\beta^{a,i,j} = G^a\bar{\beta}^{a,i,j}$ follows immediately.

The identities in Eq. 27 and 28 are proved through straightforward algebraic manipulation, based on the following properties:

- By the inductive hypothesis, $\beta^{b,k,j} = G^b \bar{\beta}^{b,k,j}$ and $\beta^{b,i,k} = G^b \bar{\beta}^{b,i,k}$.
- By correctness of the inside terms, as shown earlier in this proof, it holds that $\alpha^{c,k,i-1} = \bar{\alpha}^{c,k,i-1}(G^c)^{-1}$ and $\alpha^{c,j+1,k} = \bar{\alpha}^{c,j+1,k}(G^c)^{-1}$.
- By the assumptions in the theorem,

$$C^{a \to b\ c}(y^1, y^2) = \left(T^{a \to b\ c}(y^1 G^b, y^2 G^c)\right)(G^a)^{-1}.$$

It follows (see Lemma 11) that

$$\begin{aligned} C^{b \to c\ a}_{(1,2)}(\beta^{b,k,j}, \alpha^{c,k,i-1}) &= G^a \left(T^{b \to c\ a}_{(1,2)}((G^b)^{-1}\beta^{b,k,j}, \alpha^{c,k,i-1} G^c)\right) \\ &= G^a \left(T^{b \to c\ a}_{(1,2)}(\bar{\beta}^{b,k,j}, \bar{\alpha}^{c,k,i-1})\right) \end{aligned}$$

and

$$C^{b \to a\ c}_{(1,3)}(\beta^{b,i,k}, \alpha^{c,j+1,k}) = G^a \left(T^{b \to a\ c}_{(1,3)}(\bar{\beta}^{b,i,k}, \bar{\alpha}^{c,j+1,k})\right)$$

■

Finally, we give the following Lemma, as used above:

Lemma 11 *Assume we have tensors $C \in \mathbb{R}^{m \times m \times m}$ and $T \in \mathbb{R}^{m \times m \times m}$ such that for any y^2, y^3,*

$$C(y^2, y^3) = \left(T(y^2 A, y^3 B)\right) D$$

where A, B, D are matrices in $\mathbb{R}^{m \times m}$. Then for any y^1, y^2,

$$C_{(1,2)}(y^1, y^2) = B\left(T_{(1,2)}(Dy^1, y^2 A)\right) \tag{29}$$

and for any y^1, y^3,

$$C_{(1,3)}(y^1, y^3) = A\left(T_{(1,3)}(Dy^1, y^3 B)\right). \tag{30}$$

Proof: Consider first Eq. 29. We will prove the following statement:

$$\forall y^1, y^2, y^3, \quad y^3 C_{(1,2)}(y^1, y^2) = y^3 B\left(T_{(1,2)}(Dy^1, y^2 A)\right)$$

This statement is equivalent to Eq. 29.

First, for all y^1, y^2, y^3, by the assumption that $C(y^2, y^3) = \left(T(y^2 A, y^3 B)\right) D$,

$$C(y^2, y^3) y^1 = T(y^2 A, y^3 B) D y^1$$

hence

$$\sum_{i,j,k} C_{i,j,k} y^1_i y^2_j y^3_k = \sum_{i,j,k} T_{i,j,k} z^1_i z^2_j z^3_k \tag{31}$$

where $z^1 = Dy^1$, $z^2 = y^2 A$, $z^3 = y^3 B$.

In addition, it is easily verified that

$$y^3 C_{(1,2)}(y^1, y^2) = \sum_{i,j,k} C_{i,j,k} y^1_i y^2_j y^3_k \quad (32)$$

$$y^3 B\left(T_{(1,2)}(Dy^1, y^2 A)\right) = \sum_{i,j,k} T_{i,j,k} z^1_i z^2_j z^3_k \quad (33)$$

where again $z^1 = Dy^1$, $z^2 = y^2 A$, $z^3 = y^3 B$. Combining Eqs. 31, 32, and 33 gives

$$y^3 C_{(1,2)}(y^1, y^2) = y^3 B\left(T_{(1,2)}(Dy^1, y^2 A)\right),$$

thus proving the identity in Eq. 29.

The proof of the identity in Eq. 30 is similar, and is omitted for brevity. ■

A.2 Proof of the Identity in Eq. 17

We now prove the identity in Eq. 17, repeated here:

$$D^{a \to b\ c}(y^1, y^2) = \left(T^{a \to b\ c}(y^1 G^b, y^2 G^c)\right) \mathrm{diag}(\gamma^a)(K^a)^\top.$$

Recall that

$$D^{a \to b\ c} = \mathbf{E}\left[[\![R_1 = a \to b\ c]\!] Z Y_2^\top Y_3^\top | A_1 = a\right],$$

or equivalently

$$D^{a \to b\ c}_{i,j,k} = \mathbf{E}\left[[\![R_1 = a \to b\ c]\!] Z_i Y_{2,j} Y_{3,k} | A_1 = a\right].$$

Using the chain rule, and marginalizing over hidden variables, we have

$$\begin{aligned} D^{a \to b\ c}_{i,j,k} &= \mathbf{E}\left[[\![R_1 = a \to b\ c]\!] Z_i Y_{2,j} Y_{3,k} | A_1 = a\right] \\ &= \sum_{h_1, h_2, h_3 \in [m]} p(a \to b\ c, h_1, h_2, h_3 | a) \mathbf{E}\left[Z_i Y_{2,j} Y_{3,k} | R_1 = a \to b\ c, h_1, h_2, h_3\right]. \end{aligned}$$

By definition, we have

$$p(a \to b\ c, h_1, h_2, h_3 | a) = \gamma^a_{h_1} \times t(a \to b\ c, h_2, h_3 | h_1, a)$$

In addition, under the independence assumptions in the L-PCFG, and using the definitions of K^a and G^a, we have

$$\begin{aligned} & \mathbf{E}\left[Z_i Y_{2,j} Y_{3,k} | R_1 = a \to b\ c, h_1, h_2, h_3\right] \\ = \ & \mathbf{E}\left[Z_i | A_1 = a, H_1 = h_1\right] \times \mathbf{E}\left[Y_{2,j} | A_2 = b, H_2 = h_2\right] \times \mathbf{E}\left[Y_{3,k} | A_3 = c, H_3 = h_3\right] \\ = \ & K^a_{i,h_1} \times G^b_{j,h_2} \times G^c_{k,h_3}. \end{aligned}$$

Putting this all together gives

$$\begin{aligned} D^{a \to b\ c}_{i,j,k} &= \sum_{h_1, h_2, h_3 \in [m]} \gamma^a_{h_1} \times t(a \to b\ c, h_2, h_3 | h_1, a) \times K^a_{i,h_1} \times G^b_{j,h_2} \times G^c_{k,h_3} \\ &= \sum_{h_1 \in [m]} \gamma^a_{h_1} \times K^a_{i,h_1} \times \sum_{h_2, h_3 \in [m]} t(a \to b\ c, h_2, h_3 | h_1, a) \times G^b_{j,h_2} \times G^c_{k,h_3}. \end{aligned}$$

By the definition of tensors,

$$\begin{aligned}
& [D^{a\to b\ c}(y^1, y^2)]_i \\
= & \sum_{j,k} D^{a\to b\ c}_{i,j,k} y^1_j y^2_k \\
= & \sum_{h_1\in[m]} \gamma^a_{h_1} \times K^a_{i,h_1} \times \sum_{h_2,h_3\in[m]} t(a \to b\ c, h_2, h_3|h_1, a) \times \left(\sum_j y^1_j G^b_{j,h_2}\right) \times \left(\sum_k y^2_k G^c_{k,h_3}\right) \\
= & \sum_{h_1\in[m]} \gamma^a_{h_1} \times K^a_{i,h_1} \times \left[T^{a\to b\ c}(y^1 G^b, y^2 G^c)\right]_{h_1}. \qquad (34)
\end{aligned}$$

The last line follows because by the definition of tensors,

$$\left[T^{a\to b\ c}(y^1 G^b, y^2 G^c)\right]_{h_1} = \sum_{h_2,h_3} T^{a\to b\ c}_{h_1,h_2,h_3} \left[y^1 G^b\right]_{h_2} \left[y^2 G^c\right]_{h_3}$$

and we have

$$\begin{aligned}
T^{a\to b\ c}_{h_1,h_2,h_3} &= t(a \to b\ c, h_2, h_3|h_1, a) \\
\left[y^1 G^b\right]_{h_2} &= \sum_j y^1_j G^b_{j,h_2} \\
\left[y^2 G^c\right]_{h_3} &= \sum_k y^2_k G^c_{k,h_3}.
\end{aligned}$$

Finally, the required identity

$$D^{a\to b\ c}(y^1, y^2) = \left(T^{a\to b\ c}(y^1 G^b, y^2 G^c)\right) \mathrm{diag}(\gamma^a)(K^a)^\top$$

follows immediately from Eq. 34. ■

A.3 Proof of the Identity in Eq. 18

We now prove the identity in Eq. 18, repeated below:

$$d^\infty_{a\to x} = q_{a\to x} \mathrm{diag}(\gamma^a)(K^a)^\top.$$

Recall that by definition

$$d^\infty_{a\to x} = \mathbf{E}\left[[\![R_1 = a \to x]\!] Z^\top | A_1 = a\right],$$

or equivalently

$$[d^\infty_{a\to x}]_i = \mathbf{E}\left[[\![R_1 = a \to x]\!] Z_i | A_1 = a\right].$$

Marginalizing over hidden variables, we have

$$\begin{aligned}
[d^\infty_{a\to x}]_i &= \mathbf{E}\left[[\![R_1 = a \to x]\!] Z_i | A_1 = a\right] \\
&= \sum_h p(a \to x, h|a) \mathbf{E}[Z_i | H_1 = h, R_1 = a \to x].
\end{aligned}$$

By definition, we have

$$p(a \to x, h|a) = \gamma_h^a q(a \to x|h, a) = \gamma_h^a \left[q_{a\to x}\right]_h .$$

In addition, by the independence assumptions in the L-PCFG, and the definition of K^a,

$$\mathbf{E}[Z_i|H_1 = h, R_1 = a \to x] = \mathbf{E}[Z_i|H_1 = h, A_1 = a] = K_{i,h}^a.$$

Putting this all together gives

$$\left[d_{a\to x}^\infty\right]_i = \sum_h \gamma_h^a \left[q_{a\to x}\right]_h K_{i,h}^a$$

from which the required identity

$$d_{a\to x}^\infty = q_{a\to x}\text{diag}(\gamma^a)(K^a)^\top$$

follows immediately. ■

A.4 Proof of the Identity in Eq. 19

We now prove the identity in Eq. 19, repeated below:

$$\Sigma^a = G^a\text{diag}(\gamma^a)(K^a)^\top$$

Recall that by definition

$$\Sigma^a = \mathbf{E}[Y_1 Z^\top|A_1 = a]$$

or equivalently

$$\left[\Sigma^a\right]_{i,j} = \mathbf{E}[Y_{1,i} Z_j|A_1 = a]$$

Marginalizing over hidden variables, we have

$$\begin{aligned}\left[\Sigma^a\right]_{i,j} &= \mathbf{E}[Y_{1,i} Z_j|A_1 = a] \\ &= \sum_h p(h|a)\mathbf{E}[Y_{1,i} Z_j|H_1 = h, A_1 = a]\end{aligned}$$

By definition, we have

$$\gamma_h^a = p(h|a)$$

In addition, under the independence assumptions in the L-PCFG, and using the definitions of K^a and G^a, we have

$$\begin{aligned}\mathbf{E}[Y_{1,i} Z_j|H_1 = h, A_1 = a] &= \mathbf{E}[Y_{1,i}|H_1 = h, A_1 = a] \times \mathbf{E}[Z_j|H_1 = h, A_1 = a] \\ &= G_{i,h}^a K_{j,h}^a\end{aligned}$$

Putting all this together gives

$$\left[\Sigma^a\right]_{i,j} = \sum_h \gamma_h^a G_{i,h}^a K_{j,h}^a$$

from which the required identity

$$\Sigma^a = G^a\text{diag}(\gamma^a)(K^a)^\top$$

follows immediately. ■

A.5 Proof of the Identity in Eq. 20

We now prove the identity in Eq. 19, repeated below:

$$c^1_a = G^a \pi^a.$$

Recall that by definition

$$c^1_a = \mathbf{E}\left[[\![A_1 = a]\!] Y_1 | B = 1\right],$$

or equivalently

$$[c^1_a]_i = \mathbf{E}\left[[\![A_1 = a]\!] Y_{1,i} | B = 1\right].$$

Marginalizing over hidden variables, we have

$$\begin{aligned}[c^1_a]_i &= \mathbf{E}\left[[\![A_1 = a]\!] Y_{1,i} | B = 1\right] \\ &= \sum_h P(A_1 = a, H_1 = h | B = 1) \mathbf{E}\left[Y_{1,i} | A_1 = a, H_1 = h, B = 1\right].\end{aligned}$$

By definition we have

$$P(A_1 = a, H_1 = h | B = 1) = \pi(a, h)$$

By the independence assumptions in the PCFG, and the definition of G^a, we have

$$\begin{aligned}\mathbf{E}\left[Y_{1,i} | A_1 = a, H_1 = h, B = 1\right] &= \mathbf{E}\left[Y_{1,i} | A_1 = a, H_1 = h\right] \\ &= G^a_{i,h}.\end{aligned}$$

Putting this together gives

$$[c^1_a]_i = \sum_h \pi(a, h) G^a_{i,h}$$

from which the required identity

$$c^1_a = G^a \pi^a$$

follows. ■

Appendix B. Proof of Theorem 8

In this section we give a proof of Theorem 8. The proof relies on three lemmas:

- In Section B.1 we give a lemma showing that if estimates $\hat{C}^{a \to b\ c}$, $\hat{c}_{a \to x}$ and $\hat{c}^1_a$ are close (up to linear transforms) to the parameters of an L-PCFG, then the distribution defined by the parameters is close (in l_1-norm) to the distribution under the L-PCFG.
- In Section B.2 we give a lemma showing that if the estimates $\hat{\Omega}^a$, $\hat{D}^{a \to b\ c}$, $\hat{d}^{\infty}_{a \to x}$ and $\hat{c}^1_a$ are close to the underlying values being estimated, the estimates $\hat{C}^{a \to b\ c}$, $\hat{c}_{a \to x}$ and $\hat{c}^1_a$ are close (up to linear transforms) to the parameters of the underlying L-PCFG.
- In Section B.3 we give a lemma relating the number of samples in the estimation algorithm to the errors in estimating $\hat{\Omega}^a$, $\hat{D}^{a \to b\ c}$, $\hat{d}^{\infty}_{a \to x}$ and $\hat{c}^1_a$.

The proof of the theorem is then given in Section B.4.

B.1 A Bound on How Errors Propagate

In this section we show that if estimated tensors and vectors $\hat{C}^{a \to b\ c}$, $\hat{c}^{\infty}_{a \to x}$ and $\hat{c}^{1}_{a}$ are sufficiently close to the underlying parameters $T^{a \to b\ c}$, $q^{\infty}_{a \to x}$, and π^a of an L-PCFG, then the distribution under the estimated parameters will be close to the distribution under the L-PCFG. Section B.1.1 gives assumptions and definitions; Lemma 12 then gives the main lemma; the remainder of the section gives proofs.

B.1.1 Assumptions and Definitions

We make the following assumptions:

- Assume we have an L-PCFG with parameters $T^{a \to b\ c} \in \mathbb{R}^{m \times m \times m}$, $q_{a \to x} \in \mathbb{R}^m$, $\pi^a \in \mathbb{R}^m$. Assume in addition that we have an invertible matrix $G^a \in \mathbb{R}^{m \times m}$ for each $a \in \mathcal{N}$. For convenience define $H^a = (G^a)^{-1}$ for all $a \in \mathcal{N}$.
- We assume that we have parameters $\hat{C}^{a \to b\ c} \in \mathbb{R}^{m \times m \times m}$, $\hat{c}^{\infty}_{a \to x} \in \mathbb{R}^{1 \times m}$ and $\hat{c}^{1}_{a} \in \mathbb{R}^{m \times 1}$ that satisfy the following conditions:
  - There exists some constant $\Delta > 0$ such that for all rules $a \to b\ c$, for all $y^1, y^2 \in \mathbb{R}^m$,
  $$||\hat{C}^{a \to b\ c}(y^1 H^b, y^2 H^c) G^a - T^{a \to b\ c}(y^1, y^2)||_\infty \leqslant \Delta ||y^1||_2 ||y^2||_2.$$
  - There exists some constant $\delta > 0$ such that for all $a \in \mathcal{P}$, for all $h \in [m]$,
  $$\sum_x |[\hat{c}^{\infty}_{a \to x} G^a]_h - [q^{\infty}_{a \to x}]_h| \leqslant \delta.$$
  - There exists some constant $\kappa > 0$ such that for all a,
  $$||(G^a)^{-1} \hat{c}^{1}_{a} - \pi^a||_1 \leqslant \kappa.$$

  We give the following definitions:

- For any skeletal tree $t = r_1 \dots r_N$, define $b^i(t)$ to be the quantities computed by the algorithm in Figure 4 with t together with the parameters $T^{a \to b\ c}$, $q^{\infty}_{a \to x}$, π^a as input. Define $\hat{f}^i(t)$ to be the quantities computed by the algorithm in Figure 4 with t together with the parameters $\hat{C}^{a \to b\ c}$, $\hat{c}^{\infty}_{a \to x}$, $\hat{c}^{1}_{a}$ as input. Define
  $$\xi(t) = b^1(t),$$
  and
  $$\hat{\xi}(t) = f^1(t) G^{a_1}.$$
  where as before a_1 is the non-terminal on the left-hand-side of rule r_1. Define $\hat{p}(t)$ to be the value returned by the algorithm in Figure 4 with t together with the parameters $\hat{C}^{a \to b\ c}$, $\hat{c}^{\infty}_{a \to x}$, $\hat{c}^{1}_{a}$ as input. Define $p(t)$ to be the value returned by the algorithm in Figure 4 with t together with the parameters $T^{a \to b\ c}$, $q^{\infty}_{a \to x}$, π^a as input.
- Define $\mathcal{T}(a, N)$ to be the set of of all skeletal trees with N binary rules (hence $2N + 1$ rules in total), with non-terminal a at the root of the tree.

- Define

$$\begin{aligned} Z(a,h,N) &= \sum_{t \in \mathcal{T}(a,N)} [\xi(t)]_h, \\ D(a,h,N) &= \sum_{t \in \mathcal{T}(a,N)} |[\hat{\xi}(t)]_h - [\xi(t)]_h|, \\ F(a,h,N) &= \frac{D(a,h,N)}{Z(a,h,N)}. \end{aligned}$$

- Define

$$\gamma = \min_{a,b,c \in \mathcal{N}, h_1,h_2,h_3 \in [m]} t(a \rightarrow b\ c, h_2, h_3 | a, h_1).$$

- For any $a \rightarrow b\ c$ define the tensor

$$\hat{T}^{a \rightarrow b\ c}(y^1, y^2) = \hat{C}^{a \rightarrow b\ c}(y^1 H^b, y^2 H^c) G^a.$$

B.1.2 The Main Lemma

Lemma 12 *Given the assumptions in Section B.1.1, for any a, N,*

$$\sum_{t \in \mathcal{T}(a,N)} |\hat{p}(t) - p(t)| \leqslant m \left((1+\kappa) \left(1 + \frac{\Delta}{\gamma} \right)^{N-1} (1+\delta)^N - 1 \right). \tag{35}$$

Proof: By definition we have

$$\begin{aligned} \sum_{t \in \mathcal{T}(a,N)} |\hat{p}(t) - p(t)| &= \sum_{t \in \mathcal{T}(a,N)} \left| \sum_h [\hat{\xi}(t)]_h [(G^a)^{-1} \hat{c}^1_a]_h - \sum_h [\xi(t)]_h \pi^a_h \right| \\ &= \sum_{t \in \mathcal{T}(a,N)} \left| \hat{\xi}(t) \cdot [(G^a)^{-1} \hat{c}^1_a] - \xi(t) \cdot \pi^a \right|. \end{aligned}$$

Define $e = [(G^a)^{-1} \hat{c}^1_a] - \pi^a$. Then by the triangle inequality,

$$\left| \hat{\xi}(t) \cdot [(G^a)^{-1} \hat{c}^1_a] - \xi(t) \cdot \pi^a \right| \leqslant |\hat{\xi}(t) \cdot \pi^a - \xi(t) \cdot \pi^a| + |\hat{\xi}(t) \cdot e - \xi(t) \cdot e| + |\xi(t) \cdot e|$$

We bound each of the three terms as follows:

$$|\hat{\xi}(t) \cdot \pi^a - \xi(t) \cdot \pi^a| \leqslant ||\hat{\xi}(t) - \xi(t)||_\infty ||\pi^a||_1 \leqslant ||\hat{\xi}(t) - \xi(t)||_\infty \leqslant \sum_h \left| [\hat{\xi}(t)]_h - [\xi(t)]_h \right|$$

$$|\hat{\xi}(t) \cdot e - \xi(t) \cdot e| \leqslant ||\hat{\xi}(t) - \xi(t)||_\infty ||e||_1 \leqslant \kappa ||\hat{\xi}(t) - \xi(t)||_\infty \leqslant \kappa \sum_h \left| [\hat{\xi}(t)]_h - [\xi(t)]_h \right|$$

$$|\xi(t) \cdot e| \leqslant ||\xi(t)||_\infty ||e||_1 \leqslant \kappa ||\xi(t)||_\infty \leqslant \kappa \sum_h [\xi(t)]_h.$$

Combining the above gives

$$\begin{aligned}\sum_{t\in\mathcal{T}(a,N)}|\hat{p}(t)-p(t)| &\leqslant (1+\kappa)\sum_{t\in\mathcal{T}(a,N)}\sum_h\left|[\hat{\xi}(t)]_h-[\xi(t)]_h\right|+\kappa\sum_{t\in\mathcal{T}(a,N)}\sum_h[\xi(t)]_h\\ &\leqslant m(1+\kappa)\left(\left(1+\frac{\Delta}{\gamma}\right)^N(1+\delta)^{N+1}-1\right)+m\kappa\\ &= m\left((1+\kappa)\left(1+\frac{\Delta}{\gamma}\right)^N(1+\delta)^{N+1}-1\right)\end{aligned}$$

where the second inequality follows because $\sum_{t\in\mathcal{T}(a,N)}\sum_h[\xi(t)]_h\leqslant m$, and because Lemma 13 gives

$$\sum_{t\in\mathcal{T}(a,N)}\sum_h\left|[\hat{\xi}(t)]_h-[\xi(t)]_h\right|\leqslant m\left(\left(1+\frac{\Delta}{\gamma}\right)^N(1+\delta)^{N+1}-1\right).$$

■

We now give a crucial lemma used in the previous proof:

Lemma 13 *Given the assumptions in Section B.1.1, for any a, h, N,*

$$D(a,h,N)=\sum_{t\in\mathcal{T}(a,N)}\left|[\hat{\xi}(t)]_h-[\xi(t)]_h\right|\leqslant Z(a,h,N)\left(\left(1+\frac{\Delta}{\gamma}\right)^N(1+\delta)^{N+1}-1\right).$$

Proof: A key identity is the following, which holds for any $N\geqslant 1$ (recall that $F(a,h,N)=D(a,h,N)/Z(a,h,N)$):

$$\begin{aligned}&F(a,h,N)\\ \leqslant\ &-1+\sum_{k=0}^{N-1}\sum_{b,c}\sum_{h_1,h_2}g(a,b,c,k,h_1,h_2)(1+F(b,h_1,k))(1+F(c,h_2,N-k-1))\\ &+\Delta\frac{Y(N)}{Z(a,h,N)}\sum_{k=0}^{N-1}\sum_{b,c}\sum_{h_1,h_2}h(b,c,k,h_1,h_2)(1+F(b,h_1,k))(1+F(c,h_2,N-k-1)),\end{aligned}\tag{36}$$

where

$$\begin{aligned}g(a,b,c,k,h_1,h_2) &= t(a\to b\ c,h_1,h_2|a,h)\frac{Z(b,h_1,k)Z(c,h_2,N-k-1)}{Z(a,h,N)}\\ Y(N) &= \sum_{k=0}^{N-1}\sum_{b,c}\sum_{h_1,h_2}Z(b,h_1,k)Z(c,h_2,N-k-1)\\ h(b,c,k,h_1,h_2) &= \frac{Z(b,h_1,k)Z(c,h_2,N-k-1)}{Y(N)}.\end{aligned}$$

The proof of Eq. 36 is in Section B.1.3. Note that we have

$$\sum_{k=0}^{N-1} \sum_{b,c} \sum_{h_1,h_2} g(a,b,c,k,h_1,h_2) = \sum_{k=0}^{N-1} \sum_{b,c} \sum_{h_1,h_2} h(b,c,k,h_1,h_2) = 1.$$

The rest of the proof follows through induction. For the base case, for $N = 0$ we have

$$Z(a,h,N)\left(\left(1+\frac{\Delta}{\gamma}\right)^N (1+\delta)^{N+1} - 1\right) = \delta Z(a,h,N) = \delta$$

where the last equality follows because $Z(a,h,0) = 1$ for any a, h. For $N = 0$ we also have

$$\sum_{t \in \mathcal{T}(a,N)} \left| [\hat{\xi}(t)]_h - [\xi(t)]_h \right| = \sum_x |[\hat{c}^{\infty}_{a \to x} G^a]_h - [q^{\infty}_{a \to x}]_h| \leqslant \delta.$$

The base case follows immediately.

For the recursive case, by the inductive hypothesis we have

$$1 + F(b,h_1,k) \leqslant \left(1+\frac{\Delta}{\gamma}\right)^k (1+\delta)^{k+1}$$

and

$$1 + F(c,h_2,N-k-1) \leqslant \left(1+\frac{\Delta}{\gamma}\right)^{N-k-1} (1+\delta)^{N-k}.$$

It follows from Eq. 36 that

$$\begin{aligned} F(a,h,N) &\leqslant -1 + \left(1 + \Delta \frac{Y(N)}{Z(a,h,N)}\right)\left(1+\frac{\Delta}{\gamma}\right)^{N-1} (1+\delta)^{N+1} \\ &\leqslant -1 + \left(1+\frac{\Delta}{\gamma}\right)^N (1+\delta)^{N+1} \end{aligned}$$

where the second inequality follows because

$$\frac{Y(N)}{Z(a,h,N)} = \frac{\sum_{k=0}^{N-1} \sum_{b,c} \sum_{h_1,h_2} Z(b,h_1,k) Z(c,h_2,N-k-1)}{\sum_{k=0}^{N-1} \sum_{b,c} \sum_{h_1,h_2} t(a \to b\ c, h_1, h_2 | a, h) Z(b,h_1,k) Z(c,h_2,N-k-1)} \leqslant \frac{1}{\gamma}.$$

This completes the proof. ■

B.1.3 Proof of Eq. 36

Any tree $t \in \mathcal{T}(a,N)$ where $N \geqslant 1$ can be decomposed into the following: 1) A choice b, c, implying the rule $a \to b\ c$ is at the root; 2) A choice of $0 \leqslant k \leqslant N-1$, implying that the tree dominated by b is of size k, the tree dominated by c is of size $N-1-k$; 3) A choice of trees $t_1 \in \mathcal{T}(b,k)$ and $t_2 \in \mathcal{T}(c,N-1-k)$. The resulting tree has $\xi_h(t) = T_h^{a \to b\ c}(\xi(t_1), \xi(t_2))$.

Define $d(t) = \xi(t) - \hat{\xi}(t)$. We then have the following:

$$\begin{aligned}
& \sum_{t \in \mathcal{T}(a,N)} |\hat{\xi}_h(t) - \xi_h(t)| \\
= & \sum_{k=0}^{N-1} \sum_{b,c} \sum_{t_1 \in \mathcal{T}(b,k)} \sum_{t_2 \in \mathcal{T}(c,N-1-k)} |\hat{T}_h^{a \to b\ c}(\hat{\xi}(t_1), \hat{\xi}(t_2)) - T_h^{a \to b\ c}(\xi(t_1), \xi(t_2))| \\
\leqslant & \Delta \sum_{k=0}^{N-1} \sum_{b,c} \sum_{t_1 \in \mathcal{T}(b,k)} \sum_{t_2 \in \mathcal{T}(c,N-1-k)} (||\xi(t_1)||_2 + ||d(t_1)||_2)(||\xi(t_2)||_2 + ||d(t_2)||_2) \\
& + \sum_{k=0}^{N-1} \sum_{b,c} \sum_{t_1 \in \mathcal{T}(b,k)} \sum_{t_2 \in \mathcal{T}(c,N-1-k)} |T_h^{a \to b\ c}(\xi(t_1), d(t_2))| \\
& + \sum_{k=0}^{N-1} \sum_{b,c} \sum_{t_1 \in \mathcal{T}(b,k)} \sum_{t_2 \in \mathcal{T}(c,N-1-k)} |T_h^{a \to b\ c}(d(t_1), \xi(t_2))| \\
& + \sum_{k=0}^{N-1} \sum_{b,c} \sum_{t_1 \in \mathcal{T}(b,k)} \sum_{t_2 \in \mathcal{T}(c,N-1-k)} |T_h^{a \to b\ c}(d(t_1), d(t_2))|. \qquad (37)
\end{aligned}$$

The inequality follows because by Lemma 14,

$$\begin{aligned}
& |\hat{T}_h^{a \to b\ c}(\hat{\xi}(t_1), \hat{\xi}(t_2)) - T_h^{a \to b\ c}(\xi(t_1), \xi(t_2))| \\
\leqslant & \Delta(||\xi(t_1)||_2 + ||d(t_1)||_2)(||\xi(t_2)||_2 + ||d(t_2)||_2) \\
& + |T_h^{a \to b\ c}(\xi(t_1), d(t_2))| + |T_h^{a \to b\ c}(d(t_1), \xi(t_2))| + |T_h^{a \to b\ c}(d(t_1), d(t_2))|.
\end{aligned}$$

We first derive an upper bound on the last three terms of Eq. 37. Note that we have the identity

$$\begin{aligned}
& Z(a, h, N) \\
= & \sum_{k=0}^{N-1} \sum_{b,c} \sum_{h_1,h_2} t(a \to b\ c, h_1, h_2 | a, h) \sum_{t_1 \in \mathcal{T}(b,k)} \xi_{h_1}(t_1) \sum_{t_2 \in \mathcal{T}(c,N-1-k)} \xi_{h_2}(t_2) \\
= & \sum_{k=0}^{N-1} \sum_{b,c} \sum_{h_1,h_2} t(a \to b\ c, h_1, h_2 | a, h) Z(b, h_1, k) Z(c, h_2, N-k-1).
\end{aligned}$$

It follows that

$$
\begin{aligned}
&\sum_{k=0}^{N-1}\sum_{b,c}\sum_{t_1\in\mathcal{T}(b,k)}\sum_{t_2\in\mathcal{T}(c,N-1-k)}\big(|T_h^{a\to b\ c}(\xi(t_1),d(t_2))|+|T_h^{a\to b\ c}(d(t_1),\xi(t_2))|\\
&\qquad\qquad +|T_h^{a\to b\ c}(d(t_1),d(t_2))|\big)\\
=\ &\sum_{k=0}^{N-1}\sum_{b,c}\sum_{h_1,h_2}t(a\to b\ c,h_1,h_2|a,h)\sum_{t_1\in\mathcal{T}(b,k)}\xi(t_1)_{h_1}\sum_{t_2\in\mathcal{T}(c,N-1-k)}|d(t_2)_{h_2}|\\
&+\sum_{k=0}^{N-1}\sum_{b,c}\sum_{h_1,h_2}t(a\to b\ c,h_1,h_2|a,h)\sum_{t_1\in\mathcal{T}(b,k)}|d(t_1)_{h_1}|\sum_{t_2\in\mathcal{T}(c,N-1-k)}\xi(t_2)_{h_2}\\
&+\sum_{k=0}^{N-1}\sum_{b,c}\sum_{h_1,h_2}t(a\to b\ c,h_1,h_2|a,h)\sum_{t_1\in\mathcal{T}(b,k)}|d(t_1)_{h_1}|\sum_{t_2\in\mathcal{T}(c,N-1-k)}|d(t_2)_{h_2}|\\
=\ &\Bigg(\sum_{k=0}^{N-1}\sum_{b,c}\sum_{h_1,h_2}t(a\to b\ c,h_1,h_2|a,h)\\
&\qquad\times\Bigg(\sum_{t_1\in\mathcal{T}(b,k)}\sum_{t_2\in\mathcal{T}(c,N-1-k)}(\xi(t_1)_{h_1}+|d(t_1)_{h_1}|)(\xi(t_2)_{h_2}+|d(t_2)_{h_2}|)\Bigg)\Bigg)-Z(a,h,N)\\
=\ &\Bigg(\sum_{k=0}^{N-1}\sum_{b,c}\sum_{h_1,h_2}t(a\to b\ c,h_1,h_2|a,h)(Z(b,h_1,k)+\\
&\qquad D(b,h_1,k))(Z(c,h_2,N-k-1)+D(c,h_2,N-k-1))\Bigg)-Z(a,h,N)\\
=\ &\Bigg(\sum_{k=0}^{N-1}\sum_{b,c}\sum_{h_1,h_2}t(a\to b\ c,h_1,h_2|a,h)Z(b,h_1,k)Z(c,h_2,N-k-1)\\
&\qquad\times(1+\frac{D(b,h_1,k)}{Z(b,h_1,k)})(1+\frac{D(c,h_2,N-k-1)}{Z(c,h_2,N-k-1)})\Bigg)-Z(a,h,N)\\
=\ &Z(a,h,N)\left(\sum_{k=0}^{N-1}\sum_{b,c}\sum_{h_1,h_2}g(a,b,c,k,h_1,h_2)(1+F(b,h_1,k))(1+F(c,h_2,N-k-1))\right)\\
&-Z(a,h,N)
\end{aligned}
\tag{38}
$$

where $g(a,b,c,k,h_1,h_2)=\frac{t(a\to b\ c,h_1,h_2|a,h)Z(b,h_1,k)Z(c,h_2,N-k-1)}{Z(a,h,n)}$.

We next derive a bound on the first term as follows:

$$\begin{aligned}
& \Delta \sum_{k=0}^{N-1} \sum_{b,c} \sum_{t_1 \in \mathcal{T}(b,k)} \sum_{t_2 \in \mathcal{T}(c,N-1-k)} (||\xi(t_1)||_2 + ||d(t_1)||_2)(||\xi(t_2)||_2 + ||d(t_2)||_2) \\
\leqslant & \Delta \sum_{k=0}^{N-1} \sum_{b,c} \sum_{t_1 \in \mathcal{T}(b,k)} \sum_{t_2 \in \mathcal{T}(c,N-1-k)} (||\xi(t_1)||_1 + ||d(t_1)||_1)(||\xi(t_2)||_1 + ||d(t_2)||_1) \\
= & \Delta \sum_{k=0}^{N-1} \sum_{b,c} \sum_{h_1,h_2} (Z(b,h_1,k) + D(b,h_1,k))(Z(c,h_2,N-k-1) + D(c,h_2,N-k-1)) \\
= & \Delta \sum_{k=0}^{N-1} \sum_{b,c} \sum_{h_1,h_2} Z(b,h_1,k)Z(c,h_2,N-k-1)(1+F(b,h_1,k))(1+F(c,h_2,N-k-1)) \\
= & \Delta Y(N) \sum_{k=0}^{N-1} \sum_{b,c} \sum_{h_1,h_2} h(k,b,c,h_1,h_2)(1+F(b,h_1,k))(1+F(c,h_2,N-k-1)) \qquad (39)
\end{aligned}$$

where

$$h(k,b,c,h_1,h_2) = \frac{Z(b,h_1,k)Z(c,h_2,N-k-1)}{Y(N)}$$

and $Y(N) = \sum_{k=0}^{N-1} \sum_{b,c} \sum_{h_1,h_2} Z(b,h_1,k)Z(c,h_2,N-k-1)$.

Combining Eqs. 37, 38 and 39 gives the inequality in Eq. 36, repeated below:

$$\begin{aligned}
& F(a,h,N) \\
\leqslant & -1 \\
& + \sum_{k=0}^{N-1} \sum_{b,c} \sum_{h_1,h_2} g(a,b,c,k,h_1,h_2)(1+F(b,h_1,k))(1+F(c,h_2,N-k-1)) \\
& + \Delta \frac{Y(N)}{Z(a,h,N)} \sum_{k=0}^{N-1} \sum_{b,c} \sum_{h_1,h_2} h(b,c,k,h_1,h_2)(1+F(b,h_1,k))(1+F(c,h_2,N-k-1)).
\end{aligned}$$

■

The following lemma was used in the previous proof:

Lemma 14 *Assume we have tensors $\hat{T}$ and T and that there is some constant Δ such that for any $y^1, y^2 \in \mathbb{R}^m$,*

$$||\hat{T}(y^1,y^2) - T(y^1,y^2)||_\infty \leqslant \Delta ||y^1||_2 ||y^2||_2$$

Then for any $y^1, y^2, \hat{y}^1, \hat{y}^2$, for any h, it follows that

$$\begin{aligned}
|\hat{T}_h(\hat{y}^1,\hat{y}^2) - T_h(y^1,y^2)| \leqslant & \Delta(||y^1||_2 + ||d^1||_2)(||y^2||_2 + ||d^2||_2) \\
& + |T_h(y^1,d^2)| + |T_h(d^1,d^2)| + |T_h(d^1,y^2)|
\end{aligned}$$

where $d^1 = \hat{y}^1 - y^1$, and $d^2 = \hat{y}^2 - y^2$.

Proof: Define

$$\hat{g}(y^1) = \hat{T}_h(y^1, \hat{y}^2),$$
$$g(y^1) = T_h(y^1, y^2).$$

Define $d^1 = (\hat{y}^1 - y^1)$, $d^2 = (\hat{y}^2 - y^2)$. For any $v \in \mathbb{R}^m$,

$$\begin{aligned} |\hat{g}(v) - g(v)| &= |\hat{T}_h(v, \hat{y}^2) - T_h(v, y^2)| \\ &\leqslant |\hat{T}_h(v, y^2) - T_h(v, y^2)| + |\hat{T}_h(v, d^2) - T_h(v, d^2)| + |T_h(v, d^2)|. \end{aligned}$$

We can then derive the following bound:

$$\begin{aligned} &|\hat{T}_h(\hat{y}^1, \hat{y}^2) - T_h(y^1, y^2)| = |\hat{g}(\hat{y}^1) - g(y^1)| \\ &\leqslant |\hat{g}(y^1) - g(y^1)| + |\hat{g}(d^1) - g(d^1)| + |g(d^1)| \\ &\leqslant |\hat{T}_h(y^1, y^2) - T_h(y^1, y^2)| + |\hat{T}_h(y^1, d^2) - T_h(y^1, d^2)| + |T_h(y^1, d^2)| \\ &\quad + |\hat{T}_h(d^1, y^2) - T_h(d^1, y^2)| + |\hat{T}_h(d^1, d^2) - T_h(d^1, d^2)| + |T_h(d^1, d^2)| \\ &\quad + |T_h(d^1, y^2)| \\ &\leqslant \Delta(||y^1||_2 + ||d^1||_2)(||y^2||_2 + ||d^2||_2) \\ &\quad + |T_h(y^1, d^2)| + |T_h(d^1, d^2)| + |T_h(d^1, y^2)|. \end{aligned}$$

■

B.2 Relating Δ, δ, κ to Estimation Errors

We now give a lemma that relates estimation errors in the algorithm to the values for Δ, δ and κ as defined in the previous section.

Throughout this section, in addition to the estimates $\hat{D}^{a \to b\ c}$, $\hat{d}^{\infty}_{a \to x}$, $\hat{\Sigma}^a$, $\hat{C}^{a \to b\ c}$, $\hat{c}^{\infty}_{a \to x}$, $\hat{c}^1_a$ computed by the algorithm in Figure 7, we define quantities

$$\begin{aligned} \Sigma^a &= \mathbf{E}[Y_1 Z^\top | A_1 = a] \\ D^{a \to b\ c} &= \mathbf{E}\left[[\![R_1 = a \to b\ c]\!] Z Y_2^\top Y_3^\top | A_1 = a\right] \\ d^{\infty}_{a \to x} &= \mathbf{E}\left[[\![R_1 = a \to x]\!] Z^\top | A_1 = a\right] \\ C^{a \to b\ c}(y^1, y^2) &= D^{a \to b\ c}(y^1, y^2)(\Sigma^a)^{-1} \\ c^{\infty}_{a \to x} &= d^{\infty}_{a \to x}(\Sigma^a)^{-1} \\ c^1_a &= \mathbf{E}\left[[\![A_1 = a]\!] Y_1 | B = 1\right] \end{aligned}$$

where

$$Y_1 = (\hat{U}^{a_1})^\top \phi(T_1) \quad Z = (\hat{V}^{a_1})^\top \psi(O)$$
$$Y_2 = (\hat{U}^{a_2})^\top \phi(T_2) \quad Y_3 = (\hat{U}^{a_3})^\top \phi(T_3).$$

Note that these definitions are identical to those given in Section 7.2, with the additional detail that the projection matrices used to define random variables Y_1, Y_2, Y_3, Z are $\hat{U}^a$ and $\hat{V}^a$, that is, the projection matrices estimated in the first step of the algorithm in Figure 7.

The lemma is as follows:

Lemma 15 *Assume that under a run of the algorithm in Figure 7 there are constants* $\epsilon_\Omega^1, \epsilon_\Omega^2, \epsilon_D, \epsilon_d, \epsilon_\pi$ *such that*

$$\begin{aligned}
\forall a \in \mathcal{P}, \quad & ||\hat{\Omega}^a - \Omega^a||_F \leqslant \epsilon_\Omega^1 \\
\forall a \in \mathcal{I}, \quad & ||\hat{\Omega}^a - \Omega^a||_F \leqslant \epsilon_\Omega^2 \\
\forall a \to b\ c, \quad & ||\hat{D}^{a \to b\ c} - D^{a \to b\ c}||_F \leqslant \epsilon_D \\
\forall a \in \mathcal{P}, \quad & \sqrt{\sum_x ||\hat{d}^\infty_{a \to x} - d^\infty_{a \to x}||_2^2} \leqslant \epsilon_d \\
\forall a, \quad & ||\hat{c}_a^1 - c_a^1||_2 \leqslant \epsilon_\pi.
\end{aligned}$$

Assume in addition that $\epsilon_\Omega^1 \leqslant \min_{a \in \mathcal{P}} \frac{\sigma_m(\Omega^a)}{3}$ *and* $\epsilon_\Omega^2 \leqslant \min_{a \in \mathcal{I}} \frac{\sigma_m(\Omega^a)}{3}$. *For all* a *define* $G^a = (\hat{U}^a)^\top I^a$ *and* $H^a = (G^a)^{-1}$. *Then:*

- *For all* a, G^a *is invertible.*
- *For all* $y^1, y^2 \in \mathbb{R}^m$, *for all rules of the form* $a \to b\ c$

$$||\hat{C}^{a \to b\ c}(y^1 H^b, y^2 H^c) G^a - C^{a \to b\ c}(y^1 H^b, y^2 H^c) G^a||_\infty \leqslant \Delta ||y^1||_2 ||y^2||_2$$

  where

$$\Delta = \frac{16}{3} \frac{1}{\sigma_m(I^b)\sigma_m(I^c)} \left(\frac{\epsilon_\Omega^2}{\sigma_m(\Omega^a)^2} + \frac{\epsilon_D}{3\sigma_m(\Omega^a)} \right).$$

- *For all* $a \in \mathcal{P}$, *for all* $h \in [m]$,

$$\sum_x |[\hat{c}^\infty_{a \to x} G^a]_h - [c^\infty_{a \to x} G^a]_h| \leqslant \delta$$

  where

$$\delta = 4 \left(\frac{\epsilon_\Omega^1}{\sigma_m(\Omega^a)^2} + \frac{\epsilon_d \sqrt{n}}{3\sigma_m(\Omega^a)} \right).$$

- *For all* a,

$$||(G^a)^{-1} \hat{c}_a^1 - (G^a)^{-1} c_a^1||_1 \leqslant \kappa$$

  where

$$\kappa = \frac{2}{\sqrt{3}} \frac{\sqrt{m}}{\sigma_m(\Omega^a)} \epsilon_\pi.$$

B.2.1 Proof of Lemma 15

We first prove three necessary lemmas, then give a proof of Lemma 15.

Lemma 16 *Assume we have vectors and matrices* $d \in \mathbb{R}^{1 \times m}$, $\Sigma \in \mathbb{R}^{m \times m}$, $\hat{d} \in \mathbb{R}^{1 \times m}$, $\hat{\Sigma} \in \mathbb{R}^{m \times m}$, $U \in \mathbb{R}^{d \times m}$, $I \in \mathbb{R}^{d \times m}$. *We assume that* Σ, $\hat{\Sigma}$, *and* $(U^\top I)$ *are invertible.*

In addition, define

$$\begin{aligned}
c &= d\Sigma^{-1} \\
\hat{c} &= \hat{d}\hat{\Sigma}^{-1} \\
G^a &= U^\top I.
\end{aligned}$$

We assume:

- *For $h = 1 \ldots m$, $||I_h||_2 \leqslant 1$, where I_h is the h'th column of I^a.*
- *$||U||_{2,o} \leqslant 1$ where $||U||_{2,o}$ is the spectral norm of the matrix U.*
- $||\hat{\Sigma} - \Sigma||_{2,o} \leqslant \epsilon_1$

It follows that

$$||\hat{c}G^a - cG^a||_\infty \leqslant \frac{1+\sqrt{5}}{2} \frac{\epsilon_1 ||\hat{d}||_2}{\min\{\sigma_m(\Sigma), \sigma_m(\hat{\Sigma})\}^2} + \frac{||\hat{d} - d||_2}{\sigma_m(\Sigma)}.$$

Proof:

$$\begin{aligned}
& ||\hat{c}G^a - cG^a||_\infty \\
= \; & ||(\hat{c} - c)U^\top I||_\infty \\
& \text{(By definition } G^a = U^\top I) \\
\leqslant \; & ||(\hat{c} - c)U^\top||_2 \\
& \text{(By } ||I_h||_2 \leqslant 1) \\
\leqslant \; & ||\hat{c} - c||_2 \\
& \text{(By } ||U||_{2,o} \leqslant 1) \\
= \; & ||\hat{d}\hat{\Sigma}^{-1} - d\Sigma^{-1}||_2 \\
& \text{(By definitions of } c, \hat{c}) \\
\leqslant \; & ||\hat{d}(\hat{\Sigma}^{-1} - \Sigma^{-1})||_2 + ||(\hat{d} - d)\Sigma^{-1}||_2 \\
& \text{(By triangle inequality)} \\
\leqslant \; & ||\hat{d}||_2 ||\hat{\Sigma}^{-1} - \Sigma^{-1}||_{2,o} + ||\hat{d} - d||_2 ||\Sigma^{-1}||_{2,o} \\
& \text{(By definition of } ||.||_{2,o}) \\
\leqslant \; & ||\hat{d}||_2 \frac{1+\sqrt{5}}{2} \frac{\epsilon_1}{\min\{\sigma_m(\Sigma), \sigma_m(\hat{\Sigma})\}^2} + \frac{||\hat{d} - d||_2}{\sigma_m(\Sigma)} \\
& \text{(By Lemma 23 of Hsu et al. 2009, and } ||\Sigma^{-1}||_{2,o} = 1/\sigma_m(\Sigma))
\end{aligned}$$

Lemma 17 *Assume we have vectors $c, \hat{c} \in \mathbb{R}^{m \times 1}$, and we have a matrix $G^a \in \mathbb{R}^{m \times m}$ that is invertible. It follows that*

$$||(G^a)^{-1}\hat{c} - (G^a)^{-1}c||_1 \leqslant \frac{\sqrt{m}||\hat{c} - c||_2}{\sigma_m(G^a)}.$$

Proof:

$$||(G^a)^{-1}\hat{c} - (G^a)^{-1}c||_1 \leqslant \sqrt{m}||(G^a)^{-1}\hat{c} - (G^a)^{-1}c||_2 \leqslant \frac{\sqrt{m}||\hat{c} - c||_2}{\sigma_m(G^a)}.$$

The first inequality follows because $||.||_1 \leqslant \sqrt{m}||.||_2$. The second inequality follows because $||(G^a)^{-1}||_{2,o} = 1/\sigma_m(G^a)$.

Lemma 18 *Assume we have matrices and tensors $D \in \mathbb{R}^{m \times m \times m}$, $\Sigma \in \mathbb{R}^{m \times m}$, $\hat{D} \in \mathbb{R}^{m \times m \times m}$, $\hat{\Sigma} \in \mathbb{R}^{m \times m}$, $U \in \mathbb{R}^{d \times m}$, $I \in \mathbb{R}^{d \times m}$, $G^b \in \mathbb{R}^{m \times m}$, $G^c \in \mathbb{R}^{m \times m}$. We assume that Σ, $\hat{\Sigma}$, G^b, G^c, and $U^\top I$ are invertible.*

In addition define

$$\begin{aligned} C(y^1, y^2) &= D(y^1, y^2)\Sigma^{-1} \\ \hat{C}(y^1, y^2) &= \hat{D}(y^1, y^2)\hat{\Sigma}^{-1} \\ G^a &= U^\top I \\ H^b &= (G^b)^{-1} \\ H^c &= (G^c)^{-1} \end{aligned}$$

We assume:

- *For $h = 1 \ldots m$, $||I_h||_2 \leqslant 1$, where I_h is the h'th column of I^a.*
- $||U||_{2,o} \leqslant 1$
- $||\hat{\Sigma} - \Sigma||_{2,o} \leqslant \epsilon_1$

It follows that for any $y^1, y^2 \in \mathbb{R}^m$,

$$\begin{aligned} & ||\hat{C}(y^1 H^b, y^2 H^c)G^a - C(y^1 H^b, y^2 H^c)G^a||_\infty \\ \leqslant \; & \frac{||y^1||_2 ||y^2||_2}{\sigma_m(G^b)\sigma_m(G^c)} \left(\frac{1+\sqrt{5}}{2} \times \frac{\epsilon_1 ||\hat{D}||_F}{\min\{\sigma_m(\Sigma), \sigma_m(\hat{\Sigma})\}^2} + \frac{||\hat{D} - D||_F}{\sigma_m(\Sigma)} \right). \end{aligned}$$

Proof:

$$\begin{aligned} & ||\hat{C}(y^1 H^b, y^2 H^c)G^a - C(y^1 H^b, y^2 H^c)G^a||_\infty \\ \leqslant \; & ||\hat{D}(y^1 H^b, y^2 H^c)||_2 \frac{1+\sqrt{5}}{2} \frac{\epsilon_1}{\min\{\sigma_m(\Sigma), \sigma_m(\hat{\Sigma})\}^2} + \frac{||\hat{D}(y^1 H^b, y^2 H^c) - D(y^1 H^b, y^2 H^c)||_2}{\sigma_m(\Sigma)} \\ & \text{(By Lemma 16, using } \hat{d} = \hat{D}(y^1 H^b, y^2 H^c),\ d = D(y^1 H^b, y^2 H^c).) \\ \leqslant \; & ||y^1 H^b||_2 ||y^2 H^c||_2 \left(||\hat{D}||_F \frac{1+\sqrt{5}}{2} \frac{\epsilon_1}{\min\{\sigma_m(\Sigma), \sigma_m(\hat{\Sigma})\}^2} + \frac{||\hat{D} - D||_F}{\sigma_m(\Sigma)} \right) \\ & \text{(By } ||D(v^1, v^2)||_2 \leqslant ||D||_F ||v^1||_2 ||v^2||_2 \text{ for any tensor } D, \text{ vectors } v^1, v^2.) \\ \leqslant \; & \frac{||y^1||_2 ||y^2||_2}{\sigma_m(G^b)\sigma_m(G^c)} \left(||\hat{D}||_F \frac{1+\sqrt{5}}{2} \frac{\epsilon_1}{\min\{\sigma_m(\Sigma), \sigma_m(\hat{\Sigma})\}^2} + \frac{||\hat{D} - D||_F}{\sigma_m(\Sigma)} \right) \\ & \text{(By } H^b = (G^b)^{-1} \text{ hence } ||H^b||_{2,o} = 1/\sigma_m(G^b). \text{ Similar for } H^c.) \end{aligned}$$

Proof of Lemma 15: By Lemma 9 of Hsu et al. (2009), assuming that $\epsilon_\Omega \leqslant \min_a \frac{\sigma_m(\Omega^a)}{3}$ gives for all a

$$\sigma_m(\hat{\Sigma}^a) \geqslant \frac{2}{3}\sigma_m(\Omega^a)$$

$$\sigma_m(\Sigma^a) \geqslant \frac{\sqrt{3}}{2}\sigma_m(\Omega^a)$$

$$\sigma_m(G^a) \geqslant \frac{\sqrt{3}}{2}\sigma_m(I^a)$$

The condition that $\sigma_m(I^a) > 0$ implies that $\sigma_m(G^a) > 0$ and hence G^a is invertible. The values for Δ and κ follow from lemmas 18 and and 17 respectively.

The value for δ is derived as follows. By Lemma 16 we have for any rule $a \to x$, for any $h \in [m]$,

$$|[\hat{c}^{\infty}_{a\to x}G^a]_h - [c^{\infty}_{a\to x}G^a]_h| \leqslant \frac{1+\sqrt{5}}{2}\frac{\epsilon_1||\hat{d}^{\infty}_{a\to x}||_2}{\min\{\sigma_m(\Sigma^a), \sigma_m(\hat{\Sigma^a})\}^2} + \frac{||\hat{d}^{\infty}_{a\to x} - d^{\infty}_{a\to x}||_2}{\sigma_m(\Sigma^a)}. \quad (40)$$

By definition

$$\hat{d}^{\infty}_{a\to x} = \left(\frac{\sum_{i=1}^{M}[\![r^{(i,1)} = a \to x]\!]}{\sum_{i=1}^{M}[\![a_i = a]\!]}\right) \times \left(\frac{\sum_{i=1}^{M}[\![r^{(i,1)} = a \to x]\!](z^{(i)})^\top}{\sum_{i=1}^{M}[\![r^{(i,1)} = a \to x]\!]}\right)$$

In addition $z^{(i)} = (\hat{V}^{a_i})^\top\psi(t^{(i,1)})$ and $||\hat{V}^{a_i}||_{2,o} \leqslant 1$, $||\psi(t^{(i,1)})||_2 \leqslant 1$, hence $||z^{(i)}||_2 \leqslant 1$, and

$$||\hat{d}^{\infty}_{a\to x}||_2 \leqslant \frac{\sum_{i=1}^{M}[\![r^{(i,1)} = a \to x]\!]}{\sum_{i=1}^{M}[\![a_i = a]\!]}.$$

It follows that

$$\sum_x ||\hat{d}^{\infty}_{a\to x}||_2 \leqslant 1. \quad (41)$$

In addition we have

$$\sum_x ||\hat{d}^{\infty}_{a\to x} - d^{\infty}_{a\to x}||_2 \leqslant \sqrt{n}\sqrt{\sum_x ||\hat{d}^{\infty}_{a\to x} - d^{\infty}_{a\to x}||_2^2} \leqslant \sqrt{n}\epsilon_d. \quad (42)$$

Combining Eqs. 41, 42 and 40 gives for any $a \in \mathcal{P}$, for any $h \in [m]$,

$$\sum_x |[\hat{c}^{\infty}_{a\to x}G^a]_h - [c^{\infty}_{a\to x}G^a]_h| \leqslant \frac{1+\sqrt{5}}{2}\frac{\epsilon_1}{\min\{\sigma_m(\Sigma^a), \sigma_m(\hat{\Sigma^a})\}^2} + \frac{\sqrt{n}\sqrt{\sum_x ||\hat{d}^{\infty}_{a\to x} - d^{\infty}_{a\to x}||_2^2}}{\sigma_m(\Sigma^a)}$$

from which the lemma follows.

■

B.3 Estimation Errors

The next lemma relates estimation errors to the number of samples in the algorithm in Figure 4:

Lemma 19 *Consider the algorithm in Figure 7. With probability at least $1-\delta$, the following statements hold:*

$$\forall a \in \mathcal{I}, \sqrt{\sum_{b,c} ||\hat{D}^{a\to b\ c} - D^{a\to b\ c}||_F^2} \leqslant \sqrt{\frac{1}{M_a}} + \sqrt{\frac{2}{M_a}\log\frac{2|\mathcal{N}|+1}{\delta}}$$

$$\forall a \in \mathcal{P}, \sqrt{\sum_x ||\hat{d}^{\infty}_{a \to x} - d^{\infty}_{a \to x}||_2^2} \leqslant \sqrt{\frac{1}{M_a}} + \sqrt{\frac{2}{M_a} \log \frac{2|\mathcal{N}| + 1}{\delta}}$$

$$\forall a \in \mathcal{N}, ||\hat{\Omega}^a - \Omega^a||_F \leqslant \sqrt{\frac{1}{N_a}} + \sqrt{\frac{2}{N_a} \log \frac{2|\mathcal{N}| + 1}{\delta}}$$

$$\sqrt{\sum_a ||\hat{c}^1_a - c^1_a||_2^2} \leqslant \sqrt{\frac{1}{R}} + \sqrt{\frac{2}{R} \log \frac{2|\mathcal{N}| + 1}{\delta}}$$

B.3.1 Proof of Lemma 19

We first need the following lemma:

Lemma 20 *Assume i.i.d. random vectors $X_1 \dots X_N$ where each $X_i \in \mathbb{R}^d$, and for all i with probability 1, $||X_i||_2 \leqslant 1$. Define*

$$q = \mathbf{E}[X_i]$$

for all i and

$$\hat{Q} = \frac{\sum_{i=1}^N X_i}{N}.$$

Then for any $\epsilon > 0$,

$$\mathbf{P}(||\hat{Q} - q||_2 \geqslant 1/\sqrt{N} + \epsilon) \leqslant e^{-N\epsilon^2/2}.$$

Proof: The proof is very similar to the proof of proposition 19 of Hsu et al. (2009). Consider two random samples $x_1 \dots x_n$ and $y_1 \dots y_n$ where $x_i = y_i$ for all $i \neq k$. define

$$\hat{q} = \frac{\sum_{i=1}^N x_i}{N}$$

and

$$\hat{p} = \frac{\sum_{i=1}^N y_i}{N}.$$

Then

$$||\hat{q} - q||_2 - ||\hat{p} - q||_2 \leqslant ||\hat{q} - \hat{p}||_2 = \frac{||x_k - y_k||_2}{N} \leqslant \frac{||x_k||_2 + ||y_k||_2}{N} \leqslant \frac{2}{N}.$$

It follows through McDiarmid's inequality (McDiarmid, 1989) that

$$Pr(||\hat{Q} - q||_2 \geqslant \mathbf{E}||\hat{Q} - q||_2 + \epsilon) \leqslant e^{-N\epsilon^2/2}$$

In addition,

$$\begin{aligned}
& \mathbf{E}\left[||\hat{Q} - q||_2\right] \\
= \;& \mathbf{E}\left[||\frac{\sum_{i=1}^N X_i}{N} - q||_2\right] \\
= \;& \frac{1}{N}\mathbf{E}\left[||\sum_{i=1}^N (X_i - q)||_2\right] \\
\leqslant \;& \frac{1}{N}\sqrt{\mathbf{E}\left[||\sum_{i=1}^N (X_i - q)||_2^2\right]} \\
& \text{(By Jensen's inequality)} \\
= \;& \frac{1}{N}\sqrt{\sum_{i=1}^N \mathbf{E}\left[||(X_i - q)||_2^2\right]} \\
& \text{(By independence of the } X_i\text{'s)} \\
= \;& \frac{1}{N}\sqrt{\sum_{i=1}^N \mathbf{E}\left[||X_i||_2^2\right] - N||q||_2^2} \\
\leqslant \;& \frac{1}{N}\sqrt{N(1 - ||q||_2^2)} \\
& \text{(By } ||X_i||_2 \leqslant 1.\text{)} \\
\leqslant \;& \frac{1}{\sqrt{N}},
\end{aligned}$$

which completes the proof. ■

Proof of Lemma 19: For each $a \to b\ c$, $i, j, k \in [m]$, define a random variable

$$A_{i,j,k}^{a \to b\ c} = [\![R_1 = a \to b\ c]\!] Z_i Y_j^2 Y_k^3.$$

It follows that

$$D_{i,j,k}^{a \to b\ c} = \mathbf{E}[A_{i,j,k}^{a \to b\ c} | A_1 = a].$$

Note that

$$||Z||_2 = ||(V^a)^\top \psi(O)||_2 \leqslant 1$$

because $||V^a||_{2,o} \leqslant 1$, and $||\psi(O)||_2 \leqslant 1$. Similarly $||Y^2||_2 \leqslant 1$ and $||Y^3||_2 \leqslant 1$.

In addition we have for all $a \in \mathcal{I}$,

$$\begin{aligned}
\sum_{b,c}\sum_{i=1}^m\sum_{j=1}^m\sum_{k=1}^m |A_{i,j,k}^{a \to b\ c}|^2 &= \sum_{b,c}\sum_{i=1}^m\sum_{j=1}^m\sum_{k=1}^m |Z_i|^2 |Y_j^2|^2 |Y_k^3|^2 [\![R_1 = a \to b\ c]\!]^2 \\
&= ||Z||_2^2 ||Y^2||_2^2 ||Y^3||_2^2 (\sum_{b,c} [\![R_1 = a \to b\ c]\!]^2) \leqslant 1
\end{aligned}$$

It follows by an application of Lemma 20 that for the definitions of $D^{a\to b\ c}$ and $\hat{D}^{a\to b\ c}$ in Figure 7, for all a,

$$\mathbf{P}(\sqrt{\sum_{b,c}\sum_{i,j,k}|\hat{D}^{a\to b\ c}_{i,j,k}-D^{a\to b\ c}_{i,j,k}|^2}\geqslant 1/\sqrt{M_a}+\epsilon_1)\leqslant e^{-M_a\epsilon_1^2/2},$$

or equivalently,

$$\mathbf{P}\left(\sqrt{\sum_{b,c}||\hat{D}^{a\to b\ c}-D^{a\to b\ c}||_F^2}\geqslant\frac{1}{\sqrt{M_a}}+\sqrt{\frac{2}{M_a}\log\frac{2|\mathcal{N}|+1}{\delta}}\right)\leqslant\frac{\delta}{2|\mathcal{N}|+1}. \tag{43}$$

By a similar argument, if for each $a\in\mathcal{P}$, $x\in[n]$, $i\in[m]$ we define the random variable

$$B_i^{a\to x}=Z_i[\![R_1=a\to x]\!]$$

then

$$d^{\infty}_{a\to x}=\mathbf{E}[B_i^{a\to x}|A_1=a]$$

and

$$\sum_x\sum_{i=1}^m|B_i^{a\to x}|^2=\sum_x\sum_{i=1}^m|Z_i|^2[\![R_1=a\to x]\!]^2\leqslant 1$$

It follows by an application of Lemma 20 that for the definitions of $d^{\infty}_{a\to x}$ and $\hat{d}^{\infty}_{a\to x}$ in Figure 7, for all a,

$$\mathbf{P}(\sqrt{\sum_x\sum_i|[\hat{d}^{\infty}_{a\to x}]_i-[d^{\infty}_{a\to x}]_i|^2}\geqslant 1/\sqrt{N_a}+\epsilon_2)\leqslant e^{-M_a\epsilon_2^2/2}$$

or equivalently

$$\mathbf{P}\left(\sqrt{\sum_x||\hat{d}^{\infty}_{a\to x}-d^{\infty}_{a\to x}||_2^2}\geqslant\frac{1}{\sqrt{M_a}}+\sqrt{\frac{2}{M_a}\log\frac{2|\mathcal{N}|+1}{\delta}}\right)\leqslant\frac{\delta}{2|\mathcal{N}|+1}. \tag{44}$$

A similar argument can be used to show that for all a, for the definitions of Ω^a and $\hat{\Omega}^a$ in Figure 7,

$$\mathbf{P}(\sqrt{\sum_{i,j}|\hat{\Omega}^a_{i,j}-\Omega^a_{i,j}|^2}\geqslant 1/\sqrt{N_a}+\epsilon_3)\leqslant e^{-N_a\epsilon_3^2/2}$$

or equivalently

$$\mathbf{P}\left(||\hat{\Omega}^a-\Omega^a||_F\geqslant\frac{1}{\sqrt{N_a}}+\sqrt{\frac{2}{N_a}\log\frac{2|\mathcal{N}|+1}{\delta}}\right)\leqslant\frac{\delta}{2|\mathcal{N}|+1} \tag{45}$$

Finally, if we define the random variable

$$F_i^a=Y_i^1[\![A_1=a]\!]$$

then

$$\sum_a \sum_i |F_i^a|^2 = \sum_a \sum_i |Y_i^1|^2 [\![A_1 = a]\!]^2 \leqslant 1.$$

In addition

$$c_a^1 = \mathbf{E}[F_i^a | B = 1].$$

It follows by an application of Lemma 20 that for the definitions of c_a^1 and $\hat{c}_a^1$ in Figure 7,

$$\mathbf{P}(\sqrt{\sum_a \sum_i |[\hat{c}_a^1]_i - [c_a^1]_i|^2} \geqslant 1/\sqrt{R} + \epsilon_4) \leqslant e^{-R\epsilon_4^2/2}$$

or equivalently

$$\mathbf{P}\left(\sqrt{\sum_a ||\hat{c}_a^1 - c_a^1||_2^2} \geqslant \frac{1}{\sqrt{R}} + \sqrt{\frac{2}{R} \log \frac{2|\mathcal{N}|+1}{\delta}}\right) \leqslant \frac{\delta}{2|\mathcal{N}|+1}. \tag{46}$$

Finally, applying the union bound to the $2|\mathcal{N}|+1$ events in Eqs. 43, 44, 45 and 46 proves the theorem. ■

B.4 Proof of Theorem 8

Under the assumptions of the theorem, we have constants C_1, C_2, C_3, C_4 and C_5 such that

$$\forall a \in \mathcal{I}, N_a \geqslant L \times \left(C_1 \frac{N}{\gamma\epsilon} \frac{m}{\xi^2\sigma^2}\right)^2 \quad \forall a \in \mathcal{P}, N_a \geqslant L \times \left(\frac{C_2 N m}{\epsilon\sigma^2}\right)^2$$

$$\forall a \in \mathcal{I}, M_a \geqslant L \times \left(C_3 \frac{N}{\gamma\epsilon} \frac{m}{\xi^2\sigma}\right)^2 \quad \forall a \in \mathcal{P}, M_a \geqslant L \times \left(C_4 \frac{Nm\sqrt{n}}{\epsilon\sigma}\right)^2$$

$$R \geqslant L \times \left(C_5 \frac{Nm\sqrt{m}}{\epsilon\sigma}\right)^2$$

It follows from Lemma 19 that with probability at least $1-\delta$,

$$\begin{aligned}
\forall a \in \mathcal{I}, &\quad ||\hat{\Omega}^a - \Omega^a||_F \leqslant \epsilon_\Omega^1 \\
\forall a \in \mathcal{P}, &\quad ||\hat{\Omega}^a - \Omega^a||_F \leqslant \epsilon_\Omega^2 \\
\forall a \to b\ c, &\quad ||\hat{D}^{a \to b\ c} - D^{a \to b\ c}||_F \leqslant \epsilon_D \\
\forall a \in \mathcal{P}, &\quad \sqrt{\sum_x ||\hat{d}_{a \to x}^{\infty} - d_{a \to x}^{\infty}||_2} \leqslant \epsilon_d \\
\forall a, &\quad ||\hat{c}_a^1 - c_a^1||_2 \leqslant \epsilon_\pi
\end{aligned}$$

where

$$\epsilon_\Omega^1 \leqslant 3 \times \frac{1}{C_2} \times \sigma^2 \times \frac{\epsilon}{Nm}$$

$$\epsilon_\Omega^2 \leqslant 3 \times \frac{1}{C_1} \times \xi^2\sigma^2 \times \frac{\gamma\epsilon}{Nm}$$

$$\epsilon_D \leqslant 3 \times \frac{1}{C_3} \times \xi^2\sigma \times \frac{\gamma\epsilon}{Nm}$$

$$\epsilon_d \leqslant 3 \times \frac{1}{C_4} \times \sigma \times \frac{\epsilon}{\sqrt{n}Nm}$$
$$\epsilon_\pi \leqslant 3 \times \frac{1}{C_5} \times \frac{\sigma}{\sqrt{m}} \times \frac{\epsilon}{Nm}.$$

It follows from Lemma 15 that with suitable choices of $C_1 \dots C_5$, the inequalities in Lemma 15 hold with values

$$\Delta \leqslant \frac{\gamma\epsilon}{4Nm}$$
$$\delta \leqslant \frac{\epsilon}{4Nm}$$
$$\kappa \leqslant \frac{\epsilon}{4Nm}.$$

It follows from Lemma 12 that

$$\sum_{t \in \mathcal{T}(a,N)} |\hat{p}(t) - p(t)| \leqslant m \left(\left(1 + \frac{\epsilon}{4Nm} \right)^{2N} - 1 \right) \leqslant \epsilon$$

where the second inequality follows because $(1 + a/t)^t \leqslant 1 + 2a$ for $a \leqslant 1/2$.

References

S. Arora, R. Ge, Y. Halpern, D. M. Mimno, A. Moitra, D. Sontag, Y. Wu, and M. Zhu. A practical algorithm for topic modeling with provable guarantees. In *Proceedings of ICML*, 2013.

R. Bailly, A. Habrar, and F. Denis. A spectral approach for probabilistic grammatical inference on trees. In *Proceedings of ALT*, 2010.

R. Bailly, Carreras P. X., F. M. Luque, and A. J. Quattoni. Unsupervised spectral learning of WCFG as low-rank matrix completion. In *Proceedings of EMNLP*, 2013.

J. Baker. Trainable grammars for speech recognition. In *Proceedings of ASA*, 1979.

B. Balle, A. Quattoni, and X. Carreras. A spectral learning algorithm for finite state transducers. In *Proceedings of ECML*, 2011.

P.J. Bickel and K.A. Doksum. *Mathematical Statistics: Basic Ideas And Selected Topics.* Mathematical Statistics: Basic Ideas and Selected Topics. Pearson Prentice Hall, 2006.

E. Charniak. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of AAAI-IAAI*, 1997.

S. B. Cohen and M. Collins. A provably correct learning algorithm for latent-variable PCFGs. In *Proceedings of ACL*, 2014.

S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. Spectral learning of latent-variable PCFGs. In *Proceedings of ACL*, 2012.

S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL*, 2013.

M. Collins. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL*, 1997.

S. Dasgupta. Learning mixtures of Gaussians. In *Proceedings of FOCS*, 1999.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

P. Dhillon, D. Foster, and L. Ungar. Multi-view learning of word embeddings via CCA. In *Proceedings of NIPS*, 2011.

P. Dhillon, J. Rodu, M. Collins, D. P. Foster, and L. H. Ungar. Spectral dependency parsing with latent variables. In *Proceedings of EMNLP*, 2012.

D. P. Foster, J. Rodu, and L. H. Ungar. Spectral dimensionality reduction for HMMs. arXiv:1203.6130, 2012.

J. Goodman. Parsing algorithms and metrics. In *Proceedings of ACL*, 1996.

D. Hsu, S. M. Kakade, and T. Zhang. A spectral algorithm for learning hidden Markov models. In *Proceedings of COLT*, 2009.

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

M. Johnson. PCFG models of linguistic tree representations.

D. Klein and C.D. Manning. Accurate Unlexicalized Parsing. In *Proceedings of ACL*, pages 423–430, 2003.

F. M. Luque, A. Quattoni, B. Balle, and X. Carreras. Spectral learning for non-deterministic dependency parsing. In *Proceedings of EACL*, 2012.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

T. Matsuzaki, Y. Miyao, and J. Tsujii. Probabilistic CFG with latent annotations. In *Proceedings of ACL*, 2005.

C. McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, pages 148–188, 1989.

A. Moitra and G. Valiant. Settling the polynomial learnability of mixtures of Gaussians. *IEEE Annual Symposium on Foundations of Computer Science*, pages 93–102, 2010. ISSN 0272-5428.

A. Parikh, L. Song, and E. P. Xing. A spectral algorithm for latent tree graphical models. In *Proceedings of ICML*, 2011.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL*, 2006.

S. Siddiqi, B. Boots, and G. Gordon. Reduced-rank hidden Markov models. *Journal of Machine Learning Research*, 9:741–748, 2010.

L. Song, B. Boots, S. M. Siddiqi, G. J. Gordon, and A. J. Smola. Hilbert space embeddings of hidden Markov models. In *Proceedings of ICML*, 2010.

L. Song, A. P. Parikh, and E. P. Xing. Kernel embeddings of latent tree graphical models. In *NIPS*, pages 2708–2716, 2011.

K. Stratos, A. M. Rush, S. B. Cohen, and M. Collins. Spectral learning of refinement HMMs. In *Proceedings of CoNLL*, 2013.

S. A. Terwijn. On the learnability of hidden Markov models. In *Grammatical Inference: Algorithms and Applications (Amsterdam, 2002)*, volume 2484 of *Lecture Notes in Artificial Intelligence*, pages 261–268, Berlin, 2002. Springer.

A. Tropp, N. Halko, and P. G. Martinsson. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. In *Technical Report No. 2009-05*, 2009.

L. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

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

Journal of Machine Learning Research 15 (2014) 2451-2487 Submitted 7/13; Revised 1/14; Published 7/14

On Multilabel Classification and Ranking with Bandit Feedback

Claudio Gentile CLAUDIO.GENTILE@UNINSUBRIA.IT
DiSTA, Università dell'Insubria
via Mazzini 5
21100 Varese, Italy

Francesco Orabona FRANCESCO@ORABONA.COM
Toyota Technological Institute at Chicago
6045 South Kenwood Avenue
60637 Chicago, IL, USA

Editor: Peter Auer

Abstract

We present a novel multilabel/ranking algorithm working in partial information settings. The algorithm is based on 2nd-order descent methods, and relies on upper-confidence bounds to trade-off exploration and exploitation. We analyze this algorithm in a partial adversarial setting, where covariates can be adversarial, but multilabel probabilities are ruled by (generalized) linear models. We show $O(T^{1/2} \log T)$ regret bounds, which improve in several ways on the existing results. We test the effectiveness of our upper-confidence scheme by contrasting against full-information baselines on diverse real-world multilabel data sets, often obtaining comparable performance.

Keywords: contextual bandits, structured prediction, ranking, online learning, regret bounds, generalized linear

1. Introduction

Consider a book recommendation system. Given a customer's profile, the system recommends a few possible books to the user by means of, e.g., a limited number of banners placed at different positions on a webpage. The system's goal is to select books that the user likes and possibly purchases. Typical feedback in such systems is the actual action of the user or, in particular, what books he has bought/preferred, if any. The system cannot observe what would have been the user's actions had other books got recommended, or had the same book ads been placed in a different order within the webpage.

Such problems are collectively referred to as learning with partial feedback. As opposed to the full information case, where the system (the learning algorithm) knows the outcome of each possible response (e.g., the user's action for each and every possible book recommendation placed in the largest banner ad), in the partial feedback setting the system only observes the response to very limited options and, specifically, the option that was actually recommended.

In this and many other examples of this sort, it is reasonable to assume that recommended options are not given the same treatment by the system, e.g., large banners which

are displayed on top of the page should somehow be more committing as a recommendation than smaller ones placed elsewhere. Moreover, it is often plausible to interpret the user feedback as a preference (if any) *restricted to* the displayed alternatives.

In this paper, we consider instantiations of this problem in the multilabel and learning-to-rank settings. Learning proceeds in rounds: in round t, the algorithm receives an instance $\boldsymbol{x}_t$ and outputs an ordered subset $\hat{Y}_t$ of labels from a finite set of possible labels $[K] = \{1, 2, \ldots, K\}$. Restrictions might apply to the size of $\hat{Y}_t$ (due, e.g., to the number of available slots in the webpage, or to the specifics of the targeted user). The set $\hat{Y}_t$ corresponds to the aforementioned recommendations, and is intended to approximate the true set of preferences associated with $\boldsymbol{x}_t$. However, the latter set is never observed. In its stead, the algorithm receives $Y_t \cap \hat{Y}_t$, where $Y_t \subseteq [K]$ is a *noisy version* of the true set of user preferences on $\boldsymbol{x}_t$. When we are restricted to $|\hat{Y}_t| = 1$ for all t, this becomes a multiclass classification problem with bandit feedback—see below.

1.1 Related Work

This paper lies at the intersection between online learning with partial feedback and multilabel classification/ranking. Both fields include a substantial amount of work, so we can hardly do it justice here. In the sequel, we outline some of the main contributions in the two fields, with an emphasis on those we believe are the most related to this paper.

A well-known tool for facing the problem of partial feedback in online learning is to trade off exploration and exploitation through upper confidence bounds. This technique has been introduced by Lai and Robbins (1985), and can by now be considered a standard tool. In the so-called *bandit* setting with contextual information (sometimes called bandits with side information or bandits with covariates, e.g., Auer 2002; Dani et al. 2008; Filippi et al. 2010; Crammer and Gentile 2011; Krause and Ong 2011, and references therein) an online algorithm receives at each time step a *context* (typically, in the form of a feature vector $\boldsymbol{x}$) and is compelled to select an action (e.g., a label), whose goodness is quantified by a predefined loss function. Full information about the loss function (one that would perhaps allow to minimize the total loss over the contexts seen so far) is not available. The specifics of the interaction model determines which pieces of loss will be observed by the algorithm, e.g., the actual value of the loss on the chosen action, some information on more profitable directions on the action space, noisy versions thereof, etc. The overall goal is to compete against classes of functions that map contexts to (expected) losses in a regret sense, that is, to obtain *sublinear* cumulative regret bounds.

All these algorithms share the common need to somehow trade off an exploratory attitude for gathering loss information on unchosen directions of the context-action space, and an exploitatory attitude for choosing actions that are deemed best according to the available data. For instance, Auer (2002); Dani et al. (2008); Filippi et al. (2010); Abbasi-Yadkori et al. (2011) work in a finite action space where the mappings context-to-loss for each action are linear (or generalized linear, as Filippi et al., 2010's) functions of the features. They all obtain $T^{1/2}$-like regret bounds, where T is the time horizon. This is extended by Krause and Ong (2011), where the loss function is modeled as a sample from a Gaussian process over the joint context-action space. We are using a similar (generalized) linear modeling here. An earlier (but somehow more general) setting that models such mappings by VC-classes

is considered by Langford and Zhang (2008), where a $T^{2/3}$ regret bound has been proven under i.i.d. assumptions. Linear multiclass classification problems with bandit feedback are considered by, e.g., Kakade et al. (2008); Crammer and Gentile (2011); Hazan and Kale (2011), where either $T^{2/3}$ or $T^{1/2}$ or even logarithmic regret bounds are proven, depending on the noise model and the underlying loss functions.

All the above papers do not consider *structured* action spaces, where the learner is allowed to select *sets* of actions, which is more suitable to multilabel and ranking problems. Along these lines are the papers by Hazan and Kale (2009); Streeter et al. (2009); Kale et al. (2010); Slivkins et al. (2010); Shivaswamy and Joachims (2012); Amin et al. (2011). The general problem of online minimization of a submodular loss function under both full and bandit information without covariates is considered by Hazan and Kale (2009), achieving a regret $T^{2/3}$ in the bandit case. Streeter et al. (2009) consider the problem of online learning of assignments, where at each round an algorithm is requested to assign positions (e.g., rankings) to sets of items (e.g., ads) with given constraints on the set of items that can be placed in each position. Their problem shares similar motivations as ours but, again, the bandit version of their algorithm does not explicitly take side information into account, and leads to a $T^{2/3}$ regret bound. Another paper with similar goals but a different mathematical model is by Kale et al. (2010), where the aim is to learn a suitable ordering (an "ordered slate") of the available actions. Among other things, the authors prove a $T^{1/2}$ regret bound in the bandit setting with a multiplicative weight updating scheme. Yet, no contextual information is incorporated. Slivkins et al. (2010) motivate the ability of selecting sets of actions by a problem of diverse retrieval in large document collections which are meant to live in a general metric space. In contrast to our paper, that approach does not lead to strong regret guarantees for specific (e.g., smooth) loss functions. Shivaswamy and Joachims (2012) use a simple linear model for the hidden utility function of users interacting with a web system and providing partial feedback in any form that allows the system to make significant progress in learning this function (this is called an α-informative feedback by the authors). Under these assumptions, a regret bound of $T^{1/2}$ is again provided that depends on the degree of informativeness of the feedback, as measured by the progress made during the learning process. It is experimentally argued that this feedback is typically made available by a user that clicks on relevant URLs out of a list presented by a search engine. Despite the neatness of the argument, no formal effort is put into relating this information to the context information at hand or, more generally, to the way data are generated. The recent paper by Amin et al. (2011) investigates classes of graphical models for contextual bandit settings that afford richer interaction between contexts and actions leading again to a $T^{2/3}$ regret bound.

Finally, further interesting recent works that came to our attention at the time of writing this extended version of our conference paper (Gentile and Orabona, 2012) are the papers by Bartók and Szepesvári (2012), by Bartók (2013), and by Agarwal (2013). In Bartók and Szepesvári (2012), the authors provide sufficient conditions ("local observability") that insure rates of the form $T^{1/2}$ in partial monitoring games with side information. Partial monitoring is an attempt to formalize through a unifying language the partial information settings where the algorithm is observing only partial information about the loss of its action, in the form of some kind of feedback or "signal". The results presented by Bartók and Szepesvári (2012) do not seem to conveniently extend to the structured action space

setting we are interested in (or, if they do, we do not see it in the current version of their paper). Moreover, being very general in scope, that paper is missing a tight dependence of the regret bound on the number of available actions, which can be very large in structured action spaces. Progress in this directions has very recently been made by Bartók (2013), where the dependence on the number of actions is replaced by a quantity depending on the structure of the action space in the locally observable game. Yet, no side information is considered in that paper. The paper by Agarwal (2013) investigates multiclass selective sampling settings (similar to Cavallanti et al., 2011; Cesa-Bianchi et al., 2009; Dekel et al., 2012; Orabona and Cesa-Bianchi, 2011) with essentially the same generalized linear models as the ones we consider here. As such, that paper is close to ours only from a technical viewpoint.

The literature on multilabel learning and learning to rank is overwhelming. The wide attention this literature attracts is often motivated by its web-search-engine or recommender-system applications, and many of the papers are experimental in nature. Relevant references include the work by Tsoumakas et al. (2011); Furnkranz et al. (2008); Dembczynski et al. (2012), along with references therein. Moreover, when dealing with multilabel, the typical assumption is full supervision, an important concern being modeling correlations among classes. In contrast to that, the specific setting we are considering here need not face such a modeling issue (Dembczynski et al., 2012). The more recent work by Wang et al. (2012) reduces any online algorithm working on pairwise loss functions (like a ranking loss) to a batch algorithm with generalization bound guarantees. But, again, only fully supervised settings are considered. Other related references are the papers by Herbrich et al. (2000); Freund et al. (2003), where learning is by pairs of examples. Yet, these approaches need i.i.d. assumptions on the data, and typically deliver batch learning procedures. Finally, more recent efforts related to proving consistency of pairwise ranking methods are Clémençon et al. (2005); Cossock and Zhang (2006); Duchi et al. (2010); Buffoni et al. (2011); Lan et al. (2012) where, unlike this paper, multi-level user ratings are assumed to be available.

To summarize, whereas we are technically closer to the linear modeling approaches by Auer (2002); Dani et al. (2008); Dekel et al. (2012); Crammer and Gentile (2011); Filippi et al. (2010); Abbasi-Yadkori et al. (2011); Krause and Ong (2011); Bartók and Szepesvári (2012); Agarwal (2013), from a motivational standpoint we are perhaps closest to Streeter et al. (2009); Kale et al. (2010); Shivaswamy and Joachims (2012).

1.2 Our Results

We investigate the multilabel and learning-to-rank problems in a partial feedback scenario with contextual information, where we assume a probabilistic linear model over the labels, although the contexts can be chosen by an adaptive adversary. We consider two families of loss functions, one is a cost-sensitive multilabel loss that generalizes the standard Hamming loss in several respects, the other is a kind of (unnormalized) ranking loss. In both cases, the learning algorithm is maintaining a (generalized) linear predictor for the probability that a given label occurs, the ranking being produced by upper confidence-corrected estimated probabilities. In such settings, we prove $T^{1/2} \log T$ cumulative regret bounds, which are essentially optimal (up to log factors) in some cases. A distinguishing feature of our user feedback model is that, unlike previous papers (e.g., Hazan and Kale 2009; Streeter et al.

2009; Abbasi-Yadkori et al. 2011; Krause and Ong 2011), we are not assuming the algorithm is observing a noisy version of the risk function on the currently selected action. In fact, when a generalized linear model is adopted, the mapping context-to-risk turns out to be nonconvex in the parameter space. Furthermore, when operating on structured action spaces this more traditional form of bandit model does not seem appropriate to capture the typical user preference feedback. Our approach is based on having the loss decoupled from the label generating model, the user feedback being a noisy version of the gradient of a *surrogate* convex loss associated with the model itself. As a consequence, the algorithm is not directly dealing with the original loss when making exploration. In this sense, we are more similar to the multiclass bandit algorithm by Crammer and Gentile (2011). Yet, our work is a substantial departure from Crammer and Gentile's (2011) in that we lift their machinery to nontrivial structured action spaces, and we do so by means of generalized linear models. On one hand, these extensions pose several extra technical challenges; on the other, they provide additional modeling power and practical advantage.

Though the emphasis is on theoretical results, we also validate our algorithms on real-world multilabel data sets under several experimental conditions: data set size, label set size, loss functions, training mode and performance (online vs. batch), label generation model (linear vs. logistic). Under all such conditions, our algorithms are contrasted against the corresponding multilabel/ranking baselines that operate with full information, often showing (surprisingly enough) comparable prediction performance.

1.3 Structure of the Paper

The paper is organized as follows. In Section 2 we introduce our learning model, our first loss function, the label generation model, and some preliminary results and notation used throughout the rest of the paper. In Section 3 we describe our partial feedback algorithm working under the loss function introduced in Section 2, along with the associated regret analysis. In Section 4 we show that a very similar machinery applies to ranking with partial feedback, where the loss function is a kind of pairwise ranking loss (with partial feedback). Similar regret bounds are then presented that work under additional modeling restrictions. In Section 5 we provide our experimental comparison. Section 6 gives proof ideas and technical details. The paper is concluded with Section 7, where possible directions for future research are mentioned.

2. Model and Preliminaries

We consider a setting where the algorithm receives at time t the side information vector $\boldsymbol{x}_t \in \mathbb{R}^d$, is allowed to output a (possibly ordered) subset[1] $\hat{Y}_t \subseteq [K]$ of the set of possible labels, then the subset of labels $Y_t \subseteq [K]$ associated with $\boldsymbol{x}_t$ is generated, and the algorithm gets as feedback $\hat{Y}_t \cap Y_t$. The loss suffered by the algorithm may take into account several things: the *distance* between Y_t and $\hat{Y}_t$ (both viewed as sets), as well as the *cost* for playing $\hat{Y}_t$. The cost $c(\hat{Y}_t)$ associated with $\hat{Y}_t$ might be given by the sum of costs suffered on each class $i \in \hat{Y}_t$, where we possibly take into account the *order* in which i occurs within $\hat{Y}_t$ (viewed as an ordered list of labels). Specifically, given constant $a \in [0, 1]$ and costs

1. An ordered subset is like a list with *no repeated* items.

$c = \{c(i,s), i = 1, \ldots, s, s \in [K]\}$, such that $1 \geq c(1,s) \geq c(2,s) \geq \ldots c(s,s) \geq 0$, for all $s \in [K]$, we consider the loss function

$$\ell_{a,c}(Y_t, \hat{Y}_t) = a\,|Y_t \setminus \hat{Y}_t| + (1-a) \sum_{i \in \hat{Y}_t \setminus Y_t} c(j_i, |\hat{Y}_t|),$$

where j_i is the position of class i in $\hat{Y}_t$, and $c(j_i, \cdot)$ depends on $\hat{Y}_t$ only through its size $|\hat{Y}_t|$. In the above, the first term accounts for the false negative mistakes, hence there is no specific ordering of labels therein. The second term collects the loss contribution provided by all false positive classes, taking into account through the costs $c(j_i, |\hat{Y}_t|)$ the order in which labels occur in $\hat{Y}_t$. The constant a serves as weighting the relative importance of false positive vs. false negative mistakes.[2] As a specific example, suppose that $K = 10$, the costs $c(i,s)$ are given by $c(i,s) = (s - i + 1)/s, i = 1, \ldots, s$, the algorithm plays the ordered list $\hat{Y}_t = (4, 3, 6)$, but Y_t is the (unordered) set $\{1, 3, 8\}$. In this case, $|Y_t \setminus \hat{Y}_t| = 2$, and $\sum_{i \in \hat{Y}_t \setminus Y_t} c(j_i, |\hat{Y}_t|) = 3/3 + 1/3$, i.e., the cost for mistakenly playing class 4 in the top slot of $\hat{Y}_t$ is more damaging than mistakenly playing class 6 in the third slot. In the special case when all costs are unitary, there is no longer need to view $\hat{Y}_t$ as an ordered collection, and the above loss reduces to a standard Hamming-like loss between sets Y_t and $\hat{Y}_t$, i.e., $a\,|Y_t \setminus \hat{Y}_t| + (1-a)\,|\hat{Y}_t \setminus Y_t|$. Notice that the partial feedback $\hat{Y}_t \cap Y_t$ allows the algorithm to know which of the chosen classes in $\hat{Y}_t$ are good or bad (and to what extent, because of the selected ordering within $\hat{Y}_t$).

The reader should also observe the asymmetry between the label set $\hat{Y}_t$ produced by the algorithm and the true label set Y_t: the algorithm predicts an ordered set of labels, but the true set of labels is unordered. In fact, it is often the case in, e.g., recommender system practice, that the user feedback does not contain preference information in the form of an ordered set of items. Still, in such systems we would like to get back to the user with an appropriate ranking over the items.

Working with the above loss function makes the algorithm's output $\hat{Y}_t$ become a ranked list of classes, where ranking is *restricted* to the deemed relevant classes only. In this sense, the above problem can be seen as a partial information version of the multilabel ranking problem (see the work by Furnkranz et al., 2008, and references therein). In a standard multilabel ranking problem a classifier has to provide for any given instance $\boldsymbol{x}_t$, both a separation between relevant and irrelevant classes and a ranking of the classes within the two sets (or, perhaps, over the whole set of classes, as long as ranking is consistent with the relevance separation). In our setting, instead, ranking applies to the selected classes only, but the information gathered by the algorithm while training is partial. That is, only a relevance feedback among the selected classes is observed (the set $Y_t \cap \hat{Y}_t$), but no supervised ranking information (e.g., in the form of pairwise preferences) is provided to the algorithm within this set. Alternatively, we can think of a ranking framework where restrictions on the size of $\hat{Y}_t$ are set by an exogenous (and possibly time-varying) parameter of the problem, and the algorithm is required to provide a ranking complying with these restrictions. In this sense, an alternative interpretation of the ranking-sensitive term $\sum_{i \in \hat{Y}_t \setminus Y_t} c(j_i, |\hat{Y}_t|)$ in $\ell_{a,c}(Y_t, \hat{Y}_t)$ is a Discounted Cumulative Gain (DCG) difference between the optimal ranking

2. Parameter a is not redundant here, since the costs $c(i,s)$ have been normalized to [0,1].

(i.e., the one sorting the $|Y_t|$ classes in Y_t according to decreasing value of $c(i, |\hat{Y}_t|)$) and the actual ranking contained in $\hat{Y}_t$, the discounting function being just the coefficients $c(i, |\hat{Y}_t|), i = 1, \dots |\hat{Y}_t|$. DCG is a standard metric for measuring the effectiveness of Web search engine algorithms (e.g., Jarvelin and Kekalainen, 2002).

Another important concern we would like to address with our loss function $\ell_{a,c}$ is to avoid combinatorial explosions due to the exponential number of possible choices for $\hat{Y}_t$. As we shall see below, this is guaranteed by the chosen structure for costs $c(i, s)$. Another loss function providing similar guarantees (though with additional modeling restrictions) is the (pairwise) ranking loss considered in Section 4, where more on the connection to the ranking setting with partial feedback is given.

The problem arises as to which noise model we should adopt so as to encompass significant real-world settings while at the same time affording *efficient implementation* of the resulting algorithms. For any subset $Y_t \subseteq [K]$, we let $(y_{1,t}, \dots, y_{K,t}) \in \{0,1\}^K$ be the corresponding indicator vector. Then it is easy to see that

$$\begin{aligned}\ell_{a,c}(Y_t, \hat{Y}_t) &= a \sum_{i \notin \hat{Y}_t} y_{i,t} + (1-a) \sum_{i \in \hat{Y}_t} c(j_i, |\hat{Y}_t|)\,(1 - y_{i,t}) \\ &= a \sum_{i=1}^{K} y_{i,t} + (1-a) \sum_{i \in \hat{Y}_t} \left(c(j_i, |\hat{Y}_t|) - \left(\tfrac{a}{1-a} + c(j_i, |\hat{Y}_t|) \right) y_{i,t} \right).\end{aligned}$$

Moreover, because the first sum does not depend on $\hat{Y}_t$, for the sake of optimizing over $\hat{Y}_t$ (but also for the sake of defining the regret R_T—see below) we can equivalently define

$$\ell_{a,c}(Y_t, \hat{Y}_t) = (1-a) \sum_{i \in \hat{Y}_t} \left(c(j_i, |\hat{Y}_t|) - \left(\tfrac{a}{1-a} + c(j_i, |\hat{Y}_t|) \right) y_{i,t} \right). \tag{1}$$

Note that the algorithm can evaluate the value of this loss, using the feedback received. Let $\mathbb{P}_t(\cdot)$ be a shorthand for the conditional probability $\mathbb{P}(\cdot \,|\, \boldsymbol{x}_t)$, where the side information vector $\boldsymbol{x}_t$ can in principle be generated by an adaptive adversary as a function of the past. Then

$$\mathbb{P}_t(y_{1,t}, \dots, y_{K,t}) = \mathbb{P}(y_{1,t}, \dots, y_{K,t} \,|\, \boldsymbol{x}_t).$$

We will assume that the marginals $\mathbb{P}_t(y_{i,t} = 1)$ satisfy[3]

$$\mathbb{P}_t(y_{i,t} = 1) = \frac{g(-\boldsymbol{u}_i^\top \boldsymbol{x}_t)}{g(\boldsymbol{u}_i^\top \boldsymbol{x}_t) + g(-\boldsymbol{u}_i^\top \boldsymbol{x}_t)}, \qquad i = 1, \dots, K, \tag{2}$$

for some K vectors $\boldsymbol{u}_1, \dots, \boldsymbol{u}_K \in \mathcal{R}^d$, and a (known) function $g : D \subseteq \mathcal{R} \to \mathcal{R}^+$, that is the negative derivative of a suitable convex and nonincreasing function. The model is well defined if $\boldsymbol{u}_i^\top \boldsymbol{x} \in D$ for all i and all $\boldsymbol{x} \in \mathcal{R}^d$ chosen by the adversary. We assume for the sake of simplicity that $||\boldsymbol{x}_t|| = 1$ for all t. Notice that here the variables $y_{i,t}$ *need not* be conditionally independent. We are only defining a family of allowed joint distributions

3. The reader familiar with generalized linear models will recognize the derivative of the function $p(\Delta) = \frac{g(-\Delta)}{g(\Delta)+g(-\Delta)}$ as the (inverse) link function of the associated canonical exponential family of distributions (McCullagh and Nelder, 1989).

$\mathbb{P}_t(y_{1,t},\dots,y_{K,t})$ through the properties of their marginals $\mathbb{P}_t(y_{i,t})$. A classical result in the theory of copulas (Sklar, 1959) makes one derive all allowed joint distributions starting from the corresponding one-dimensional marginals. It is also important to point out the arbitrary dependence of $\boldsymbol{x}_t$ on the past, since the typical scenarios we are modeling here (human interaction) are producing data sequences which are nonstationary in nature, implying that traditional statistical inference methods (e.g., empirical risk minimization) should be used cautiously.

Our algorithm will be based on the loss function L, which is such that the function g above is equal to the negative derivative of L. For instance, if L is the square loss $L(\Delta) = (1-\Delta)^2/2$, then $g(\Delta) = 1-\Delta$, resulting in $\mathbb{P}_t(y_{i,t}=1) = (1+\boldsymbol{u}_i^\top \boldsymbol{x}_t)/2$, under the assumption $D=[-1,1]$. If L is the logistic loss $L(\Delta) = \ln(1+e^{-\Delta})$, then $g(\Delta) = \frac{1}{e^{\Delta}+1}$, and $\mathbb{P}_t(y_{i,t}=1) = e^{\boldsymbol{u}_i^\top \boldsymbol{x}_t}/(e^{\boldsymbol{u}_i^\top \boldsymbol{x}_t}+1)$, with domain $D=\mathcal{R}$. Observe that in both cases $\mathbb{P}_t(y_{i,t}=1)$ is an increasing function of $\boldsymbol{u}_i^\top \boldsymbol{x}_t$. This will be true in general.

Set for brevity $\Delta_{i,t} = \boldsymbol{u}_i^\top \boldsymbol{x}_t$. Taking into account (1), this model allows us to write the (conditional) expected loss of the algorithm playing $\hat{Y}_t$ as

$$\mathbb{E}_t[\ell_{a,c}(Y_t,\hat{Y}_t)] = (1-a)\sum_{i\in\hat{Y}_t}\left(c(j_i,|\hat{Y}_t|) - \left(\tfrac{a}{1-a} + c(j_i,|\hat{Y}_t|)\right)\,p_{i,t}\right), \tag{3}$$

where we introduced the shorthands

$$p_{i,t} = p(\Delta_{i,t}), \qquad\qquad p(\Delta) = \frac{g(-\Delta)}{g(\Delta)+g(-\Delta)},$$

and the expectation $\mathbb{E}_t$ in (3) is w.r.t. the generation of labels Y_t, conditioned on both $\boldsymbol{x}_t$, and all previous $\boldsymbol{x}$ and Y.

A key aspect of this formalization is that the Bayes optimal ordered subset

$$Y_t^* = \mathrm{argmin}_{Y=(j_1,j_2,\dots,j_{|Y|})\subseteq[K]}\mathbb{E}_t[\ell_{a,c}(Y_t,Y)]$$

can be computed efficiently when knowing $\Delta_{1,t},\dots,\Delta_{K,t}$. This is handled by the next lemma. In words, this lemma says that, in order to minimize (3), it suffices to try out all possible sizes $s=0,1,\dots,K$ for Y_t^* and, for each such value, determine the sequence $Y_{s,t}^*$ that minimizes (3) over all sequences of size s. In turn, $Y_{s,t}^*$ can be computed just by sorting classes $i\in[K]$ in decreasing order of $p_{i,t}$, sequence $Y_{s,t}^*$ being given by the first s classes in this sorted list.

Lemma 1 *With the notation introduced so far, let $p_{i_1,t}\geq p_{i_2,t}\geq\dots p_{i_K,t}$ be the sequence of $p_{i,t}$ sorted in nonincreasing order. Then we have that*

$$Y_t^* = \mathrm{argmin}_{s=0,1,\dots K}\mathbb{E}_t[\ell_{a,c}(Y_t,Y_{s,t}^*)],$$

where $Y_{s,t}^* = (i_1,i_2,\dots,i_s)$, and $Y_{0,t}^*=\emptyset$.

Proof First observe that, for any given size s, the sequence $Y_{s,t}^*$ must contain the s top-ranked classes in the sorted order of $p_{i,t}$. This is because, for any candidate sequence $Y_s = \{j_1,j_2,\dots,j_s\}$, we have $\mathbb{E}_t[\ell_{a,c}(Y_t^*,Y_s)] = (1-a)\sum_{i\in Y_s}\left(c(j_i,s) - \left(\frac{a}{1-a}+c(j_i,s)\right)\,p_{i,t}\right)$. If

there exists $i \in Y_s$ which is not among the s-top ranked ones, then we could replace class i in position j_i within Y_s with class $k \notin Y_s$ such that $p_{k,t} > p_{i,t}$ obtaining a smaller loss.

Next, we show that the optimal ordering within $Y^*_{s,t}$ is precisely ruled by the nonincreasing order of $p_{i,t}$. By the sake of contradiction, assume there are i and k in $Y^*_{s,t}$ such that i precedes k in $Y^*_{s,t}$ but $p_{k,t} > p_{i,t}$. Specifically, let i be in position j_1 and k be in position j_2 with $j_1 < j_2$ and such that $c(j_1, s) > c(j_2, s)$. Then, disregarding the common $(1-a)$-factor, switching the two classes within $Y^*_{s,t}$ yields an expected loss difference of

$$\begin{aligned}
&c(j_1, s) - \left(\tfrac{a}{1-a} + c(j_1, s)\right) p_{i,t} + c(j_2, s) - \left(\tfrac{a}{1-a} + c(j_2, s)\right) p_{k,t} \\
&- \left(c(j_1, s) - \left(\tfrac{a}{1-a} + c(j_1, s)\right) p_{k,t}\right) - \left(c(j_2, s) - \left(\tfrac{a}{1-a} + c(j_2, s)\right) p_{i,t}\right) \\
&= (p_{k,t} - p_{i,t}) \left(c(j_1, s) - c(j_2, s)\right) > 0,
\end{aligned}$$

since $p_{k,t} > p_{i,t}$ and $c(j_1, s) > c(j_2, s)$. Hence switching would get a smaller loss which leads as a consequence to $Y^*_{s,t} = (i_1, i_2, \ldots, i_s)$. ■

Notice the way costs $c(i, s)$ influence the Bayes optimal computation. We see from (3) that placing class i within $\hat{Y}_t$ in position j_i is beneficial (i.e., it leads to a reduction of loss) if and only if $p_{i,t} > c(j_i, |\hat{Y}_t|)/(\frac{a}{1-a} + c(j_i, |\hat{Y}_t|))$. Hence, the higher is the slot i_j in $\hat{Y}_t$ the larger should be $p_{i,t}$ in order for this inclusion to be convenient.[4]

It is Y^*_t above that we interpret as the true set of user preferences on $\boldsymbol{x}_t$. We would like to compete against Y^*_t in a cumulative regret sense, i.e., we would like to bound

$$R_T = \sum_{t=1}^{T} \mathbb{E}_t[\ell_{a,c}(Y_t, \hat{Y}_t)] - \mathbb{E}_t[\ell_{a,c}(Y_t, Y^*_t)]$$

with high probability.

We use a similar but largely more general analysis than Crammer and Gentile (2011)'s to devise an online second-order descent algorithm whose updating rule makes the comparison vector $U = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K) \in \mathcal{R}^{dK}$ defined through (2) be Bayes optimal w.r.t. a surrogate convex loss $L(\cdot)$ such that $g(\Delta) = -L'(\Delta)$. Observe that the expected loss function defined in (3) is, generally speaking, nonconvex in the margins $\Delta_{i,t}$ (consider, for instance the logistic case $g(\Delta) = \frac{1}{e^{\Delta}+1}$). Thus, we cannot directly minimize this expected loss.

3. Algorithm and Regret Bounds

In Figure 2 is our bandit algorithm for (ordered) multiple labels. In order to acquaint the reader with this algorithm, a simplified version of it is first presented (Figure 1) which applies to the linear model $p(\Delta) = \frac{1+\Delta}{2}$, $g(\Delta) = 1 - \Delta$, under the simplifying assumption $||\boldsymbol{u}_i|| \leq 1$, for $i \in [K]$.

4. Notice that this depends on the actual size of $\hat{Y}_t$, so we cannot decompose this problem into K independent problems. The decomposition does occur if the costs $c(i, s)$ are constants, independent of i and s, the criterion for inclusion becoming $p_{i,t} \geq \theta$, for some constant threshold θ.

Parameters:

- Loss parameters $a \in [0,1]$, and cost values $c(i,s)$;
- Confidence level $\delta \in [0,1]$.

Initialization: $A_{i,0} = I \in \mathcal{R}^{d\times d}$, $i = 1,\dots,K$, $\boldsymbol{w}_{i,1} = 0 \in \mathcal{R}^d$, $i = 1,\dots,K$;

For $t = 1, 2 \dots, T$:

1. Get instance $\boldsymbol{x}_t \in \mathcal{R}^d$: $||\boldsymbol{x}_t|| = 1$;
2. For $i \in [K]$, set $\widehat{\Delta}'_{i,t} = \boldsymbol{x}_t^\top \boldsymbol{w}'_{i,t}$, where
$$\boldsymbol{w}'_{i,t} = \begin{cases} \boldsymbol{w}_{i,t} & \text{if } \boldsymbol{w}_{i,t}^\top \boldsymbol{x}_t \in [-1,1], \\ \boldsymbol{w}_{i,t} - \left(\frac{\boldsymbol{w}_{i,t}^\top \boldsymbol{x}_t - 1}{\boldsymbol{x}_t^\top A_{i,t-1}^{-1} \boldsymbol{x}_t}\right) A_{i,t-1}^{-1} \boldsymbol{x}_t & \text{if } \boldsymbol{w}_{i,t}^\top \boldsymbol{x}_t > 1, \\ \boldsymbol{w}_{i,t} - \left(\frac{\boldsymbol{w}_{i,t}^\top \boldsymbol{x}_t + 1}{\boldsymbol{x}_t^\top A_{i,t-1}^{-1} \boldsymbol{x}_t}\right) A_{i,t-1}^{-1} \boldsymbol{x}_t & \text{if } \boldsymbol{w}_{i,t}^\top \boldsymbol{x}_t < -1; \end{cases}$$
3. Output
$$\hat{Y}_t = \operatorname*{argmin}_{Y=(j_1,j_2,\dots j_{|Y|})\subseteq[K]} \left(\sum_{i\in Y} \left(c(j_i,|Y|) - \left(\tfrac{a}{1-a} + c(j_i,|Y|)\right) \widehat{p}_{i,t} \right) \right),$$
where
$$\widehat{p}_{i,t} = \frac{1 + [\widehat{\Delta}'_{i,t} + \epsilon_{i,t}]_{[-1,1]}}{2},$$
$$\epsilon_{i,t}^2 = \boldsymbol{x}_t^\top A_{i,t-1}^{-1} \boldsymbol{x}_t \left(1 + 4\,d\,\ln\left(1 + \frac{t-1}{d}\right) + 48\,\ln\frac{K(t+4)}{\delta} \right) ;$$
4. Get feedback $Y_t \cap \hat{Y}_t$;
5. For $i \in [K]$, update:
$$A_{i,t} = A_{i,t-1} + |s_{i,t}| \boldsymbol{x}_t \boldsymbol{x}_t^\top, \qquad \boldsymbol{w}_{i,t+1} = \boldsymbol{w}'_{i,t} - A_{i,t}^{-1} \nabla_{i,t},$$
where
$$s_{i,t} = \begin{cases} 1 & \text{if } i \in Y_t \cap \hat{Y}_t, \\ -1 & \text{if } i \in \hat{Y}_t \setminus Y_t = \hat{Y}_t \setminus (Y_t \cap \hat{Y}_t), \\ 0 & \text{otherwise;} \end{cases}$$
and
$$\nabla_{i,t} = (s_{i,t}\,\widehat{\Delta}'_{i,t} - 1)\, s_{i,t}\, \boldsymbol{x}_t.$$

Figure 1: The partial feedback algorithm in the (ordered) multiple label setting—the linear model case.

Both algorithms are based on replacing the unknown model vectors $\boldsymbol{u}_1, \dots, \boldsymbol{u}_K$ with prototype vectors $\boldsymbol{w}'_{1,t}, \dots, \boldsymbol{w}'_{K,t}$, being $\boldsymbol{w}'_{i,t}$ the time-t approximation to $\boldsymbol{u}_i$, satisfying sim-

Parameters:

- Loss parameters $a \in [0,1]$, and cost values $c(i,s)$;
- Interval $D = [-R, R]$, function $g : D \to \mathcal{R}$;
- Confidence level $\delta \in [0,1]$, and norm upper bound $U > 0$.

Initialization: $A_{i,0} = I \in \mathcal{R}^{d\times d}$, $i = 1,\ldots,K$, $\boldsymbol{w}_{i,1} = 0 \in \mathcal{R}^d$, $i = 1,\ldots,K$;

For $t = 1, 2 \ldots, T$:

1. Get instance $\boldsymbol{x}_t \in \mathcal{R}^d$: $||\boldsymbol{x}_t|| = 1$;
2. For $i \in [K]$, set $\widehat{\Delta}'_{i,t} = \boldsymbol{x}_t^\top \boldsymbol{w}'_{i,t}$, where

$$\boldsymbol{w}'_{i,t} = \begin{cases} \boldsymbol{w}_{i,t} & \text{if } \boldsymbol{w}_{i,t}^\top \boldsymbol{x}_t \in [-R, R], \\ \boldsymbol{w}_{i,t} - \left(\frac{\boldsymbol{w}_{i,t}^\top \boldsymbol{x}_t - R}{\boldsymbol{x}_t^\top A_{i,t-1}^{-1} \boldsymbol{x}_t}\right) A_{i,t-1}^{-1} \boldsymbol{x}_t & \text{if } \boldsymbol{w}_{i,t}^\top \boldsymbol{x}_t > R, \\ \boldsymbol{w}_{i,t} - \left(\frac{\boldsymbol{w}_{i,t}^\top \boldsymbol{x}_t + R}{\boldsymbol{x}_t^\top A_{i,t-1}^{-1} \boldsymbol{x}_t}\right) A_{i,t-1}^{-1} \boldsymbol{x}_t & \text{if } \boldsymbol{w}_{i,t}^\top \boldsymbol{x}_t < -R; \end{cases}$$

3. Output

$$\hat{Y}_t = \operatorname*{argmin}_{Y=(j_1,j_2,\ldots j_{|Y|})\subseteq[K]} \left(\sum_{i\in Y} \left(c(j_i, |Y|) - \left(\tfrac{a}{1-a} + c(j_i, |Y|)\right) \widehat{p}_{i,t} \right) \right),$$

where

$$\widehat{p}_{i,t} = p\left([\widehat{\Delta}'_{i,t} + \epsilon_{i,t}]_D\right) = \frac{g\left(-[\widehat{\Delta}'_{i,t} + \epsilon_{i,t}]_D\right)}{g\left([\widehat{\Delta}'_{i,t} + \epsilon_{i,t}]_D\right) + g\left(-[\widehat{\Delta}'_{i,t} + \epsilon_{i,t}]_D\right)},$$

$$\epsilon_{i,t}^2 = \boldsymbol{x}_t^\top A_{i,t-1}^{-1} \boldsymbol{x}_t \left(U^2 + \frac{d\, c'_L}{(c''_L)^2} \ln\left(1 + \frac{t-1}{d}\right) + \frac{12}{c''_L}\left(\frac{c'_L}{c''_L} + 3L(-R)\right) \ln\frac{K(t+4)}{\delta} \right) ;$$

4. Get feedback $Y_t \cap \hat{Y}_t$;
5. For $i \in [K]$, update:

$$A_{i,t} = A_{i,t-1} + |s_{i,t}| \boldsymbol{x}_t \boldsymbol{x}_t^\top, \qquad \boldsymbol{w}_{i,t+1} = \boldsymbol{w}'_{i,t} - \frac{1}{c''_L} A_{i,t}^{-1} \nabla_{i,t},$$

where

$$s_{i,t} = \begin{cases} 1 & \text{if } i \in Y_t \cap \hat{Y}_t, \\ -1 & \text{if } i \in \hat{Y}_t \setminus Y_t = \hat{Y}_t \setminus (Y_t \cap \hat{Y}_t), \\ 0 & \text{otherwise;} \end{cases}$$

and

$$\nabla_{i,t} = \nabla_{\boldsymbol{w}} L(s_{i,t}\, \boldsymbol{w}^\top \boldsymbol{x}_t)|_{\boldsymbol{w}=\boldsymbol{w}'_{i,t}} = -g(s_{i,t}\, \widehat{\Delta}'_{i,t})\, s_{i,t}\, \boldsymbol{x}_t.$$

Figure 2: The partial feedback algorithm in the (ordered) multiple label setting—the *generalized* linear model case.

ilar constraints we set for the $\boldsymbol{u}_i$ vectors. For the sake of brevity, we let $\widehat{\Delta}'_{i,t} = \boldsymbol{x}_t^\top \boldsymbol{w}'_{i,t}$, and $\Delta_{i,t} = \boldsymbol{u}_i^\top \boldsymbol{x}_t$, $i \in [K]$.

The algorithms use $\widehat{\Delta}'_{i,t}$ as proxies for the underlying $\Delta_{i,t}$ according to the (upper confidence) approximation scheme $\Delta_{i,t} \approx [\widehat{\Delta}'_{i,t} + \epsilon_{i,t}]_D$, where $\epsilon_{i,t} \geq 0$ is a suitable upper-confidence level for class i at time t, and $[\cdot]_D$ denotes the clipping-to-D operation: if $D = [-R, R]$, then

$$[x]_D = \begin{cases} R & \text{if } x > R \\ x & \text{if } -R \leq x \leq R \\ -R & \text{if } x < -R. \end{cases}$$

The algorithms' prediction at time t has the same form as the computation of the Bayes optimal sequence Y_t^*, where we replace the true (and unknown) $p_{i,t} = p(\Delta_{i,t})$ with the corresponding upper confidence proxy

$$\widehat{p}_{i,t} = p([\widehat{\Delta}'_{i,t} + \epsilon_{i,t}]_D),$$

being

$$\hat{Y}_t = \operatorname*{argmin}_{Y=(j_1,j_2,\dots j_{|Y|}) \subseteq [K]} \left(\sum_{i \in Y} \left(c(j_i, |Y|) - \left(\tfrac{a}{1-a} + c(j_i, |Y|) \right) \widehat{p}_{i,t} \right) \right).$$

Computing $\hat{Y}_t$ above can be done by mimicking the computation of the Bayes optimal ordered subset Y_t^* (just replace $p_{i,t}$ by $\widehat{p}_{i,t}$). From a computational viewpoint, this essentially amounts to sorting classes $i \in [K]$ in decreasing value of $\widehat{p}_{i,t}$, i.e., order of $K \log K$ running time per prediction. Thus the algorithms are producing a ranked list of relevant classes based on upper-confidence-corrected scores $\widehat{p}_{i,t}$. Class i is deemed relevant and ranked high among the relevant ones when either $\widehat{\Delta}'_{i,t}$ is a good approximation to $\Delta_{i,t}$ and $p_{i,t}$ is large, or when the algorithms are not very confident on their own approximation about i (that is, the upper confidence level $\epsilon_{i,t}$ is large).

Specifically, the algorithm in Figure 1 receives in input the loss parameters a and $c(i, s)$, and the desired confidence level δ, and maintains both K positive definite matrices $A_{i,t}$ of dimension d (initially set to the $d \times d$ identity matrix), and K weight vectors $\boldsymbol{w}_{i,t} \in \mathcal{R}^d$ (initially set to the zero vector). At each time step t, upon receiving the d-dimensional instance vector $\boldsymbol{x}_t$ the algorithm uses the weight vectors $\boldsymbol{w}_{i,t}$ to compute the prediction vectors $\boldsymbol{w}'_{i,t}$. These vectors can easily be seen as the result of projecting $\boldsymbol{w}_{i,t}$ onto interval $[-1, 1]$ w.r.t. the distance function $d_{i,t-1}$, i.e.,

$$\boldsymbol{w}'_{i,t} = \operatorname*{argmin}_{\boldsymbol{w} \in \mathcal{R}^d \,:\, \boldsymbol{w}^\top \boldsymbol{x}_t \in [-1,1]} d_{i,t-1}(\boldsymbol{w}, \boldsymbol{w}_{i,t}), \qquad i \in [K],$$

where

$$d_{i,t-1}(\boldsymbol{u}, \boldsymbol{w}) = (\boldsymbol{u} - \boldsymbol{w})^\top A_{i,t-1} (\boldsymbol{u} - \boldsymbol{w}).$$

Vectors $\boldsymbol{w}'_{i,t}$ are then used to produce prediction values $\widehat{\Delta}'_{i,t}$ involved in the upper-confidence calculation of the predicted ordered subset $\hat{Y}_t \subseteq [K]$. Next, the feedback $Y_t \cap \hat{Y}_t$ is observed, and the algorithm in Figure 1 promotes all classes $i \in Y_t \cap \hat{Y}_t$ (sign $s_{i,t} = 1$), demotes all

classes $i \in \hat{Y}_t \setminus Y_t$ (sign $s_{i,t} = -1$), and leaves all remaining classes $i \notin \hat{Y}_t$ unchanged (sign $s_{i,t} = 0$). Promotion of class i on $\boldsymbol{x}_t$ implies that if the new vector $\boldsymbol{x}_{t+1}$ is close to $\boldsymbol{x}_t$ then i will be ranked higher on $\boldsymbol{x}_{t+1}$. The update $\boldsymbol{w}'_{i,t} \to \boldsymbol{w}_{i,t+1}$ is based on the gradients $\nabla_{i,t}$ of the square loss function $L(\Delta) = (1-\Delta)^2/2$. On the other hand, the update $A_{i,t-1} \to A_{i,t}$ uses the rank-one matrix[5] $\boldsymbol{x}_t \boldsymbol{x}_t^\top$. The matrix $A_{i,t-1}$ is used to calculate the upper confidence level on each prediction. Matrix $A_{i,t-1}$ is the empirical covariance matrix of the samples on which we received some feedback, either positive ($s_{i,t} = 1$) or negative ($s_{i,t} = -1$), and is used in the expression for the confidence $\epsilon_{i,t}^2$ involving the quadratic form $\boldsymbol{x}_t^\top A_{i,t-1}^{-1} \boldsymbol{x}_t$. Notice that $\epsilon_{i,t}^2$ will be small when the current sample $\boldsymbol{x}_t$ is in the span of the previous samples on which we received feedback, and will be large otherwise. In both the update of $\boldsymbol{w}'_{i,t}$ and the one involving $A_{i,t-1}$, the reader should observe the role played by the signs $s_{i,t}$.

The algorithm contained in Figure 2 is just a more general version of the one in Figure 1, where we also receive in input the specifics of the generalized linear model through the model function $g(\cdot)$ and the associated margin domain $D = [-R, R]$, and the norm upper bound U, such that $\|\boldsymbol{u}_i\| \leq U$ for all $i \in [K]$. The update $\boldsymbol{w}'_{i,t} \to \boldsymbol{w}_{i,t+1}$ in Figure 2 is based on the gradients $\nabla_{i,t}$ of a loss function $L(\cdot)$ satisfying $L'(\Delta) = -g(\Delta)$. On the other hand, the update $A_{i,t-1} \to A_{i,t}$ uses again the rank-one matrix $\boldsymbol{x}_t \boldsymbol{x}_t^\top$. The constants c'_L and c''_L occurring in the expression for $\epsilon_{i,t}^2$ in Figure 2 are related to smoothness properties of $L(\cdot)$. In particular, $\epsilon_{i,t}^2$ in Figure 1 is obtained from $\epsilon_{i,t}^2$ in Figure 2 by setting $R = 1$, $L(-R) = L(-1) = 0$, along with $c'_L = 4$ and $c''_L = 1$, as explained in the next theorem.[6]

Theorem 2 *Let $L : D = [-R, R] \subseteq \mathcal{R} \to \mathcal{R}^+$ be a $C^2(D)$ convex and nonincreasing function of its argument, $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K) \in \mathcal{R}^{dK}$ be defined in (2) with $g(\Delta) = -L'(\Delta)$ for all $\Delta \in D$, and such that $\|\boldsymbol{u}_i\| \leq U$ for all $i \in [K]$. Assume there are positive constants c_L, c'_L and c''_L such that*

i. $\frac{L'(\Delta)\, L''(-\Delta) + L''(\Delta)\, L'(-\Delta)}{(L'(\Delta) + L'(-\Delta))^2} \geq -c_L$,

ii. $(L'(\Delta))^2 \leq c'_L$,

iii. $L''(\Delta) \geq c''_L$

simultaneously hold for all $\Delta \in D$. Then the cumulative regret R_T of the algorithm in Figure 2 satisfies, with probability at least $1 - \delta$,

$$R_T = O\left((1-a)\, c_L\, K \sqrt{T\, C\, d\, \ln\left(1 + \frac{T}{d}\right)} \right),$$

where

$$C = O\left(U^2 + \frac{d\, c'_L}{(c''_L)^2}\, \ln\left(1 + \frac{T}{d}\right) + \left(\frac{c'_L}{(c''_L)^2} + \frac{L(-R)}{c''_L} \right) \ln \frac{KT}{\delta} \right).$$

5. The rank-one update is based on $\boldsymbol{x}_t \boldsymbol{x}_t^\top$ rather than $\nabla_{i,t} \nabla_{i,t}^\top$, as in , e.g., the paper by Hazan et al. (2007). This is due to technical reasons that will be made clear in Section 6. This feature tells this algorithm slightly apart from the Online Newton step algorithm (Hazan et al., 2007), which is the starting point of our analysis. The very same comment applies to the algorithm in Figure 2.

6. The proof is given in Section 6.

It is easy to see that when $L(\cdot)$ is the square loss $L(\Delta) = (1-\Delta)^2/2$ and $D = [-1, 1]$, we have $c_L = 1/2$, $c'_L = 4$ and $c''_L = 1$; when $L(\cdot)$ is the logistic loss $L(\Delta) = \ln(1 + e^{-\Delta})$ and $D = [-R, R]$, we have $c_L = 1/4$, $c'_L \le 1$ and $c''_L = \frac{1}{2(1+\cosh(R))}$, where $\cosh(x) = \frac{e^x + e^{-x}}{2}$.

The following remarks are in order at this point.

Remark 3 *A drawback of Theorem 2 is that, in order to properly set the upper confidence levels $\epsilon_{i,t}$, we assume prior knowledge of the norm upper bound U. Because this information is often unavailable, we present here a simple modification to the algorithm that copes with this limitation, similar to the one proposed in Orabona and Cesa-Bianchi (2011). We change the definition of $\epsilon^2_{i,t}$ in Figure 2 to*

$$\epsilon^2_{i,t} = \max\left\{ \boldsymbol{x}^\top A^{-1}_{i,t-1} \boldsymbol{x} \left(\frac{2\,d\,c'_L}{(c''_L)^2} \ln\left(1 + \frac{t-1}{d}\right) + \frac{12}{c''_L} \left(\frac{c'_L}{c''_L} + 3L(-R) \right) \ln \frac{K(t+4)}{\delta} \right), 4\,R^2 \right\},$$

that is, we substitute U^2 by $\frac{d\,c'_L}{(c''_L)^2} \ln\left(1 + \frac{t-1}{d}\right)$, and cap the maximal value of $\epsilon^2_{i,t}$ to $4\,R^2$. This immediately leads to the following result.[7]

Theorem 4 *With the same assumptions and notation as in Theorem 2, if we replace $\epsilon^2_{i,t}$ as explained above we have that, with probability at least $1-\delta$, R_T satisfies*

$$R_T = O\left((1-a)\,c_L\,K \sqrt{T\,C\,d \ln\left(1 + \frac{T}{d}\right)} + (1-a)\,c_L\,K\,R\,d \left(\exp\left(\frac{(c''_L)^2\,U^2}{c'_L\,d} \right) - 1 \right) \right).$$

Remark 5 *From a computational standpoint, the most demanding operation in Figure 2 is computing the upper confidence levels $\epsilon_{i,t}$ involving the inverse matrices $A^{-1}_{i,t-1}$, $i \in [K]$. Note that the matrices can be safely inverted because they are full rank, being initialized with identity matrices. The matrix inversion can be done incrementally in $\mathcal{O}(K\,d^2)$ time per round. This can be hardly practical if both d and K are large. In practice (as explained, e.g., by Crammer and Gentile, 2011), one can use an approximated version of the algorithm which maintains* diagonal *matrices $A_{i,t}$ instead of full ones. All the steps remain the same except Step 5 of Algorithm 2 where one defines the rth diagonal element of matrix $A_{i,t}$ as $(A_{i,t})_{r,r} = (A_{i,t-1})_{r,r} + x^2_{r,t}$, being $\boldsymbol{x}_t = (x_{1,t}, x_{2,t}, \ldots, x_{r,t}, \ldots, x_{K,t})^\top$. The resulting running time per round (including prediction and update) becomes $\mathcal{O}(dK + K \log K)$. In fact, when a limitation on the size of $\hat{Y}_t$ is given, the running time may be further reduced, see Remark 8.*

4. On Ranking with Partial Feedback

As Lemma 1 points out, when the cost values $c(i, s)$ in the loss function $\ell_{a,c}$ are *strictly* decreasing i.e., $c(1, s) > c(2, s) > \ldots > c(s, s)$, for all $s \in [K]$, then the Bayes optimal ordered sequence Y^*_t on $\boldsymbol{x}_t$ is unique can be obtained by sorting classes in decreasing values of $p_{i,t}$, and then decide on a cutoff point[8] induced by the loss parameters, so as to tell relevant classes apart from irrelevant ones. In turn, because $p(\Delta) = \frac{g(-\Delta)}{g(\Delta)+g(-\Delta)}$ is increasing in Δ,

7. The proof is deferred to Section 6.
8. This is called the *zero point* by Furnkranz et al. (2008).

this ordering corresponds to sorting classes in decreasing values of $\Delta_{i,t}$. Now, if parameter a in $\ell_{a,c}$ is very close[9] to 1, then $|Y_t^*| = K$, and the algorithm itself will produce ordered subsets $\hat{Y}_t$ such that $|\hat{Y}_t| = K$. Moreover, it does so by receiving *full* feedback on the relevant classes at time t (since $Y_t \cap \hat{Y}_t = Y_t$). As is customary (e.g., Dembczynski et al. 2012), one can view any multilabel assignment $Y = (y_1, \ldots, y_K) \in \{0,1\}^K$ as a ranking among the K classes in the most natural way: i precedes j if and only if $y_i > y_j$. The (unnormalized) ranking loss function $\ell_{rank}(Y, f)$ between the multilabel Y and a ranking function $f : \mathcal{R}^d \to \mathcal{R}^K$, representing degrees of class relevance sorted in a decreasing order $f_{j_1}(\boldsymbol{x}_t) \geq f_{j_2}(\boldsymbol{x}_t) \geq \ldots \geq f_{j_K}(\boldsymbol{x}_t) \geq 0$, counts the number of class pairs that disagree in the two rankings:

$$\ell_{rank}(Y, f) = \sum_{i,j \in [K] \,:\, y_i > y_j} \left(\{ f_i(\boldsymbol{x}_t) < f_j(\boldsymbol{x}_t) \} + \tfrac{1}{2} \{ f_i(\boldsymbol{x}_t) = f_j(\boldsymbol{x}_t) \} \right),$$

where $\{\ldots\}$ is the indicator function of the predicate at argument. As pointed out by Dembczynski et al. (2012), the ranking function $f(\boldsymbol{x}_t) = (p_{1,t}, \ldots, p_{K,t})$ is also Bayes optimal w.r.t. $\ell_{rank}(Y, f)$, *no matter if* the class labels y_i are conditionally independent or not. Hence we can use the algorithm in Figure 2 with a close to 1 for tackling ranking problems derived from multilabel ones, when the measure of choice is ℓ_{rank} and the feedback is full.

We now consider a partial information version of the above ranking problem. Suppose that at each time t, the environment discloses both $\boldsymbol{x}_t$ and a maximal *size* S_t for the ordered subset $\hat{Y}_t = (j_1, j_2, \ldots, j_{|\hat{Y}_t|})$ (both $\boldsymbol{x}_t$ and S_t can be chosen adaptively by an adversary). Here S_t might be the number of available slots in a webpage or the maximal number of URLs returned by a search engine in response to query $\boldsymbol{x}_t$. Then it is plausible to compete in a regret sense against the best time-t offline ranking of the form

$$f^*(\boldsymbol{x}_t) = f^*(\boldsymbol{x}_t; S_t) = (f_1^*(\boldsymbol{x}_t), f_2^*(\boldsymbol{x}_t), \ldots, f_K^*(\boldsymbol{x}_t)),$$

where the number of strictly positive $f_i^*(\boldsymbol{x}_t)$ values is at most S_t. Further, the ranking loss could be reasonably restricted to count the number of class pairs disagreeing within $\hat{Y}_t$ plus a quantity related to the number of false negative mistakes. If $\hat{Y}_t$ is the sequence of length S_t associated with a ranking function f, we consider the loss function $\ell_{p-rank,t}$ ("partial information ℓ_{rank} at time t")

$$\ell_{p-rank,t}(Y, f) = \sum_{i,j \in \hat{Y}_t \,:\, y_i > y_j} \left(\{ f_i(\boldsymbol{x}_t) < f_j(\boldsymbol{x}_t) \} + \tfrac{1}{2} \{ f_i(\boldsymbol{x}_t) = f_j(\boldsymbol{x}_t) \} \right) + S_t \, |Y_t \setminus \hat{Y}_t|.$$

In this loss function, the factor S_t multiplying $|Y_t \setminus \hat{Y}_t|$ serves as balancing the contribution of the double sum $\sum_{i,j \in \hat{Y}_t \,:\, y_i > y_j}$ (potentially involving a quadratic number of terms) with the contribution of false negative mistakes $|Y_t \setminus \hat{Y}_t|$. As for loss $\ell_{a,c}$, we can rewrite $\ell_{p-rank,t}(Y, f)$ as

$$\ell_{p-rank,t}(Y, f) = \sum_{i,j \in \hat{Y}_t \,:\, y_i > y_j} \left(\{ f_i(\boldsymbol{x}_t) < f_j(\boldsymbol{x}_t) \} + \tfrac{1}{2} \{ f_i(\boldsymbol{x}_t) = f_j(\boldsymbol{x}_t) \} \right) - S_t \, |Y_t \cap \hat{Y}_t| + S_t \, |Y_t|,$$

9. If $a = 1$, the algorithm only cares about false negative mistakes, the best strategy being always predicting $\hat{Y}_t = [K]$. Unsurprisingly, this yields zero regret in both Theorems 2 and 4.

where the first two terms can be calculated by the algorithm, and the last one does not depend on $\hat{Y}_t$. For convenience, we will interchangeably use the notations $\ell_{p-rank,t}(Y, f)$ and $\ell_{p-rank,t}(Y, \hat{Y}_t)$, whenever it is clear from the surrounding context that $\hat{Y}_t$ is the sequence corresponding to f.

The next lemma[10] is the ranking counterpart to Lemma 1. It shows that the Bayes optimal ranking for $\ell_{p-rank,t}$ is given by

$$f^*(\boldsymbol{x}_t; S_t) = (p'_{1,t}, p'_{2,t}, \ldots, p'_{K,t}),$$

where $p'_{j,t} = p_{j,t}$ if $p_{j,t}$ is among the S_t largest values in the sequence $(p_{1,t}, \ldots, p_{K,t})$, and 0 otherwise. That is, $f^*(\boldsymbol{x}_t; S_t)$ is the function that ranks classes according to decreasing values of $p_{i,t}$ and cuts off exactly at position S_t. This is in contrast to what happens for loss $\ell_{a,c}$, where, depending on the cost parameters $c(i, s)$, the cut off point can even be smaller than the total number of available slots—see Lemma 1 and surrounding comments. In order for this result to go through, we need to restrict model (2) to the case of conditionally independent classes, i.e., to the case when

$$\mathbb{P}_t(y_{1,t}, \ldots, y_{K,t}) = \prod_{i \in [K]} p_{i,t}\,. \tag{4}$$

This is a significant departure from the full information setting, where the Bayes optimal ranking only depends on the marginal distribution values $p_{i,t}$ (Dembczynski et al., 2012). Due to the interaction between the two terms in the definition of $\ell_{p-rank,t}$, the Bayes optimal ranking for $\ell_{p-rank,t}$ turns out to depend on both marginal and pairwise correlation values of the joint class distribution. Assumption (4) may be avoided by maintaining $O(K^2)$ upper confidence values $\epsilon_{i,j}$, one for each pair $(i, j), i < j$, leading to an extra computational burden which can become prohibitive even in the presence of a moderate number of classes K.

Lemma 6 *With the notation introduced so far, let the joint distribution $\mathbb{P}_t(y_{1,t}, \ldots, y_{K,t})$ factorize as in (4). Then $f^*(\boldsymbol{x}_t; S_t)$ introduced above satisfies*

$$f^*(\boldsymbol{x}_t; S_t) = \operatorname*{argmin}_{Y=(i_1,i_2,\ldots i_h)\,, h \le S_t} \mathbb{E}_t[\ell_{p-rank,t}(Y_t, Y)].$$

If we add to the argmin of our algorithm (Step 3 in Figure 2) the further constraint $|Y| \le S_t$ (notice that the resulting computation is still about sorting classes according to decreasing values of $\widehat{p}_{i,t}$), we are defining a partial information ranking algorithm that ranks classes according to decreasing values of $\widehat{p}_{i,t}$ up to position S_t (i.e., $|\hat{Y}_t| = S_t$). Let $\widehat{f}(\boldsymbol{x}_t, S_t)$ be the resulting ranking. We can then define the cumulative regret R_T w.r.t. $\ell_{p-rank,t}$ as

$$R_T = \sum_{t=1}^{T} \mathbb{E}_t[\ell_{p-rank,t}(Y_t, \widehat{f}(\boldsymbol{x}_t, S_t))] - \mathbb{E}_t[\ell_{p-rank,t}(Y_t, f^*(\boldsymbol{x}_t, S_t)], \tag{5}$$

that is, the extent to which the conditional $\ell_{p-rank,t}$-risk of $\widehat{f}(\boldsymbol{x}_t, S_t)$ exceeds the one of the Bayes optimal ranking $f^*(\boldsymbol{x}_t; S_t)$, accumulated over time.

We have the following ranking counterpart to Theorem 2.

10. We postpone its lengthy proof to Section 6.

Theorem 7 *With the same assumptions and notation as in Theorem 2, combined with the independence assumption (4), let the cumulative regret R_T w.r.t. $\ell_{p-rank,t}$ be defined as in (5). Then, with probability at least $1-\delta$, we have that the algorithm in Figure 2 working with $a \to 1$ and strictly decreasing cost values $c(i,s)$ (i.e., the algorithm computing in round t the ranking function $\widehat{f}(\boldsymbol{x}_t, S_t)$) achieves*

$$R_T = O\left(c_L \sqrt{S\,K\,T\,C\,d\,\ln\left(1+\frac{T}{d}\right)}\right),$$

where $S = \max_{t=1,\ldots,T} S_t$.

The proof (see Section 6) is very similar to the one of Theorem 2. This suggests that, to some extent, we are decoupling the label generating model from the loss function ℓ under consideration.

Remark 8 *As is typical in many multilabel classification settings, the number of classes K can be very large and/or have an inner structure (e.g., a hierarchical or DAG-like structure). It is often the case that in such a large label space, many classes are relatively rare. This has lead researchers to consider methods that are specifically tailored to leverage the label sparsity of the chosen classifier (e.g., Hsu et al. 2009 and references therein) and/or the specific structure of the set of labels (e.g., Cesa-Bianchi et al. 2006a; Bi and Kwok 2011, and references therein). Though our algorithm is not designed to exploit the label structure, we would like to stress that the restriction $|\hat{Y}_t| \leq S_t \leq S$ in Theorem 7 allows us to replace the linear dependence on the total number of classes K (which is often much larger than S) by $\sqrt{SK}$. It is very easy to see that this restriction would bring similar benefits to Theorem 2.*

In fact, the above restriction is not only beneficial from a "statistical" point of view, but also from a computational one. As is by now standard, algorithms like the one in Figure 2 can easily be cast in dual variables (i.e., in a RKHS). This comes with at least two consequences:

1. *We can depart from the (generalized) linear modeling assumption (2), and allow for more general nonlinear dependencies of $p_{i,t}$ on the input vectors $\boldsymbol{x}_t$, possibly resorting to the universal approximation properties of Gaussian RKHS (e.g., Steinwart, 2002).*

2. *We can maintain a dual variable representation for margins $\widehat{\Delta}'_{i,t}$ and quadratic forms $\boldsymbol{x}_t^\top A_{i,t-1}^{-1} \boldsymbol{x}_t$, so that computing each one of them takes $O(N_{i,t-1}^2)$ inner products, where $N_{i,t}$ is the number of times class i has been updated up to time t, each inner product being $O(d)$. Now, each of the (at most $S_t \leq S$) updates is $O(N_{i,t-1}^2)$. Hence, the overall running time in round t is coarsely overapproximated by $O(d \sum_{i\in[K]} N_{i,T}^2 + K \log K)$. From $\sum_{i\in[K]} N_{i,T} \leq ST$, we see that when S is small compared to K, then $N_{i,t-1}$ tends to be small as well. For instance, if $S \leq \sqrt{K}$ this leads to a running time per round of the form SdT^2, which can be smaller than the bound Kd^2 mentioned in Remark 5.*

Finally, observe that one can also combine Theorem 7 with the argument contained in Remark 1.

Task	Train+Test	d	K	Avg	Avg + std	95%	99%
Mediamill	30,993+12,914	120	101	5	7	8	10
Sony	16,452+16,519	98	632	38	44	48	52
Yeast	1,500+917	103	14	5	6	7	8

Table 1: Main statistics related to the three data sets used in our experiments. The last four columns give information on the distribution of the number of labels per instance. "Avg" denotes the (rounded) average number of labels over the training examples, and "Avg+std" gives the average augmented by one unit of standard deviation. So, for instance, in the Mediamill data set, the average number of labels per instance in the training set is 5, with a standard deviation of 2. The columns tagged "95%" and "99%" give an idea of the quantiles of this distribution. E.g., on Mediamill, 95% of the training examples have at most 8 classes (out of 101), on the Sony data set, 99% of the training examples have at most 52 classes (out of 632).

5. Experiments

The experiments we report here are meant to validate the exploration-exploitation tradeoff implemented by our algorithm along different axes: data set size, label set size, loss function, label generation model, training mode of operation, and restrictions on the total number of classes predicted. Moreover, we explicitly tested the effectiveness of ranking classes based on upper confidence-corrected probability estimates.

5.1 Data Sets

We used three diverse multilabel data sets, intended to represent different real-world conditions. The first one, called Mediamill, was introduced in a video annotation challenge (Snoek et al., 2006). It comprises $30,993$ training samples and $12,914$ test ones. The number of features d is 120, and the number of classes K is 101. The second data set is the music annotated Sony CSL Paris data set (Pachet and Roy, 2009), made up of $16,452$ training samples and 16,519 test samples, each sample being described by $d = 98$ features. The number of classes K is 632, which is significantly larger than Mediamill's. The third one is the smaller Yeast data set (Elisseeff and Weston, 2002), made up of $1,500$ training samples, 917 test samples, with $d = 103$ and $K = 14$. In all cases, the feature vectors have been normalized to unit Euclidean norm. Table 1 summarizes relevant statistics about these data sets. This table also gives an idea of the distribution of the number of classes per instance.

5.2 Parameter Setting and Loss Measures

For the practical implementation of the algorithm in Figure 2, we simplified the formula for $\epsilon_{i,t}^2$. This is justified by the fact that the actual constants in the definition of $\epsilon_{i,t}^2$ are artifacts of our high-probability upper bounds. Hence, we used

$$\epsilon_{i,t}^2 = \alpha \, \boldsymbol{x}_t^\top A_{i,t-1}^{-1} \boldsymbol{x}_t \, \log(t+1),$$

where α is a parameter that we found by cross-validation on each data set across the range $\alpha = 2^{-8}, 2^{-7}, \ldots, 2^{7}, 2^{8}$, for each choice of the label-generation model, loss setting, and value of S—see below. We have considered two different loss functions L, the square loss and the logistic loss (denoted by "Log Loss" in our plots). Correspondingly, the two label-generation models we tested are the linear model $\mathbb{P}_t(y_{i,t} = 1) = (1 + \boldsymbol{u}_i^\top \boldsymbol{x}_t)/2$ with domain $D = [-1, 1]$, and the logistic model $\mathbb{P}_t(y_{i,t} = 1) = e^{\boldsymbol{u}_i^\top \boldsymbol{x}_t}/(e^{\boldsymbol{u}_i^\top \boldsymbol{x}_t} + 1)$. In the logistic case, it makes sense in practice not to place any restrictions on the margin domain D, so that we set $R = \infty$. Again, because our upper bounding analysis would yield as a consequence $c_L'' = 0$, we instead set c_L'' to a small positive constant, specifically $c_L'' = 0.1$, with no special attention to its fine-tuning. The setting of the cost function $c(i, s)$ depends on the task at hand, and we decided to evaluate two possible settings. The first one, denoted by "decreasing" is $c(i, s) = \frac{s-i+1}{s}, i = 1, \ldots, s$, the second one, denoted by "constant", is $c(i, s) = 1$, for all i and s. In all experiments with $\ell_{a,c}$, the a parameter was set to 0.5 (so that $\ell_{a,c}$ with constant c reduces to half the Hamming loss). In the decreasing c scenario, we evaluated the performance of the algorithm on the loss $\ell_{a,c}$ that the algorithm is minimizing, but also its ability to produce meaningful (partial) rankings through $\ell_{p-rank,t}$. In the constant c scenario, we only evaluated the Hamming loss, its natural loss function.

As is typical of multilabel problems, the label *density* of our data sets, i.e., the average fraction of labels associated with the examples, is quite small. Hence, it is clearly beneficial to our learning algorithm to bias its inference process so as to produce short ranked lists $\hat{Y}_t$. We did so by imposing, for all t, an upper bound $S_t = S$ on $|\hat{Y}_t|$. For each of the three data sets, we tried out the four different values of S reported in the last four columns of Table 1: the average number of labels; the average plus one standard deviation, the number of labels that covers 95% of the examples, and the number of labels that covers 99% of the examples, all figures only referring to the corresponding training sets.

5.3 Baselines

As a baseline, we considered a full information version of Algorithm 2, denoted by "Full Info", that receives after each prediction the full array of true labels Y_t for each sample. Comparing to full information algorithms stresses the effectiveness of the exploration/exploitation rule above and beyond the details of underlying generalized linear predictor. We also compared against the random predictor (denoted by "Random") that simply outputs at time t a ranked list $\hat{Y}_t$ made up of S labels chosen (and ranked) at random. Finally, an interesting ranking baseline which targets the ranking ability of our algorithm is one that lets our partial feedback algorithm select which classes to include in $\hat{Y}_t$, and then shuffles them at random within $\hat{Y}_t$ to produce the ranked list. This baseline we only used with the ranking loss $\ell_{p-rank,t}$, and is denoted by "Shuffled" in our plots.

5.4 Results

Our results are summarized in Figures 3, 4, and 5. The top row of each figure shows the results in the online setting, while the bottom row is for the batch setting. Each column corresponds to a different data set. In both the online and batch cases, the algorithms were fed with the training set in a sequential fashion, sweeping over it only *once*.

The plots report online or batch loss measures as a function of S,[11] averaged over 5 random permutation of the training sequence. Specifically, whereas the online measure of performance ("Final Average ... Loss") is the cumulative loss accumulated during training, divided by the number of samples in the training set, the batch measure ("Test ...") is simply the average loss over the test set achieved by the last solution produced by training. For the partial-feedback algorithms ("Square Loss", "Log Loss" and, in the ranking case, also "Square Loss Shuffled" and "Log Loss Shuffled"), only the best α-cross-validated performances are shown. Moreover, in the ranking experiments, because of the explicit dependence of $\ell_{p-rank,t}$ on S, we instead considered the scaled version of the loss $\ell_{p-rank,t}/S$. Notice that the theoretical results contained in Section 4 still apply to this scaled loss function.

The first thing to observe from the evidence we collected is that performance in the batch setting closely follows the one in the online setting, across all the data sets, conditions and losses. In a sense, this is to be expected, since the order of samples in the training set is randomly shuffled.

The optimal value of S that allows us to best balance exploration and the exploitation of the algorithm seems to be depending on the particular data set and task at hand. So, for instance, on Mediamill with Hamming loss, this value is $S = 8$, corresponding to the 95% coverage of the training set, while on Yeast it is the average value $S = 5$, covering around 50% of the training examples. When the loss is $\ell_{a,c}$, the best value of S clearly depends on the costs c. In the ranking case, performance increases as S gets larger, but this is very likely to be due to the scaling factor $1/S$ in the loss we plotted. Notice that, from our theoretical analysis in Section 3, the algorithm (e.g., in the special case on Hamming loss) should in principle be able to determine the best size of $\hat{Y}_t$ at each round, so that setting $S_t = K$ for all t is still a fair choice. Yet, this conservative setting makes the algorithm face an unnecessarily large action space (of size $K!$), and correspondingly a harder inference problem, rather than the substantially smaller space (of size $K(K-1)(K-2)\dots(K-S+1)$) obtained by setting $S_t = S$. This is evinced by the fact that all plots (regarding both partial and full information algorithms) in Figure 4 tend to be increasing with S. For the very sake of this inference, the fact that all algorithms see the examples only once seems to be a severe limitation.[12]

The performance of our partial information algorithms are always pretty close to those of the corresponding full information algorithms. This empirically validates the exploration/exploitation scheme we used. Also, in all cases, all algorithms clearly outperform the random predictor. In most of the experiments, the linear model ("Square Loss") seems to deliver slightly better results in the bandit setting than the logistic model ("Log Loss"), while the performance of the two models is very similar in the full information case. Exceptions are the constant and the decreasing cost settings in the batch case on the Yeast data set (Figure 3, bottom right, and Figure 4, bottom right), where the bandit algorithm has an even better performance than the full information one. This is perhaps due to the

11. The plots are actually piecewise linear interpolations with knots corresponding to the 4 values of S mentioned in the main text.

12. Training for a single epoch is a restriction needed to carry out a fair comparison between full and partial information algorithms: Cycling more than once on a training set may turn a partial information algorithm into a full information one.

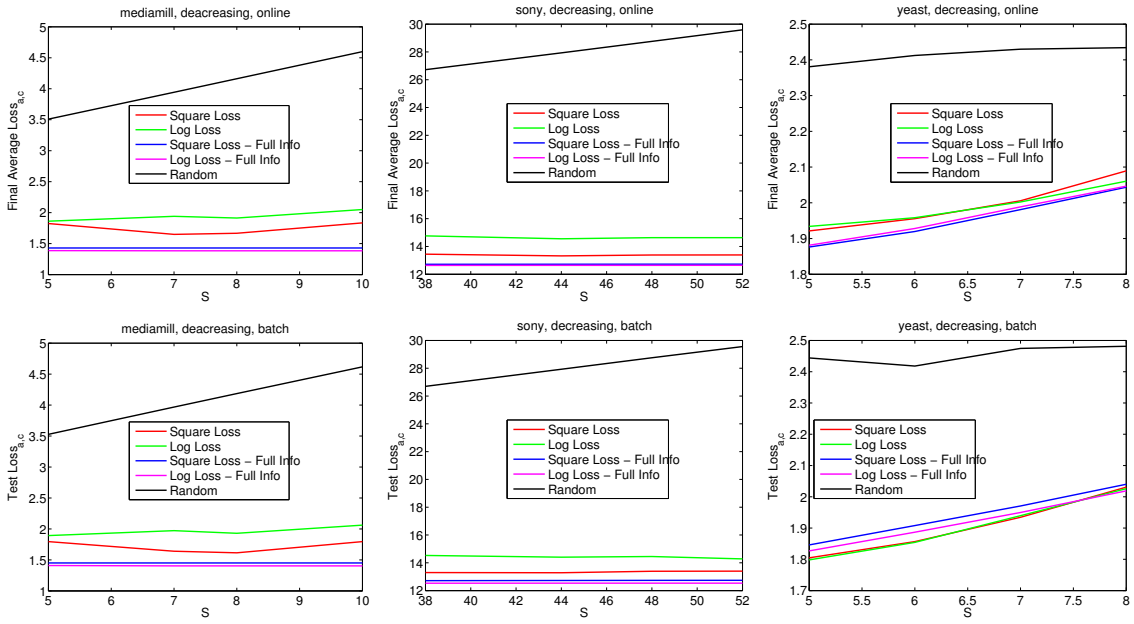


Figure 3: Experiments with $\ell_{a,c}$ and decreasing costs.

noise introduced during exploration, that acts as a kind of regularization, improving generalization performance in such a small data set. In general, however, the comparison linear vs. logistic is somewhat mixed.

In the ranking setting (Figure 5) we also show the performance of our algorithm when the order of predicted labels is randomly permuted ("Shuffled"). It is shown that, uniformly over all settings, shuffling causes performance degradation, thereby proving that our algorithm is indeed learning a meaningful ranking over the labels in the set $\hat{Y}_t$, even without receiving any ranking feedback within this set from the user.

6. Technical Details

This section contains all proofs missing from the main text, along with ancillary results and comments.

The algorithm in Figure 2 works by updating through the gradients $\nabla_{i,t}$ of a modular margin-based loss function $\sum_{i=1}^{K} L(\boldsymbol{w}_i^\top \boldsymbol{x})$ associated with the label generation model (2), i.e., associated with function g, so as to make the parameters $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K) \in \mathcal{R}^{dK}$ therein achieve the Bayes optimality condition

$$(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K) = \arg \min_{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K \,:\, \boldsymbol{w}_i^\top \boldsymbol{x}_t \in D} \mathbb{E}_t \left[\sum_{i=1}^{K} L(s_{i,t}\, \boldsymbol{w}_i^\top \boldsymbol{x}_t) \right], \tag{6}$$

where $\mathbb{E}_t[\cdot]$ above is over the generation of Y_t in producing the sign value $s_{i,t} \in \{-1, 0, +1\}$, conditioned on the past (in particular, conditioned on $\hat{Y}_t$). The requirement in (6) is akin to the classical construction of *proper scoring rules* in the statistical literature (e.g., Savage, 1973).

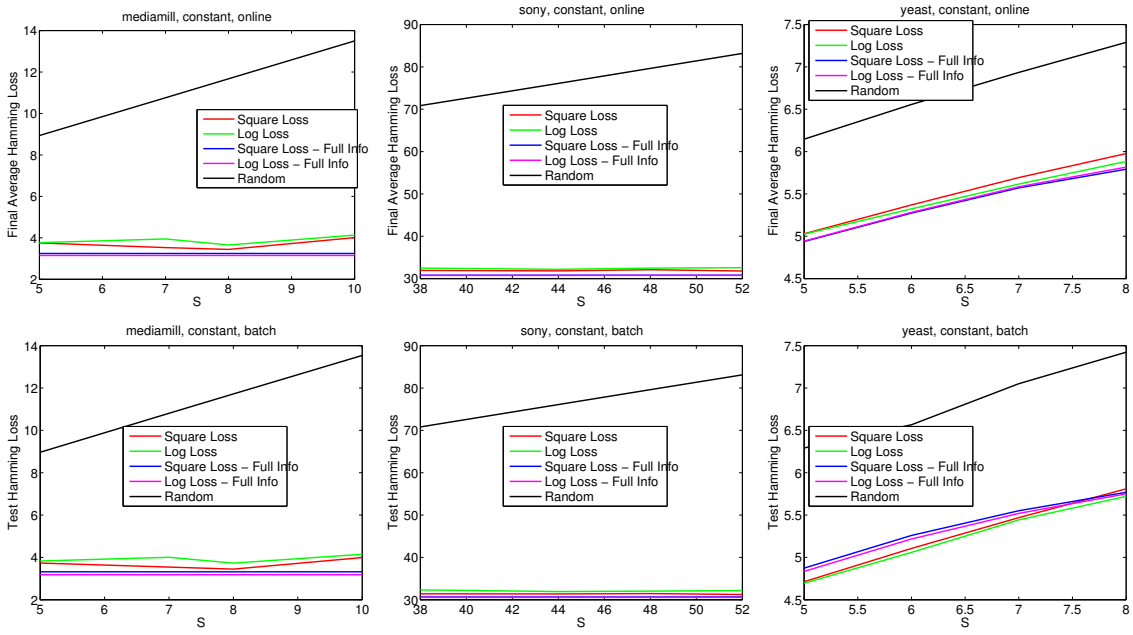


Figure 4: Experiments with $\ell_{a,c}$ and constant costs (Hamming loss).

The above is combined with the ability of the algorithm to guarantee the high probability convergence of the prototype vectors $\boldsymbol{w}'_{i,t}$ to the corresponding $\boldsymbol{u}_i$ (Lemma 13). The rate of convergence is ruled by the fact that the associated upper confidence values $\epsilon_{i,t}$ shrink to zero as $\frac{1}{\sqrt{t}}$ when t grows large. In order for this convergence to take place, it is important to insure that the algorithm is observing informative feedback (either "correct", i.e., $s_{i,t} = 1$, or "mistaken", i.e., $s_{i,t} = -1$) for each class i contained in the selected $\hat{Y}_t$. This in turn implies regret bounds for both $\ell_{a,c}$ (Lemma 11) and $\ell_{p-rank,t}$ (Lemma 12).

The following lemma faces the problem of hand-crafting a convenient loss function $L(\cdot)$ such that (6) holds.

Lemma 9 *Let $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K \in \mathcal{R}^{dK}$ be arbitrary weight vectors such that $\boldsymbol{w}_i^\top \boldsymbol{x}_t \in D$, $i \in [K]$, $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K) \in \mathcal{R}^{dK}$ be defined in (2), $s_{i,t}$ be the updating signs computed by the algorithm at the end (Step 5) of time t, $L : D = [-R, R] \subseteq \mathcal{R} \to \mathcal{R}^+$ be a convex and differentiable function of its argument, with $g(\Delta) = -L'(\Delta)$. Then for any t we have*

$$\mathbb{E}_t \left[\sum_{i=1}^{K} L(s_{i,t}\, \boldsymbol{w}_i^\top \boldsymbol{x}_t) \right] \geq \mathbb{E}_t \left[\sum_{i=1}^{K} L(s_{i,t}\, \boldsymbol{u}_i^\top \boldsymbol{x}_t) \right],$$

i.e., (6) holds.

Proof Let us introduce the shorthands $\Delta_i = \boldsymbol{u}_i^\top \boldsymbol{x}_t$, $\widehat{\Delta}_i = \boldsymbol{w}_{i,t}^\top \boldsymbol{x}_t$, $s_i = s_{i,t}$, and $p_i = \mathbb{P}(y_{i,t} = 1 \,|\, \boldsymbol{x}_t) = \frac{L'(-\Delta_i)}{L'(\Delta_i)+L'(-\Delta_i)} = \frac{g(-\Delta_i)}{g(\Delta_i)+g(-\Delta_i)}$. Moreover, let $\mathbb{P}_t(\cdot)$ be an abbreviation for the conditional probability $\mathbb{P}(\cdot \,|\, (y_1, \boldsymbol{x}_1), \ldots, (y_{t-1}, \boldsymbol{x}_{t-1}), \boldsymbol{x}_t)$. Recalling the way $s_{i,t}$ is

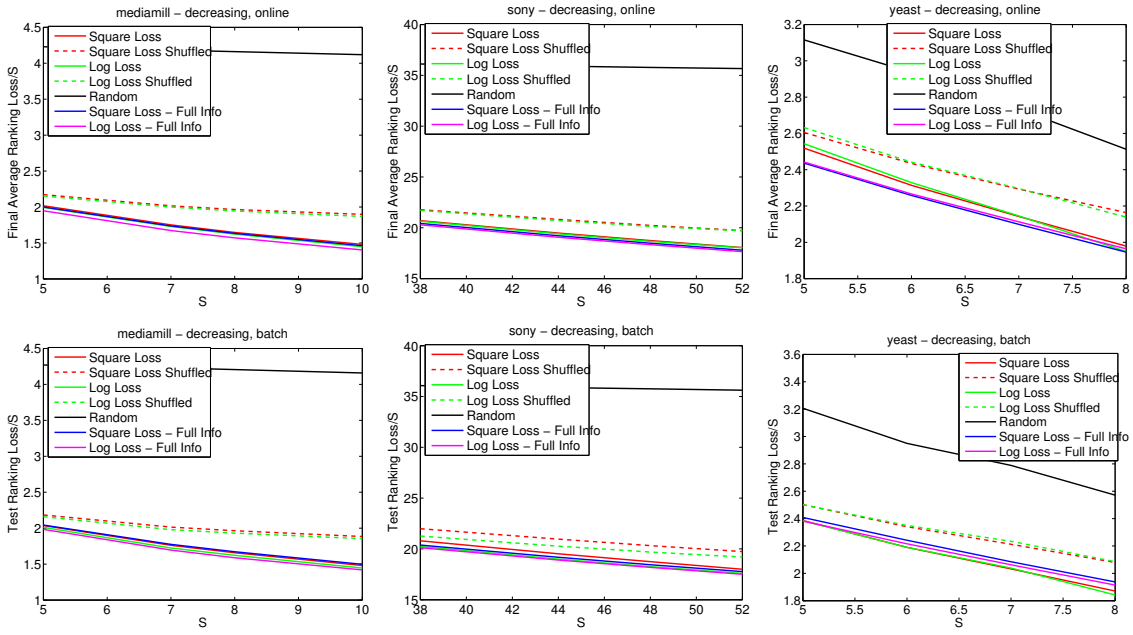


Figure 5: Experiments with the ranking loss $\ell_{p-rank,t}$. In order to obtain "scale-independent" results, in this figure we actually used $\ell_{p-rank,t}/S$ rather than $\ell_{p-rank,t}$ itself.

constructed (Figure 2), we can write

$$\begin{aligned}\mathbb{E}_t\left[\sum_{i=1}^{K} L(s_{i,t}\,\widehat{\Delta}_i)\right] &= \sum_{i\in\hat{Y}_t}\Big(\mathbb{P}_t(s_{i,t}=1)\,L(\widehat{\Delta}_i)+\mathbb{P}_t(s_{i,t}=-1)\,L(-\widehat{\Delta}_i)\Big)+(K-|\hat{Y}_t|)\,L(0)\\ &= \sum_{i\in\hat{Y}_t}\Big(p_i\,L(\widehat{\Delta}_i)+(1-p_i)\,L(-\widehat{\Delta}_i)\Big)+(K-|\hat{Y}_t|)\,L(0),\end{aligned}$$

For similar reasons,

$$\mathbb{E}_t\left[\sum_{i=1}^{K} L(s_{i,t}\,\Delta_i)\right] = \sum_{i\in\hat{Y}_t}\left(p_i\,L(\Delta_i)+(1-p_i)\,L(-\Delta_i)\right)+(K-|\hat{Y}_t|)\,L(0).$$

Since $L(\cdot)$ is convex, so is $\mathbb{E}_t\left[\sum_{i=1}^{K} L(s_{i,t}\,\widehat{\Delta}_i)\right]$ when viewed as a function of the $\widehat{\Delta}_i$. We have that $\frac{\partial\,\mathbb{E}_t\left[\sum_{i=1}^{K} L(s_{i,t}\,\widehat{\Delta}_i)\right]}{\partial\widehat{\Delta}_i}=0$ if and only if for all $i\in\hat{Y}_t$ we have that $\widehat{\Delta}_i$ satisfies

$$p_i=\frac{L'(-\widehat{\Delta}_i)}{L'(\widehat{\Delta}_i)+L'(-\widehat{\Delta}_i)}.$$

Since $p_i=\frac{L'(-\Delta_i)}{L'(\Delta_i)+L'(-\Delta_i)}$, we have that $\mathbb{E}_t\left[\sum_{i=1}^{K} L(s_{i,t}\,\widehat{\Delta}_i)\right]$ is minimized when $\widehat{\Delta}_i=\Delta_i$ for all $i\in[K]$. The claimed result immediately follows. ■

Let now $Var_t(\cdot)$ be a shorthand for $Var(\cdot \,|\, (y_1, \boldsymbol{x}_1), \ldots, (y_{t-1}, \boldsymbol{x}_{t-1}), \boldsymbol{x}_t)$. The following lemma shows that under additional assumptions on the loss $L(\cdot)$, we can bound the variance of a difference of losses $L(\cdot)$ by the expectation of this difference. This will be key to proving the fast rates of convergence contained in the subsequent Lemma 13.

Lemma 10 *Let $(\boldsymbol{w}'_{1,t}, \ldots, \boldsymbol{w}'_{K,t}) \in \mathcal{R}^{dK}$ be the weight vectors computed by the algorithm in Figure 2 at the beginning (Step 2) of time t, $s_{i,t}$ be the updating signs computed at the end (Step 5) of time t, and $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K) \in \mathcal{R}^{dK}$ be the comparison vectors defined through (2). Let $L : D = [-R, R] \subseteq \mathcal{R} \to \mathcal{R}^+$ be a $C^2(D)$ convex function of its argument, with $g(\Delta) = -L'(\Delta)$ and such that there are positive constants c'_L and c''_L with $(L'(\Delta))^2 \le c'_L$ and $L''(\Delta) \ge c''_L$ for all $\Delta \in D$. Then for any $i \in \hat{Y}_t$*

$$0 \le Var_t\left(L(s_{i,t}\, \boldsymbol{x}_t^\top \boldsymbol{w}'_{i,t}) - L(s_{i,t}\, \boldsymbol{u}_i^\top \boldsymbol{x}_t)\right) \le \frac{2c'_L}{c''_L}\, \mathbb{E}_t\left[L(s_{i,t}\, \boldsymbol{x}_t^\top \boldsymbol{w}'_{i,t}) - L(s_{i,t}\, \boldsymbol{u}_i^\top \boldsymbol{x}_t)\right].$$

Proof Let us introduce the shorthands $\Delta_i = \boldsymbol{x}_t^\top \boldsymbol{u}_i$, $\widehat{\Delta}_i = \boldsymbol{x}_t^\top \boldsymbol{w}'_{i,t}$, $s_i = s_{i,t}$, and $p_i = \mathbb{P}(y_{i,t} = 1 \,|\, \boldsymbol{x}_t) = \frac{L'(-\Delta_i)}{L'(\Delta_i)+L'(-\Delta_i)} = \frac{g(-\Delta_i)}{g(\Delta_i)+g(-\Delta_i)}$. Then, for any $i \in [K]$,

$$Var_t\left(L(s_{i,t}\, \boldsymbol{x}_t^\top \boldsymbol{w}'_{i,t}) - L(s_{i,t}\, \boldsymbol{u}_i^\top \boldsymbol{x}_t)\right) \le \mathbb{E}_t\left(\left(L(s_i\, \widehat{\Delta}_i) - L(s_i\, \Delta_i)\right)^2\right) \le c'_L\, (\widehat{\Delta}_i - \Delta_i)^2. \quad (7)$$

Moreover, for any $i \in \hat{Y}_t$ we can write

$$\begin{aligned}
\mathbb{E}_t\left[L(s_i\, \widehat{\Delta}_i) - L(s_i\, \Delta_i)\right] &= p_i\, (L(\widehat{\Delta}_i) - L(\Delta_i)) + (1 - p_i)\, (L(-\widehat{\Delta}_i) - L(-\Delta_i)) \\
&\ge p_i \left(L'(\Delta_i)(\widehat{\Delta}_i - \Delta_i) + \frac{c''_L}{2}(\widehat{\Delta}_i - \Delta_i)^2\right) \\
&\quad + (1 - p_i) \left(L'(-\Delta_i)(\Delta_i - \widehat{\Delta}) + \frac{c''_L}{2}(\widehat{\Delta}_i - \Delta_i)^2\right) \\
&= p_i\, \frac{c''_L}{2}(\widehat{\Delta}_i - \Delta_i)^2 + (1 - p_i)\, \frac{c''_L}{2}(\widehat{\Delta}_i - \Delta_i)^2 \\
&= \frac{c''_L}{2}(\widehat{\Delta}_i - \Delta_i)^2, \qquad (8)
\end{aligned}$$

where the second equality uses the definition of p_i. Combining (7) with (8) gives the desired bound. ■

We continue by showing a one-step regret bound *for our original* loss $\ell_{a,c}$. The precise connection to loss $L(\cdot)$ will be established with the help of a later lemma (Lemma 13).

Lemma 11 *Let $L : D = [-R, R] \subseteq \mathcal{R} \to \mathcal{R}^+$ be a convex, twice differentiable, and nonincreasing function of its argument. Let $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K) \in \mathcal{R}^{dK}$ be defined in (2) with $g(\Delta) = -L'(\Delta)$ for all $\Delta \in D$. Let also c_L be a positive constant such that*

$$\frac{L'(\Delta)\, L''(-\Delta) + L''(\Delta)\, L'(-\Delta)}{(L'(\Delta) + L'(-\Delta))^2} \ge -c_L$$

holds for all $\Delta \in D$. Finally, let $\Delta_{i,t}$ denote $\boldsymbol{u}_i^\top \boldsymbol{x}_t$, and $\widehat{\Delta}'_{i,t}$ denote $\boldsymbol{x}_t^\top \boldsymbol{w}'_{i,t}$, where $\boldsymbol{w}'_{i,t}$ is the i-the weight vector computed by the algorithm at the beginning (Step 2) of time t. If time t is such that $|\Delta_{i,t} - \widehat{\Delta}'_{i,t}| \le \epsilon_{i,t}$ for all $i \in [K]$, then

$$\mathbb{E}_t[\ell_{a,c}(Y_t, \hat{Y}_t)] - \mathbb{E}_t[\ell_{a,c}(Y_t, Y_t^*)] \le 2\,(1-a)\,c_L \sum_{i \in \hat{Y}_t} \epsilon_{i,t}.$$

Proof Recall the shorthand notation $p(\Delta) = \frac{g(-\Delta)}{g(\Delta)+g(-\Delta)}$. We can write

$$\begin{aligned}
&\mathbb{E}_t[\ell_{a,c}(Y_t, \hat{Y}_t)] - \mathbb{E}_t[\ell_{a,c}(Y_t, Y_t^*)] \\
&\qquad = (1-a) \sum_{i \in \hat{Y}_t} \Big(c(\hat{j}_i, |\hat{Y}_t|) - \Big(\tfrac{a}{1-a} + c(\hat{j}_i, |\hat{Y}_t|) \Big)\, p(\Delta_{i,t}) \Big) \\
&\qquad\quad - (1-a) \sum_{i \in Y_t^*} \Big(c(j_i^*, |Y_t^*|) - \Big(\tfrac{a}{1-a} + c(j_i^*, |Y_t^*|) \Big)\, p(\Delta_{i,t}) \Big),
\end{aligned}$$

where $\hat{j}_i$ denotes the position of class i in $\hat{Y}_t$ and j_i^* is the position of class i in Y_t^*. Now,

$$p'(\Delta) = \frac{-g'(-\Delta)\,g(\Delta) - g'(\Delta)\,g(-\Delta)}{(g(\Delta)+g(-\Delta))^2} = \frac{-L'(\Delta)\,L''(-\Delta) - L'(-\Delta)\,L''(\Delta)}{(L'(\Delta)+L'(-\Delta))^2} \ge 0$$

since $g(\Delta) = -L'(\Delta)$, and $L(\cdot)$ is convex and nonincreasing. Hence $p(\Delta)$ is itself a nondecreasing function of Δ. Moreover, the extra condition on L involving L' and L'' is a Lipschitz condition on $p(\Delta)$ via a uniform bound on $p'(\Delta)$. Hence, from $|\Delta_{i,t} - \widehat{\Delta}'_{i,t}| \le \epsilon_{i,t}$ and the definition of $\hat{Y}_t$ we can write

$$\begin{aligned}
&\mathbb{E}_t[\ell_{a,c}(Y_t, \hat{Y}_t)] - \mathbb{E}_t[\ell_{a,c}(Y_t, Y_t^*)] \\
&\qquad \le (1-a) \sum_{i \in \hat{Y}_t} \Big(c(\hat{j}_i, |\hat{Y}_t|) - \Big(\tfrac{a}{1-a} + c(\hat{j}_i, |\hat{Y}_t|) \Big)\, p([\widehat{\Delta}'_{i,t} - \epsilon_{i,t}]_D) \Big) \\
&\qquad\quad - (1-a) \sum_{i \in Y_t^*} \Big(c(j_i^*, |Y_t^*|) - \Big(\tfrac{a}{1-a} + c(j_i^*, |Y_t^*|) \Big)\, p([\widehat{\Delta}'_{i,t} + \epsilon_{i,t}]_D) \Big) \\
&\qquad \le (1-a) \sum_{i \in \hat{Y}_t} \Big(c(\hat{j}_i, |\hat{Y}_t|) - \Big(\tfrac{a}{1-a} + c(\hat{j}_i, |\hat{Y}_t|) \Big)\, p([\widehat{\Delta}'_{i,t} - \epsilon_{i,t}]_D) \Big) \\
&\qquad\quad - (1-a) \sum_{i \in \hat{Y}_t} \Big(c(\hat{j}_i, |\hat{Y}_t|) - \Big(\tfrac{a}{1-a} + c(\hat{j}_i, |\hat{Y}_t|) \Big)\, p([\widehat{\Delta}'_{i,t} + \epsilon_{i,t}]_D) \Big) \\
&\qquad = (1-a) \sum_{i \in \hat{Y}_t} \Big(c(\hat{j}_i, |\hat{Y}_t|) \Big(p([\widehat{\Delta}'_{i,t} + \epsilon_{i,t}]_D) - p([\widehat{\Delta}'_{i,t} - \epsilon_{i,t}]_D) \Big) \Big) \\
&\qquad \le 2\,(1-a)\,c_L \sum_{i \in \hat{Y}_t} \epsilon_{i,t},
\end{aligned}$$

the last inequality deriving from $c(i,s) \le 1$ for all $i \le s \le K$, and

$$p([\widehat{\Delta}'_{i,t} + \epsilon_{i,t}]_D) - p([\widehat{\Delta}'_{i,t} - \epsilon_{i,t}]_D) \le c_L \big([\widehat{\Delta}'_{i,t} + \epsilon_{i,t}]_D - [\widehat{\Delta}'_{i,t} - \epsilon_{i,t}]_D \big) \le 2\,c_L\,\epsilon_{i,t}.$$

■

Now, we first give a proof of Lemma 6, and then provide a one step regret for the partial information ranking loss.

Proof [Lemma 6] Recall the notation $\mathbb{P}_t(\cdot) = \mathbb{P}(\cdot \,|\, \boldsymbol{x}_t)$, and $p_{i,t} = p(\Delta_{i,t}) = \frac{g(-\Delta_{i,t})}{g(\Delta_{i,t})+g(-\Delta_{i,t})}$. For notational convenience, in this proof we drop subscript t from $p_{i,t}$, S_t, $y_{i,t}$, $\hat{Y}_t$, and $\ell_{p-rank,t}$. A simple adaptation of Dembczynski et al. (2012) (proof of Theorem 1 therein) shows that for a generic sequence $\widehat{a} = (\widehat{a}_1, \ldots, \widehat{a}_K)$ with at most S nonzero values $\widehat{a}_i$ and associated set of indices $\hat{Y}$, one has

$$\mathbb{E}_t[\ell_{p-rank}(Y_t, \widehat{a})] = \sum_{i,j \in \hat{Y},\, i<j} (\widehat{r}_{i,j} + \widehat{r}_{j,i}) + S \left(\sum_{i \in [K]} p_i - \sum_{i \in \hat{Y}} p_i \right)$$

where

$$\widehat{r}_{i,j} = \widehat{r}_{i,j}(\widehat{a}) = \mathbb{P}_t(y_i > y_j) \left(\{\widehat{a}_i < \widehat{a}_j\} + \tfrac{1}{2} \{\widehat{a}_i = \widehat{a}_j\} \right) .$$

Moreover, if p^* denotes the sequence made up of at most S nonzero values taken from $\{p_i \,, i \in [K]\}$, *where i ranges again in $\hat{Y}$*, we have

$$\mathbb{E}_t[\ell_{p-rank}(Y_t, p^*)] = \sum_{i,j \in \hat{Y},\, i<j} (r_{i,j} + r_{j,i}) + S \left(\sum_{i \in [K]} p_i - \sum_{i \in \hat{Y}} p_i \right)$$

with

$$r_{i,j} = r_{i,j}(p^*) = \mathbb{P}_t(y_i > y_j) \left(\{p_i < p_j\} + \tfrac{1}{2} \{p_i = p_j\} \right) .$$

Hence

$$\mathbb{E}_t[\ell_{p-rank}(Y_t, \widehat{a})] - \mathbb{E}_t[\ell_{p-rank}(Y_t, p^*)] = \sum_{i,j \in \hat{Y},\, i<j} (\widehat{r}_{i,j} - r_{i,j} + \widehat{r}_{j,i} - r_{j,i}) \,.$$

Since

$$\mathbb{P}_t(y_i > y_j) - \mathbb{P}_t(y_j > y_i) = \mathbb{P}_t(y_i = 1) - \mathbb{P}_t(y_j = 1) = p_i - p_j,$$

a simple (but lengthy) case analysis reveals that

$$\widehat{r}_{i,j} - r_{i,j} + \widehat{r}_{j,i} - r_{j,i} = \begin{cases} \frac{1}{2}(p_i - p_j) & \text{If } \widehat{a}_i < \widehat{a}_j,\ p_i = p_j \text{ or } \widehat{a}_i = \widehat{a}_j,\ p_i > p_j \\ \frac{1}{2}(p_j - p_i) & \text{If } \widehat{a}_i = \widehat{a}_j,\ p_i < p_j \text{ or } \widehat{a}_i > \widehat{a}_j,\ p_i = p_j \\ p_i - p_j & \text{If } \widehat{a}_i < \widehat{a}_j,\ p_i > p_j \\ p_j - p_i & \text{If } \widehat{a}_i > \widehat{a}_j,\ p_i < p_j \,. \end{cases}$$

Notice that the above quantity is always nonnegative, and is strictly positive if the p_i are all different. The nonnegativity implies that *whatever set of indices $\hat{Y}$ we select*, the best way to sort them within $\hat{Y}$ in order to minimize $\mathbb{E}_t[\ell_{p-rank}(Y_t, \cdot)]$ is by following the ordering of the corresponding p_i.

We are left to show that the best choice for $\hat{Y}$ is to collect the S largest[13] values in $\{p_i\,, i \in [K]\}$. To this effect, consider again $\mathbb{E}_t[\ell_{p-rank}(Y_t, p^*)] = \mathbb{E}_t[\ell_{p-rank}(Y_t, \hat{Y})]$, and introduce the shorthand $p_{i,j} = p_i\, p_j = p_i - \mathbb{P}_t(y_i > y_j)$. Disregarding the term $S \sum_{i\in[K]} p_i$, which is independent of $\hat{Y}$, we can write

$$\begin{aligned}
\mathbb{E}_t[\ell_{p-rank}(Y_t, \hat{Y})] = & \sum_{i,j\in\hat{Y},\, i<j} \mathbb{P}_t(y_i > y_j)\, \left(\{p_i < p_j\} + \tfrac{1}{2}\,\{p_i = p_j\}\right) \\
& + \sum_{i,j\in\hat{Y},\, i<j} \mathbb{P}_t(y_j > y_i)\, \left(\{p_j < p_i\} + \tfrac{1}{2}\,\{p_j = p_i\}\right) - S \sum_{i\in\hat{Y}} p_i \\
= & \sum_{i,j\in\hat{Y},\, i<j} (p_i - p_{i,j})\{p_i < p_j\} + (p_i - p_{i,j})\tfrac{1}{2}\,\{p_i = p_j\} \\
& + \sum_{i,j\in\hat{Y},\, i<j} (p_j - p_{i,j})\{p_j < p_i\} + (p_j - p_{i,j})\tfrac{1}{2}\,\{p_j = p_i\} - S \sum_{i\in\hat{Y}} p_i \\
= & \sum_{i,j\in\hat{Y},\, i<j} (p_i - p_j)\{p_i < p_j\} + \tfrac{1}{2}\,(p_i - p_j)\,\{p_i = p_j\} + p_j - p_{i,j} - S \sum_{i\in\hat{Y}} p_i \\
= & \sum_{i,j\in\hat{Y},\, i<j} (\min\{p_i, p_j\} - p_i p_j) - S \sum_{i\in\hat{Y}} p_i
\end{aligned}$$

which can be finally seen to be equal to

$$-\sum_{i\in\hat{Y}} (S + 1 - \hat{j}_i)\, p_i - \sum_{i,j\in\hat{Y},\, i<j} p_i\, p_j\,, \tag{9}$$

where $\hat{j}_i$ is the position of class i within $\hat{Y}_t$ in decreasing order of p_i.

Now, rename the indices in $\hat{Y}$ as $1, 2, \ldots, S$, in such a way that $p_1 > p_2 > \ldots > p_S$ (so that $\hat{j}_i = i$), and consider the way to increase (9) by adding to $\hat{Y}$ item $k \notin \hat{Y}$ such that $p_S > p_k$ and removing from $\hat{Y}$ the item in position ℓ. Denote the resulting sequence by $\hat{Y}'$. From (9), it is not hard to see that

$$\begin{aligned}
& \mathbb{E}_t[\ell_{p-rank}(Y_t, \hat{Y})] - \mathbb{E}_t[\ell_{p-rank}(Y_t, \hat{Y}')] \\
& \quad = (\ell - 1)\, p_\ell + \sum_{i=\ell+1}^{S} p_i - \sum_{i=1}^{\ell-1} p_i\, p_\ell - \sum_{i=\ell+1}^{S} p_\ell\, p_i - (S-1)\, p_k + \sum_{i=1, i\neq\ell}^{S} p_i\, p_k - S(p_\ell - p_k) \\
& \quad = (\ell - 1)\, p_\ell + \sum_{i=\ell+1}^{S} p_i - (p_\ell - p_k) \sum_{i=1, i\neq\ell}^{S} p_i - (S-1)\, p_k - S(p_\ell - p_k) \\
& \quad \leq (S-1)\, p_\ell - (p_\ell - p_k) \sum_{i=1, i\neq\ell}^{S} p_i - (S-1)\, p_k - S(p_\ell - p_k) \\
& \quad = (p_k - p_\ell) \left(1 + \sum_{i=1, i\neq\ell}^{S} p_i\right)
\end{aligned} \tag{10}$$

13. It is at this point that we need the conditional independence assumption over the classes.

which is smaller than zero since, by assumption, $p_\ell > p_k$. Reversing the direction, if we maintain a sequence $\hat{Y}$ of size S, we can always reduce (9) by removing its smallest element and replacing it with a larger element outside the sequence. We continue until no element outside the current sequence exists which is larger than the smallest one in the sequence. Clearly, we end up collecting the S largest elements in $\{p_i\,, i \in [K]\}$.

Finally, from (9) it is very clear that removing an element from a sequence $\hat{Y}$ of length $h \leq S$ can only increase the value of (9). Since this holds for an arbitrary $\hat{Y}$ and an arbitrary $h \leq S$, this shows that, no matter which set $\hat{Y}$ we start off from, we always converge to the same set containing exactly the S largest elements in $\{p_i\,, i \in [K]\}$. This concludes the proof. ■

Lemma 12 *Under the same assumptions and notation as in Lemma 11, combined with the independence assumption (4), let the Algorithm in Figure 2 be working with $a \to 1$ and strictly decreasing cost values $c(i,s)$, i.e., the algorithm is computing in round t the ranking function $\widehat{f}(\boldsymbol{x}_t; S_t)$ defined in Section 4. Let $\boldsymbol{w}'_{i,t}$ be the i-th weight vector computed by this algorithm at the beginning (Step 2) of time t. If time t is such that $|\Delta_{i,t} - \widehat{\Delta}'_{i,t}| \leq \epsilon_{i,t}$ for all $i \in [K]$, then*

$$\mathbb{E}_t[\ell_{rank,t}(Y_t, \widehat{f}(\boldsymbol{x}_t; S_t)] - \mathbb{E}_t[\ell_{rank,t}(Y_t, f^*(\boldsymbol{x}_t; S_t)] \leq 4\, S_t\, c_L \sum_{i \in \hat{Y}_t} \epsilon_{i,t}.$$

Proof We use the same notation as in the proof of Lemma 6, where $\widehat{a}$ is now $\hat{Y}_t$, the sequence produced by ranking $\widehat{f}(\boldsymbol{x}_t; S_t)$ operating on $\widehat{p}_{i,t}$. Denote by Y_t^* the sequences determined by $f^*(\boldsymbol{x}_t; S_t)$, and let $\hat{j}_i$ and j_i^* be the position of class i in decreasing order of $p_{i,t}$ within $\hat{Y}_t$ and Y_t^*, respectively.

Proceeding as in Lemma 11 and recalling (9) we can write

$$\begin{aligned}
&\mathbb{E}_t[\ell_{p-rank,t}(Y_t, \widehat{f}(\boldsymbol{x}_t; S_t))] - \mathbb{E}_t[\ell_{p-rank,t}(Y_t, f^*(\boldsymbol{x}_t; S_t)] \\
&\quad = \sum_{i \in Y_t^*} (S_t + 1 - j_i^*)\, p_i + \sum_{i,j \in Y_t^*,\, i<j} p_i\, p_j - \sum_{i \in \hat{Y}_t} (S_t + 1 - \hat{j}_i)\, p_i - \sum_{i,j \in \hat{Y}_t,\, i<j} p_i\, p_j \\
&\quad \leq \sum_{i \in Y_t^*} (S_t + 1 - j_i^*)\, p([\widehat{\Delta}'_{i,t} + \epsilon_{i,t}]_D) + \sum_{i,j \in Y_t^*,\, i<j} p([\widehat{\Delta}'_{i,t} + \epsilon_{i,t}]_D)\, p([\widehat{\Delta}'_{j,t} + \epsilon_{j,t}]_D) \\
&\qquad - \sum_{i \in \hat{Y}_t} (S_t + 1 - \hat{j}_i)\, p([\widehat{\Delta}'_{i,t} - \epsilon_{i,t}]_D) - \sum_{i,j \in \hat{Y}_t,\, i<j} p([\widehat{\Delta}'_{i,t} - \epsilon_{i,t}]_D)\, p([\widehat{\Delta}'_{j,t} - \epsilon_{j,t}]_D) \\
&\quad \leq \sum_{i \in \hat{Y}_t} (S_t + 1 - \hat{j}_i)\, \Big(p([\widehat{\Delta}'_{i,t} + \epsilon_{i,t}]_D) - p([\widehat{\Delta}'_{i,t} - \epsilon_{i,t}]_D) \Big) \\
&\qquad + \sum_{i,j \in \hat{Y}_t,\, i<j} \Big(p([\widehat{\Delta}'_{i,t} + \epsilon_{i,t}]_D)\, p([\widehat{\Delta}'_{j,t} + \epsilon_{j,t}]_D) - p([\widehat{\Delta}'_{i,t} - \epsilon_{i,t}]_D)\, p([\widehat{\Delta}'_{j,t} - \epsilon_{j,t}]_D) \Big)\,.
\end{aligned}$$

This, in turn, can be upper bounded by

$$2S_t c_L \sum_{i\in\hat{Y}_t} \epsilon_{i,t} + \sum_{i,j\in\hat{Y}_t,\, i<j} 2c_L\,(\epsilon_{i,t}+\epsilon_{j,t}) = 2\,S_t\,c_L \sum_{i\in\hat{Y}_t} \epsilon_{i,t} + 2\,(S_t-1)\,c_L \sum_{i\in\hat{Y}_t} \epsilon_{i,t}$$
$$< 4\,S_t\,c_L \sum_{i\in\hat{Y}_t} \epsilon_{i,t}\,,$$

as claimed. ■

Lemma 13 *Let $L : D = [-R, R] \subseteq \mathcal{R} \to \mathcal{R}^+$ be a $C^2(D)$ convex and nonincreasing function of its argument, $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K) \in \mathcal{R}^{dK}$ be defined in (2) with $g(\Delta) = -L'(\Delta)$ for all $\Delta \in D$, and such that $\|\boldsymbol{u}_i\| \le U$ for all $i \in [K]$. Assume there are positive constants c'_L and c''_L with $(L'(\Delta))^2 \le c'_L$ and $L''(\Delta) \ge c''_L$ for all $\Delta \in D$. With the notation introduced in Figure 2, we have that*

$$(\boldsymbol{x}^\top \boldsymbol{w}'_{i,t} - \boldsymbol{u}_i^\top \boldsymbol{x})^2 \le \boldsymbol{x}^\top A_{i,t-1}^{-1} \boldsymbol{x} \left(U^2 + \frac{d\,c'_L}{(c''_L)^2} \ln\left(1 + \frac{t-1}{d}\right) + \frac{12}{c''_L}\left(\frac{c'_L}{c''_L} + 3L(-R)\right) \ln\frac{K(t+4)}{\delta}\right)$$

holds with probability at least $1-\delta$ for any $\delta < 1/e$, uniformly *over $i \in [K]$, $t = 1, 2, \ldots$, and $\boldsymbol{x} \in \mathcal{R}^d$.*

Proof For any given class i, the time-t update rule $\boldsymbol{w}'_{i,t} \to \boldsymbol{w}_{i,t+1} \to \boldsymbol{w}'_{i,t+1}$ in Figure 2 allows us to start off from the paper by Hazan et al. (2007) (proof of Theorem 2 therein), from which one can extract the following inequality

$$d_{i,t-1}(\boldsymbol{u}_i, \boldsymbol{w}'_{i,t})$$
$$\le U^2 + \frac{1}{(c''_L)^2} \sum_{k=1}^{t-1} r_{i,k} - \frac{2}{c''_L} \sum_{k=1}^{t-1} \left(\nabla_{i,k}^\top (\boldsymbol{w}'_{i,k} - \boldsymbol{u}_i) - \frac{c''_L}{2} \left(s_{i,k}\, \boldsymbol{x}_k^\top (\boldsymbol{w}'_{i,k} - \boldsymbol{u}_i)\right)^2 \right), \quad (11)$$

where we set $r_{i,k} = \nabla_{i,k}^\top A_{i,k}^{-1} \nabla_{i,k}$.

We now observe that we can construct a quadratic lower bound to L, using the lower bound on the second derivative of L. More explicitly, using the Taylor expansion of L, we have

$$L(x) \ge L(y) + L'(y)(x-y) + \frac{c''_L}{2}(x-y)^2,$$

for any x, y in D. Hence, setting $y = s_{i,k}\, \boldsymbol{x}_k^\top \boldsymbol{w}'_{i,k}$ and $x = s_{i,k}\, \boldsymbol{u}_i^\top \boldsymbol{x}_k$, we have

$$L(s_{i,k}\, \boldsymbol{x}_k^\top \boldsymbol{w}'_{i,k}) - L(s_{i,k}\, \boldsymbol{u}_i^\top \boldsymbol{x}_k)$$
$$\le L'(s_{i,k}\, \boldsymbol{x}_k^\top \boldsymbol{w}'_{i,k})(s_{i,k} \boldsymbol{x}_k^\top \boldsymbol{w}'_{i,k} - s_{i,k}\, \boldsymbol{u}_i^\top \boldsymbol{x}_k) - \frac{c''_L}{2}(s_{i,k}\, \boldsymbol{x}_k^\top \boldsymbol{w}'_{i,k} - s_{i,k}\, \boldsymbol{u}_i^\top \boldsymbol{x}_k)^2$$
$$= \nabla_{i,k}^\top (\boldsymbol{w}'_{i,k} - \boldsymbol{u}_i) - \frac{c''_L}{2} \left(s_{i,k}\, \boldsymbol{x}_k^\top (\boldsymbol{w}'_{i,k} - \boldsymbol{u}_i)\right)^2.$$

Plugging back into (11) yields

$$d_{i,t-1}(\boldsymbol{u}_i, \boldsymbol{w}'_{i,t}) \leq U^2 + \frac{1}{(c''_L)^2} \sum_{k=1}^{t-1} r_{i,k} - \frac{2}{c''_L} \sum_{k=1}^{t-1} \Big(L(s_{i,k}\, \boldsymbol{x}_k^\top \boldsymbol{w}'_{i,k}) - L(s_{i,k}\, \boldsymbol{u}_i^\top \boldsymbol{x}_k) \Big) . \qquad (12)$$

We now borrow a proof technique from Dekel et al. (2012) (see also the papers by Crammer and Gentile 2011; Abbasi-Yadkori et al. 2011 and references therein). Define

$$L_{i,k} = L(s_{i,k}\, \boldsymbol{x}_k^\top \boldsymbol{w}'_{i,k}) - L(s_{i,k}\, \boldsymbol{u}_i^\top \boldsymbol{x}_k),$$

and $L'_{i,k} = \mathbb{E}_k[L_{i,k}] - L_{i,k}$. Notice that the sequence of random variables $L'_{i,1}$, $L'_{i,2}, \ldots$, forms a martingale difference sequence such that, for any $i \in \hat{Y}_k$:

i. $\mathbb{E}_k[L_{i,k}] \geq 0$, by Lemma 10 (or Lemma 9);

ii. $|L'_{i,k}| \leq 2L(-R)$, since $L(\cdot)$ is nonincreasing over D, and $s_{i,k}\, \boldsymbol{x}_k^\top \boldsymbol{w}'_{i,k}$, $s_{i,k}\, \boldsymbol{u}_i^\top \boldsymbol{x}_k \in D$;

iii. $Var_k(L'_{i,k}) = Var_k(L_{i,k}) \leq \frac{2c'_L}{c''_L}\, \mathbb{E}_k[L_{i,k}]$ (again, because of Lemma 10).

On the other hand, when $i \notin \hat{Y}_k$ then $s_{i,k} = 0$, and the above three properties are trivially satisfied. Under the above conditions, we are in a position to apply any fast concentration result for bounded martingale difference sequences. For instance, setting for brevity $B = B(t, \delta) = 3 \ln \frac{K(t+4)}{\delta}$, a result contained in the paper by Kakade and Tewari (2009) allows us derive the inequality

$$\sum_{k=1}^{t-1} \mathbb{E}_k[L_{i,k}] - \sum_{k=1}^{t-1} L_{i,k} \geq \max \left\{ \sqrt{\frac{8c'_L}{c''_L}\, B \sum_{k=1}^{t-1} \mathbb{E}_k[L_{i,k}]}, 6L(-R)\, B \right\},$$

that holds with probability at most $\frac{\delta}{Kt(t+1)}$ for any $t \geq 1$. We use the inequality $\sqrt{cb} \leq \frac{1}{2}(c+b)$ with $c = \frac{4c'_L}{c''_L}\, B$, and $b = 2 \sum_{k=1}^{t-1} \mathbb{E}_k[L_{i,k}]$, and simplify. This gives

$$-\sum_{k=1}^{t-1} L_{i,k} \leq \left(\frac{2c'_L}{c''_L} + 6L(-R) \right) B$$

with probability at least $1 - \frac{\delta}{Kt(t+1)}$. Using the Cauchy-Schwarz inequality

$$(\boldsymbol{x}^\top \boldsymbol{w}'_{i,t} - \boldsymbol{u}_i^\top \boldsymbol{x})^2 \leq \boldsymbol{x}^\top A_{i,t-1}^{-1}\, \boldsymbol{x}\, d_{i,t-1}(\boldsymbol{u}_i, \boldsymbol{w}'_{i,t})$$

holding for any $\boldsymbol{x} \in \mathcal{R}^d$, and replacing back into (12) allows us to conclude that

$$(\boldsymbol{x}^\top \boldsymbol{w}'_{i,t} - \boldsymbol{u}_i^\top \boldsymbol{x})^2 \leq \boldsymbol{x}^\top A_{i,t-1}^{-1} \boldsymbol{x} \left(U^2 + \frac{1}{(c''_L)^2} \sum_{k=1}^{t-1} r_{i,k} + \frac{12}{c''_L} \left(\frac{c'_L}{c''_L} + 3L(-R) \right) \ln \frac{K(t+4)}{\delta} \right) \qquad (13)$$

holds with probability at least $1 - \frac{\delta}{Kt(t+1)}$, *uniformly* over $\boldsymbol{x} \in \mathcal{R}^d$.

The bounds on $\sum_{k=1}^{t-1} r_{i,k}$ can be obtained in a standard way. Applying known inequalities (Azoury and Warmuth, 2001; Cesa-Bianchi et al., 2002, 2009; Cavallanti et al., 2011; Hazan et al., 2007; Dekel et al., 2012), and using the fact that $\nabla_{i,k} = L'(s_{i,k}\,\boldsymbol{x}_k^\top \boldsymbol{w}'_{i,k})\,s_{i,k}\boldsymbol{x}_k$ we have

$$\begin{aligned}\sum_{k=1}^{t-1} r_{i,k} &= \sum_{k=1}^{t-1} |s_{i,j}|\,(L'(s_{i,k}\,\boldsymbol{x}_k^\top \boldsymbol{w}'_{i,k}))^2\,\boldsymbol{x}_k^\top A_{i,k}^{-1}\boldsymbol{x}_k \le c'_L \sum_{k=1}^{t-1} |s_{i,k}|\boldsymbol{x}_k^\top A_{i,k}^{-1}\boldsymbol{x}_k \\ &\le c'_L \sum_{k=1}^{t-1} \ln\frac{|A_{i,k}|}{|A_{i,k-1}|} = c'_L\,\ln\frac{|A_{i,t-1}|}{|A_{i,0}|} \le d\,c'_L\,\ln\left(1+\frac{t-1}{d}\right).\end{aligned}$$

Combining as in (13) and stratifying over $t = 1, 2, \ldots$, and $i \in [K]$ concludes the proof. ■

We are now ready to put all pieces together.

Proof [Theorem 2] From Lemma 11 and Lemma 13, we see that with probability at least $1-\delta$,

$$R_T \le 2\,(1-a)\,c_L \sum_{t=1}^{T} \sum_{i\in\hat{Y}_t} \epsilon_{i,t}\,, \tag{14}$$

when $\epsilon_{i,t}^2$ is the one given in Figure 2. We continue by proving a pointwise upper bound on the sum in the RHS. More in detail, we will find an upper bound on $\sum_{t=1}^{T}\sum_{i\in\hat{Y}_t}\epsilon_{i,t}^2$, and then derive a resulting upper bound on the RHS of (14).

From Lemma 13 and the update rule (Step 5) of the algorithm we can write[14]

$$\begin{aligned}\epsilon_{i,t}^2 &\le C\,\boldsymbol{x}_t^\top A_{i,t-1}^{-1}\boldsymbol{x}_t = C\,\frac{\boldsymbol{x}_t^\top (A_{i,t-1}+|s_{i,t}|\,\boldsymbol{x}_t\boldsymbol{x}_t^\top)^{-1}\boldsymbol{x}_t}{1-|s_{i,t}|\boldsymbol{x}_t^\top (A_{i,t-1}+|s_{i,t}|\,\boldsymbol{x}_t\boldsymbol{x}_t^\top)^{-1}\boldsymbol{x}_t} \\ &= C\,\frac{\boldsymbol{x}_t^\top A_{i,t}^{-1}\boldsymbol{x}_t}{1-|s_{i,t}|\boldsymbol{x}_t^\top (A_{i,t-1}+|s_{i,t}|\,\boldsymbol{x}_t\boldsymbol{x}_t^\top)^{-1}\boldsymbol{x}_t} \\ &\le C\,\frac{\boldsymbol{x}_t^\top A_{i,t}^{-1}\boldsymbol{x}_t}{1-|s_{i,t}|\boldsymbol{x}_t^\top (A_0+|s_{i,t}|\,\boldsymbol{x}_t\boldsymbol{x}_t^\top)^{-1}\boldsymbol{x}_t} = C\,\frac{\boldsymbol{x}_t^\top A_{i,t}^{-1}\boldsymbol{x}_t}{1-\frac{1}{2}} = 2\,C\,\boldsymbol{x}_t^\top A_{i,t}^{-1}\boldsymbol{x}_t.\end{aligned}$$

Hence, if we set $r_{i,t} = \boldsymbol{x}_t^\top A_{i,t}^{-1}\boldsymbol{x}_t$ and proceed as in the proof of Lemma 13, we end up with the upper bound $\sum_{t=1}^{T}\epsilon_{i,t}^2 \le 2\,C\,d\,\ln\left(1+\frac{T}{d}\right)$, holding for all $i \in [K]$. Denoting by M the quantity $2\,C\,d\,\ln\left(1+\frac{T}{d}\right)$, we conclude from (14) that

$$R_T \le 2\,(1-a)\,c_L\,\max\left\{\sum_{i\in[K]}\sum_{t=1}^{T}\epsilon_{i,t}\,\Big|\,\sum_{t=1}^{T}\epsilon_{i,t}^2 \le M,\;\; i\in[K]\right\} = 2\,(1-a)\,c_L\,K\,\sqrt{T\,M},$$

as claimed. ■

14. It is in this chain of inequalities that we exploit the rank-one update of $A_{i,t-1}$ based on $\boldsymbol{x}_t\boldsymbol{x}_t^\top$ rather than $\nabla_{i,t}\nabla_{i,t}^\top$. Here we need to lower bound the eigenvalue of the rank-one matrix used in the update. Using the $\nabla_{i,t}\nabla_{i,t}^\top$ (as in the worst-case analysis by Hazan et al. 2007), the lower bound would be zero. This is due to the presence of the multiplicative factor $g(s_{i,t}\widehat{\Delta}'_{i,t})$ (Step 5 in Figure 2) which can be arbitrarily small.

Proof [Theorem 4] As we said, we change the definition of $\epsilon_{i,t}^2$ in the Algorithm in Figure 2 to

$$\epsilon_{i,t}^2 = \max\left\{ \boldsymbol{x}^\top A_{i,t-1}^{-1} \boldsymbol{x} \left(\frac{2\,d\,c'_L}{(c''_L)^2} \ln\left(1 + \frac{t-1}{d}\right) + \frac{12}{c''_L} \left(\frac{c'_L}{c''_L} + 3L(-R) \right) \ln \frac{K(t+4)}{\delta} \right), 4\,R^2 \right\}.$$

First, notice that the $4R^2$ cap seamlessly applies, since $(\boldsymbol{x}^\top \boldsymbol{w}'_{i,t} - \boldsymbol{u}_i^\top \boldsymbol{x})^2$ in Lemma 13 is bounded by $4\,R^2$ anyway. With this modification, we have that Theorem 2 only holds for t such that $\frac{d\,c'_L}{(c''_L)^2} \ln\left(1 + \frac{t-1}{d}\right) \geq U^2$, i.e., for $t \geq d\left(\exp\left(\frac{(c''_L)^2\,U^2}{c'_L\,d}\right) - 1\right) + 1$, while for $t < d\left(\exp\left(\frac{(c''_L)^2\,U^2}{c'_L\,d}\right) - 1\right) + 1$ we have in the worst-case scenario the maximum amount of regret at each step. From Lemma 11 we see that this maximum amount (the cap on $\epsilon_{i,t}^2$ is needed here) can be bounded by $4\,(1-a)\,c_L\,|\hat{Y}_t|\,R \leq 4\,(1-a)\,c_L\,K\,R$. ■

Proof [Theorem 7] We start from the one step-regret delivered by Lemma 12, and proceed as in the proof of Theorem 2. This yields

$$R_T \leq 4\,c_L \sum_{t=1}^{T} S_t \sum_{i \in \hat{Y}_t} \epsilon_{i,t} \leq 4\,S\,c_L \sum_{t=1}^{T} \sum_{i \in \hat{Y}_t} \epsilon_{i,t} \leq 4\,S\,c_L \sum_{t=1}^{T} \sum_{i \in [K]} \epsilon_{i,t} = 4\,S\,c_L \sum_{i \in [K]} \sum_{t=1}^{T} \epsilon_{i,t}\,,$$

with probability at least $1-\delta$, where $\epsilon_{i,t}^2$ is the one given in Figure 2. Let M be as in the proof of Theorem 2. We have that $\sum_{t=1}^{T} \epsilon_{i,t}^2 \leq M$. If $N_{i,T}$ denotes the total number of times class i occurs in $\hat{Y}_t$, this implies $\sum_{t=1}^{T} \epsilon_{i,t} \leq \sqrt{N_{i,T}\,M}$ for all $i \in [K]$. Moreover, from $\sum_{i \in [K]} N_{i,T} \leq ST$ we can write

$$R_T \leq 4\,S\,c_L \sum_{i \in K]} \sqrt{N_{i,T}\,M} \leq 4\,c_L \sqrt{M\,S\,K\,T}\,,$$

as claimed. ■

7. Conclusions and Open Questions

In this paper, we have used generalized linear models to formalize the exploration-exploitation tradeoff in a multilabel/ranking setting with partial feedback, providing $T^{1/2}$-like regret bounds under semi-adversarial settings. Our analysis decouples the multilabel/ranking loss at hand from the label-generation model, improving in various ways on the existing literature. Thanks to the usage of calibrated score values $\widehat{p}_{i,t}$, our algorithm is capable of automatically inferring where to split the ranking between relevant and nonrelevant classes (Furnkranz et al., 2008), the split depending on the loss function under consideration. We considered two partial-feedback loss functions: $\ell_{a,c}$ and $\ell_{p-rank,t}$. The former can be seen

as a Discounted Cumulative Gain difference, the latter a version of the standard (unnormalized) ranking loss, both being restricted to the chosen ranked list $\hat{Y}_t$. These two losses are inherently different: whereas $\ell_{p-rank,t}$ has a pairwise component, $\ell_{a,c}$ does not; whereas the Bayes optimal Y_t^* w.r.t. $\ell_{p-rank,t}$ has the maximal allowed length, the Bayes optimal Y_t^* w.r.t. $\ell_{a,c}$ need not be full length; whereas Bayes optimality solely based on $p_{i,t}$ does not require conditional independence assumptions when the loss is $\ell_{a,c}$, such condition is needed when the loss is $\ell_{p-rank,t}$. Yet, both losses depend in a similar fashion on the classes contained in $\hat{Y}_t$, as well as on the way such classes are ranked within $\hat{Y}_t$.

We have investigated the practically important case when $\hat{Y}_t$ has to satisfy length constraints $|\hat{Y}_t| \leq S_t$, which is a typical prior knowledge in the presence of large multilabel action spaces. When $S_t \leq S$ for all t, our regret bounds turn the linear dependence on K into a linear dependence on $\sqrt{SK}$.

Finally, we have presented experiments aimed at validating our upper-confidence-based ranking scheme against several real-world conditions and modeling assumptions.

There are many directions along which this work could be extended. In what follows, we briefly mention three of them.

- Multilabel and ranking algorithms are usually evaluated using an array of loss measures, including 0/1, Average Precision, F-measure, AUC, normalized ranking losses, etc. It would be nice to extend the theory contained in this paper to such measures. However, many of these losses are likely to require modeling pairwise correlations among classes.

- In the case when $S_t \leq S$, we showed regret bounds of the form $\sqrt{SK}\,\sqrt{T}$. Is it possible to modify our theoretical arguments (possibly combining with the compressed sensing machinery used by Hsu et al. 2009) so as to obtain the information-theoretic bound $(S \log K)\,\sqrt{T}$, instead? Clearly enough, it would be most interesting to do so via computationally efficient algorithms.

- As a broader goal, it would be interesting to extend this theory to other practically relevant structured action spaces. For instance, an interesting extension is to the case when class labels $y_{i,t}$ are not binary, but real valued. Such values can in fact be the results of click aggregations over time. In this case, we may want to interpret Y_t as a ranked list as well, and come up with appropriate (partial-information) losses between pairs of such lists. Another interesting extension is to (multilabel) hierarchical classification. To this effect, the Bayes optimality arguments developed by Cesa-Bianchi et al. (2006a,b) may be of some relevance.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments that helped us to improve the presentation of this paper. We would also like to thank Koby Crammer, Krzysztof Dembczyński, and Willem Waegeman for early discussions regarding this paper. The first author has been partially supported by the PASCAL2 Network of Excellence under EC grant 216886. This publication only reflects the authors views.

References

Y. Abbasi-Yadkori, D. Pal, and C. Szepesvári. Improved algorithms for linear stochastic bandits. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2312–2320. Curran Associates, Inc., 2011.

A. Agarwal. Selective sampling algorithms for cost-sensitive multiclass prediction. In *ICML (3)*, volume 28 of *JMLR Proceedings*, pages 1220–1228. JMLR.org, 2013.

K. Amin, M. Kearns, and U. Syed. Graphical models for bandit problems. In F.G. Cozman and A. Pfeffer, editors, *UAI*, pages 1–10. AUAI Press, 2011.

P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

K. S. Azoury and M. K. Warmuth. Relative loss bounds for online density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211–246, 2001.

G. Bartók. A near-optimal algorithm for finite partial-monitoring games against adversarial opponents. In S. Shalev-Shwartz and I. Steinwart, editors, *COLT*, volume 30 of *JMLR Proceedings*, pages 696–710. JMLR.org, 2013.

G. Bartók and C. Szepesvári. Partial monitoring with side information. In N.H. Bshouty, G. Stoltz, N. Vayatis, and T. Zeugmann, editors, *ALT*, volume 7568 of *Lecture Notes in Computer Science*, pages 305–319. Springer, 2012.

W. Bi and J. Kwok. Multilabel classification on tree- and DAG-structured hierarchies. In L. Getoor and T. Scheffer, editors, *ICML*, pages 17–24. Omnipress, 2011.

D. Buffoni, C. Calauzènes, P. Gallinari, and N. Usunier. Learning scoring functions with order-preserving losses and standardized supervision. In L. Getoor and T. Scheffer, editors, *ICML*, pages 825–832. Omnipress, 2011.

G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Learning noisy linear classifiers via adaptive and selective sampling. *Machine Learning*, 83:71–102, 2011.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order Perceptron algorithm. In J. Kivinen and R. H. Sloan, editors, *COLT*, volume 2375 of *Lecture Notes in Computer Science*, pages 121–137. Springer, 2002.

N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Incremental algorithms for hierarchical classification. *Journal of Machine Learning Research*, 7:31–54, 2006a.

N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Hierarchical classification: combining Bayes with SVM. In W.W. Cohen and A. Moore, editors, *ICML*, volume 148 of *ACM International Conference Proceeding Series*, pages 177–184. ACM, 2006b.

N. Cesa-Bianchi, C. Gentile, and F. Orabona. Robust bounds for classification via selective sampling. In A.P. Danyluk, L. Bottou, and M.L. Littman, editors, *ICML*, volume 382 of *ACM International Conference Proceeding Series*. ACM, 2009.

S. Clémençon, G. Lugosi, and N. Vayatis. Ranking and scoring using empirical risk minimization. In P. Auer and R. Meir, editors, *COLT*, volume 3559 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2005.

D. Cossock and T. Zhang. Subset ranking using regression. In G. Lugosi and H.-U. Simon, editors, *COLT*, volume 4005 of *Lecture Notes in Computer Science*, pages 605–619. Springer, 2006.

K. Crammer and C. Gentile. Multiclass classification with bandit feedback using adaptive regularization. In L. Getoor and T. Scheffer, editors, *ICML*, pages 273–280. Omnipress, 2011.

V. Dani, T. Hayes, and S. Kakade. Stochastic linear optimization under bandit feedback. In R.A. Servedio and T. Zhang, editors, *COLT*, pages 355–366. Omnipress, 2008.

O. Dekel, C. Gentile, and K. Sridharan. Selective sampling and active learning from single and multiple teachers. *Journal of Machine Learning Research*, 13:2655–2697, 2012.

K. Dembczynski, W. Waegeman, W. Cheng, and E. Hullermeier. On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88:5–45, 2012.

J. C. Duchi, L. W. Mackey, and M. I. Jordan. On the consistency of ranking algorithms. In J. Frnkranz and T. Joachims, editors, *ICML*, pages 327–334. Omnipress, 2010.

A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, volume 14, pages 681–687. MIT Press, 2002.

S. Filippi, O. Cappé, A. Garivier, and C. Szepesvári. Parametric bandits: The generalized linear case. In J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 586–594. Curran Associates, Inc., 2010.

Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

J. Furnkranz, E. Hullermeier, E. Loza Menca, and K. Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 73:133–153, 2008.

C. Gentile and F. Orabona. On multilabel classification and ranking with partial feedback. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1151–1159. Curran Associates, Inc., 2012.

E. Hazan and S. Kale. Beyond convexity: Online submodular minimization. In Y. Bengio, D. Schuurmans, J.D. Lafferty, C.K.I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 700–708. Curran Associates, Inc., 2009.

E. Hazan and S. Kale. Newtron: an efficient bandit algorithm for online multiclass prediction. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 891–899. Curran Associates, Inc., 2011.

E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69:169–192, 2007.

R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers, MIT Press*, 2000.

D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In Y. Bengio, D. Schuurmans, J.D. Lafferty, C.K.I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 772–780. Curran Associates, Inc., 2009.

K. Jarvelin and J. Kekalainen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20:422–446, 2002.

S. Kakade and A. Tewari. On the generalization ability of online strongly convex programming algorithms. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 801–808. Curran Associates, Inc., 2009.

S. Kakade, S. Shalev-Shwartz, and A. Tewari. Efficient bandit algorithms for online multiclass prediction. In W.W. Cohen, A. McCallum, and S.T. Roweis, editors, *ICML*, volume 307 of *ACM International Conference Proceeding Series*, pages 440–447. ACM, 2008.

S. Kale, L. Reyzin, and R. Schapire. Non-stochastic bandit slate problems. In J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1054–1062. Curran Associates, Inc., 2010.

A. Krause and C. S. Ong. Contextual Gaussian process bandit optimization. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2447–2455. Curran Associates, Inc., 2011.

T. H. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.*, 6, 1985.

Y. Lan, J. Guo, X. Cheng, and T. Liu. Statistical consistency of ranking methods in a rank-differentiable probability space. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1232–1240. Curran Associates, Inc., 2012.

J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In J.C. Platt, D. Koller, Y. Singer, and S.T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 817–824. Curran Associates, Inc., 2008.

P. McCullagh and J.A. Nelder. *Generalized Linear Models*. Chapman and Hall, 1989.

F. Orabona and N. Cesa-Bianchi. Better algorithms for selective sampling. In L. Getoor and T. Scheffer, editors, *ICML*, pages 433–440. Omnipress, 2011.

F. Pachet and P. Roy. Improving multilabel analysis of music titles: A large-scale validation of the correction approach. *IEEE Trans. on Audio, Speech, and Lang. Proc.*, 17(2):335–343, 2009.

L.J. Savage. Elicitation of personal probabilities and expectations. *Journal of the American Statistical Association*, 336:783–801, 1973.

P. Shivaswamy and T. Joachims. Online structured prediction via coactive learning. In *ICML*. icml.cc / Omnipress, 2012.

A. Sklar. Fonctions de répartition à n dimensions et leurs marges. *Publ. Inst. Statist. Univ. Paris*, 8:229–231, 1959.

A. Slivkins, F. Radlinski, and S. Gollapudi. Learning optimally diverse rankings over large document collections. pages 983–990. Omnipress, 2010.

C.G.M. Snoek, M. Worring, J.C. van Gemert, J.-M. Geusebroek, and A.W.M. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In K. Nahrstedt, M. Turk, Y. Rui, W. Klas, and K. Mayer-Patel, editors, *ACM Multimedia*, pages 421–430. ACM, 2006.

I. Steinwart. Support vector machines are universally consistent. *J. Complexity*, 18(3): 768–791, 2002.

M. Streeter, D. Golovin, and A. Krause. Online learning of assignments. In Y. Bengio, D. Schuurmans, J.D. Lafferty, C.K.I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1794–1802. Curran Associates, Inc., 2009.

G. Tsoumakas, I. Katakis, and I. Vlahavas. Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 23:1079–1089, 2011.

Y. Wang, R. Khardon, D. Pechyony, and R. Jones. Generalization bounds for online learning algorithms with pairwise loss functions. In S. Mannor, N. Srebro, and R.C. Williamson, editors, *COLT*, volume 23 of *JMLR Proceedings*, pages 13.1–13.22. JMLR.org, 2012.

Journal of Machine Learning Research 15 (2014) 2489-2512 Submitted 4/13; Revised 3/14; Published 7/14

Beyond the Regret Minimization Barrier: Optimal Algorithms for Stochastic Strongly-Convex Optimization

Elad Hazan* EHAZAN@IE.TECHNION.AC.IL
Technion - Israel Inst. of Tech.
Haifa 32000, Israel

Satyen Kale SATYEN@YAHOO-INC.COM
Yahoo! Labs
111 W 40th St, 9th Floor, New York, NY 10018

Editor: Nicolo Cesa-Bianchi

Abstract

We give novel algorithms for stochastic strongly-convex optimization in the gradient oracle model which return a $O(\frac{1}{T})$-approximate solution after T iterations. The first algorithm is deterministic, and achieves this rate via gradient updates and historical averaging. The second algorithm is randomized, and is based on pure gradient steps with a random step size.

This rate of convergence is optimal in the gradient oracle model. This improves upon the previously known best rate of $O(\frac{\log(T)}{T})$, which was obtained by applying an online strongly-convex optimization algorithm with regret $O(\log(T))$ to the batch setting.

We complement this result by proving that any algorithm has expected regret of $\Omega(\log(T))$ in the online stochastic strongly-convex optimization setting. This shows that any online-to-batch conversion is inherently suboptimal for stochastic strongly-convex optimization. This is the first formal evidence that online convex optimization is strictly more difficult than batch stochastic convex optimization.[1]

Keywords: stochastic gradient descent, convex optimization, regret minimization, online learning

1. Introduction

Stochastic convex optimization has an inherently different flavor than standard convex optimization. In the stochastic case, a crucial resource is the number of data samples from the function to be optimized. This resource limits the precision of the output: given few samples there is simply not enough information to compute the optimum up to a certain precision. The error arising from this lack of information is called the *estimation error*.

The estimation error is independent of the choice of optimization algorithm, and it is reasonable to choose an optimization method whose precision is of the same order of magnitude as the sampling error: lesser precision is suboptimal, whereas much better precision is pointless. This issue is extensively discussed by Bottou and Bousquet (2007) and by

∗. Supported by ISF Grant 810/11 and the Microsoft-Technion EC Center.

1. An extended abstract of this work appeared in COLT 2011 (Hazan and Kale, 2011). In this version we have included a new randomized algorithm which is based on pure gradient steps, and extended the results to strong convexity with respect to general norms.

Shalev-Shwartz and Srebro (2008). This makes first-order methods ideal for stochastic convex optimization: their error decreases as a polynomial in the number of iterations, usually make only one iteration per data point, and each iteration is extremely efficient.

In this paper we consider first-order methods for stochastic convex optimization. Formally, the problem of stochastic convex optimization is the minimization of a convex (possibly non-smooth) function on a convex domain $\mathcal{K}$:

$$\min_{\mathbf{x}\in\mathcal{K}} F(\mathbf{x}).$$

The stochasticity is in the access model: the only access to F is via a stochastic subgradient oracle, which given any point $\mathbf{x} \in \mathcal{K}$, produces a random vector $\hat{\mathbf{g}}$ whose expectation is a subgradient of F at the point $\mathbf{x}$, i.e., $\mathbb{E}[\hat{\mathbf{g}}] \in \partial F(\mathbf{x})$, where $\partial F(\mathbf{x})$ denotes the subdifferential set of F at $\mathbf{x}$.

We stress that F may be non-smooth. This is important for the special case when $F(\mathbf{x}) = \mathbb{E}_Z[f(\mathbf{x}, Z)]$ (the expectation being taken over a random variable Z), where for every fixed z, $f(\mathbf{x}, z)$ is a convex function of $\mathbf{x}$. The goal is to minimize F while given a sample $z_1, z_2, \ldots$ drawn independently from the unknown distribution of Z. A prominent example of this formulation is the problem of support vector machine (SVM) training (see Shalev-Shwartz et al., 2009). For SVM training, the function F is convex but non-smooth.

An algorithm for stochastic convex optimization is allowed a budget of T calls to the gradient oracle. It sequentially queries the gradient oracle at consecutive points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$, and produces an approximate solution $\bar{\mathbf{x}}$. The *rate of convergence* of the algorithm is the expected excess cost of the point $\bar{\mathbf{x}}$ over the optimum, i.e. $\mathbb{E}[F(\bar{\mathbf{x}})] - \min_{\mathbf{x}\in\mathcal{K}} F(\mathbf{x})$, where the expectation is taken over the randomness in the gradient oracle and the internal random seed of the algorithm. The paramount parameter for measuring this rate is in terms of T, the number of gradient oracle calls.

Our first and main contribution is the first algorithm to attain the optimal rate of convergence in the case where F is λ-strongly convex, and the gradient oracle is G-bounded (see precise definitions in Section 2.1). After T gradient updates, the algorithm returns a solution which is $O(\frac{1}{T})$-close in cost to the optimum. Formally, we prove the following theorem.

Theorem 1 *Assume that F is λ-strongly convex and the gradient oracle is G-bounded. Then there exists a deterministic algorithm that after at most T gradient updates returns a vector $\bar{\mathbf{x}}$ such that for any $\mathbf{x}^\star \in \mathcal{K}$ we have*

$$\mathbb{E}[F(\bar{\mathbf{x}})] - F(\mathbf{x}^\star) \leq O\left(\frac{G^2}{\lambda T}\right).$$

This matches the lower bound of Agarwal et al. (2012) up to constant factors.

The previously best known rate was $O(\frac{\log(T)}{T})$, and follows by converting a more general online convex optimization algorithm of Hazan et al. (2007) to the batch setting. This standard online-to-batch reduction works as follows. In the online convex optimization setting, in each round $t = 1, 2, \ldots, T$, a decision maker (represented by an algorithm $\mathcal{A}$) chooses a point $\mathbf{x}_t$ in convex domain $\mathcal{K}$, and incurs a cost $f_t(\mathbf{x}_t)$ for an adversarially chosen

convex cost function f_t. In this model performance is measured by the *regret*, defined as

$$\text{Regret}(\mathcal{A}) := \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x}). \tag{1}$$

A regret minimizing algorithm is one that guarantees that the regret grows like $o(T)$. Given such an algorithm, one can perform batch stochastic convex optimization by setting f_t to be the function $f(\cdot, z_t)$. A simple analysis then shows that the cost of the average point, $\bar{\mathbf{x}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{x}_t$, converges to the optimum cost at the rate of the *average* regret, which converges to zero.

The best previously known convergence rates for stochastic convex optimization were obtained using this online-to-batch reduction, and thus these rates were equal to the average regret of the corresponding online convex optimization algorithm. While it is known that for general convex optimization, this online-to-batch reduction gives the optimal rate of convergence, such a result was not known for stochastic strongly-convex functions. In this paper we show that for stochastic strongly-convex functions, minimizing regret is strictly more difficult than batch stochastic strongly-convex optimization.

More specifically, the best known regret bound for λ-strongly-convex cost functions with gradients bounded in norm by G is $O(\frac{G^2\log(T)}{\lambda})$ (Hazan et al., 2007). This regret bound holds even for adversarial, not just stochastic, strongly-convex cost functions. A matching lower bound was obtained by Takimoto and Warmuth (2000) for the adversarial setting.

Our second contribution in this paper is a matching lower bound for strongly-convex cost functions that holds *even in the stochastic setting*, i.e., if the cost functions are sampled i.i.d from an unknown distribution. Formally:

Theorem 2 *For any online decision-making algorithm $\mathcal{A}$, there is a distribution over λ-strongly-convex cost functions with norms of gradients bounded by G such that*

$$\mathbb{E}[\textit{Regret}(\mathcal{A})] = \Omega\left(\frac{G^2\log(T)}{\lambda}\right).$$

Hence, our new rate of convergence of $O(\frac{G^2}{\lambda T})$ is the first to separate the complexity of stochastic and online strongly-convex optimization. The following table summarizes our contribution with respect to the previously known bounds. The setting is assumed to be stochastic λ-strongly-convex functions with expected subgradient norms bounded by G.

	Previously known bound	New bound here
Convergence rate	$O\left(\frac{G^2\log(T)}{\lambda T}\right)$ (Hazan et al., 2007)	$O\left(\frac{G^2}{\lambda T}\right)$
Regret	$\Omega\left(\frac{G^2}{\lambda}\right)$ (Trivial bound[2])	$\Omega\left(\frac{G^2\log(T)}{\lambda}\right)$

2. The lower bound follows from the work of Agarwal et al. (2012), but a simple lower bound example is the following. Consider an adversary that plays a fixed function, either $\frac{\lambda}{2}x^2$ or $\frac{\lambda}{2}(x-\frac{G}{\lambda})^2$, for all rounds, with $\mathcal{K} = [0, \frac{G}{\lambda}]$. On the first round, the loss of the algorithm's point x_1 for one of these two functions is at least $\frac{G^2}{8\lambda}$: this is because $\frac{\lambda}{2}x_1^2 + \frac{\lambda}{2}(x_1 - \frac{G}{\lambda})^2 = \lambda(x_1 - \frac{G}{2\lambda})^2 + \frac{G^2}{4\lambda} \geq \frac{G^2}{4\lambda}$. Clearly the best point in hindsight has 0 loss, so the regret of the algorithm is at least $\frac{G^2}{8\lambda}$ for one of the two functions.

We also sharpen our results: Theorem 1 bounds the expected excess cost of the solution over the optimum by $O(\frac{1}{T})$. We can also show high probability bounds. In situations where it is possible to evaluate F at any given point efficiently, simply repeating the algorithm a number of times and taking the best point found bounds the excess cost by $O(\frac{G^2 \log(\frac{1}{\delta})}{\lambda T})$ with probability at least $1-\delta$. In more realistic situations where it is not possible to evaluate F efficiently, we can still modify the algorithm so that with high probability, the actual excess cost of the solution is bounded by $O(\frac{\log\log(T)}{T})$:

Theorem 3 *Assume that F is λ-strongly convex, and the gradient oracle is strongly G-bounded. Then for any $\delta > 0$, there exists an algorithm that after at most T gradient updates returns a vector $\bar{\mathbf{x}}$ such that with probability at least $1-\delta$, for any $\mathbf{x}^\star \in \mathcal{K}$ we have*

$$F(\bar{\mathbf{x}}) - F(\mathbf{x}^\star) \leq O\left(\frac{G^2(\log(\frac{1}{\delta}) + \log\log(T))}{\lambda T}\right).$$

The algorithm attaining the convergence rate claimed in Theorem 1 is deterministic, albeit not a pure gradient-step algorithm: it proceeds in epochs; each epoch performs gradient steps only. However, the initialization of any epoch is given by the *average* iterate of the previous epoch. A natural question that arises is whether there exists a pure gradient step algorithm, that performs only gradient steps with carefully controlled step size. We also give an algorithm achieving this (although using random step sizes).

1.1 Related Work

For an in depth discussion of first-order methods, the reader is referred to the book by Bertsekas (1999).

The study of lower bounds for stochastic convex optimization was undertaken by Nemirovski and Yudin (1983), and recently extended and refined by Agarwal et al. (2012).

Online convex optimization was introduced by Zinkevich (2003). Optimal lower bounds for the convex case, even in the stochastic setting, of $\Omega(\sqrt{T})$ are simple and given in the book by Cesa-Bianchi and Lugosi (2006). For exp-concave cost functions, Ordentlich and Cover (1998) give a $\Omega(\log T)$ lower bound on the regret, even when the cost functions are sampled according to a known distribution. For strongly convex functions, no non-trivial stochastic lower bound was known. Takimoto and Warmuth (2000) give a $\Omega(\log T)$ lower bound in the regret for adaptive adversaries. Abernethy et al. (2009) put this lower bound in a general framework for min-max regret minimization.

It has been brought to our attention that Juditsky and Nesterov (2010) and Ghadimi and Lan (2010) have recently published technical reports that have very similar results to ours, and also obtain an $O(\frac{1}{T})$ convergence rate. Our work was done independently and a preliminary version was published on arXiv (Hazan and Kale, 2010) before the technical reports of Juditsky and Nesterov (2010) and Ghadimi and Lan (2010) were made available. Note that the high probability bound in this paper has better dependence on T than the result of Ghadimi and Lan (2010): we lose an additional $\log\log T$ factor vs. the $\log^2 \log T$ factor lost in the paper of Ghadimi and Lan (2010). Our lower bound on the regret for stochastic online strongly-convex optimization is entirely new.

Following our work, a number of other works have appeared which obtain the optimal $O(\frac{1}{T})$ convergence rate using other methods. Rakhlin et al. (2012) show that for strongly

convex cost functions that are also smooth, a $O(\frac{1}{T})$ rate is attainable by vanilla stochastic gradient descent (SGD), and further that SGD with special averaging of the last iterates recovers this optimal rate even in the non-smooth case. They also show that empirically, our algorithm indeed performs better than vanilla averaged SGD; though it is slightly worse than the suffix-averaging variant of SGD in their paper. Shamir and Zhang (2013) later considered the last iterate of vanilla SGD, for which they show $O(\frac{\log T}{T})$ convergence rate in the strongly convex case. This complements the bound of $O(\frac{1}{T})$ on the suboptimality of a random iterate from the random SGD variant we give in this paper.

2. Setup and Background

In this section we give basic definitions and describe the optimization framework for our results.

2.1 Stochastic Convex Optimization

We work in a Euclidean space[3] $\mathcal{H}$ with norm $\|\cdot\|$ with the dual norm $\|\cdot\|_\star$. For $\mathbf{x}, \mathbf{w} \in \mathcal{H}$, let $\mathbf{w} \cdot \mathbf{x}$ denote their inner product. For a convex and differentiable function f, we denote by ∇f its gradient at a given point. Consider the setting of stochastic convex optimization of a convex (possibly non-smooth) function F over a convex (possibly non-compact) set $\mathcal{K} \subseteq \mathcal{H}$. Let $\mathbf{x}^\star$ be a point in $\mathcal{K}$ where F is minimized. We make the following assumptions:

1. We assume that we have a convex and differentiable function $\mathcal{R} : \mathcal{H} \to \mathbb{R} \cup \{-\infty, +\infty\}$ with its corresponding Bregman divergence defined as:

$$B_{\mathcal{R}}(\mathbf{y}, \mathbf{x}) \;:=\; \mathcal{R}(\mathbf{y}) - \mathcal{R}(\mathbf{x}) - \nabla\mathcal{R}(\mathbf{x}) \cdot (\mathbf{y} - \mathbf{x}).$$

   By direct substitution, this definition implies that for any vectors $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{H}$,

$$(\nabla\mathcal{R}(\mathbf{z}) - \nabla\mathcal{R}(\mathbf{y})) \cdot (\mathbf{x} - \mathbf{y}) = B_{\mathcal{R}}(\mathbf{x}, \mathbf{y}) - B_{\mathcal{R}}(\mathbf{x}, \mathbf{z}) + B_{\mathcal{R}}(\mathbf{y}, \mathbf{z}). \tag{2}$$

   We assume further that $\mathcal{R}$ is strongly-convex w.r.t. the norm $\|\cdot\|$, i.e., for any two points $\mathbf{x}, \mathbf{y} \in \mathcal{H}$, we have

$$B_{\mathcal{R}}(\mathbf{y}, \mathbf{x}) \;\geq\; \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|^2.$$

2. We assume that F is **λ-strongly convex w.r.t.** $B_{\mathcal{R}}$: i.e., for any two points $\mathbf{x}, \mathbf{y} \in \mathcal{K}$ and any $\alpha \in [0, 1]$, we have

$$F(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) \;\leq\; \alpha F(\mathbf{x}) + (1-\alpha)F(\mathbf{y}) - \lambda\alpha(1-\alpha)B_{\mathcal{R}}(\mathbf{y}, \mathbf{x}).$$

   A sufficient condition for F to be λ-strongly-convex w.r.t. $B_{\mathcal{R}}$ is if $F(\mathbf{x}) = \mathbb{E}_Z[f(\mathbf{x}, Z)]$ and $f(\cdot, z)$ is λ-strongly-convex w.r.t. $B_{\mathcal{R}}$ for every z in the support of Z.

   This implies F satisfies the following inequality:

$$F(\mathbf{x}) - F(\mathbf{x}^\star) \;\geq\; \lambda B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}). \tag{3}$$

3. In this paper, we work in a Euclidean space for simplicity. Our results extend without change to any real Banach space $\mathcal{B}$ with norm $\|\cdot\|$ with the dual space $\mathcal{B}^\star$ and the dual norm $\|\cdot\|_\star$, with the additional assumption that $\mathcal{K}$ is compact.

This follows by setting $\mathbf{y} = \mathbf{x}^\star$, dividing by α, taking the limit as $\alpha \to 0^+$, and using the fact that $\mathbf{x}^\star$ is the minimizer of F. This inequality holds even if $\mathbf{x}^\star$ is on the boundary of $\mathcal{K}$. In fact, (3) is the *only* requirement on the strong convexity of F for the analysis to work; we will simply assume that (3) holds.

3. Assume that we have a stochastic subgradient oracle for F, i.e., we have black-box access to an algorithm that computes an unbiased estimator $\hat{\mathbf{g}}$ of some subgradient of F at any point $\mathbf{x}$, i.e., $\mathbb{E}[\hat{\mathbf{g}}] \in \partial F(\mathbf{x})$. We assume that each call to the oracle uses randomness that is independent of all previously made calls. Further, we assume that at any point $\mathbf{x} \in \mathcal{K}$, the stochastic subgradient $\hat{\mathbf{g}}$ output by the oracle satisfies one of the assumptions below:

   (a) $\mathbb{E}[\|\hat{\mathbf{g}}\|_\star^2] \leq G^2$.

   (b) $\mathbb{E}\left[\exp\left(\frac{\|\hat{\mathbf{g}}\|_\star^2}{G^2}\right)\right] \leq \exp(1)$.

   It is easy to see that assumption 3b implies assumption 3a by Jensen's inequality. We will need the stronger assumption 3b to prove high probability bounds. We call an oracle satisfying the weaker assumption 3a **G-bounded**, and an oracle satisfying the stronger assumption 3b **strongly G-bounded**. For a G-bounded oracle, note that by Jensen's inequality, we also have that $\|\mathbb{E}[\hat{\mathbf{g}}]\|_\star^2 \leq G^2$, so in particular, at all points $\mathbf{x} \in \mathcal{K}$, there is a subgradient of F with $\|\cdot\|_\star$ norm bounded by G.

   For example, in the important special case $F(\mathbf{x}) = \mathbb{E}_Z[f(\mathbf{x}, Z)]$ where $f(\cdot, z)$ is convex for every z in the support of Z, we can obtain such a stochastic subgradient oracle simply by taking a subgradient of $f(\cdot, z)$.

4. The Fenchel conjugate of $\mathcal{R}$ is the function $\mathcal{R}^\star : \mathcal{H} \to \mathbb{R} \cup \{-\infty, +\infty\}$

$$\mathcal{R}^\star(\mathbf{w}) \;:=\; \sup_{\mathbf{x}} \mathbf{w} \cdot \mathbf{x} - \mathcal{R}(\mathbf{x}).$$

   By the properties of Fenchel conjugacy (see Borwein and Lewis, 2006, for more details), we have that $\nabla \mathcal{R}^\star = \nabla \mathcal{R}^{-1}$. We assume that the following "Bregman update and projection" operations can be carried out efficiently over the domain $\mathcal{K}$, for any $\mathbf{x}, \mathbf{g} \in \mathcal{H}$:

$$\begin{aligned} \mathbf{y} &= \nabla \mathcal{R}^\star(\nabla \mathcal{R}(\mathbf{x}) - \eta \mathbf{g}). \\ \mathbf{x}' &= \arg\min_{\mathbf{z} \in \mathcal{K}} \left\{ B_{\mathcal{R}}(\mathbf{z}, \mathbf{y}) \right\}. \end{aligned}$$

   In general this is a convex optimization problem and can be solved efficiently; however the method described in this paper is really useful when this operation can be carried very efficiently (say linear time).

   For example, if $\mathcal{R}(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|_2^2$, where $\|\cdot\|_2$ is the usual Euclidean ℓ_2 norm, then $B_{\mathcal{R}}(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2$, and the Bregman update and projection operations reduce to the usual projected gradient algorithm:

$$\mathbf{x}' \;=\; \arg\min_{\mathbf{z} \in \mathcal{K}} \frac{1}{2}\|(\mathbf{x} - \eta \mathbf{g}) - \mathbf{z}\|_2^2.$$

The above assumptions imply the following lemma:

Lemma 4 *For all $\mathbf{x} \in \mathcal{K}$, and $\mathbf{x}^\star$ the minimizer of F, we have $F(\mathbf{x}) - F(\mathbf{x}^\star) \leq \frac{2G^2}{\lambda}$.*

Proof For any $\mathbf{x} \in \mathcal{K}$, let $\mathbf{g} \in \partial F(\mathbf{x})$ be a subgradient of F at $\mathbf{x}$ such that $\|\mathbf{g}\|_\star \leq G$ (the existence of $\mathbf{g}$ is guaranteed by assumption 3a). Then by the convexity of F, we have $F(\mathbf{x}) - F(\mathbf{x}^\star) \leq \mathbf{g} \cdot (\mathbf{x} - \mathbf{x}^\star)$, so that by the Cauchy-Schwarz inequality, we have $F(\mathbf{x}) - F(\mathbf{x}^\star) \leq G\|\mathbf{x} - \mathbf{x}^\star\|$. But assumption 1 and 2 imply that

$$F(\mathbf{x}) - F(\mathbf{x}^\star) \;\geq\; \lambda B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}) \;\geq\; \frac{\lambda}{2}\|\mathbf{x}^\star - \mathbf{x}\|^2.$$

Putting these together, we get that $\|\mathbf{x} - \mathbf{x}^\star\| \leq \frac{2G}{\lambda}$. Finally, we have

$$F(\mathbf{x}) - F(\mathbf{x}^\star) \;\leq\; G\|\mathbf{x} - \mathbf{x}^\star\| \;\leq\; \frac{2G^2}{\lambda}.$$

■

2.2 Online Convex Optimization and Regret

Recall the setting of online convex optimization given in the introduction. In each round $t = 1, 2, \ldots, T$, a decision-maker needs to choose a point $\mathbf{x}_t \in \mathcal{K}$, a convex set. Then nature provides a convex cost function $f_t : \mathcal{K} \to \mathbb{R}$, and the decision-maker incurs the cost $f_t(\mathbf{x}_t)$. The (adversarial) regret of the decision-maker is defined to be

$$\text{AdversarialRegret} \;:=\; \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x}). \tag{4}$$

When the cost functions f_t are drawn i.i.d. from some unknown distribution D, (stochastic) regret is traditionally defined measured with respect to the expected cost function, $F(\mathbf{x}) = \mathbb{E}_D[f_1(\mathbf{x})]$:

$$\text{StochasticRegret} \;:=\; \mathop{\mathbb{E}}_{D}\left[\sum_{t=1}^{T} F(\mathbf{x}_t)\right] - T \min_{\mathbf{x}\in\mathcal{K}} F(\mathbf{x}). \tag{5}$$

In either case, if the decision-making algorithm is randomized, then we measure the performance by the expectation of the regret taken over the random seed of the algorithm in addition to any other randomness.

When cost functions are drawn i.i.d. from an unknown distribution D, it is easy to check that

$$\mathop{\mathbb{E}}_{D}\left[\min_{\mathbf{x}\in\mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x})\right] \;\leq\; \min_{\mathbf{x}\in\mathcal{K}} \mathop{\mathbb{E}}_{D}\left[\sum_{t=1}^{T} f_t(\mathbf{x})\right],$$

by considering the point $\mathbf{x}^\star = \arg\min_{\mathbf{x}\in\mathcal{K}} \mathbb{E}_D\left[\sum_{t=1}^{T} f_t(\mathbf{x})\right]$. So

$$\mathop{\mathbb{E}}_{D}[\text{AdversarialRegret}] \;\geq\; \text{StochasticRegret}.$$

Thus, for the purpose of proving lower bounds on the regret (expected regret in the case of randomized algorithms), it suffices to prove such bounds for StochasticRegret. We prove such lower bounds in Section 5. For notational convenience, henceforth the term "regret" refers to StochasticRegret.

3. The Optimal Algorithm and its Analysis

Our algorithm is an extension of stochastic gradient descent. The new feature is the introduction of "epochs" inside of which standard stochastic gradient descent is used, but in each consecutive epoch the learning rate decreases exponentially.

Algorithm 1 Epoch-GD

Input: parameters η_1, T_1 and total time T.
Initialize $\mathbf{x}_1^1 \in \mathcal{K}$ arbitrarily, and set $k = 1$.
while $\sum_{i=1}^{k} T_i \leq T$ **do**
  // Start epoch k
  for $t = 1$ to T_k **do**
    Query the gradient oracle at $\mathbf{x}_t^k$ to obtain $\hat{\mathbf{g}}_t$
    Update

$$\mathbf{y}_{t+1}^k = \nabla \mathcal{R}^\star(\nabla \mathcal{R}(\mathbf{x}_t^k) - \eta_k \hat{\mathbf{g}}_t),$$

$$\mathbf{x}_{t+1}^k = \arg\min_{\mathbf{x} \in \mathcal{K}} \left\{ B_{\mathcal{R}}(\mathbf{x}, \mathbf{y}_{t+1}^k) \right\}.$$

  end for
  Set $\mathbf{x}_1^{k+1} = \frac{1}{T_k} \sum_{t=1}^{T_k} \mathbf{x}_t^k$
  Set $T_{k+1} \leftarrow 2T_k$ and $\eta_{k+1} \leftarrow \eta_k/2$.
  Set $k \leftarrow k + 1$
end while
return $\mathbf{x}_1^k$.

Our main result is the following theorem, which immediately implies Theorem 1.

Theorem 5 *Set the parameters $T_1 = 4$ and $\eta_1 = \frac{1}{\lambda}$ in the* Epoch-GD *algorithm. The final point $\mathbf{x}_1^k$ returned by the algorithm has the property that*

$$\mathbb{E}[F(\mathbf{x}_1^k)] - F(\mathbf{x}^\star) \leq \frac{16G^2}{\lambda T}.$$

The total number of gradient updates is at most T.

The intra-epoch use of online mirror decent is analyzed using the following lemma, which follows the ideas of Zinkevich (2003); Bartlett et al. (2007), and given here for completeness:

Lemma 6 *Starting from an arbitrary point $\mathbf{x}_1 \in \mathcal{K}$, apply T iterations of the update*

$$\mathbf{y}_{t+1} = \nabla \mathcal{R}^\star(\nabla \mathcal{R}(\mathbf{x}_t) - \eta \hat{\mathbf{g}}_t),$$

$$\mathbf{x}_{t+1} \;=\; \arg\min_{\mathbf{x}\in\mathcal{K}} \; \{B_{\mathcal{R}}(\mathbf{x}, \mathbf{y}_{t+1})\}\,.$$

Then for any point $\mathbf{x}^\star \in \mathcal{K}$, we have

$$\sum_{t=1}^{T} \hat{\mathbf{g}}_t \cdot (\mathbf{x}_t - \mathbf{x}^\star) \;\le\; \frac{\eta}{2}\sum_{t=1}^{T} \|\hat{\mathbf{g}}_t\|_\star^2 + \frac{B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}_1)}{\eta}.$$

Proof Since $\nabla\mathcal{R}^\star = \nabla\mathcal{R}^{-1}$, we have $\nabla\mathcal{R}(\mathbf{y}_{t+1}) = \nabla\mathcal{R}(\mathbf{x}_t) - \eta\hat{\mathbf{g}}_t$. Thus, we have

$$\begin{aligned}
\hat{\mathbf{g}}_t \cdot (\mathbf{x}_t - \mathbf{x}^\star) \;&=\; \frac{1}{\eta}(\nabla\mathcal{R}(\mathbf{y}_{t+1}) - \nabla\mathcal{R}(\mathbf{x}_t)) \cdot (\mathbf{x}^\star - \mathbf{x}_t) \\
&=\; \frac{1}{\eta}[B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}_t) - B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{y}_{t+1}) + B_{\mathcal{R}}(\mathbf{x}_t, \mathbf{y}_{t+1})] \qquad \text{via (2)} \\
&\le\; \frac{1}{\eta}[B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}_t) - B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}_{t+1}) + B_{\mathcal{R}}(\mathbf{x}_t, \mathbf{y}_{t+1})],
\end{aligned}$$

where the last inequality follows from the Pythagorean Theorem for Bregman divergences (see Bregman, 1967): since $\mathbf{x}_{t+1}$ is the Bregman projection of $\mathbf{y}_{t+1}$ on the convex set $\mathcal{K}$, and $\mathbf{x}^\star \in K$, we have $B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}_{t+1}) \le B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{y}_{t+1})$. Summing over all iterations, and using the non-negativity of the Bregman divergence, we get

$$\begin{aligned}
\sum_{t=1}^{T} \hat{\mathbf{g}}_t \cdot (\mathbf{x}_t - \mathbf{x}^\star) \;&\le\; \frac{1}{\eta}[B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}_1) - B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}_{T+1})] + \frac{1}{\eta}\sum_{t=1}^{T} B_{\mathcal{R}}(\mathbf{x}_t, \mathbf{y}_{t+1})] \\
&\le\; \frac{1}{\eta}B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}_1) + \frac{1}{\eta}\sum_{t=1}^{T} B_{\mathcal{R}}(\mathbf{x}_t, \mathbf{y}_{t+1}). \qquad (6)
\end{aligned}$$

We proceed to bound $B_{\mathcal{R}}(\mathbf{x}_t, \mathbf{y}_{t+1})$. By the definition of Bregman divergence, we get

$$\begin{aligned}
B_{\mathcal{R}}(\mathbf{x}_t, \mathbf{y}_{t+1}) + B_{\mathcal{R}}(\mathbf{y}_{t+1}, \mathbf{x}_t) \;&=\; (\nabla\mathcal{R}(\mathbf{x}_t) - \nabla\mathcal{R}(\mathbf{y}_{t+1})) \cdot (\mathbf{x}_t - \mathbf{y}_{t+1}) \\
&=\; \eta\hat{\mathbf{g}}_t \cdot (\mathbf{x}_t - \mathbf{y}_{t+1}) \\
&\le\; \frac{1}{2}\eta^2\|\hat{\mathbf{g}}_t\|_\star^2 + \frac{1}{2}\|\mathbf{x}_t - \mathbf{y}_{t+1}\|^2.
\end{aligned}$$

The last inequality uses the fact that since $\|\cdot\|$ and $\|\cdot\|_\star$ are dual norms, we have

$$\mathbf{w}\cdot\mathbf{v} \;\le\; \|\mathbf{w}\|_\star\|\mathbf{v}\| \;\le\; \frac{1}{2}\|\mathbf{w}\|_\star^2 + \frac{1}{2}\|\mathbf{v}\|^2.$$

Thus, by our assumption $B_{\mathcal{R}}(\mathbf{x}, \mathbf{y}) \ge \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|^2$, we have

$$B_{\mathcal{R}}(\mathbf{x}_t, \mathbf{y}_{t+1}) \;\le\; \frac{1}{2}\eta^2\|\hat{\mathbf{g}}_t\|_\star^2 + \frac{1}{2}\|\mathbf{x}_t - \mathbf{y}_{t+1}\|^2 - B_{\mathcal{R}}(\mathbf{y}_{t+1}, \mathbf{x}_t) \;\le\; \frac{\eta^2}{2}\|\hat{\mathbf{g}}_t\|_\star^2.$$

Plugging this bound into (6), we get the required bound. ■

Lemma 7 *Starting from an arbitrary point $\mathbf{x}_1 \in \mathcal{K}$, apply T iterations of the update*

$$\begin{aligned}\mathbf{y}_{t+1} &= \nabla\mathcal{R}^\star(\nabla\mathcal{R}(\mathbf{x}_t) - \eta\hat{\mathbf{g}}_t),\\ \mathbf{x}_{t+1} &= \arg\min_{\mathbf{x}\in\mathcal{K}} \left\{B_{\mathcal{R}}(\mathbf{x}, \mathbf{y}_{t+1})\right\},\end{aligned}$$

where $\hat{\mathbf{g}}_t$ is an unbiased estimator for a subgradient $\mathbf{g}_t$ of F at $\mathbf{x}_t$ satisfying assumption 3a. Then for any point $\mathbf{x}^\star \in \mathcal{K}$, we have

$$\frac{1}{T}\,\mathbb{E}\left[\sum_{t=1}^{T} F(\mathbf{x}_t)\right] - F(\mathbf{x}^\star) \;\le\; \frac{\eta G^2}{2} + \frac{B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}_1)}{\eta T}.$$

By convexity of F, we have the same bound for $\mathbb{E}[F(\bar{\mathbf{x}})] - F(\mathbf{x}^\star)$, where $\bar{\mathbf{x}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_t$.

Proof For a random variable X measurable w.r.t. the randomness until round t, let $\mathbb{E}_{t-1}[X]$ denote its expectation conditioned on the randomness until round $t-1$. By the convexity of F, we get

$$F(\mathbf{x}_t) - F(\mathbf{x}^\star) \;\le\; \mathbf{g}_t \cdot (\mathbf{x}_t - \mathbf{x}^\star) = \mathop{\mathbb{E}}_{t-1}[\hat{\mathbf{g}}_t \cdot (\mathbf{x}_t - \mathbf{x}^\star)],$$

since $\mathbb{E}_{t-1}[\hat{\mathbf{g}}_t] = \mathbf{g}_t$ and $\mathbb{E}_{t-1}[\mathbf{x}_t] = \mathbf{x}_t$. Taking expectations of the inequality, we get that

$$\mathbb{E}[F(\mathbf{x}_t)] - F(\mathbf{x}^\star) \;\le\; \mathbb{E}[\hat{\mathbf{g}}_t \cdot (\mathbf{x}_t - \mathbf{x}^\star)].$$

Summing up over all $t = 1, 2, \ldots, T$, and taking the expectation on both sides of the inequality in Lemma 6, we get the required bound. ∎

Define $V_k = \frac{G^2}{2^{k-2}\lambda}$ and $\Delta_k = F(\mathbf{x}_1^k) - F(\mathbf{x}^\star)$. The choice of initial parameters $T_1 = 4$ and $\eta_1 = \frac{1}{\lambda}$ was specified in Theorem 5, and by definition $T_k = T_1 2^{k-1}$ and $\eta_k = \eta_1 2^{-(k-1)}$. Using Lemma 7 we prove the following key lemma:

Lemma 8 *For any k, we have $\mathbb{E}[\Delta_k] \le V_k$.*

Proof We prove this by induction on k. The claim is true for $k = 1$ since $\Delta_k \le \frac{2G^2}{\lambda}$ by Lemma 4. Assume that $\mathbb{E}[\Delta_k] \le V_k$ for some $k \ge 1$ and now we prove it for $k+1$. For a random variable X measurable w.r.t. the randomness defined up to epoch $k+1$, let $\mathbb{E}_k[X]$ denote its expectation conditioned on all the randomness up to epoch k. By Lemma 7 we have

$$\begin{aligned}\mathop{\mathbb{E}}_{k}[F(\mathbf{x}_1^{k+1})] - F(\mathbf{x}^\star) &\le \frac{\eta_k G^2}{2} + \frac{B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}_1^k)}{\eta_k T_k}\\ &\le \frac{\eta_k G^2}{2} + \frac{\Delta_k}{\eta_k T_k \lambda},\end{aligned}$$

since $\Delta_k = F(\mathbf{x}_1^k) - F(\mathbf{x}^\star) \ge \lambda B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}_1^k)$ by λ-strong convexity of F with respect to $B_{\mathcal{R}}$. Hence, we get

$$\mathbb{E}[\Delta_{k+1}] \;\le\; \frac{\eta_k G^2}{2} + \frac{\mathbb{E}[\Delta_k]}{\eta_k T_k \lambda} \;\le\; \frac{\eta_k G^2}{2} + \frac{V_k}{\eta_k T_k \lambda} \;=\; \frac{\eta_1 G^2}{2^k} + \frac{V_k}{\eta_1 T_1 \lambda} \;=\; V_{k+1},$$

as required. The second inequality uses the induction hypothesis, and the last two equalities use the definition of V_k, the equalities $T_k = T_1 2^{k-1}$ and $\eta_k = \eta_1 2^{-(k-1)}$, and the initial values $T_1 = 4$ and $\eta_1 = \frac{1}{\lambda}$. ■

We can now prove our main theorem:

Proof [Proof of Theorem 5.] The number of epochs made are given by the largest value of k satisfying $\sum_{i=1}^{k} T_i \leq T$, i.e.,

$$\sum_{i=1}^{k} 2^{i-1} T_1 \;=\; (2^k - 1) T_1 \;\leq\; T.$$

This value is $k^\dagger = \lfloor \log_2(\frac{T}{T_1}+1) \rfloor$. The final point output by the algorithm is $\mathbf{x}_1^{k^\dagger+1}$. Applying Lemma 8 to $k^\dagger + 1$ we get

$$\mathbb{E}[F(\mathbf{x}_1^{k^\dagger+1})] - F(\mathbf{x}^\star) \;=\; \mathbb{E}[\Delta_{k^\dagger+1}] \;\leq\; V_{k^\dagger+1} \;=\; \frac{G^2}{2^{k^\dagger-1}\lambda} \;\leq\; \frac{4T_1G^2}{\lambda T} \;=\; \frac{16G^2}{\lambda T},$$

as claimed. The while loop in the algorithm ensures that the total number of gradient updates is naturally bounded by T. ■

3.1 A Randomized Stopping Variant

In this section we describe a pure stochastic gradient descent algorithm with random step sizes that has the same (expected) rate of convergence.

Our main theorem of this section is:

Theorem 9 *Set the parameters $T_1 = 4$ and $\eta_1 = \frac{1}{\lambda}$ in the* Random-Step-GD *algorithm. The final point $\mathbf{x}_t$ returned by the algorithm has the property that*

$$\mathbb{E}[F(\mathbf{x}_t)] - F(\mathbf{x}^\star) \leq \frac{16G^2}{\lambda T}$$

where the expectation is taken over the gradient estimates as well as the internal randomization of the algorithm.

Proof The proof of this theorem is on the same lines as before. In particular, we divide up the entire time period into (possibly overlapping) epochs. For $k = 1, 2, \ldots$, epoch k consists of the following sequence of T_k rounds: $\{B_k, B_k + 1, \ldots, B_k + T_k - 1\}$. Note that B_{k+1} is a uniformly random time in the above sequence. The behavior of the algorithm in rounds $B_k, B_k + 1, \ldots, B_{k+1} - 1$ can be simulated by the following thought-experiment: starting with $\mathbf{x}_{B_k}$, run T_k iterations of stochastic mirror descent, i.e.,

$$\nabla \mathcal{R}(\mathbf{y}_{t+1}) = \nabla \mathcal{R}(\mathbf{x}_t) - \eta_k \hat{\mathbf{g}}_t,$$

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x} \in \mathcal{K}} \left\{ B_{\mathcal{R}}(\mathbf{x}, \mathbf{y}_{t+1}) \right\},$$

Algorithm 2 Random-Step-GD

Input: parameters η_1, T_1 and total time T.
Initialize $\mathbf{x}_1 \in \mathcal{K}$ arbitrarily, and set $k = 1$, $B_1 = 1$, $B_2 \in \{1, 2, \ldots, T_1\}$ uniformly at random.
for $t = 1, 2, \ldots$ **do**
  if $t = B_{k+1}$ **then**
    Set $k \leftarrow k + 1$.
    Set $T_k \leftarrow 2T_{k-1}$ and $\eta_k \leftarrow \eta_{k-1}/2$.
    Set $B_{k+1} \in \{B_k, B_k + 1, \ldots, B_k + T_k - 1\}$ uniformly at random.
    if $B_{k+1} > T$ **then**
      Break **for** loop.
    end if
  end if
  Query the gradient oracle at $\mathbf{x}_t$ to obtain $\hat{\mathbf{g}}_t$.
  Update

$$\begin{aligned} \mathbf{y}_{t+1} &= \nabla \mathcal{R}^\star(\nabla \mathcal{R}(\mathbf{x}_t) - \eta_k \hat{\mathbf{g}}_t) \\ \mathbf{x}_{t+1} &= \arg\min_{\mathbf{x} \in \mathcal{K}} \{B_{\mathcal{R}}(\mathbf{x}, \mathbf{y}_{t+1})\} \end{aligned}$$

end for
return $\mathbf{x}_t$.

for $t = B_k, \ldots, B_k + T_k - 1$, and return $\mathbf{x}_{B_{k+1}}$. Conditioning on $\mathbf{x}_{B_{k-1}}$, and taking expectations, since B_{k+1} was chosen uniformly at random from a sequence of T_k rounds, we get

$$\mathbb{E}[F(\mathbf{x}_{B_{k+1}})] = \frac{1}{T_k} \sum_{t=B_k}^{B_k+T_k-1} \mathbb{E}[F(\mathbf{x}_t)].$$

Now, by Lemma 7, we conclude that

$$\mathbb{E}[F(\mathbf{x}_{B_{k+1}})] - F(\mathbf{x}^\star) \leq \frac{\eta_k G^2}{2} + \frac{B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}_{B_k})}{\eta_k T_k}. \tag{7}$$

Now, just as before, we define $V_k = \frac{G^2}{2^{k-2}\lambda}$ and $\Delta_k = F(\mathbf{x}_{B_k}) - F(\mathbf{x}^\star)$. Recall the choice of initial parameters $T_1 = 4$ and $\eta_1 = \frac{1}{\lambda}$ as specified in Theorem 9. Now, arguing exactly as in Lemma 8

Lemma 10 *For any k, we have $\mathbb{E}[\Delta_k] \leq V_k$.*

Proof We prove this by induction on k. The claim is true for $k = 1$ since $\Delta_k \leq \frac{2G^2}{\lambda}$ by Lemma 4. Assume that $\mathbb{E}[\Delta_k] \leq V_k$ for some $k \geq 1$ and now we prove it for $k + 1$. For a random variable X measurable w.r.t. the randomness defined up to epoch $k + 1$, let $\mathbb{E}_k[X]$ denote its expectation conditioned on all the randomness up to epoch k. By Lemma 7 we

have

$$\begin{aligned}\mathop{\mathbb{E}}_{k}[F(\mathbf{x}_1^{k+1})] - F(\mathbf{x}^\star) &\leq \frac{\eta_k G^2}{2} + \frac{B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}_1^k)}{\eta_k T_k} \\ &\leq \frac{\eta_k G^2}{2} + \frac{\Delta_k}{\eta_k T_k \lambda},\end{aligned}$$

since $\Delta_k = F(\mathbf{x}_1^k) - F(\mathbf{x}^\star) \geq \lambda B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}_1^k)$ by λ-strong convexity of F with respect to $\mathcal{R}$. Hence, we get

$$\mathbb{E}[\Delta_{k+1}] \leq \frac{\eta_k G^2}{2} + \frac{\mathbb{E}[\Delta_k]}{\eta_k T_k \lambda} \leq \frac{\eta_k G^2}{2} + \frac{V_k}{\eta_k T_k \lambda} = \frac{\eta_1 G^2}{2^k} + \frac{V_k}{\eta_1 T_1 \lambda} = V_{k+1},$$

as required. As before, the second inequality above uses the induction hypothesis, and the last two equalities use the definition of V_k, the equalities $T_k = T_1 2^{k-1}$ and $\eta_k = \eta_1 2^{-(k-1)}$, and the initial values $T_1 = 4$ and $\eta_1 = \frac{1}{\lambda}$. ■

Now just as in the proof of Theorem 5, since we output $\mathbf{x}_t = \mathbf{x}_{B_{k^\dagger+1}}$, where $k^\dagger$, the number of epochs, is at least[4] $\lfloor \log_2(\frac{T}{T_1} + 1) \rfloor$, we conclude that $\mathbb{E}[F(\mathbf{x}_t)] - F(\mathbf{x}^\star) \leq \frac{16G^2}{\lambda T}$ as required. ■

4. High Probability Bounds

While Epoch-GD algorithm has a $O(\frac{1}{T})$ rate of convergence, this bound is only on the expected excess cost of the final solution. In applications we usually need the rate of convergence to hold with high probability. Markov's inequality immediately implies that with probability $1-\delta$, the actual excess cost is at most a factor of $\frac{1}{\delta}$ times the stated bound. While this guarantee might be acceptable for not too small values of δ, it becomes useless when δ gets really small.

There are two ways of remedying this. The easy way applies if it is possible to evaluate F efficiently at any given point. Then we can divide the budget of T gradient updates into $\ell = \log_2(1/\delta)$ consecutive intervals of $\frac{T}{\ell}$ rounds each, and run independent copies of Epoch-GD in each. Finally, we take the ℓ solutions obtained, and output the best one (i.e., the one with the minimum F value). Applying Markov's inequality to every run of Epoch-GD, with probability at least $1/2$, we obtain a point with excess cost at most $\frac{64G^2\ell}{\lambda T} = \frac{64G^2 \log_2(1/\delta)}{\lambda T}$, and so with probability at least $1 - 2^{-\ell} = 1 - \delta$, the best point has excess cost at most $\frac{64G^2 \log_2(1/\delta)}{\lambda T}$. This finishes the description of the easy way to obtain high probability bounds.

The easy way fails if it is not possible to evaluate F efficiently at any given point. For this situation, we now describe how using essentially the same algorithm with slightly different parameters, we can get a high probability guarantee on the quality of the solution. To prove

4. Here we have an inequality rather than an equality as in the previous algorithm since we may have more epochs due to the random early stopping of epochs.

the high probability bound, we need to make the stronger assumption 3b, i.e., for all points $\mathbf{x} \in \mathcal{K}$, the stochastic subgradient $\hat{\mathbf{g}}$ output by the oracle satisfies $\mathbb{E}[\exp(\frac{\|\hat{\mathbf{g}}\|_\star^2}{G^2})] \leq e$.

The only differences in the new algorithm, dubbed Epoch-GD-Proj, are as follows. The algorithm takes a new parameter, D_1. The update in line 7 requires a projection onto a smaller set, and becomes

$$\begin{aligned} \mathbf{y}_{t+1}^k &= \nabla\mathcal{R}^\star(\nabla\mathcal{R}(\mathbf{x}_t^k) - \eta_k\hat{\mathbf{g}}_t), \\ \mathbf{x}_{t+1}^k &= \operatorname*{arg\,min}_{\mathbf{x}\in\mathcal{K}\cap\mathcal{B}(\mathbf{x}_1^k, D_k)} \left\{B_{\mathcal{R}}(\mathbf{x}, \mathbf{y}_{t+1}^k)\right\}. \end{aligned} \tag{8}$$

Here $\mathcal{B}(\mathbf{x}, D) = \{\mathbf{y} : \|\mathbf{y} - \mathbf{x}\| \leq D\}$ denotes the ball of radius D around the point $\mathbf{x}$, and D_k is computed in the algorithm. The update in line 10 now becomes:

$$\text{Set } T_{k+1} \leftarrow 2T_k,\ \eta_{k+1} \leftarrow \eta_k/2, \text{ and } D_{k+1} \leftarrow D_k/\sqrt{2}.$$

Since the intersection of two convex sets is also a convex set, the above projection can be computed via a convex program.[5] A completely analogous version of Random-Step-GD is an easy extension; it enjoys the same high probability bound as given below. We prove the following high probability result, which in turn directly implies Theorem 3.

Theorem 11 *Given $\delta > 0$ for success probability $1-\delta$, set $\tilde{\delta} = \frac{\delta}{k^\dagger}$ for $k^\dagger = \lfloor \log_2(\frac{T}{450}+1)\rfloor$. Set the parameters $T_1 = 450$, $\eta_1 = \frac{1}{3\lambda}$, and $D_1 = 2G\sqrt{\frac{\log(2/\tilde{\delta})}{\lambda}}$ in the* Epoch-GD-Proj *algorithm. The final point $\mathbf{x}_1^k$ returned by the algorithm has the property that with probability at least $1-\delta$, we have*

$$F(\mathbf{x}_1^k) - F(\mathbf{x}^\star) \leq \frac{1800G^2\log(2/\tilde{\delta})}{\lambda T}.$$

The total number of gradient updates is at most T.

The following lemma is analogous to Lemma 7, but provides a high probability guarantee.

Lemma 12 *For any given $\mathbf{x}^\star \in \mathcal{K}$, let D be an upper bound on $\|\mathbf{x}_1 - \mathbf{x}^\star\|$. Apply T iterations of the update*

$$\begin{aligned} \mathbf{y}_{t+1} &= \nabla\mathcal{R}^\star(\nabla\mathcal{R}(\mathbf{x}_t) - \eta\hat{\mathbf{g}}_t), \\ \mathbf{x}_{t+1} &= \operatorname*{arg\,min}_{\mathbf{x}\in\mathcal{K}\cap\mathcal{B}(\mathbf{x}_1, D)} \{B_{\mathcal{R}}(\mathbf{x}, \mathbf{y}_{t+1})\}. \end{aligned}$$

5. It was suggested to us by a referee that in practice, computing x_{t+1}^k by taking a Bregman projection on $\mathcal{K}\cap B'(\mathbf{x}_1^k, D_k)$, where $B'(\mathbf{x}, r) = \{\mathbf{y} : B_{\mathcal{R}}(\mathbf{y}, \mathbf{x}) \leq D^2/2\}$ is the "Bregman ball of radius D around the point $\mathbf{x}$", might be more efficient than a projection on $\mathcal{K}\cap\mathcal{B}(\mathbf{x}_1^k, D_k)$. This depends on the application, but it is easy to see that all the proofs (and thus the high-probability guarantees) go through simply because the Bregman balls are a subset of the norm $\|\cdot\|$ balls, i.e., $B'(\mathbf{x}, D) \subseteq \mathcal{B}(\mathbf{x}, D)$, by the strong-convexity of $\mathcal{R}$ w.r.t. the norm $\|\cdot\|$. We prefer to leave the update in terms of the norm $\|\cdot\|$ balls since generally speaking projections on larger sets are easier; the specific choice can be tailored to the application.

where $\hat{\mathbf{g}}_t$ is an unbiased estimator for the subgradient of F at $\mathbf{x}_t$ satisfying assumption 3b. Then for any $\delta \in (0,1)$, with probability at least $1-\delta$ we have

$$\frac{1}{T}\sum_{t=1}^{T} F(\mathbf{x}_t) - F(\mathbf{x}^\star) \;\le\; \frac{\eta G^2 \log(2/\delta)}{2} + \frac{B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}_1)}{\eta T} + \frac{4GD\sqrt{3\log(2/\delta)}}{\sqrt{T}}.$$

By the convexity of F, the same bound also holds for $F(\bar{\mathbf{x}}) - F(\mathbf{x}^\star)$, where $\bar{\mathbf{x}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_t$.

Proof First, note that since the oracle uses independent randomness in every call to it, we conclude that that for all t, $\hat{\mathbf{g}}_t$ is independent of $\hat{\mathbf{g}}_1, \ldots, \hat{\mathbf{g}}_{t-1}$ given $\mathbf{x}_t$, and thus by assumption 3b we have

$$\mathop{\mathbb{E}}_{t}\left[\exp\left(\frac{\|\hat{\mathbf{g}}_t\|_\star^2}{G^2}\right)\right] \;=\; \mathbb{E}\left[\exp\left(\frac{\|\hat{\mathbf{g}}_t\|_\star^2}{G^2}\right)\middle|\,\mathbf{x}_t\right] \;\le\; \exp(1). \tag{9}$$

The proof proceeds on similar lines as that of Lemma 7, except that we use high-probability bounds rather than expected bounds. Using the same notation as in the proof of Lemma 7, let $\mathbb{E}_{t-1}[\hat{\mathbf{g}}_t] = \mathbf{g}_t$, a subgradient of F at $\mathbf{x}_t$. We now need to bound $\sum_{t=1}^{T}\hat{\mathbf{g}}_t \cdot (\mathbf{x}_t - \mathbf{x}^\star)$ in terms of $\sum_{t=1}^{T}\mathbf{g}_t\cdot(\mathbf{x}_t-\mathbf{x}^\star)$, and $\sum_{t=1}^{T}\|\hat{\mathbf{g}}_t\|_\star^2$ in terms of G^2T.

As before, $\mathbb{E}_{t-1}[\hat{\mathbf{g}}_t\cdot(\mathbf{x}_t-\mathbf{x}^\star)] = \mathbf{g}_t\cdot(\mathbf{x}_t-\mathbf{x}^\star)$, and thus the following defines a martingale difference sequence:

$$X_t \;:=\; \mathbf{g}_t\cdot(\mathbf{x}_t-\mathbf{x}^\star) - \hat{\mathbf{g}}_t\cdot(\mathbf{x}_t-\mathbf{x}^\star).$$

Note that $\|\mathbf{g}_t\|_\star = \|\,\mathbb{E}_{t-1}[\hat{\mathbf{g}}_t]\|_\star \le \mathbb{E}_{t-1}[\|\hat{\mathbf{g}}_t\|_\star] \le G$, and so we can bound $|X_t|$ as follows:

$$|X_t| \le \|\mathbf{g}_t\|_\star\|\mathbf{x}_t-\mathbf{x}^\star\| + \|\hat{\mathbf{g}}_t\|_\star\|\mathbf{x}_t-\mathbf{x}^\star\| \;\le\; 2GD + 2D\|\hat{\mathbf{g}}_t\|_\star,$$

where the last inequality uses the fact that since $\mathbf{x}^\star, \mathbf{x}_t \in \mathcal{B}(\mathbf{x}_1, D)$, we have $\|\mathbf{x}_t-\mathbf{x}^\star\| \le \|\mathbf{x}_t-\mathbf{x}_1\| + \|\mathbf{x}_1-\mathbf{x}^\star\| \le 2D$. This implies that

$$\mathop{\mathbb{E}}_{t}\left[\exp\left(\frac{X_t^2}{16G^2D^2}\right)\right] \;\le\; \mathop{\mathbb{E}}_{t}\left[\exp\left(\frac{4D^2(2G^2+2\|\hat{\mathbf{g}}_t\|_\star^2)}{16G^2D^2}\right)\right] \;\le\; \exp(\tfrac{1}{2})\sqrt{\mathop{\mathbb{E}}_{t}\left[\exp\left(\frac{\|\hat{\mathbf{g}}_t\|_\star^2}{G^2}\right)\right]} \;\le\; \exp(1),$$

where the second inequality follows by Jensen's inequality and the inequality $(a+b)^2 \le 2a^2+2b^2$, and the last by (9).

By Lemma 14, with probability at least $1-\delta/2$, we have $\sum_{t=1}^{T}X_t \le 4GD\sqrt{3\log(2/\delta)T}$, which implies that

$$\frac{1}{T}\sum_{t=1}^{T}F(\mathbf{x}_t)-F(\mathbf{x}^\star) \;\le\; \frac{1}{T}\sum_{t=1}^{T}\mathbf{g}_t\cdot(\mathbf{x}_t-\mathbf{x}^\star) - \frac{1}{T}\sum_{t=1}^{T}\hat{\mathbf{g}}_t\cdot(\mathbf{x}_t-\mathbf{x}^\star) \;\le\; \frac{4GD\sqrt{3\log(2/\delta)}}{\sqrt{T}}, \tag{10}$$

where the first inequality follows by convexity of F.

Next, consider $\mathbb{E}[\exp(\frac{\sum_{t=1}^{T}\|\hat{\mathbf{g}}_t\|_\star^2}{G^2})]$. We can upper bound this as follows:

$$\begin{aligned}
\mathbb{E}\left[\exp\left(\frac{\sum_{t=1}^{T}\|\hat{\mathbf{g}}_t\|_\star^2}{G^2}\right)\right] \;&=\; \mathbb{E}\left[\mathop{\mathbb{E}}_{T}\left[\exp\left(\frac{\sum_{t=1}^{T}\|\hat{\mathbf{g}}_t\|_\star^2}{G^2}\right)\right]\right] \\
&=\; \mathbb{E}\left[\exp\left(\frac{\sum_{t=1}^{T-1}\|\hat{\mathbf{g}}_t\|_\star^2}{G^2}\right)\mathop{\mathbb{E}}_{T}\left[\exp\left(\frac{\|\hat{\mathbf{g}}_T\|_\star^2}{G^2}\right)\right]\right] \\
&\le\; \mathbb{E}\left[\exp\left(\frac{\sum_{t=1}^{T-1}\|\hat{\mathbf{g}}_t\|_\star^2}{G^2}\right)\cdot\exp(1)\right]
\end{aligned}$$

by (9). Continuing inductively, we conclude that $\mathbb{E}[\exp(\frac{\sum_{t=1}^{T}\|\hat{\mathbf{g}}_t\|_\star^2}{G^2})] \leq \exp(T)$, which implies (via Markov's inequality) that with probability at least $1-\delta/2$, we have

$$\sum_{t=1}^{T}\|\hat{\mathbf{g}}_t\|_\star^2 \;\leq\; G^2 T\log(2/\delta). \tag{11}$$

Then, by using Lemma 6 and inequalities (10) and (11), we get the claimed bound. ■

We now prove the analogue of Lemma 8. In this case, the result holds with high probability. As before, define $V_k = \frac{G^2}{2^{k-2}\lambda}$ and $\Delta_k = F(\mathbf{x}_1^k) - F(\mathbf{x}^\star)$. Recall the choice of initial parameters $T_1 = 450$ and $\eta_1 = \frac{1}{3\lambda}$ as specified in Theorem 3.

Lemma 13 *For any k, with probability $(1-\tilde{\delta})^{k-1}$ we have $\Delta_k \leq V_k \log(2/\tilde{\delta})$.*

Proof For notational convenience, in the following we define:

$$L \;:=\; \log(2/\tilde{\delta}).$$

We prove the lemma by induction on k. The claim is true for $k=1$ since $\Delta_k \leq \frac{2G^2L}{\lambda}$ by Lemma 4. Assume that $\Delta_k \leq V_k L$ for some $k \geq 1$ with probability at least $(1-\tilde{\delta})^{k-1}$ and now we prove the corresponding statement for $k+1$. We condition on the event that $\Delta_k \leq V_k L$. Since $\Delta_k \geq \frac{\lambda}{2}\|\mathbf{x}_1^k - \mathbf{x}^\star\|^2$ by λ-strong convexity, this conditioning implies that $\|\mathbf{x}_1^k - \mathbf{x}^\star\| \leq \sqrt{2V_kL/\lambda} = D_k$. So Lemma 12 applies with $D = D_k$ and hence we have with probability at least $1-\tilde{\delta}$,

$$\begin{aligned}
\Delta_{k+1} \;&=\; F(\mathbf{x}_1^{k+1}) - F(\mathbf{x}^\star) \\
&\leq\; \frac{\eta_k G^2 L}{2} + \frac{B_{\mathcal{R}}(\mathbf{x}^\star, \mathbf{x}_1^k)}{\eta_k T_k} + 10G\sqrt{\frac{V_kL}{\lambda T_k}} && \text{(by Lemma 12)} \\
&\leq\; \frac{\eta_k G^2 L}{2} + \frac{\Delta_k}{\eta_k T_k} + 10G\sqrt{\frac{V_kL}{\lambda T_k}} && \text{(by λ-strong convexity of F)} \\
&\leq\; \frac{\eta_k G^2 L}{2} + \frac{V_kL}{\eta_k T_k\lambda} + 10G\sqrt{\frac{V_kL}{\lambda T_k}} && \text{(by induction hypothesis)} \\
&=\; \frac{\eta_1 G^2 L}{2^k} + \frac{V_kL}{\eta_1 T_1\lambda} + 10G\sqrt{\frac{V_kL}{\lambda T_1 2^{k-1}}} && \text{(by definition of T_k, η_k)} \\
&=\; \frac{V_kL}{12} + \frac{V_kL}{150} + \frac{V_k\sqrt{L}}{3} && \text{(using values of T_1, η_1, V_k)} \\
&\leq\; \frac{V_kL}{2} \;=\; V_{k+1}L.
\end{aligned}$$

Factoring in the conditioned event, which happens with probability at least $(1-\tilde{\delta})^{k-1}$, overall, we get that $\Delta_{k+1} \leq V_{k+1}$ with probability at least $(1-\tilde{\delta})^k$. ■

We can now prove our high probability theorem:

Proof [Theorem 11] Proceeding exactly as in the proof of Theorem 1, we get that final epoch is $k^\dagger = \lfloor \log_2(\frac{T}{T_1}+1) \rfloor$. The final point output is $\mathbf{x}_1^{k^\dagger+1}$. By Lemma 13, we have with probability at least $(1-\tilde{\delta})^{k^\dagger}$ that

$$\begin{aligned} F(\mathbf{x}_1^{k^\dagger+1}) - F(\mathbf{x}^\star) \;&=\; \Delta_{k^\dagger+1} \;\leq\; V_{k^\dagger+1}\log(2/\tilde{\delta}) \\ &= \frac{G^2\log(2/\tilde{\delta})}{2^{k^\dagger-1}\lambda} \;\leq\; \frac{4T_1G^2\log(2/\tilde{\delta})}{\lambda T} \;=\; \frac{1800G^2\log(2/\tilde{\delta})}{\lambda T}, \end{aligned}$$

as claimed. Since $\tilde{\delta} = \frac{\delta}{k^\dagger}$, and hence $(1-\tilde{\delta})^{k^\dagger} \geq 1-\delta$ as needed. The while loop in the algorithm ensures that the total number of gradient updates is bounded by T. ■

In the analysis, we used the following well-known martingale inequality, a restatement of Lemma 2 of Lan et al. (2012). Here, $\mathbb{E}_t[\cdot]$ denotes the expectation at time t conditioned on all the randomness till time $t-1$.

Lemma 14 *Let $X_1, \ldots, X_T$ be a martingale difference sequence, i.e., $\mathbb{E}_t[X_t] = 0$ for all t. Suppose that for some values σ_t, for $t = 1, 2, \ldots, T$, we have $\mathbb{E}_t[\exp(\frac{X_t^2}{\sigma_t^2})] \leq \exp(1)$. Then with probability at least $1-\delta$, we have*

$$\sum_{t=1}^{T} X_t \;\leq\; \sqrt{3\log(1/\delta)\sum_{t=1}^{T}\sigma_t^2}.$$

5. Lower Bounds on Stochastic Strongly Convex Optimization

In this section we prove Theorem 2 and show that any algorithm (deterministic or randomized) for online stochastic strongly-convex optimization must have $\Omega(\log(T))$ regret on some distribution. We start by proving a $\Omega(\log T)$ lower bound for the case when the cost functions are 1-strongly convex with respect to the Euclidean norm and the gradient oracle is 1-bounded, and fine tune these parameters in the next subsection by way of reduction.

In our analysis, we need the following standard lemma, which we reprove here for completeness. Here, for two distributions P, P' defined on the same probability space, $d_{TV}(P, P')$ is the total variation distance, i.e.

$$d_{TV}(P, P') = \sup_A |P(A) - P'(A)|$$

where the supremum ranges over all events A in the probability space.

Let B_p be the Bernoulli distribution on $\{0, 1\}$ with probability of obtaining 1 equal to p. Let B_p^n denote the product measure on $\{0, 1\}^n$ induced by taking n independent Bernoulli trials according to B_p (thus, $B_p^1 = B_p$).

Lemma 15 *Let $p, p' \in [\frac{1}{4}, \frac{3}{4}]$ such that $|p' - p| \leq 1/8$. Then*

$$d_{TV}(B_p^n, B_{p'}^n) \leq \frac{1}{2}\sqrt{(p'-p)^2 n}.$$

Proof Pinsker's inequality says that $d_{TV}(B_p^n, B_{p'}^n) \leq \sqrt{\frac{1}{2}\mathrm{RE}(B_p^n \| B_{p'}^n)}$, where $\mathrm{RE}(B_p^n \| B_{p'}^n) = \mathbb{E}_{X \sim B_p^n}[\ln \frac{B_p^n(X)}{B_{p'}^n(X)}]$ is the relative entropy between B_p^n and $B_{p'}^n$. To bound $\mathrm{RE}(B_p^n \| B_{p'}^n)$, note that the additivity of the relative entropy for product measures implies that

$$\mathrm{RE}(B_p^n \| B_{p'}^n) = n\mathrm{RE}(B_p \| B_{p'}) = n\left[p \log\left(\frac{p}{p'}\right) + (1-p)\log\left(\frac{1-p}{1-p'}\right)\right]. \tag{12}$$

Without loss of generality, assume that $p' \geq p$, and let $p' = p + \varepsilon$, where $0 \leq \varepsilon \leq 1/8$. Using the Taylor series expansion of $\log(1+x)$, we get the following bound

$$p \log\left(\frac{p}{p'}\right) + (1-p)\log\left(\frac{1-p}{1-p'}\right) = \sum_{i=1}^{\infty}\left[\frac{(-1)^i}{p^{i-1}} + \frac{1}{(1-p)^{i-1}}\right]\varepsilon^i \leq \sum_{i=2}^{\infty} 4^{i-1}\varepsilon^i \leq \frac{\varepsilon^2}{2},$$

for $\varepsilon \leq 1/8$. Plugging this (12) and using Pinsker's inequality, we get the stated bound. ■

We now turn to showing our lower bound on expected regret. We consider the following online stochastic strongly-convex optimization setting: the domain is $\mathcal{K} = [0,1]$. For every $p \in [\frac{1}{4}, \frac{3}{4}]$, define a distribution over strongly-convex cost functions parameterized by p as follows: choose $X \in \{0,1\}$ from B_p, and return the cost function

$$f(x) = (x - X)^2.$$

With some abuse of notation, we use B_p to denote this distribution over cost functions.

Under distribution B_p, the expected cost function F is

$$F(x) := \mathbf{E}[f(x)] = p(x-1)^2 + (1-p)x^2 = x^2 + 2px + p = (x-p)^2 + c_p,$$

where $c_p = p - p^2$. The optimal point is therefore $x^\star = p$, with expected cost c_p. The regret for playing a point x (i.e., excess cost over the minimal expected cost) is

$$F(x) - F(x^\star) = (x-p)^2 + c_p - c_p = (x-p)^2.$$

Now let $\mathcal{A}$ be a deterministic[6] algorithm for online stochastic strongly-convex optimization. Since the cost functions until time t are specified by a bit string $X \in \{0,1\}^{t-1}$ (i.e., the cost function at time t is $(x - X_t)^2$), we can interpret the algorithm as a function that takes a variable length bit string, and produces a point in $[0,1]$, i.e., with some abuse of notation,

$$\mathcal{A} : \{0,1\}^{\leq T} \longrightarrow [0,1],$$

where $\{0,1\}^{\leq T}$ is the set of all bit strings of length up to T.

Now suppose the cost functions are drawn from B_p. Fix a round t. Let X be the $t-1$ bit string specifying the cost functions so far. Note that X has distribution B_p^{t-1}. For notational convenience, denote by $\Pr_p[\cdot]$ and $\mathbb{E}_p[\cdot]$ the probability of an event and the expectation of a random variable when the cost functions are drawn from B_p, and since these are defined by the bit string X, they are computed over the product measure B_p^{t-1}.

6. We will remove the deterministic requirement shortly and allow randomized algorithms.

Let the point played by $\mathcal{A}$ at time t be $x_t = \mathcal{A}(X)$. The regret (conditioned on the choice of X) in round t is then

$$\text{regret}_t \;:=\; (\mathcal{A}(X) - p)^2,$$

and thus the expected (over the choice of X) regret of $\mathcal{A}$ in round t is $\mathbb{E}_p[\text{regret}_t] = \mathbb{E}_p[(\mathcal{A}(X) - p)^2]$.

We now show that for any round t, for two distributions over cost functions B_p and $B_{p'}$ that are close (in terms of $|p - p'|$), but not too close, the regret of $\mathcal{A}$ on at least one of the two distributions must be large.

Lemma 16 *Fix a round t. Let $\varepsilon \le \frac{1}{8\sqrt{t}}$ be a parameter. Let $p, p' \in [\frac{1}{4}, \frac{3}{4}]$ such that $2\varepsilon \le |p - p'| \le 4\varepsilon$. Then we have*

$$\mathop{\mathbb{E}}_{p}[\mathit{regret}_t] + \mathop{\mathbb{E}}_{p'}[\mathit{regret}_t] \;\ge\; \frac{1}{4}\varepsilon^2.$$

Proof Assume without loss of generality that $p' \ge p + 2\varepsilon$. Let X and X' be $(t-1)$-bit vectors parameterizing the cost functions drawn from B_p^{t-1} and $B_{p'}^{t-1}$ respectively. Then

$$\mathop{\mathbb{E}}_{p}[\text{regret}_t] + \mathop{\mathbb{E}}_{p'}[\text{regret}_t] \;=\; \mathop{\mathbb{E}}_{p}[(\mathcal{A}(X) - p)^2] + \mathop{\mathbb{E}}_{p'}[(\mathcal{A}(X') - p')^2].$$

Now suppose the stated bound does not hold. Then by Markov's inequality, we have

$$\Pr_p[(\mathcal{A}(X) - p)^2 < \varepsilon^2] \;\ge\; 3/4,$$

or in other words,

$$\Pr_p[\mathcal{A}(X) < p + \varepsilon] \;\ge\; 3/4. \tag{13}$$

Similarly, we can show that

$$\Pr_{p'}[\mathcal{A}(X') > p + \varepsilon] \;\ge\; 3/4, \tag{14}$$

since $p' \ge p + 2\varepsilon$. Now define the event

$$A \;:=\; \{Y \in \{0,1\}^{t-1} :\; \mathcal{A}(Y) > p + \varepsilon\}.$$

Now (13) implies that $\Pr_p(A) < 1/4$ and (14) implies that $\Pr_{p'}(A) \ge 3/4$. But then by Lemma 15 we have

$$\begin{aligned}\frac{1}{2} \;<\; |\Pr_p(A) - \Pr_{p'}(A)| \;\le\; d_{TV}(B_p^{t-1}, B_{p'}^{t-1}) \;&\le\; \frac{1}{2}\sqrt{(p'-p)^2(t-1)} \\ &\le\; \frac{1}{2}\sqrt{16\varepsilon^2(t-1)} \;\le\; \frac{1}{4},\end{aligned}$$

a contradiction. ∎

We now show how to remove the deterministic requirement on $\mathcal{A}$:

Corollary 17 *The bound of Lemma 16 holds even if $\mathcal{A}$ is randomized:*

$$\mathop{\mathbb{E}}_{p,R}[\mathit{regret}_t] + \mathop{\mathbb{E}}_{p',R}[\mathit{regret}_t] \;\geq\; \frac{1}{4}\varepsilon^2,$$

where $\mathbb{E}_{p,R}[\cdot]$ denotes the expectation computed over the random seed R of the algorithm as well as the randomness in the cost functions.

Proof Fixing the random seed R of $\mathcal{A}$, we get a deterministic algorithm, and then Lemma 16 gives the following bound on the sum of the conditional expected regrets:

$$\mathop{\mathbb{E}}_{p}[\mathrm{regret}_t|R] + \mathop{\mathbb{E}}_{p'}[\mathrm{regret}_t|R] \;\geq\; \frac{1}{4}\varepsilon^2.$$

Now taking expectations over the random seed R, we get the desired bound. ■

Thus, from now on we allow $\mathcal{A}$ to be randomized. We now show the desired lower bound on the expected regret:

Theorem 18 *The expected regret for algorithm $\mathcal{A}$ is at least $\Omega(\log(T))$.*

Proof We prove this by showing that there is one value of $p \in [\frac{1}{4}, \frac{3}{4}]$ such that regret of $\mathcal{A}$ when cost functions are drawn from B_p is at least $\Omega(\log(T))$.

We assume that T is of the form $16 + 16^2 + \cdots 16^k = \frac{1}{15}(16^{k+1} - 16)$ for some integer k: if it isn't, we ignore all rounds $t > T'$, where $T' = \frac{1}{15}(16^{k^\star+1} - 16)$ for $k^\star = \lfloor \log_{16}(15T + 16) - 1 \rfloor$, and show that in the first T' rounds the algorithm can be made to have $\Omega(\log(T))$ regret. We now divide the time periods $t = 1, 2, \ldots, T'$ into consecutive epochs of length $16, 16^2, \ldots, 16^{k^\star}$. Thus, epoch k, denoted E_k, has length 16^k, and consists of the time periods $t = \frac{1}{15}(16^k - 16) + 1, \ldots, \frac{1}{15}(16^{k+1} - 16)$. We prove the following lemma momentarily:

Lemma 19 *There exists a collection of nested intervals, $[\frac{1}{4}, \frac{3}{4}] \supseteq I_1 \supseteq I_2 \supseteq I_3 \supseteq \cdots$, such that interval I_k corresponds to epoch k, with the property that I_k has length $4^{-(k+3)}$, and for every $p \in I_k$, for at least half the rounds t in epoch k, algorithm $\mathcal{A}$ has $\mathbb{E}_{p,R}[\mathit{regret}_t] \geq \frac{1}{8} \cdot 16^{-(k+3)}$.*

As a consequence of this lemma, we get that there is a value of $p \in \bigcap_k I_k$ such that in every epoch k, the total regret is

$$\sum_{t \in E_k} \frac{1}{8} \cdot 16^{-(k+3)} \geq \frac{1}{2} 16^k \cdot \frac{1}{8} \cdot 16^{-(k+3)} = \frac{1}{16^4}.$$

Thus, the regret in every epoch is $\Omega(1)$. Since there are $k^\star = \Theta(\log(T))$ epochs total, the regret of the algorithm is at least $\Omega(\log(T))$. ■

We now turn to prove Lemma 19.

Proof [Lemma 19] We build the nested collection of intervals iteratively as follows. For notational convenience, define I_0 to be some arbitrary interval of length 4^{-3} inside $[\frac{1}{4}, \frac{3}{4}]$. Suppose for some $k \geq 0$ we have found the interval $I_k = [a, a + 4^{-(k+3)}]$. We want to find

the interval I_{k+1} now. For this, divide up I_k into 4 equal quarters of length $\varepsilon = 4^{-(k+4)}$, and consider the first and fourth quarters, viz. $L = [a, a + 4^{-(k+4)}]$ and $R = [a + 3 \cdot 4^{-(k+4)}, a + 4^{-(k+3)}]$. We now show that one of L or R is a valid choice for I_{k+1}, and so the construction can proceed.

Suppose L is not a valid choice for I_{k+1}, because there is some point $p \in L$ such that for more than half the rounds t in E_{k+1}, we have $\mathbb{E}_{p,R}[\text{regret}_t] < 16^{-(k+1)}$. Then we show that R is a valid choice for I_{k+1} as follows. Let $H = \{t \in E_{k+1} : \mathbb{E}_{p,R}[\text{regret}_t] < \frac{1}{8} \cdot 16^{-(k+4)}\}$. Now, we claim that for all $p' \in R$, and all $t \in H$, we must have $\mathbb{E}_{p',R}[\text{regret}_t] > \frac{1}{8} \cdot 16^{-(k+4)}$, which would imply that R is a valid choice for I_{k+1}, since by assumption, $|H| \geq \frac{1}{2}|E_{k+1}|$.

To show this we apply Lemma 16. Fix any $p' \in R$ and $t \in H$. First, note that $\varepsilon = 4^{-(k+4)} \leq \frac{1}{8\sqrt{t}}$, since $t \leq 16^{k+2}$. Next, we have $p' - p \geq 2\varepsilon$ (since we excluded the middle two quarters of I_k), and $|p - p'| \leq 4\varepsilon$ (since I_k has length $4^{-(k+3)}$). Then Lemma 16 implies that

$$\mathop{\mathbb{E}}_{p,R}[\text{regret}_t] + \mathop{\mathbb{E}}_{p',R}[\text{regret}_t] \geq \frac{1}{4} \cdot 16^{-(k+4)},$$

which implies that $\mathbb{E}_{p',R}[\text{regret}_t] \geq \frac{1}{8} \cdot 16^{-(k+4)}$ since $\mathbb{E}_{p,R}[\text{regret}_t] < \frac{1}{8} \cdot 16^{-(k+4)}$, as required. ■

5.1 Dependence on the Gradient Bound and on Strong Convexity

A simple corollary of the previous proof gives us tight lower bounds in terms of the natural parameters of the problem: the strong-convexity parameter λ and the upper bound on the norm of the subgradients G. The following Corollary implies Theorem 2.

Corollary 20 *For any algorithm $\mathcal{A}$, there is distribution over λ-strongly convex cost functions over a bounded domain $\mathcal{K} \subset \mathbb{R}$ with gradients bounded in norm by G such that the expected regret of $\mathcal{A}$ is $\Omega\left(\frac{G^2 \log(T)}{\lambda}\right)$.*

Proof The online convex optimization setting we design is very similar: let $\lambda, G \geq 0$ be given parameters. The domain is $\mathcal{K} = [0, \frac{G}{\lambda}]$. In round t, we choose $X_t \in \{0, 1\}$ from B_p, and return

$$f_t(x) = \frac{\lambda}{2}\left(x - \frac{G}{\lambda}X_t\right)^2$$

as the cost function. Notice that the cost functions are always λ-strongly convex, and in addition, for any $x \in \mathcal{K}$, the gradient of the cost function at x is bounded in norm by G.

Denote $x' = \frac{\lambda x}{G}$ to be the scaled decision x, mapping it from $\mathcal{K}$ to $[0, 1]$. The expected cost when playing $x \in \mathcal{K}$ is given by

$$\mathbb{E}[f_t(x)] = \mathop{\mathbb{E}}_{X \sim B_p}\left[\frac{\lambda}{2}\left(x - \frac{G}{\lambda}X_t\right)^2\right] = \frac{G^2}{2\lambda}\,\mathbb{E}[(x' - X_t)^2]. \tag{15}$$

Given an algorithm $\mathcal{A}$ for this online convex optimization instance, we derive another algorithm, $\mathcal{A}'$, which plays points $x' \in \mathcal{K}' = [0, 1]$ and receives the cost function $(x' - X_t)^2$ in round t (i.e., the setting considered in Section 5). When $\mathcal{A}$ plays x_t in round t and obtains

cost function $\frac{\lambda}{2}\left(x - \frac{G}{\lambda}X_t\right)^2$, the algorithm $\mathcal{A}'$ plays the point $x'_t = \frac{\lambda}{G}x_t$ and receives the cost function $(x' - X_t)^2$.

The optimum point for the setting of $\mathcal{A}$ is $\frac{G}{\lambda}p$, with expected cost $\frac{G^2}{2\lambda}$ times the expected cost for the optimum point p for the setting of $\mathcal{A}'$. By equation (15), the cost of $\mathcal{A}$ is $\frac{G^2}{2\lambda}$ times that of $\mathcal{A}'$. Hence, the regret of $\mathcal{A}$ is $\frac{G^2}{2\lambda}$ times that of $\mathcal{A}'$.

By Theorem 18, there is a value of p such that the expected regret of $\mathcal{A}'$ is $\Omega(\log T)$, and hence the expected regret of $\mathcal{A}$ is $\Omega\left(\frac{G^2 \log(T)}{\lambda}\right)$, as required. ■

6. Conclusions

We have given an algorithm for stochastic strongly-convex optimization with an optimal rate of convergence $O(\frac{1}{T})$. The Epoch-GD algorithm has an appealing feature of returning the average of the most recent points (rather than all points visited by the algorithm as in previous approaches). This is an intuitive feature which, as demonstrated by Rakhlin et al. (2012), works well in practice for important applications such as support vector machine training.

Our analysis deviates from the common template of designing a regret minimization algorithm and then using online-to-batch conversion. In fact, we show that the latter approach is inherently suboptimal by our new lower bound on the regret of online algorithms for stochastic cost functions. This combination of results formally shows that the batch stochastic setting is strictly easier than its online counterpart, giving us tighter bounds.

A few questions remain open. The high-probability bound algorithm Epoch-GD-Proj has an extra factor of $O(\log\log(T))$ in its convergence rate. Is it possible to devise an algorithm that has $O(\frac{1}{T})$ convergence rate with high probability? We believe the answer is yes; the $O(\log\log(T))$ is just an artifact of the analysis. In fact, as we mention in Section 4, if it is possible to evaluate F efficiently at any given point, then this dependence can be removed. Also, our lower bound proof is somewhat involved. Are there easier information theoretic arguments that give similar lower bounds?

Acknowledgements

We gratefully acknowledge the help of two anonymous reviewers that gave insightful feedback and significantly helped shape the final version of this manuscript.

References

Jacob Abernethy, Alekh Agarwal, Peter L. Bartlett, and Alexander Rakhlin. A stochastic view of optimal regret through minimax duality. In *COLT*, 2009.

Alekh Agarwal, Peter L. Bartlett, Pradeep D. Ravikumar, and Martin J. Wainwright. Information-theoretic lower bounds on the oracle complexity of stochastic convex optimization. *IEEE Transactions on Information Theory*, 58(5):3235–3249, 2012.

Peter L. Bartlett, Elad Hazan, and Alexander Rakhlin. Adaptive online gradient descent. In *NIPS*, 2007.

Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, September 1999. ISBN 1886529000.

Jonathan M. Borwein and Adrian S. Lewis. *Convex Analysis and Nonlinear Optimization*. Springer, 2006.

Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *NIPS*, 2007.

Lev M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217, 1967.

Nicolò Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

Saeed Ghadimi and Guanghui Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization. In *Optimization Online*, 2010. URL `http://www.optimization-online.org/DB_HTML/2010/07/2669.html`.

Elad Hazan and Satyen Kale. An optimal algorithm for stochastic strongly-convex optimization. In *arXiv:1006.2425v1*, June 2010. URL `http://arxiv.org/abs/1006.2425`.

Elad Hazan and Satyen Kale. Beyond the regret minimization barrier: an optimal algorithm for stochastic strongly-convex optimization. In *COLT*, 2011.

Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

Anatoli Juditsky and Yuri Nesterov. Primal-dual subgradient methods for minimizing uniformly convex functions. August 2010. URL `http://hal.archives-ouvertes.fr/docs/00/50/89/33/PDF/Strong-hal.pdf`.

Guanghui Lan, Arkadi Nemirovski, and Alexander Shapiro. Validation analysis of mirror descent stochastic approximation method. *Math. Program.*, 134(2):425–458, 2012.

Arkadi S. Nemirovski and David B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. John Wiley UK/USA, 1983.

Erik Ordentlich and Thomas M. Cover. The cost of achieving the best portfolio in hindsight. *Mathematics of Operations Research*, 23:960–982, November 1998.

Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *ICML*, 2012.

Shai Shalev-Shwartz and Nathan Srebro. SVM optimization: inverse dependence on training set size. In *ICML*, 2008.

Shai Shalev-Shwartz, Ohad Shamir, Karthik Sridharan, and Nati Srebro. Stochastic convex optimization. In *COLT*, 2009.

Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *ICML*, 2013.

Eiji Takimoto and Manfred K. Warmuth. The minimax strategy for Gaussian density estimation. In *COLT*, 2000.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, 2003.

Journal of Machine Learning Research 15 (2014) 2513-2532 Submitted 5/13; Revised 2/14; Published 7/14

One-Shot-Learning Gesture Recognition using HOG-HOF Features

Jakub Konečný* KUBO.KONECNY@GMAIL.COM
Michal Hagara MICHAL.HAGARA@GMAIL.COM
Korešpondenčný Matematický Seminár
Comenius University
Bratislava, Slovakia

Editors: Isabelle Guyon, Vassilis Athitsos, and Sergio Escalera

Abstract

The purpose of this paper is to describe one-shot-learning gesture recognition systems developed on the *ChaLearn Gesture Dataset* (ChaLearn). We use RGB and depth images and combine appearance (Histograms of Oriented Gradients) and motion descriptors (Histogram of Optical Flow) for parallel temporal segmentation and recognition. The Quadratic-Chi distance family is used to measure differences between histograms to capture cross-bin relationships. We also propose a new algorithm for trimming videos—to remove all the unimportant frames from videos. We present two methods that use a combination of HOG-HOF descriptors together with variants of a Dynamic Time Warping technique. Both methods outperform other published methods and help narrow the gap between human performance and algorithms on this task. The code is publicly available in the MLOSS repository.

Keywords: ChaLearn, histogram of oriented gradients, histogram of optical flow, dynamic time warping

1. Introduction

Gesture recognition can be seen as a way for computers to understand human body language. Improving state-of-the-art algorithms for gesture recognition facilitates human-computer communication beyond primitive text user interfaces or GUIs (graphical user interfaces). With rapidly improving comprehension of human gestures we can start building NUIs (natural user interfaces) for controlling computers or robots. With the availability of such technologies, conventional input devices, such as a keyboard or mouse, could be replaced in situations in which they are inconvenient in future. Other applications of gesture recognition include sign language recognition, socially assistive robotics and game technology.

In this paper, we focus on the one-shot learning gesture recognition problem, in particular the *ChaLearn Gesture Dataset* (ChaLearn). The data set was released jointly with a competition, where the goal was to develop a system capable of learning to recognize new categories of gestures from a single training example of each gesture. The large data set of hand and arm gestures was pre-recorded using an infrared sensor, Kinect^{TM}, providing both RGB and depth images (Guyon et al., 2012, 2013).

∗. Current affiliation: University of Edinburgh

The purpose of this work is to describe methods developed during the *ChaLearn Gesture Challenge* by the Turtle Tamers team (authors of this paper). We finished in 2^{nd} place in round 2 and were invited to present our solution at the International Conference on Pattern Recognition 2012, Tsukuba, Japan. The code has been made publicly available in the MLOSS repository.[1]

Since the goal of the challenge was to provide solid baseline methods for this data set, our methods were specifically tailored for this particular competition and data set. Hence, they lack a certain generality, and we discuss and suggest changes for more general settings later.

The rest of this work is organised as follows. Related work is summarized in Section 2. In Section 3 we describe the data set and the problem in detail. In Section 4 we focus on the preprocessing needed to overcome some of the problems in the data set. Section 5 covers feature representation, using Histogram of Oriented Gradients and Histogram of Optical Flow, as well as a method used to compare similarities between these representations. In Section 6 we describe the actual algorithms, and in Section 7 we briefly describe algorithms of other participants and compare their results with ours, as well as with other published works. In Section 8 we summarize our paper and suggest an area for future work.

2. Related Work

In this section we provide a brief literature review in the area of gesture and action recognition and motivate our choices of models.

One possible approach to the problem of gesture recognition consists of analyzing motion descriptors obtained from video. Ikizler and Forsyth (2007) use the output of Human Motion Capture systems in combination with Hidden Markov Models. Wu et al. (2012) use Extended Motion History Image as a motion descriptor and apply the method to the *ChaLearn Gesture Dataset*. They fuse dual modalities inherent in the Kinect sensor using Multiview Spectral Embedding (Xia et al., 2010) in a physically meaningful manner.

A popular recent approach is to use Conditional Random Fields (CRF). Wang et al. (2006) introduce the discriminative hidden state approach, in which they combine the ability of CRFs to use long range dependencies and the ability of Hidden Markov Models to model latent structure. More recent work (Chatzis et al., 2012) describes joint segmentation and classification of sequences in the framework of CRFs. The method outperforms other popular related approaches with no sacrifices in terms of the imposed computational costs.

An evolution of Bag-of-Words (Lewis, 1998), a method used in document analysis, where each document is represented using the apparition frequency of each word in a dictionary, is one of the most popular in Computer Vision. In the image domain, these words become visual elements of a certain visual vocabulary. First, each image is decomposed into a large set of patches, obtaining a numeric descriptor. This can be done, for example, using SIFT (Lowe, 1999), or SURF (Bay et al., 2006). A set of N representative visual words are selected by means of a clustering process over the descriptors in all images. Once the visual vocabulary is defined, each image can be represented by a global histogram containing the frequencies of visual words. Finally, this histogram can be used as input for any classification technique. Extensions to image sequences have been proposed, the most popular being

1. The code is available at https://mloss.org/software/view/448.

Space-Time Interest Points (Laptev, 2005). Wang et al. (2009) have evaluated a number of feature descriptors and bag-of-features models for action recognition. This study concluded that different sampling strategies and feature descriptors were needed to achieve the best results on alternative action data sets. Recently an extension of these models to the RGB-D images, with a new depth descriptor was introduced by Hernández-Vela et al. (2012).

The methods outlined above usually ignore particular spatial position of a descriptor. We wanted to exploit the specifics of the data set, particularly the fact that user position does not change within the same batch, thus also the important parts of the same gestures will occur roughly at the same place. We use a combination of appearance descriptor, Histogram of Oriented Gradients (Dalal and Triggs, 2005) and local motion direction descriptor, Histogram of Optical Flow (Kanade and Lucas, 1981). We adopted Quadratic-Chi distance (Pele and Werman, 2010) to measure differences between these histograms. This approach only works well at high resolutions of descriptors. An alternative may be to use a non-linear support vector machine with a χ^2 kernel (Laptev et al., 2008). Another possible feature descriptor that includes spatio-temporal position of features could be HOG3D (Klaser and Marszalek, 2008), which was applied to this specific data set by Fanello et al. (2013).

3. Data and Problem Setting

In this section, we discuss the easy and difficult aspects of the data set and state the goal of the competition.

The purpose of the *ChaLearn Gesture Challenge*[2] was to develop an automated system capable of learning to recognize new categories of gestures from a single training example of each gesture. A large data set of gestures was collected before the competition, which includes more than $50,000$ gestures recorded with the Kinect^{TM} sensor, providing both RGB and depth videos. The resolution of these videos is 240×320 pixels, at 10 frames per second. The gestures are grouped into more than 500 batches of 100 gestures, each batch including 47 sequences of 1 to 5 gestures drawn from small gesture vocabularies from 8 to 14 gestures. The gestures come from over 30 different gesture vocabularies, and were performed by 20 different users.

During the challenge, development batches devel01-480 were available, with truth labels of gestures provided. Batches valid01-20 and final01-40 were provided with labels for only one example of each gesture class in each batch (training set). These batches were used for evaluation purposes. The goal is to automatically predict the gesture labels for the unlabelled gesture sequences (test set). The gesture vocabularies were selected from nine categories corresponding to various settings or applications, such as body language gestures, signals or pantomimes.

Easy aspects of the data set include the use of a fixed camera and the availability of the depth data. Within each batch, there is a single user, only homogeneous recording conditions and a small vocabulary. In every sequence, different gestures are separated by the user returning to a resting position. Gestures are usually performed by hands and arms. In particular, we made use of the fact that the user is always at the same position within one batch.

2. Details and website: http://gesture.chalearn.org/.

The challenging aspects of the data are that within a single batch there is only one labelled example of each gesture. Between different batches there are variations in recording conditions, clothing, skin color and lightning. Some users are less skilled than others, thus there are some errors or omissions in performing the gestures. And in some batches, parts of the body may be occluded.

For the evaluation of results the Levenshtein distance was used, provided as the metric for the competition. That is the minimum number of edit operations (insertion, deletion or substitution) needed to be performed to go from one vector to another. For each unlabelled video, the distance $D(T, L)$ was computed, where T is the truth vector of labels, and L is our predicted vector of labels. This distance is also known as the "edit distance". For example, $D([1, 2], [1]) = 1$, $D([1, 2, 3], [2, 4]) = 2$, $D([1, 2, 3], [3, 2]) = 2$.

The overall score for a batch was computed as a sum of Levenshtein distances divided by the total number of gestures performed in the batch. This is similar to an error rate (but can exceed 1). We multiply the result by a factor of 100 to resemble the fail percentage. For simplicity, in the rest of this work, we call it the error rate.

4. Preprocessing

In this Section we describe how we overcame some of the challenges with the given data set as well as the solutions we propose. In Section 4.1 we focus on depth noise removal. Later we describe the need for trimming the videos—removing set of frames—and the method employed.

4.1 Depth Noise Removal

One of the problems with the given data set is the noise (or missing values) in the depth data. Whenever the Kinect sensor does not receive a response from a particular point, the sensor outputs a 0, resulting in the black areas shown in Figure 1. This noise usually occurs along the edges of objects or, particularly in this data set, humans. The noise is also visible if the object is out of the range of the sensor (0.8 to 3.5 meters).

Figure 1: Examples of depth images with various levels of noise

The level of noise is usually the same within a single batch. However, there is a big difference in the noise level across different batches. If the level is not too high, it looks like 'salt and pepper' noise.

Later, in Section 5, we use Histograms of Oriented Gradients (HOGs), which work best with sharp edges, so we need a filter that preserves the edges. One of the best filters for removing this kind of noise is the median filter, and also has our desired property. Median filter replaces every pixel with the median of pixels in small area around itself. The effect of the median filter is shown in Figure 2. We can see this filter does not erase big areas of noise, however, this is not a problem in our methods. As mentioned earlier, HOG features are sensitive to the edges, but these large areas usually occur along the edges, so the difference in computed features will not be significant.

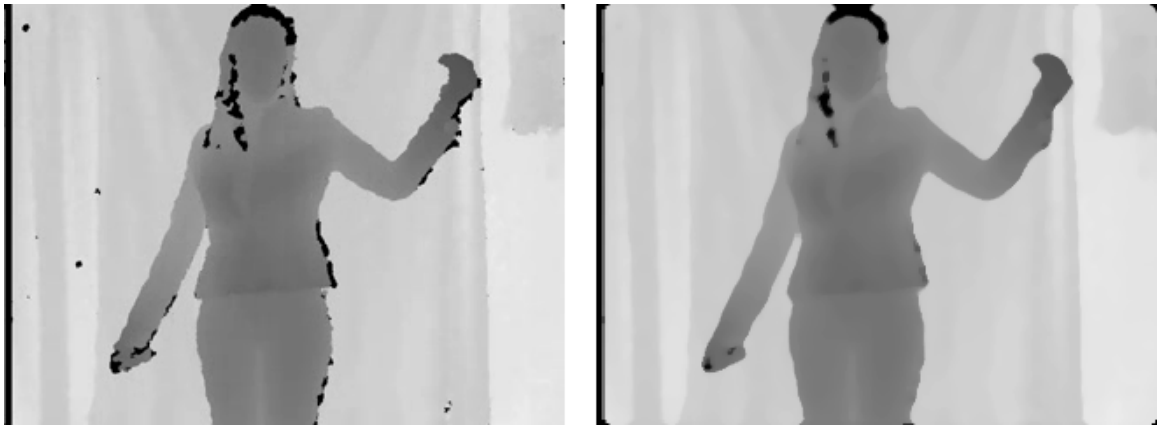


Figure 2: Effect of median filter on depth image

4.2 Trimming

In most batches we can find videos with quite long parts, at the beginning or at the end of the video, where nothing important happens. Sometimes the user is not moving at all, sometimes trying to turn on/off the recorder.[3] Another problem occurring less often is in batches, where gestures are rather static. There is often variation in time the user stays in a particular gesture setting.[4] This is a problem for most possible approaches for tackling the one-shot-learning problem. A solution can be to remove frames from the beginning and end of the videos, as well as any part with too much inactivity.

One possible approach to removing parts of inactivity can be to watch the amount of motion in the video, and remove parts where nothing happens. This is the idea we employed.

A naive but effective way is to take the depth video and compute differences for every pixel between two consecutive frames. Taking depth videos allows us to ignore problems of texture of clothing or background. We then simply count the number of pixels whose change exceeds a given threshold, or we can simply sum the differences. After numerous experiments we ended up with Algorithm 1. Suppose we have a video, n frames long. First we remove the background[5] from individual frames and apply the median filter. Then we do

3. An example is batch devel12, video 23.
4. An example is batch devel39, particularly video 18.
5. Using an algorithm *bgremove* provided in sample code of the Challenge (ChaLearn).

not compute differences of consecutive frames, but rather between frames i and $i+3$. This is to make the motion curve smoother and thus the method more robust. We also found that it was important to even out the amount of motion between, for instance, hand in front of body and hand in front of background. To that end, we set an upper boundary constraint on the difference at 15 (on a scale 0 to 255). Then we computed the actual motion as an average of differences between the chosen frames, as previously described, *above* particular frame, for example

$$\begin{aligned} motion(2) &\leftarrow (mot(1) + mot(2))/2, \\ motion(12) &\leftarrow (mot(9) + mot(10) + mot(11) + mot(12))/4. \end{aligned} \tag{1}$$

In the *mot* variable we store the average change across all pixels. Then we scaled the motion to range $[0, 1]$.

Algorithm 1 Trimming a video

$n \leftarrow length(video)$
$gap \leftarrow 3 \quad maxDiff \leftarrow 15 \quad threshold \leftarrow 0.1 \quad minTrim \leftarrow 5$
for $i = 1 \rightarrow n$ **do**
  $video(i) \leftarrow bgremove(video(i))$ ▷ Background removal
  $video(i) \leftarrow medfilt(video(i))$ ▷ Median filter
end for
for $i = 1 \rightarrow (n - gap)$ **do**
  $diff(i) \leftarrow abs(video(i) - video(i + gap))$
  $diff(i) \leftarrow min\{diff(i), maxDiff\}$
  $mot(i) \leftarrow mean(diff(i))$ ▷ Mean across all pixels
end for
$motion \leftarrow avgMotion(mot)$ ▷ As in Equation 1
$motion \leftarrow scale(motion)$ ▷ Scale motion so its range is 0 to 1
$frames \leftarrow vector(1 : n)$
if $|beginSequence(motion < threshold)| \geq minTrim$ **then**
  $frames \leftarrow trimBegin(frames)$ ▷ Remove all frames
end if
if $|endSequence(motion < threshold)| \geq minTrim$ **then**
  $frames \leftarrow trimEnd(frames)$ ▷ Remove all frames
end if
for all $|sequence(motion < threshold)| > minTrim$ **do**
  $frames \leftarrow trimMiddle(sequence, frames)$ ▷ Remove all frames but $minTrim$
end for
return $video(frames)$

Once we have the motion in the expected range, we can start actually removing frames. At first, we remove sequences from the beginning and the end of the video with motion below a *threshold* (set to 0.1), under the condition that they are of length at least *minTrim* (set to 5) frames. Then we find all sequences in the middle of the video with motion below the *threshold* of length more than 5, and uniformly choose 5 frames to remain in the video.

For example if we were to trim a sequence of length 13, only frames $\{1, 4, 7, 10, 13\}$ would remain. Then we return the video with the remaining frames. Figure 3 illustrates the threshold and the motion computed by this algorithm on a particular video.

One possible modification of this algorithm is in the step in which we scale the motion to the range of $[0, 1]$. In this case, we simply subtract $min(motion)$, and divide by $(max(motion) - min(motion))$. However, especially in videos with 4 or 5 gestures, sometimes large outliers cause problems, because the threshold is too big. Since the motion curve tends to be relatively smooth, instead of choosing $max(motion)$ we could choose the value of the second highest local maximum. This scaling performs slightly better on long videos, but does not work well on short videos. Since, we do not know how many gestures to expect in advance, we used the simpler method.

It is not straightforward to generalize this approach to color videos, since there is no easy way to distinguish the background from the foreground. Additionally, the texture of clothing could cause big problems to this approach. This could be overcome by adding an algorithm that would subtract the background after seeing the whole video, but we have not tried this.

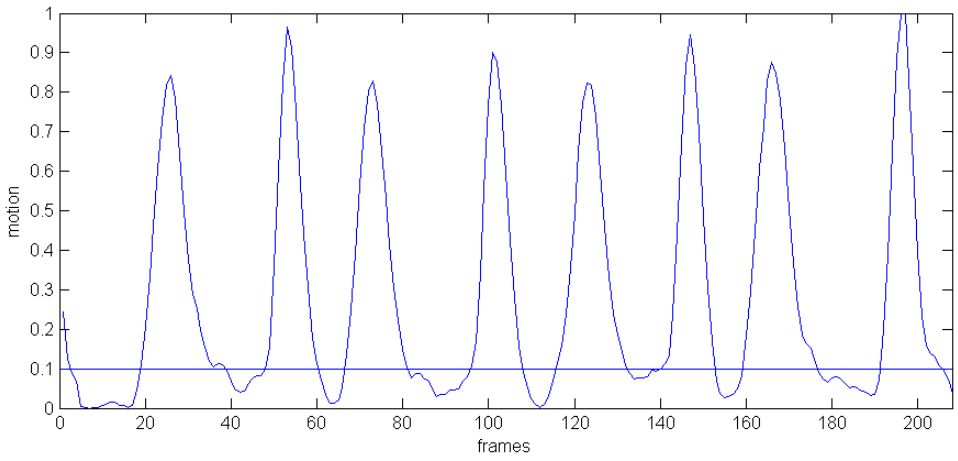


Figure 3: Example of a motion graph, batch devel11, video 32

5. Feature Representation and Distance Measure

In this section, we briefly describe the tools we propose for extracting features. Gestures differ from each other, both in appearance and the amount of motion while performing a particular gesture. A good descriptor of the static part of a gesture is the Histogram of Oriented Gradients, proposed by Dalal and Triggs (2005). A good method for capturing the size and direction of motion is computing the Optical Flow using the Lucas-Kanade method (Kanade and Lucas, 1981; Lucas, 1984) and creating a histogram of flow. Motivation for these choices is explained in Section 2. Finally, we describe the Quadratic-Chi distance family proposed by Pele and Werman (2010) for measuring distances between histograms.

5.1 Histogram of Oriented Gradients

In this section we briefly describe the HOG features. The underlying idea is that the appearance and shape of a local object can often be characterized rather well by the distribution of local intensity gradient (or edge) directions, even without precise knowledge of the corresponding gradient (or edge) positions. In practice this is implemented by dividing the image window into small spatial regions ("cells"), for each cell accumulating a local 1-D histogram of gradient directions (or edge orientations) over the pixels of the cell. It is also useful to contrast-normalize the local responses before using them. This can be done by accumulating a measure of local histogram "energy" over larger spatial regions ("blocks") and using the results to normalize all of the cells in the block.

We used a simple $[-1, 0, 1]$ gradient filter, applied in both directions and discretized the gradient orientations into 16 orientation bins between 0° and 180°. We had cells of size 40×40 pixels and blocks of size 80×80 pixels, each containing 4 cells. The histogram in each cell is normalized with sum of Euclidean norms of histograms in the whole block. Each cell (except cells on the border) belongs to 4 blocks, thus for one cell we have 4 locally normalized histograms, the sum of which is used as the resulting histogram for the cell. Since this method cannot be used to normalize histograms of marginal cells, from 240×320 image we get only 4×6 spatial cells of 16 orientation bins each. Figure 4 provides a visual example of the HOG features at their actual resolution. The space covered is smaller than the original image, but that is not a problem, since the gestures from the data set are not performed on the border of the frames. Dalal and Triggs (2005) conclude, that fine-scale gradients, fine orientation binning, relatively coarse spatial cells, and high-quality local contrast normalization in overlapping descriptor blocks are all important for obtaining good performance.

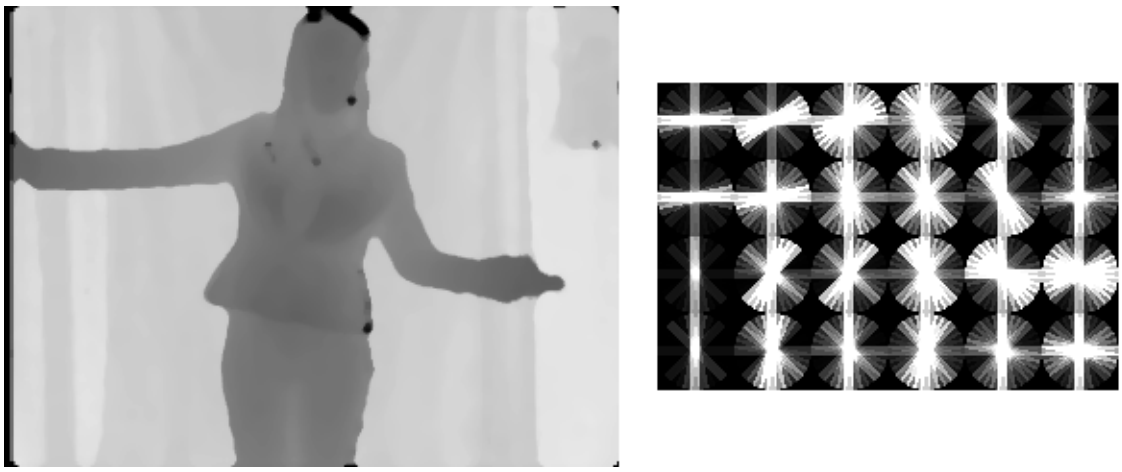


Figure 4: Example visualisation of the HOG features

As in Figure 4, we computed the HOG features from depth images, since it captures only the edges we are interested in, and not textures of clothing and so on. We used the

efficient implementation from Piotr's toolbox (Dollár), function
`hog(image, 40, 16).`

5.2 Histogram of Optical Flow

In this section we describe the general optical flow principle and the Lucas-Kanade method (Kanade and Lucas, 1981; Lucas, 1984) for estimating the actual flow. For details we refer the reader to these works. Here we present only a brief description of the method.

The optical flow methods try to estimate the motion between two images, at times t and $t + \Delta t$ at every position (in our case two consecutive frames of video). In general, the optical flow equation is formulated as a single equation with two variables. All optical flow methods introduce additional conditions for estimating the flow. The Lucas-Kanade method assumes that the flow is essentially constant in a local neighbourhood of the pixel under consideration, and solves the equation for all the pixels in the neighbourhood. The solution is obtained using the least squares principle.

After obtaining the optical flow in every point of the image we divide the image (of 240×320 pixels) to a grid of 6×8 spatial cells. We then put each optical flow vector into one of 16 orientation bins in each spatial cell, and scale them so that they sum to 1 to get a histogram of $6 \times 8 \times 16$ fields. We also tried to scale in each spatial cell separately, and the difference of error rate in our methods on all development batches was less than 0.5. We computed the optical flow from color videos, converted to grayscale, again using efficient implementation of the Flow estimation from Piotr's toolbox (Dollár), function
`optFlowLk(image1, image2, [] , 4, 2, 9e-5);`

5.3 Measuring Distance of the Histograms

Our method relies on making comparisons between pairs of frames in two videos, which requires as a component, to measure differences between histograms. The relatively simple methods based on the sum of bin-to-bin distances suffer from the following limitation: If the number of bins is too small, the measure is not discriminative and if it is too large it is not robust. Distances, that take into account cross-bin relationships, can be both robust and discriminative. With the HOG and HOF feature at the resolution that we selected, simple bin-to-bin comparisons are not robust, as exemplified in Figure 5. Thus we would like a measure that would look into surrounding orientation bins and, after experimenting, also to surrounding spatial cells. Thus we would also like a measure, that would reduce the effect of big differences, and also look into surrounding spatial cells. We adopted the following Quadratic-Chi distance family introduced by Pele and Werman (2010).

Let P and Q be two histograms. Let A be a non-negative symmetric bounded bin-similarity matrix, such that each diagonal element is bigger or equal to every other element in its row. Let $0 \leq m < 1$ be a normalization factor. A Quadratic-Chi histogram distance is defined as:

$$QC_m^A(P,Q) = \sqrt{\sum_{i,j} \left(\frac{(P_i - Q_i)}{\left(\sum_c (P_c + Q_c) A_{ci}\right)^m} \right) \left(\frac{(P_j - Q_j)}{\left(\sum_c (P_c + Q_c) A_{cj}\right)^m} \right) A_{ij}},$$

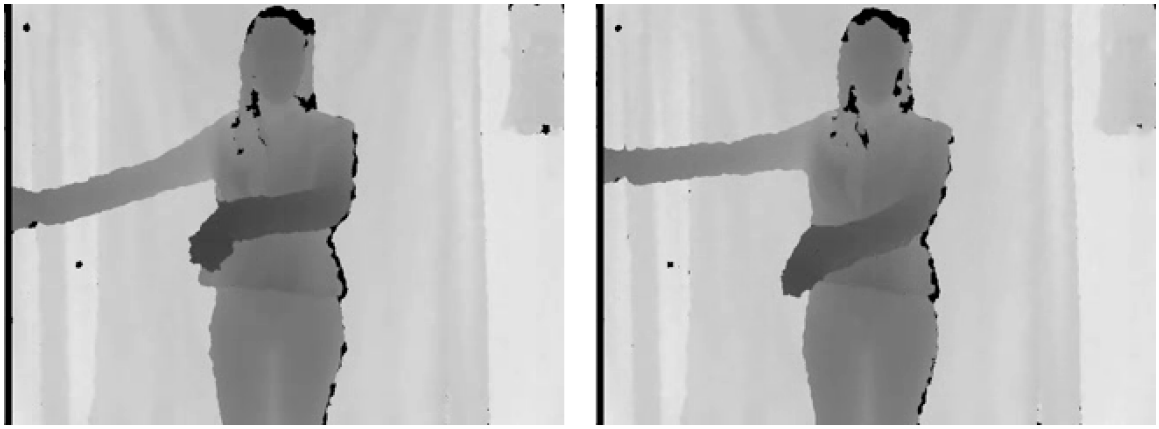


Figure 5: Example of need for cross-bin similarities: the same moment in performance of the same gesture in two different videos. The right hand stays at the same place, the left hand is moving. This illustrates how the same element can result in different neighbouring orientation bins in HOG being big in different cases.

where we define $\frac{0}{0} = 0$. The normalization factor m reduces the effect of big differences (the bigger it is, the bigger reduction; in our methods set to 0.5). While comparing the i^{th} orientation bins of two histograms, we want to look into the matching orientation bins, to 4 surrounding orientation bins (2 left, 2 right), and into the same orientation bins within 8 surrounding spatial cells. MATLAB code for creating the matrix A which captures these properties is in Appendix B.

6. Recognition

In this section we describe the two methods we propose for one-shot-learning gesture recognition. We create a single model and look for the shortest path of a new video through the model in our first method. For the second method we create a separate model for every training video and using sliding frame window to look for similar parts of training videos.

6.1 Single Model—Dynamic Time Warping

In this method (we will call it SM) we use both Histograms of Oriented Gradients and Histograms of Optical Flow and perform temporal segmentation simultaneously with recognition.

At first, we create a model illustrated in Figure 6 for the whole batch in the following way: Every row in the figure represents a single training video. Every node represents a single frame of the video. In a node we store HOG and HOF features belonging to the particular frame. Recall that the HOF needs two consecutive frames. Thus if a video has f frames, the representation of this video has $f-1$ nodes, ignoring the HOG of first frame. We add an arbitrary node, called Resting Position (RP), obtained as the average representation of first frames of each video.

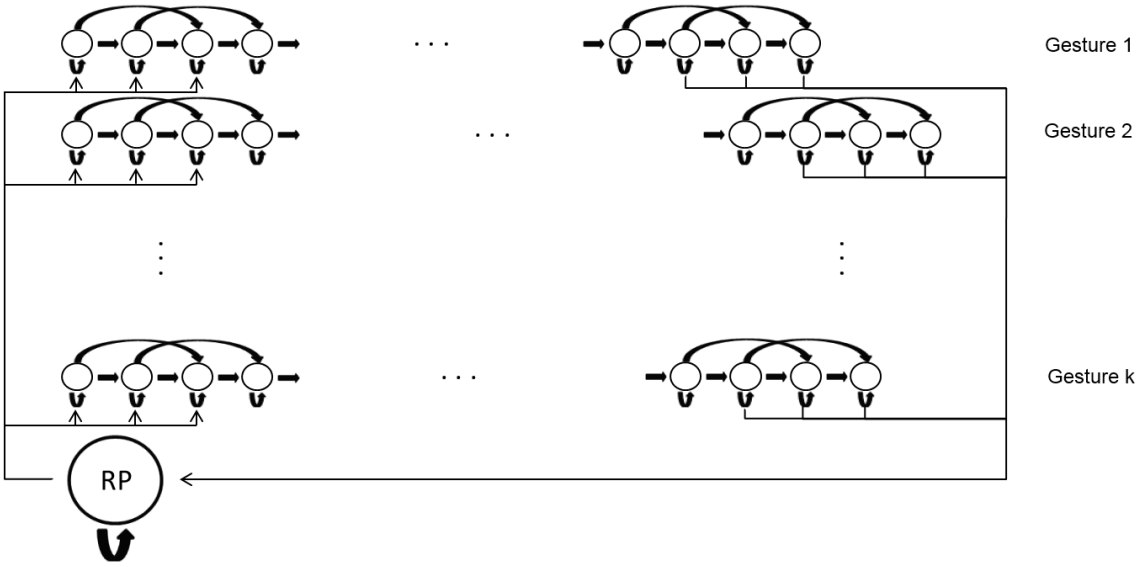


Figure 6: Model for SM—Dynamic Time Warping. Each node represents a single frame of a train video. Each row (on the Figure) represents single train video. We add a new node—RP, or Resting Position—representing state where the user is not performing any particular gesture. The arrows indicate possible transitions between states (nodes).

Since we want to capture the variation in the speed of performing the gestures, we set the transitions in the following way. When being in a particular node n at time t, moving to time $t+1$ we can either stay in the same node (slower performance), move to node $n+1$ (the same speed of performance), or move to node $n+2$ (faster performance). Experiments suggested allowing transition to node $n+3$ is not needed with the trimming described in Section 4. It even made the whole method perform worse. From the node we call RP (Resting Position) we can move to the first three nodes of any video, and from the last three nodes of every video we can move to the RP.

When we have this model, we can start inferring the gestures present in a new video. First, we compute representations of all the frames in the new video. Then we compute similarities of every node of our model with every frame representation of the new video. We compute similarities of both matching HOGs and HOFs, using the Quadratic-Chi distance described in Section 5.3, and simply sum the distances. This makes sense since the empirical distribution functions of distances of HOGs and HOFs are similar. We can represent these distances as a matrix of size $N \times (f-1)$, where N is the number of all nodes in the model, and f is the number of frames in the new video. Using the Viterbi algorithm we find the shortest path through this matrix (we constrain the algorithm to begin in RP or in any of the first three nodes of any gesture). Every column is considered a time point, and in every time point we are in one state (row of the matrix). Between neighbouring time points the

states can change only along the transitions in the model. This approach is also known as Dynamic Time Warping, Berndt and Clifford (1994).

The result of the Viterbi algorithm is a path, a sequence of nodes which correspond to states in which our new video was in time. From this path we can easily infer which gestures were present (which rows in Figure 6), and in what order. The flow of states in time is displayed in Figure 7 (the color represents the cumulative cost up to a particular point—the darker the color, the larger the cumulative cost).

Figure 7: Example of flow of states in model—devel01, video number 11—true labels are {9, 4, 4, 9}. The gray levels represent the shortest cumulative path ending in a particular point.

6.2 Multiple Models—Sliding Frame

The second method we propose is the MM. Here we used only the Histogram of Oriented Gradients and perform temporal segmentation prior to recognition. We created a similar model as in SM, but separately for every training video, illustrated in Figure 8. Again, every node represents HOG features of a single frame. Thus if we have k different gestures, we have k similar models. We do not need an RP node, since we will be looking for short sequences in these models similar to short sequences of frames of a new video. Again, the possible transitions between states in the models, capture variation in the speed of performing gestures.

Figure 8: Model for every training video in MM. Every node represents a single frame of that video. The arrows indicate possible transitions between states (nodes).

MM differs from SM mainly in its approach to inferring the gestures that are present. First, we compute all the HOG representations of a new video and compute their similarities with all the nodes in k models. Then we employ the idea of a sliding frame. The idea is to take a small sequence of the new video and monitor the parts of the training videos that it resembles. First we select frames 1 to l (we set $l = 10$) and treat this similarly as in SM. We look for the shortest path through our first model without constraint on where to begin or end. We do the same with every model. This results in numerical values representing the resemblance of a small part of our new video with any part of every training video, and optionally also the number of nodes resembling it. Then we select frames 2 to $(l+1)$, repeat the whole process, and move forward through the whole video.

Finally we obtain a matrix of size $k \times (f - l + 1)$, where k is the number of gestures and f is the number of frames in the new video. Every column represents a time instance and every row a gesture. An example of such a matrix is shown in Figure 9. Humans can fairly easily learn to recognize where and which gestures are present, but this is a bit more challenging task for a computer. We tried to treat columns as feature vectors and feed it to SM and tried to build a Hidden Markov Model to infer gestures present. We also tried to include information of what nodes of a particular model were present for every time instant, so we can prefer gestures where most of the nodes were included. That was difficult to take into account, because the start and end of most videos are very similar (Resting Position). All the methods had problems identifying two identical gestures occurring after each other, and also two similar gestures occurring after each other. We did not find satisfactory solutions to these problems without deteriorating performance.

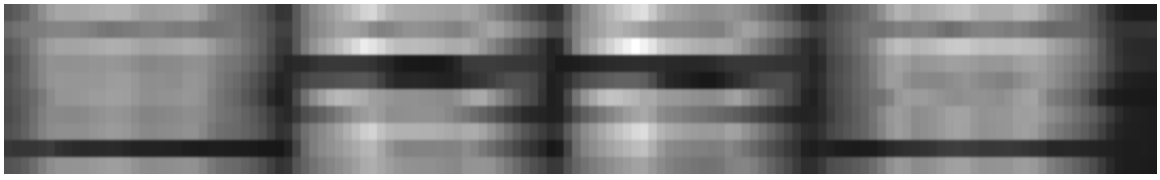


Figure 9: Example of sliding frame matrix—devel01, video number 11.

Neither of these methods manages to beat the naive approach. We resorted to first segment the video using an algorithm provided by the organizers in the sample code called *dtw_segment*. The algorithm is very fast and segments the videos very well. After segmenting, we simply summed along the rows in corresponding parts of the scores matrix and picked the minimum. An improvement was to perform a weighed sum that emphasizes the center of the video, since the important information is usually in the middle.

We used only HOG features in this method because every attempt to include HOF features resulted in considerably worse results. An explanation for this behaviour is we do not need to focus on the overall movement while looking only for short segments of videos, but it is more important to capture the static element. Thus the motion information is redundant in this setting.

7. Results

The performance of the two methods (SM & MM) on the data set is reported in this section. We also compare our results with those of other challenge participants as well as with other already published methods with experiments on this data set. Finally we summarize our contributions and suggest an area for future work.

7.1 Experimental Results

All our experiments were conducted on an Intel Core i7 3610QM processor, with 2 × 4GB DDR3 1600 MHz memory. The running time of SM was approximately 115% of real-time (takes longer to process than to record), while MM was approximately 90% of real-time. However, none of our methods could be trivially converted to an online method, since we need to have the whole video in advance.

The performance of our methods on all available data sets is presented in Table 1. The results show that our preprocessing steps positively influence the final results. The MM works better on the first 20 development batches, but performs worse overall. All other published works provides results only on the first 20 batches—too few for any reliable conclusions. Therefore we suggest providing results on all the batches for bigger relevance.

Batches	SM	MM
devel01-20	23.78	21.99
devel01-480	29.40	34.43
valid01-20	20.01	24.48
final01-20	17.02	23.08
final21-40	10.98	18.47
devel01-20 (without trimming)	26.24	22.82
devel01-20 (without medfilt)	24.70	23.92
devel01-20 (SM; only HOG)	24.53	
devel01-20 (MM; HOG and HOF)	28.73	

Table 1: Overview of our results on data sets. The numbers are normalized Levenshtein distances described in Section 3.

As mentioned in Section 2, we chose our descriptors to exploit specific properties of the data set—the user stays at the same place, and thus the important parts of gestures occur roughly in the same position within the image. Hence it is not surprising that our model is not translation nor scale invariant. Guyon et al. (2013) created 20 translated and scaled data batches, and analyzed the robustness of methods of top ranking participants. In general, the bag-of-features models have this property, but they are usually rather slow. If we wanted to incorporate translation invariance, one method could be to extract body parts from the image (the algorithm is provided within Kinect Development Toolkit[6]) and align the images so that the user is at the same position.

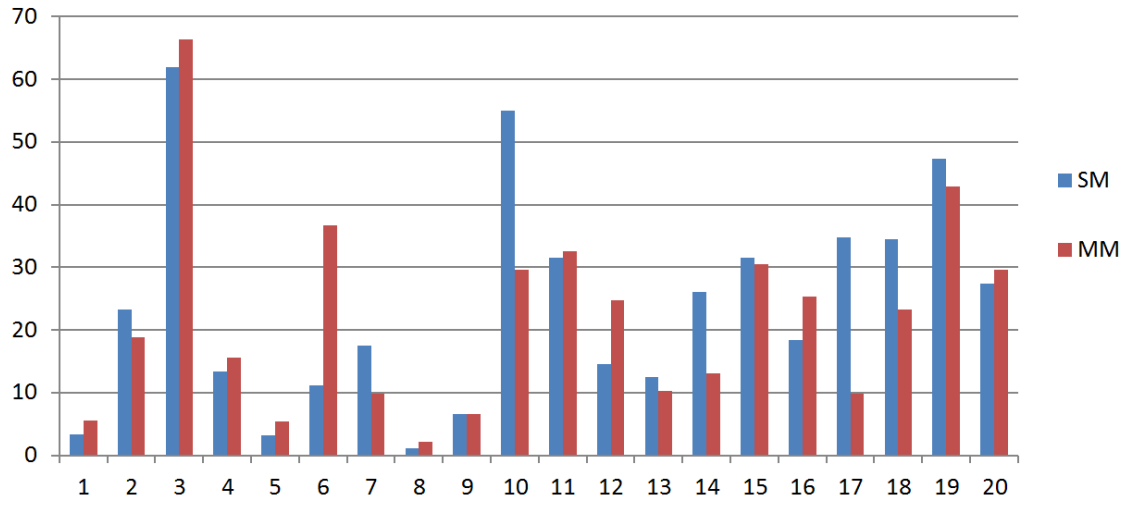


Figure 10: Scores of our methods on first 20 development batches. The numbers on y-axis are normalized Levenshtein distances described in Section 3.

The results of our method on each of the first 20 batches is displayed in Figure 10. Often our methods perform similarly, but one can spot significant differences in batches devel06 (SM—11.11, MM—36.67), devel10 (SM—54.95, MM—29.67) and devel17 (SM—34.78, MM—9.78). In batches devel10 and devel17, the gestures are only static and all occur in the same place in space. In this particular setting, the information about any motion (HOF) can be redundant. This could be a reason why MM performs better, since we do not include any motion descriptors in the representation. In devel06, the problem is, the gestures are performed very quickly, thus the videos are often very short. This is a problem since the matrix in Figure 9 has only a few columns, resulting in poor performance of MM.

The above analysis brings us to a new preprocessing step. Suppose we have many algorithms for solving this one-shot-learning task. If we knew in advance which algorithm was best at recognizing particular gestures, then we could boost the overall performance by selecting the 'best' algorithms in advance, after seeing the training videos. This is a problem we have unsuccessfully tried to solve, and which remains open for future work. If

6. Available at http://www.microsoft.com/en-us/kinectforwindows/develop/.

we always pick the better from our two methods, we would achieve score of 19.04 on the batches devel01-20.

The methods used by other challenge participants—alfnie, Pennect, Joewan (Wan et al., 2013), OneMillionMonkeys, Manavender (Malgireddy et al., 2012)—are summarized by Guyon et al. (2012, 2013). We briefly describe other published works applied on this data set. We provide a comparison of all of these methods in Table 2.

Method / team	devel01-20	valid01-20	final01-20	final21-40
SM (ours)	23.78	20.01	17.02	10.98
MM (ours)	21.99	24.48	23.08	18.47
alfnie	NA	9.51	7.34	7.10
Pennect	NA	17.97	16.52	12.31
Joewan	19.45	16.69	16.80	14.48
OneMillionMonkeys	NA	26.97	16.85	18.19
Mananender	26.34	23.32	21.64	19.25
Wu et al.	26.00	25.43	18.46	18.53
BoVDW	26.62	NA	NA	NA
Lui	28.73	NA	NA	NA
Fanello et al.	25.11	NA	NA	NA

Table 2: Comparison of results of methods from the competition as well as published methods. The numbers are normalized Levenshtein distances described in Section 3.

Wu et al. (2012) pre-segment videos and represent motions of users by Extended-Motion-History-Image and use a maximum correlation coefficient classifier. The Multi-view Spectral Embedding algorithm is used to fuse duo modalities in a physically meaningful manner.

Hernández-Vela et al. (2012) present a Bag-of-Visual-and-Depth-Words (BoVDW) model for gesture recognition, that benefits from the multimodal fusion of visual and depth features. They combine HOG and HOF features with a new proposed depth descriptor.

Tensor representation of action videos is proposed by Lui (2012). The aim of his work is to demonstrate the importance of the intrinsic geometry of tensor space which yields a very discriminating structure for action recognition. The method is assessed using three gesture databases, including Chalearn gesture challenge data set.

Fanello et al. (2013) develop a real-time learning and recognition system for RGB-D images. The proposed method relies on descriptors based on 3D Histogram of Flow, Global Histogram of Oriented Gradient and adaptive sparse coding. The effectiveness of sparse coding techniques to represent 3D actions is highlighted in their work.

7.2 Contributions

Let us now summarize our contributions. As part of the competition we managed to create solid state-of-the-art methods for the new data set—the goal of the competition—which will serve as a reference point for future works. Although the crucial elements of our methods are not novel, they provide a new perspective on the possibilities of using well studied techniques, namely capturing the cross-bin relationships using the Quadratic-Chi distance.

Further we present a novel algorithm for trimming videos, based only on depth data. As a preprocessing step we remove frames that bring little or no additional information, and thus make the method more robust. Experimental results show that this method does not only boost our performance, but also those of other published methods. Our detailed experiments with two very well performing methods suggest that different kinds of settings require different methods for the best performance. In particular, the possibility of choosing from more different types of models (like ours and bag-of-features) under different motion conditions remain unstudied and an open problem.

8. Discussion and Conclusions

In this paper we presented two methods for solving the one-shot-learning gesture recognition task introduced in the *ChaLearn Gesture Challenge* (ChaLearn). We have significantly helped narrow the gap between human and machine performance (the baseline method achieved 50% error rate on final evaluation set, our method 11%, while the human error rate is under 2%). Our methods outperform other published methods and we suggest that other authors provide results on the whole data set for greater relevance of achieved results.

We combine static—Histograms of Oriented Gradients—and dynamic—Histogram of Optical Flow—descriptors in the first method, where we create one model and perform temporal segmentation simultaneously with recognition using Dynamic Time Warping. We use only static descriptors and use pre-segmentation as a preprocessing step in the second method, where we look for similar parts in the training videos using a sliding frame.

Our first method is similar to the one developed by team Pennect in the Challenge, and also performs similarly. They also used HOG features, but at different scales, and used a one-vs-all linear classifier, while we use the Quadratic-Chi distance (Pele and Werman, 2010) to measure distances between individual frames. The recognition was also parallel with temporal segmentation using a DTW model. Surprisingly, the Pennect team used only the color images.

Bag-of-features models provide comparable (Wan et al., 2013) or slightly worse results than ours (Hernández-Vela et al., 2012). The advantage of these models is that they are scale and translation invariant - which is necessary for real-world applications like in gaming industry. On the other hand, these methods rely on presegmentation of videos to single gestures, and are considerably slower, hence are currently not applicable. An interesting property of these methods is their results seem to have lower variance—error rate at difficult data sets (for instance devel10) is smaller, but struggle to obtain strong recognition rate on easy data sets (devel08, devel09).

We present a novel video trimming technique, based on the amount of motion. Its motivation is to remove unimportant segments of videos and thus reduce the probability of confusing gestures. The method improves overall results of our methods (Table 1), and small improvement was confirmed by Wu et al. (2012)—2% and Wan et al. (2013)—0.5%.

Finally, we suggest an area for future work. Having more well working methods at our disposal, we can analyse their results on different types of gesture vocabularies, users and other settings. Overall performance could be boosted if we were able to decide which recognizer to use in advance. Especially, deeper analysis of the differences of results be-

tween Bag-of-words models and Dynamic Time Warping models is needed to obtain better description of their behaviour on different types of gesture recognition tasks.

Appendix A.

In this appendix, we analyse the computational complexity of our methods.

Let us first describe the computational complexity of the building blocks of our algorithms. Let r, c be the resolution of our videos. For this data set we have $r = 240, c = 320$. Let P denote number of pixels ($P = rc$). Computing both HOG and HOF features requires performing a fixed number of iterations for every pixel. Creating histograms in spatial cells requires a fixed number of operations with respect to the size of these cells. Thus the complexity of computing HOG and HOF descriptors for one example requires $\mathcal{O}(P)$ operations. Let m be the number of pixels used in the median filter for every pixel. Since computing the median requires ordering, the complexity of filtering an image requires $\mathcal{O}(Pm \log m)$ operations. In total, for both SM and MM, the whole training on a batch of N frames in total requires $\mathcal{O}(NPm \log(m))$ operations.

Before evaluating a new video of F frames, we have to compute the representations of the frames, which is done in $\mathcal{O}(FPm \log m)$ operations. In both methods we then perform a Viterbi search. In MM this is divided into several searches, but the total complexity stays the same. The most time consuming part is computing the Quadratic-Chi distances (Subsection 5.3) between all FN pairs of frames from the new video and model. Computing the distance needs sum over elements over sparse $H \times H$ matrix (H being the size of the histograms used) described in Algorithm 2. The number of non-zero elements is linear in H. Thus, the overall complexity of evaluating a new video is $\mathcal{O}(NPm \log(m) + NFH)$.

To summarize, the running time of our methods is linear in the number of training frames, number of frames of a new video, number of pixels of a single frame, and size of histogram (number of spatial cell times number of orientation bins). Dependence on size of the filtering region for every pixel is linearithmic since it requires sorting.

Appendix B.

In this Appendix, we provide MATLAB algorithm for creating similarity matrix used in the Quadratic-Chi distance described in Section 5.3. We have histograms of $h \times w$ spatial cells, and p orientation bins in each of the spatial bins. The size of the final matrix is $H \times H$, where $H = hwp$.

References

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370, 1994.

ChaLearn. ChaLearn Gesture Dataset (CGD2011), ChaLearn, California. `http://gesture.chalearn.org/data`, 2011.

Algorithm 2 MATLAB code producing the similarity matrix

```
gauss = fspecial('gaussian', 3, 0.56);
B = diag(ones(1,h)) + 2*(diag(ones(1, h-1), 1) + diag(ones(1, h-1), -1));
C = diag(ones(1,w)) + 2*(diag(ones(1, w-1), 1) + diag(ones(1, w-1), -1));
D = kron(C, B); % Kronecker tensor product
D(D == 1) = gauss(5);
D(D == 2) = gauss(2);
D(D == 4) = gauss(1);
A = imfilter( eye(p), gauss, 'circular');
A = sparse(kron(D, A)); % The final similarity matrix
```

Sotirios P Chatzis, Dimitrios I Kosmopoulos, and Paul Doliotis. A conditional random field-based model for joint sequence segmentation and classification. *Pattern recognition*, 2012.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893. IEEE, 2005.

Piotr Dollár. Piotr's Image and Video Matlab Toolbox (PMT). `http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html`.

Sean Ryan Fanello, Ilaria Gori, Giorgio Metta, and Francesca Odone. One-shot learning for real-time action recognition. 2013.

Isabelle Guyon, Vassilis Athitsos, Pat Jangyodsuk, Ben Hamner, and Hugo Jair Escalante. Chalearn gesture challenge: Design and first results. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1–6. IEEE, 2012.

Isabelle Guyon, Vassilis Athitsos, Pat Jangyodsuk, Hugo Jair Escalante, and Ben Hamner. Results and analysis of the chalearn gesture challenge 2012. 2013.

Antonio Hernández-Vela, Miguel Ángel Bautista, Xavier Perez-Sala, Victor Ponce, Xavier Baró, Oriol Pujol, Cecilio Angulo, and Sergio Escalera. BoVDW: Bag-of-Visual-and-Depth-Words for gesture recognition. In *International Conference on Pattern Recognition*, pages 449–452, 2012.

Nazlı Ikizler and David Forsyth. Searching video for complex activities with finite state models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2007.

Takeo Kanade and Bruce D. Lucas. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence*, 1981.

Alexander Klaser and Marcin Marszalek. A spatio-temporal descriptor based on 3d-gradients. 2008.

Ivan Laptev. On space-time interest points. *International Journal of Computer Vision*, 64 (2-3):107–123, 2005.

Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008.

David D Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Machine learning: ECML-98*, pages 4–15. Springer, 1998.

David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

Bruce D. Lucas. *Generalized Image Matching by the Method of Differences*. PhD thesis, Robotics Institute, Carnegie Mellon University, July 1984.

Yui Man Lui. Human gesture recognition on product manifolds. *Journal of Machine Learning Research*, 13:3297–3321, 2012.

Manavender R Malgireddy, Ifeoma Inwogu, and Venu Govindaraju. A temporal bayesian model for classifying, detecting and localizing activities in video sequences. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 43–48. IEEE, 2012.

Ofir Pele and Michael Werman. The quadratic-chi histogram distance family. *Computer Vision–ECCV 2010*, pages 749–762, 2010.

Jun Wan, Qiuqi Ruan, Wei Li, and Shuang Deng. One-shot Learning Gesture Recognition from RGB-D Data Using Bag of Features. *Journal of Machine Learning Research*, 14: 2549–2582, 2013. URL `http://jmlr.org/papers/v14/wan13a.html`.

Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, Cordelia Schmid, et al. Evaluation of local spatio-temporal features for action recognition. In *BMVC 2009-British Machine Vision Conference*, 2009.

Sy Bor Wang, Ariadna Quattoni, L-P Morency, David Demirdjian, and Trevor Darrell. Hidden conditional random fields for gesture recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1521–1527. IEEE, 2006.

Di Wu, Fan Zhu, and Ling Shao. One shot learning gesture recognition from rgbd images. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 7–12. IEEE, 2012.

Tian Xia, Dacheng Tao, Tao Mei, and Yongdong Zhang. Multiview spectral embedding. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 40(6):1438–1446, 2010.

Journal of Machine Learning Research 15 (2014) 2533-2568 Submitted 2/13; Revised 1/14; Published 7/14

Contextual Bandits with Similarity Information

Aleksandrs Slivkins SLIVKINS@MICROSOFT.COM
Microsoft Research New York City
641 6th Ave, 7th floor
New York, NY 10011
USA

Editor: Nicolo Cesa-Bianchi

Abstract

In a multi-armed bandit (MAB) problem, an online algorithm makes a sequence of choices. In each round it chooses from a time-invariant set of alternatives and receives the payoff associated with this alternative. While the case of small strategy sets is by now well-understood, a lot of recent work has focused on MAB problems with exponentially or infinitely large strategy sets, where one needs to assume extra structure in order to make the problem tractable. In particular, recent literature considered information on similarity between arms.

We consider similarity information in the setting of *contextual bandits*, a natural extension of the basic MAB problem where before each round an algorithm is given the *context*—a hint about the payoffs in this round. Contextual bandits are directly motivated by placing advertisements on web pages, one of the crucial problems in sponsored search. A particularly simple way to represent similarity information in the contextual bandit setting is via a *similarity distance* between the context-arm pairs which bounds from above the difference between the respective expected payoffs.

Prior work on contextual bandits with similarity uses "uniform" partitions of the similarity space, so that each context-arm pair is approximated by the closest pair in the partition. Algorithms based on "uniform" partitions disregard the structure of the payoffs and the context arrivals, which is potentially wasteful. We present algorithms that are based on *adaptive* partitions, and take advantage of "benign" payoffs and context arrivals without sacrificing the worst-case performance. The central idea is to maintain a finer partition in high-payoff regions of the similarity space and in popular regions of the context space. Our results apply to several other settings, e.g., MAB with constrained temporal change (Slivkins and Upfal, 2008) and sleeping bandits (Kleinberg et al., 2008a).

Keywords: multi-armed bandits, contextual bandits, regret, Lipschitz-continuity, metric space

1. Introduction

In a multi-armed bandit problem (henceforth, "multi-armed bandit" will be abbreviated as MAB), an algorithm is presented with a sequence of trials. In each round, the algorithm chooses one alternative from a set of alternatives (*arms*) based on the past history, and receives the payoff associated with this alternative. The goal is to maximize the total payoff of the chosen arms. The MAB setting has been introduced in Robbins (1952) and studied intensively since then in operations research, economics and computer science. This setting

is a clean model for the exploration-exploitation trade-off, a crucial issue in sequential decision-making under uncertainty.

One standard way to evaluate the performance of a bandit algorithm is *regret*, defined as the difference between the expected payoff of an optimal arm and that of the algorithm. By now the MAB problem with a small finite set of arms is quite well understood, e.g., see Lai and Robbins (1985); Auer et al. (2002b,a). However, if the arms set is exponentially or infinitely large, the problem becomes intractable unless we make further assumptions about the problem instance. Essentially, a bandit algorithm needs to find a needle in a haystack; for each algorithm there are inputs on which it performs as badly as random guessing.

Bandit problems with large sets of arms have been an active area of investigation in the past decade (see Section 2 for a discussion of related literature). A common theme in these works is to assume a certain *structure* on payoff functions. Assumptions of this type are natural in many applications, and often lead to efficient learning algorithms (Kleinberg, 2005). In particular, a line of work started in Agrawal (1995) assumes that some information on similarity between arms is available.

In this paper, we consider similarity information in the setting of *contextual bandits* (Woodroofe, 1979; Auer, 2002; Wang et al., 2005; Pandey et al., 2007; Langford and Zhang, 2007), a natural extension of the basic MAB problem where before each round an algorithm is given the *context*—a hint about the payoffs in this round. Contextual bandits are directly motivated by the problem of placing advertisements on web pages, one of the crucial problems in sponsored search. One can cast it as a bandit problem so that arms correspond to the possible ads, and payoffs correspond to the user clicks. Then the context consists of information about the page, and perhaps the user this page is served to. Furthermore, we assume that similarity information is available on both the context and the arms. Following the work in Agrawal (1995); Kleinberg (2004); Auer et al. (2007); Kleinberg et al. (2008b) on the (non-contextual) bandits, a particularly simple way to represent similarity information in the contextual bandit setting is via a *similarity distance* between the context-arm pairs, which gives an upper bound on the difference between the corresponding payoffs.

1.1 Our Model: Contextual Bandits with Similarity Information

The contextual bandits framework is defined as follows. Let X be the *context set* and Y be the *arms set*, and let $\mathcal{P} \subset X \times Y$ be the set of feasible context-arms pairs. In each round t, the following events happen in succession:

1. a context $x_t \in X$ is revealed to the algorithm,
2. the algorithm chooses an arm $y_t \in Y$ such that $(x_t, y_t) \in \mathcal{P}$,
3. payoff (reward) $\pi_t \in [0, 1]$ is revealed.

The sequence of context arrivals $(x_t)_{t \in \mathbb{N}}$ is fixed before the first round, and does not depend on the subsequent choices of the algorithm. With *stochastic payoffs*, for each pair $(x, y) \in \mathcal{P}$ there is a distribution $\Pi(x, y)$ with expectation $\mu(x, y)$, so that π_t is an independent sample from $\Pi(x_t, y_t)$. With *adversarial payoffs*, this distribution can change from round to round. For simplicity, we present the subsequent definitions for the stochastic setting only, whereas the adversarial setting is fleshed out later in the paper (Section 8).

In general, the goal of a bandit algorithm is to maximize the total payoff $\sum_{t=1}^{T} \pi_t$, where T is the *time horizon*. In the contextual MAB setting, we benchmark the algorithm's performance in terms of the context-specific "best arm". Specifically, the goal is to minimize the *contextual regret*:

$$R(T) \triangleq \sum_{t=1}^{T} \mu(x_t, y_t) - \mu^*(x_t), \quad \text{where} \quad \mu^*(x) \triangleq \sup_{y \in Y:\,(x,y)\in\mathcal{P}} \mu(x,y).$$

The context-specific best arm is a more demanding benchmark than the best arm used in the "standard" (context-free) definition of regret.

The similarity information is given to an algorithm as a metric space $(\mathcal{P}, \mathcal{D})$ which we call the *similarity space*, such that the following Lipschitz condition[1] holds:

$$|\mu(x,y) - \mu(x',y')| \leq \mathcal{D}((x,y),\ (x',y')). \tag{1}$$

Without loss of generality, $\mathcal{D} \leq 1$. The absence of similarity information is modeled as $\mathcal{D} = 1$.

An instructive special case is the *product similarity space* $(\mathcal{P}, \mathcal{D}) = (X \times Y, \mathcal{D})$, where $(X, \mathcal{D}_{\mathrm{X}})$ is a metric space on contexts (*context space*), and $(Y, \mathcal{D}_{\mathrm{Y}})$ is a metric space on arms (*arms space*), and

$$\mathcal{D}((x,y),\ (x',y')) = \min(1,\ \mathcal{D}_{\mathrm{X}}(x,x') + \mathcal{D}_{\mathrm{Y}}(y,y')). \tag{2}$$

1.2 Prior Work: Uniform Partitions

Hazan and Megiddo (2007) consider contextual MAB with similarity information on contexts. They suggest an algorithm that chooses a "uniform" partition S_{X} of the context space and approximates x_t by the closest point in S_{X}, call it x'_t. Specifically, the algorithm creates an instance $\mathcal{A}(x)$ of some bandit algorithm $\mathcal{A}$ for each point $x \in S_{\mathrm{X}}$, and invokes $\mathcal{A}(x'_t)$ in each round t. The granularity of the partition is adjusted to the time horizon, the context space, and the black-box regret guarantee for $\mathcal{A}$. Furthermore, Kleinberg (2004) provides a bandit algorithm $\mathcal{A}$ for the adversarial MAB problem on a metric space that has a similar flavor: pick a "uniform" partition S_{Y} of the arms space, and run a k-arm bandit algorithm such as EXP3 (Auer et al., 2002b) on the points in S_{Y}. Again, the granularity of the partition is adjusted to the time horizon, the arms space, and the black-box regret guarantee for EXP3.

Applying these two ideas to our setting (with the product similarity space) gives a simple algorithm which we call the *uniform algorithm*. Its contextual regret, even for adversarial payoffs, is

$$R(T) \leq O(T^{1-1/(2+d_{\mathrm{X}}+d_{\mathrm{Y}})})(\log T), \tag{3}$$

where d_{X} is the covering dimension of the context space and d_{Y} is that of the arms space.

1. In other words, μ is a Lipschitz-continuous function on $(X, \mathcal{P})$, with Lipschitz constant $K_{\mathrm{Lip}} = 1$. Assuming $K_{\mathrm{Lip}} = 1$ is without loss of generality (as long as K_{Lip} is known to the algorithm), since we can re-define $\mathcal{D} \leftarrow K_{\mathrm{Lip}}\, D$.

1.3 Our Contributions

Using "uniform" partitions disregards the potentially benign structure of expected payoffs and context arrivals. The central topic in this paper is ***adaptive partitions*** of the similarity space which are adjusted to frequently occurring contexts and high-paying arms, so that the algorithms can take advantage of the problem instances in which the expected payoffs or the context arrivals are "benign" ("low-dimensional"), in a sense that we make precise later.

We present two main results, one for stochastic payoffs and one for adversarial payoffs. For stochastic payoffs, we provide an algorithm called *contextual zooming* which "zooms in" on the regions of the context space that correspond to frequently occurring contexts, and the regions of the arms space that correspond to high-paying arms. Unlike the algorithms in prior work, this algorithm considers the context space and the arms space *jointly*—it maintains a partition of the similarity space, rather than one partition for contexts and another for arms. We develop provable guarantees that capture the "benign-ness" of the context arrivals and the expected payoffs. In the worst case, we match the guarantee (3) for the uniform algorithm. We obtain nearly matching lower bounds using the KL-divergence technique from Auer et al. (2002b); Kleinberg (2004). The lower bound is very general as it holds for every given (product) similarity space *and* for every fixed value of the upper bound.

Our stochastic contextual MAB setting, and specifically the contextual zooming algorithm, can be fruitfully applied beyond the ad placement scenario described above and beyond MAB with similarity information per se. First, writing $x_t = t$ one can incorporate "temporal constraints" (across time, for each arm), and combine them with "spatial constraints" (across arms, for each time). The analysis of contextual zooming yields concrete, meaningful bounds this scenario. In particular, we recover one of the main results in Slivkins and Upfal (2008). Second, our setting subsumes the stochastic *sleeping bandits* problem (Kleinberg et al., 2008a), where in each round some arms are "asleep", i.e., not available in this round. Here contexts correspond to subsets of arms that are "awake". Contextual zooming recovers and generalizes the corresponding result in Kleinberg et al. (2008a). Third, following the publication of a preliminary version of this paper, contextual zooming has been applied to bandit learning-to-rank in Slivkins et al. (2013).

For the adversarial setting, we provide an algorithm which maintains an adaptive partition of the context space and thus takes advantage of "benign" context arrivals. We develop provable guarantees that capture this "benign-ness". In the worst case, the contextual regret is bounded in terms of the covering dimension of the context space, matching (3). Our algorithm is in fact a *meta-algorithm*: given an adversarial bandit algorithm `Bandit`, we present a contextual bandit algorithm which calls `Bandit` as a subroutine. Our setup is flexible: depending on what additional constraints are known about the adversarial payoffs, one can plug in a bandit algorithm from the prior work on the corresponding version of adversarial MAB, so that the regret bound for `Bandit` plugs into the overall regret bound.

1.4 Discussion

Adaptive partitions (of the arms space) for context-free MAB with similarity information have been introduced in Kleinberg et al. (2008b); Bubeck et al. (2011a). This paper further

explores the potential of the zooming technique in Kleinberg et al. (2008b). Specifically, contextual zooming extends this technique to adaptive partitions of the entire similarity space, which necessitates a technically different algorithm and a more delicate analysis. We obtain a clean algorithm for contextual MAB with improved (and nearly optimal) bounds. Moreover, this algorithm applies to several other, seemingly unrelated problems and unifies some results from prior work.

One alternative approach is to maintain a partition of the context space, and run a separate instance of the zooming algorithm from Kleinberg et al. (2008b) on each set in this partition. Fleshing out this idea leads to the meta-algorithm that we present for adversarial payoffs (with `Bandit` being the zooming algorithm). This meta-algorithm is parameterized (and constrained) by a specific a priori regret bound for `Bandit`. Unfortunately, any a priori regret bound for zooming algorithm would be a pessimistic one, which negates its main strength—the ability to adapt to "benign" expected payoffs.

1.5 Map of the Paper

Section 2 is related work, and Section 3 is Preliminaries. Contextual zooming is presented in Section 4. Lower bounds are in Section 5. Some applications of contextual zooming are discussed in Section 6. The adversarial setting is treated in Section 8.

2. Related Work

A proper discussion of the literature on bandit problems is beyond the scope of this paper. This paper follows the line of work on regret-minimizing bandits; a reader is encouraged to refer to Cesa-Bianchi and Lugosi (2006); Bubeck and Cesa-Bianchi (2012) for background. A different (Bayesian) perspective on bandit problems can be found in Gittins et al. (2011).

Most relevant to this paper is the work on bandits with large sets of arms, specifically bandits with similarity information (Agrawal, 1995; Kleinberg, 2004; Auer et al., 2007; Pandey et al., 2007; Kocsis and Szepesvari, 2006; Munos and Coquelin, 2007; Kleinberg et al., 2008b; Bubeck et al., 2011a; Kleinberg and Slivkins, 2010; Maillard and Munos, 2010). Another commonly assumed structure is linear or convex payoffs (e.g., Awerbuch and Kleinberg, 2008; Flaxman et al., 2005; Dani et al., 2007; Abernethy et al., 2008; Hazan and Kale, 2009; Bubeck et al., 2012). Linear/convex payoffs is a much stronger assumption than similarity, essentially because it allows to make strong inferences about far-away arms. Other assumptions have been considered (e.g., Banks and Sundaram, 1992; Berry et al., 1997; Wang et al., 2008; Bubeck and Munos, 2010). The distinction between stochastic and adversarial payoffs is orthogonal to the structural assumption (such as Lipschitz-continuity or linearity). Papers on MAB with linear/convex payoffs typically allow adversarial payoffs, whereas papers on MAB with similarity information focus on stochastic payoffs, with notable exceptions of Kleinberg (2004) and Maillard and Munos (2010).[2]

The notion of structured adversarial payoffs in this paper is less restrictive than the one in Maillard and Munos (2010) (which in turn specializes the notion from linear/convex payoffs), in the sense that the Lipschitz condition is assumed on the expected payoffs rather

than on realized payoffs. This is a non-trivial distinction, essentially because our notion generalizes stochastic payoffs whereas the other one does not.

2.1 Contextual MAB

In Auer (2002) and Chu et al. (2011)[2] payoffs are linear in context, which is a feature vector. Woodroofe (1979); Wang et al. (2005) and Rigollet and Zeevi (2010)[2] study contextual MAB with stochastic payoffs, under the name *bandits with covariates*: the context is a random variable correlated with the payoffs; they consider the case of two arms, and make some additional assumptions. Lazaric and Munos (2009)[2] consider an online labeling problem with stochastic inputs and adversarially chosen labels; inputs and hypotheses (mappings from inputs to labels) can be thought of as "contexts" and "arms" respectively. *Bandits with experts advice* (e.g., Auer 2002) is the special case of contextual MAB where the context consists of experts' advice; the advice of a each expert is modeled as a distributions over arms. All these papers are not directly applicable to the present setting.

Experimental work on contextual MAB includes (Pandey et al., 2007) and (Li et al., 2010, 2011).[2]

Lu et al. (2010)[2] consider the setting in this paper for a product similarity space and, essentially, recover the uniform algorithm and a lower bound that matches (3). The same guarantee (3) can also be obtained as follows. The "uniform partition" described above can be used to define "experts" for a bandit-with-expert-advice algorithm such as EXP4 (Auer et al., 2002b): for each set of the partition there is an expert whose advise is simply an arbitrary arm in this set. Then the regret bound for EXP4 yields (3). Instead of EXP4 one could use an algorithm in McMahan and Streeter (2009)[2] which improves over EXP4 if the experts are not "too distinct"; however, it is not clear if it translates into concrete improvements over (3).

If the context x_t is time-invariant, our setting reduces to the Lipschitz MAB problem as defined in Kleinberg et al. (2008b), which in turn reduces to continuum-armed bandits (Agrawal, 1995; Kleinberg, 2004; Auer et al., 2007) if the metric space is a real line, and to MAB with stochastic payoffs (Auer et al., 2002a) if the similarity information is absent.

3. Preliminaries

We will use the notation from the Introduction. In particular, x_t will denote the t-th *context arrival*, i.e., the context that arrives in round t, and y_t will denote the arm chosen by the algorithm in that round. We will use $x_{(1..T)}$ to denote the sequence of the first T context arrivals $(x_1, \ldots, x_T)$. The *badness* of a point $(x,y) \in \mathcal{P}$ is defined as $\Delta(x,y) \triangleq \mu^*(x) - \mu(x,y)$. The context-specific best arm is

$$y^*(x) \in \operatorname{argmax}_{y \in Y:\,(x,y)\in\mathcal{P}}\ \mu(x,y), \tag{4}$$

where ties are broken in an arbitrary but fixed way. To ensure that the max in (4) is attained by some $y \in Y$, we will assume that the similarity space $(\mathcal{P}, \mathcal{D})$ is compact.

2. This paper is concurrent and independent work w.r.t. the preliminary publication of this paper on `arxiv.org`.

Metric spaces. Covering dimension and related notions are crucial throughout this paper. Let $\mathcal{P}$ be a set of points in a metric space, and fix $r > 0$. An *r-covering* of $\mathcal{P}$ is a collection of subsets of $\mathcal{P}$, each of diameter strictly less than r, that cover $\mathcal{P}$. The minimal number of subsets in an r-covering is called the *r-covering number*[3] of $\mathcal{P}$ and denoted $N_r(\mathcal{P})$. The *covering dimension* of $\mathcal{P}$ (with multiplier c) is the smallest d such that $N_r(\mathcal{P}) \leq c\, r^{-d}$ for each $r > 0$. In particular, if S is a subset of Euclidean space then its covering dimension is at most the linear dimension of S, but can be (much) smaller.

Covering is closely related to packing. A subset $S \subset \mathcal{P}$ is an *r-packing* of $\mathcal{P}$ if the distance between any two points in S is at least r. The maximal number of points in an r-packing is called the *r-packing number* and denoted $N_r^{\texttt{pack}}(\mathcal{P})$. It is well-known that r-packing numbers are essentially the same as r-covering numbers, namely $N_{2r}(\mathcal{P}) \leq N_r^{\texttt{pack}}(\mathcal{P}) \leq N_r(\mathcal{P})$.

The *doubling constant* $c_{\text{DBL}}(\mathcal{P})$ of $\mathcal{P}$ is the smallest k such that any ball can be covered by k balls of half the radius. The doubling constant (and *doubling dimension* $\log c_{\text{DBL}}$) was introduced in Heinonen (2001) and has been a standard notion in theoretical computer science literature since Gupta et al. (2003). It was used to characterize tractable problem instances for a variety of problems (e.g., see Talwar, 2004; Kleinberg et al., 2009; Cole and Gottlieb, 2006). It is known that $c_{\text{DBL}}(\mathcal{P}) \geq c\, 2^d$ if d is the covering dimension of $\mathcal{P}$ with multiplier c, and that $c_{\text{DBL}}(\mathcal{P}) \leq 2^d$ if $\mathcal{P}$ is a bounded subset of d-dimensional Euclidean space. A useful observation is that if distance between any two points in S is $> r$, then any ball of radius r contains at most c_{DBL} points of S.

A ball with center x and radius r is denoted $B(x, r)$. Formally, we will treat a ball as a (center, radius) pair rather than a set of points. A function $f : \mathcal{P} \to \mathbb{R}$ if a Lipschitz function on a metric space $(\mathcal{P}, \mathcal{D})$, with Lipschitz constant K_{Lip}, if the *Lipschitz condition* holds: $|f(x) - f(x')| \leq K_{\text{Lip}}\, \mathcal{D}(x, x')$ for each $x, x' \in \mathcal{P}$.

Accessing the similarity space. We assume full and computationally unrestricted access to the similarity information. While the issues of efficient representation thereof are important in practice, we believe that a proper treatment of these issues would be specific to the particular application and the particular similarity metric used, and would obscure the present paper. One clean formal way to address this issue is to assume *oracle access*: an algorithm accesses the similarity space via a few specific types of queries, and invokes an "oracle" that answers such queries.

Time horizon. We assume that the time horizon is fixed and known in advance. This assumption is without loss of generality in our setting. This is due to the well-known *doubling trick* which converts a bandit algorithm with a fixed time horizon into one that runs indefinitely and achieves essentially the same regret bound. Suppose for any fixed time horizon T there is an algorithm $\texttt{ALG}_T$ whose regret is at most $R(T)$. The new algorithm proceeds in phases $i = 1, 2, 3, \ldots$ of duration 2^i rounds each, so that in each phase i a fresh instance of $\texttt{ALG}_{2^i}$ is run. This algorithm has regret $O(\log T)R(T)$ for each round T, and $O(R(T))$ in the typical case when $R(T) \geq T^\gamma$ for some constant $\gamma > 0$.

3. The covering number can be defined via radius-r balls rather than diameter-r sets. This alternative definition lacks the appealing "robustness" property: $N_r(\mathcal{P}') \leq N_r(\mathcal{P})$ for any $\mathcal{P}' \subset \mathcal{P}$, but (other than that) is equivalent for this paper.

4. The Contextual Zooming Algorithm

In this section we consider the contextual MAB problem with stochastic payoffs. We present an algorithm for this problem, called *contextual zooming*, which takes advantage of both the "benign" context arrivals and the "benign" expected payoffs. The algorithm adaptively maintains a partition of the similarity space, "zooming in" on both the "popular" regions on the context space and the high-payoff regions of the arms space.

Contextual zooming extends the (context-free) zooming technique in Kleinberg et al. (2008b), which necessitates a somewhat more complicated algorithm. In particular, selection and activation rules are defined differently, there is a new notion of "domains" and the distinction between "pre-index" and "index". The analysis is more delicate, both the high-probability argument in Claim 3 and the subsequent argument that bounds the number of samples from suboptimal arms. Also, the key step of setting up the regret bounds is very different, especially for the improved regret bounds in Section 4.4.

4.1 Provable Guarantees

Let us define the notions that express the performance of contextual zooming. These notions rely on the packing number $N_r(\cdot)$ in the similarity space $(\mathcal{P}, \mathcal{D})$, and the more refined versions thereof that take into account "benign" expected payoffs and "benign" context arrivals.

Our guarantees have the following form, for some integer numbers $\{N_r\}_{r\in(0,1)}$:

$$R(T) \leq C_0 \inf_{r_0\in(0,1)} \left(r_0 T + \sum_{r=2^{-i}:\, i\in\mathbb{N},\, r_0\leq r\leq 1} \tfrac{1}{r}\, N_r \log T \right). \tag{5}$$

Here and thereafter, $C_0 = O(1)$ unless specified otherwise. In the pessimistic version, $N_r = N_r(\mathcal{P})$ is the r-packing number[4] of $\mathcal{P}$. The main contribution is refined bounds in which N_r is smaller.

For every guarantee of the form (5), call it N_r-*type* guarantee, prior work (e.g., Kleinberg 2004; Kleinberg et al. 2008b; Bubeck et al. 2011a) suggests a more tractable *dimension-type* guarantee. This guarantee is in terms of the *covering-type dimension* induced by N_r, defined as follows:[5]

$$d_c \triangleq \inf\{d > 0 :\ N_r \leq c\, r^{-d} \quad \forall r \in (0,1)\}. \tag{6}$$

Using (5) with $r_0 = T^{-1/(d_c+2)}$, we obtain

$$R(T) \leq O(C_0)\, (c\, T^{1-1/(2+d_c)}\, \log T) \qquad (\forall c > 0). \tag{7}$$

For the pessimistic version ($N_r = N_r(\mathcal{P})$), the corresponding covering-type dimension d_c is the covering dimension of the similarity space. The resulting guarantee (7) subsumes the bound (3) from prior work (because the covering dimension of a product similarity space is

4. Then (5) can be simplified to $R(T) \leq \inf_{r\in(0,1)} O\left(rT + \frac{1}{r}\, N_r(\mathcal{P}) \log T\right)$, asl $N_r(\mathcal{P})$ is non-increasing in r.
5. One standard definition of the covering dimension is (6) for $N_r = N_r(\mathcal{P})$ and $c = 1$. Following Kleinberg et al. (2008b), we include an explicit dependence on c in (6) to obtain a more efficient regret bound (which holds for any c).

$d_{\mathrm{X}} + d_{\mathrm{Y}}$), and extends this bound from product similarity spaces (2) to arbitrary similarity spaces.

To account for "benign" expected payoffs, instead of r-packing number of the entire set $\mathcal{P}$ we consider the r-packing number of a subset of $\mathcal{P}$ which only includes points with near-optimal expected payoffs:

$$\mathcal{P}_{\mu,r} \triangleq \{(x,y) \in \mathcal{P} : \mu^*(x) - \mu(x,y) \leq 12\, r\}. \tag{8}$$

We define the *r-zooming number* as $N_r(\mathcal{P}_{\mu,r})$, the r-packing number of $\mathcal{P}_{\mu,r}$. The corresponding covering-type dimension (6) is called the *contextual zooming dimension*.

The r-zooming number can be seen as an optimistic version of $N_r(\mathcal{P})$: while equal to $N_r(\mathcal{P})$ in the worst case, it can be much smaller if the set of near-optimal context-arm pairs is "small" in terms of the packing number. Likewise, the contextual zooming dimension is an optimistic version of the covering dimension.

Theorem 1 *Consider the contextual MAB problem with stochastic payoffs. There is an algorithm (namely, Algorithm 1 described below) whose contextual regret $R(T)$ satisfies (5) with N_r equal to $N_r(\mathcal{P}_{\mu,r})$, the r-zooming number. Consequently, $R(T)$ satisfies the dimension-type guarantee (7), where d_c is the contextual zooming dimension.*

In Theorem 1, the same algorithm enjoys the bound (7) for each $c > 0$. This is a useful trade-off since different values of c may result in drastically different values of the dimension d_c. On the contrary, the "uniform algorithm" from prior work essentially needs to take the c as input.

Further refinements to take into account "benign" context arrivals are deferred to Section 4.4.

4.2 Description of the Algorithm

The algorithm is parameterized by the time horizon T. In each round t, it maintains a finite collection $\mathcal{A}_t$ of balls in $(\mathcal{P}, \mathcal{D})$ (called *active balls*) which collectively cover the similarity space. Adding active balls is called *activating*; balls stay active once they are activated. Initially there is only one active ball which has radius 1 and therefore contains the entire similarity space.

At a high level, each round t proceeds as follows. Context x_t arrives. Then the algorithm selects an active ball B and an arm y_t such that $(x_t, y_t) \in B$, according to the "selection rule". Arm y_t is played. Then one ball may be activated, according to the "activation rule".

In order to state the two rules, we need to put forward several definitions. Fix an active ball B and round t. Let $r(B)$ be the radius of B. The *confidence radius* of B at time t is

$$\texttt{conf}_t(B) \triangleq 4\sqrt{\frac{\log T}{1 + n_t(B)}}, \tag{9}$$

where $n_t(B)$ is the number of times B has been selected by the algorithm before round t. The *domain* of ball B in round t is a subset of B that excludes all balls $B' \in \mathcal{A}_t$ of strictly smaller radius:

$$\texttt{dom}_t(B) \triangleq B \setminus \left(\bigcup_{B' \in \mathcal{A}_t:\, r(B') < r(B)} B' \right). \tag{10}$$

Algorithm 1 Contextual zooming algorithm.

Input: Similarity space $(\mathcal{P}, \mathcal{D})$ of diameter ≤ 1, $\mathcal{P} \subset X \times Y$. Time horizon T.
Data: collection $\mathcal{A}$ of "active balls" in $(\mathcal{P}, \mathcal{D})$; counters $n(B)$, $\texttt{rew}(B)$ for each $B \in \mathcal{A}$.

Init: $B \leftarrow B(p, 1)$; // center $p \in \mathcal{P}$ is arbitrary
    $\mathcal{A} \leftarrow \{B\}$; $n(B) = \texttt{rew}(B) = 0$
Main loop: for each round t // use definitions (9-12)
    Input context x_t.
        // activation rule
    $\texttt{relevant} \leftarrow \{B \in \mathcal{A} : (x_t, y) \in \texttt{dom}\,(B, \mathcal{A}) \text{ for some arm } y\}$.
    $B \leftarrow \text{argmax}_{B \in \texttt{relevant}}\ I_t(B)$. // ball B is selected
    $y \leftarrow$ any arm y such that $(x_t, y) \in \texttt{dom}\,(B, \mathcal{A})$.
    Play arm y, observe payoff π.
    Update counters: $n(B) \leftarrow n(B) + 1$, $\texttt{rew}(B) \leftarrow \texttt{rew}(B) + \pi$.
        // selection rule
    if $\texttt{conf}(B) \leq \texttt{radius}(B)$ **then**
        $B' \leftarrow B((x_t, y),\ \frac{1}{2}\,\texttt{radius}(B))$ // new ball to be activated
        $\mathcal{A} \leftarrow \mathcal{A} \cup \{B'\}$; $n(B') = \texttt{rew}(B') = 0$.

We will also denote (10) as $\texttt{dom}\,(B, \mathcal{A}_t)$. Ball B is called *relevant* in round t if $(x_t, y) \in \texttt{dom}_t(B)$ for some arm y. In each round, the algorithm selects one relevant ball B. This ball is selected according to a numerical score $I_t(B)$ called *index*. (The definition of index is deferred to the end of this subsection.)

Now we are ready to state the two rules, for every given round t.

- ***selection rule.*** Select a relevant ball B with the maximal index (break ties arbitrarily). Select an arbitrary arm y such that $(x_t, y) \in \texttt{dom}_t(B)$.
- ***activation rule.*** Suppose the selection rule selects a relevant ball B such that $\texttt{conf}_t(B) \leq r(B)$ after this round. Then, letting y be the arm selected in this round, a ball with center (x_t, y) and radius $\frac{1}{2}\,r(B)$ is activated. (B is then called the *parent* of this ball.)

See Algorithm 1 for the pseudocode.

It remains to define the index $I_t(B)$. Let $\texttt{rew}_t(B)$ be the total payoff from all rounds up to $t-1$ in which ball B has been selected by the algorithm. Then the average payoff from B is $\nu_t(B) \triangleq \frac{\texttt{rew}_t(B)}{\max(1,\, n_t(B))}$. The *pre-index* of B is defined as the average $\nu_t(B)$ plus an "uncertainty term":

$$I_t^{\text{pre}}(B) \triangleq \nu_t(B) + r(B) + \texttt{conf}_t(B). \tag{11}$$

The "uncertainty term" in (11) reflects both uncertainty due to a location in the metric space, via $r(B)$, and uncertainty due to an insufficient number of samples, via $\texttt{conf}_t(B)$.

The index of B is obtained by taking a minimum over all active balls B':

$$I_t(B) \triangleq r(B) + \min_{B' \in \mathcal{A}_t} \left(I_t^{\text{pre}}(B') + \mathcal{D}(B, B') \right), \tag{12}$$

where $\mathcal{D}(B, B')$ is the distance between the centers of the two balls.

Discussion. The meaning of index and pre-index is as follows. Both are upper confidence bound (*UCB*, for short) for expected rewards in B. Pre-index is a UCB for $\mu(B)$, the expected payoff from the center of B; essentially, it is the best UCB on $\mu(B)$ that can be obtained from the observations of B alone. The min expression in (12) is an improved UCB on $\mu(B)$, refined using observations from all other active balls. Finally, index is, essentially, the best available UCB for the expected reward of any pair $(x, y) \in B$.

Relevant balls are defined through the notion of the "domain" to ensure the following property: in each round when a parent ball is selected, some other ball is activated. This property allows us to "charge" the regret accumulated in each such round to the corresponding activated ball.

Running time. The running time is dominated by determining which active balls are relevant. Formally, we assume an oracle that inputs context x and a finite sequence $(B, B_1, \ldots, B_n)$ of balls in the similarity space, and outputs an arm y such that $(x, y) \in B \setminus \cup_{j=1}^{n} B_j$ if such arm exists, and `null` otherwise. Then each round t can be implemented via n_t oracle calls with $n < n_t$ balls each, where n_t is the current number of active balls. Letting $f(n)$ denote the running time of one oracle call in terms of n, the running time for each round the algorithm is at most $n_T\, f(n_T)$.

While implementation of the oracle and running time $f(\cdot)$ depend on the specific similarity space, we can provide some upper bounds on n_T. First, a crude upper bound is $n_T \leq T$. Second, letting $\mathcal{F}_r$ be the collection of all active balls of radius r, we prove that $|\mathcal{F}_r|$ is at most N_r, the r-zooming number of the problem instance. Third, $|\mathcal{F}_r| \leq c_{\text{DBL}}\, T r^2$, where c_{DBL} is the doubling constant of the similarity space. (This is because each active ball must be played at least r^{-2} times before it becomes a parent ball, and each parent ball can have at most c_{DBL} children.) Putting this together, we obtain $n_T \leq \sum_r \min(c_{\text{DBL}}\, T r^2,\, N_r)$, where the sum is over all $r = 2^{-j}$, $j \in \mathbb{N}$.

4.3 Analysis of the Algorithm: Proof of Theorem 1

We start by observing that the activation rule ensures several important invariants.

Claim 2 *The following invariants are maintained:*

- *(confidence) for all times t and all active balls B,*

$$\texttt{conf}_t(B) \leq r(B) \text{ if and only if } B \text{ is a parent ball.}$$

- *(covering) in each round t, the domains of active balls cover the similarity space.*
- *(separation) for any two active balls of radius r, their centers are at distance $\geq r$.*

Proof The confidence invariant is immediate from the activation rule.

For the covering invariant, note that $\cup_{B \in \mathcal{A}} \texttt{dom}\,(B, \mathcal{A}) = \cup_{B \in \mathcal{A}} B$ for any finite collection $\mathcal{A}$ of balls in the similarity space. (For each $v \in \cup_{B \in \mathcal{A}} B$, consider a smallest radius ball in $\mathcal{A}$ that contains B. Then $v \in \texttt{dom}\,(B, \mathcal{A})$.) The covering invariant then follows since $\mathcal{A}_t$ contains a ball that covers the entire similarity space.

To show the separation invariant, let B and B' be two balls of radius r such that B is activated at time t, with parent B^{par}, and B' is activated before time t. The center of B

is some point $(x_t, y_t) \in \texttt{dom}\,(B^{\text{par}}, \mathcal{A}_t)$. Since $r(B^{\text{par}}) > r(B')$, it follows that $(x_t, y_t) \notin B'$. ■

Throughout the analysis we will use the following notation. For a ball B with center $(x, y) \in \mathcal{P}$, define the expected payoff of B as $\mu(B) \triangleq \mu(x, y)$. Let $B_t^{\texttt{sel}}$ be the active ball selected by the algorithm in round t. Recall that the *badness* of $(x, y) \in \mathcal{P}$ is defined as $\Delta(x, y) \triangleq \mu^*(x) - \mu(x, y)$.

Claim 3 *If ball B is active in round t, then with probability at least $1 - T^{-2}$ we have that*

$$|\nu_t(B) - \mu(B)| \leq r(B) + \texttt{conf}_t(B). \tag{13}$$

Proof Fix ball V with center (x, y). Let S be the set of rounds $s \leq t$ when ball B was selected by the algorithm, and let $n = |S|$ be the number of such rounds. Then $\nu_t(B) = \frac{1}{n} \sum_{s \in S} \pi_s(x_s, y_s)$.

Define $Z_k = \sum (\pi_s(x_s, y_s) - \mu(x_s, y_s))$, where the sum is taken over the k smallest elements $s \in S$. Then $\{Z_{k \wedge n}\}_{k \in \mathbb{N}}$ is a martingale with bounded increments. (Note that n here is a random variable.) So by the Azuma-Hoeffding inequality with probability at least $1 - T^{-3}$ it holds that $\frac{1}{k} |Z_{k \wedge n}| \leq \texttt{conf}_t(B)$, for each $k \leq T$. Taking the Union Bound, it follows that $\frac{1}{n} |Z_n| \leq \texttt{conf}_t(B)$. Note that $|\mu(x_s, y_s) - \mu(B)| \leq r(B)$ for each $s \in S$, so $|\nu_t(B) - \mu(B)| \leq r(B) + \frac{1}{n} |Z_n|$, which completes the proof. ■

Note that (13) implies $I^{\text{pre}}(B) \geq \mu(B)$, so that $I^{\text{pre}}(B)$ is indeed a UCB on $\mu(B)$.

Call a run of the algorithm *clean* if (13) holds for each round. From now on we will focus on a clean run, and argue deterministically using (13). The heart of the analysis is the following lemma.

Lemma 4 *Consider a clean run of the algorithm. Then $\Delta(x_t, y_t) \leq 14\, r(B_t^{\texttt{sel}})$ in each round t.*

Proof Fix round t. By the covering invariant, $(x_t,\, y^*(x_t)) \in B$ for some active ball B. Recall from (12) that $I_t(B) = r(B) + I^{\text{pre}}(B') + \mathcal{D}(B, B')$ for some active ball B'. Therefore

$$\begin{aligned}
I_t(B_t^{\texttt{sel}}) &\geq I_t(B) = I^{\text{pre}}(B') + r(B) + \mathcal{D}(B, B') && \text{(selection rule, defn of index (12))} \\
&\geq \mu(B') + r(B) + \mathcal{D}(B, B') && \text{(“clean run”)} \\
&\geq \mu(B) + r(B) \geq \mu(x_t,\, y^*(x_t)) = \mu^*(x_t). && \text{(Lipschitz property (1), twice)}
\end{aligned} \tag{14}$$

On the other hand, letting B^{par} be the parent of $B_t^{\texttt{sel}}$ and noting that by the activation rule

$$\max(\mathcal{D}(B_t^{\texttt{sel}}, B^{\text{par}}),\ \texttt{conf}_t(B^{\text{par}})) \leq r(B^{\text{par}}), \tag{15}$$

we can upper-bound $I_t(B_t^{\texttt{sel}})$ as follows:

$$\begin{aligned}
I^{\text{pre}}(B^{\text{par}}) &= \nu_t(B^{\text{par}}) + r(B^{\text{par}}) + \texttt{conf}_t(B^{\text{par}}) && \text{(defn of preindex (11))} \\
&\leq \mu(B^{\text{par}}) + 2\,r(B^{\text{par}}) + 2\,\texttt{conf}_t(B^{\text{par}}) && \text{(“clean run”)} \\
&\leq \mu(B^{\text{par}}) + 4\,r(B^{\text{par}}) && \text{(“parenthood” (15))} \\
&\leq \mu(B_t^{\texttt{sel}}) + 5\,r(B^{\text{par}}) && \text{(Lipschitz property (1))} \qquad (16) \\
I_t(B_t^{\texttt{sel}}) &\leq r(B_t^{\texttt{sel}}) + I^{\text{pre}}(B^{\text{par}}) + \mathcal{D}(B_t^{\texttt{sel}}, B^{\text{par}}) && \text{(defn of index (12))} \\
&\leq r(B_t^{\texttt{sel}}) + I^{\text{pre}}(B^{\text{par}}) + r(B^{\text{par}}) && \text{(“parenthood” (15))} \\
&\leq r(B_t^{\texttt{sel}}) + \mu(B_t^{\texttt{sel}}) + 6\,r(B^{\text{par}}) && \text{(by (16))} \\
&\leq \mu(B_t^{\texttt{sel}}) + 13\,r(B_t^{\texttt{sel}}) && (r(B^{\text{par}}) = 2\,r(B_t^{\texttt{sel}})) \\
&\leq \mu(x_t, y_t) + 14\,r(B_t^{\texttt{sel}}) && \text{(Lipschitz property (1)).} \qquad (17)
\end{aligned}$$

Putting the pieces together, $\mu^*(x_t) \leq I_t(B_t^{\texttt{sel}}) \leq \mu(x_t, y_t) + 14\,r(B_t^{\texttt{sel}})$. ■

Corollary 5 *In a clean run, if ball B is activated in round t then $\Delta(x_t, y_t) \leq 10\,r(B)$.*

Proof By the activation rule, $B_t^{\texttt{sel}}$ is the parent of B. Thus by Lemma 4 we immediately have $\Delta(x_t, y_t) \leq 14\,r(B_t^{\texttt{sel}}) = 28\,r(B)$.

To obtain the constant of 10 that is claimed here, we prove a more efficient special case of Lemma 4:

$$\text{if } B_t^{\texttt{sel}} \text{ is a parent ball then } \Delta(x_t, y_t) \leq 5\,r(B_t^{\texttt{sel}}). \qquad (18)$$

To prove (18), we simply replace (17) in the proof of Lemma 4 by similar inequality in terms of $I^{\text{pre}}(B_t^{\texttt{sel}})$ rather than $I^{\text{pre}}(B^{\text{par}})$:

$$\begin{aligned}
I_t(B_t^{\texttt{sel}}) &\leq r(B_t^{\texttt{sel}}) + I^{\text{pre}}(B_t^{\texttt{sel}}) && \text{(defn of index (12))} \\
&= \nu_t(B_t^{\texttt{sel}}) + 2\,r(B_t^{\texttt{sel}}) + \texttt{conf}_t(B_t^{\texttt{sel}}) && \text{(defns of pre-index (11))} \\
&\leq \mu(B_t^{\texttt{sel}}) + 3\,r(B_t^{\texttt{sel}}) + 2\,\texttt{conf}_t(B_t^{\texttt{sel}}) && \text{(“clean run”)} \\
&\leq \mu(x_t, y_t) + 5\,r(B_t^{\texttt{sel}})
\end{aligned}$$

For the last inequality, we use the fact that $\texttt{conf}_t(B_t^{\texttt{sel}}) \leq r(B_t^{\texttt{sel}})$ whenever $B_t^{\texttt{sel}}$ is a parent ball. ■

Now we are ready for the final regret computation. For a given $r = 2^{-i}$, $i \in \mathbb{N}$, let $\mathcal{F}_r$ be the collection of all balls of radius r that have been activated throughout the execution of the algorithm. Note that in each round, if a parent ball is selected then some other ball is activated. Thus, we can partition the rounds among active balls as follows: for each ball $B \in \mathcal{F}_r$, let S_B be the set of rounds which consists of the round when B was activated and all rounds t when B was selected and was not a parent ball.[6] It is easy to see that

6. A given ball B can be selected even after it becomes a parent ball, but in such round some other ball B is activated, so this round is included in $S_{B'}$.

$|S_B| \leq O(r^{-2} \log T)$. Moreover, by Lemma 4 and Corollary 5 we have $\Delta(x_t, y_t) \leq 15\, r$ in each round $t \in S_B$.

If ball $B \in \mathcal{F}_r$ is activated in round t, then Corollary 5 asserts that its center (x_t, y_t) lies in the set $\mathcal{P}_{\mu,r}$, as defined in (8). By the separation invariant, the centers of balls in $\mathcal{F}_r$ are within distance at least r from one another. It follows that $|\mathcal{F}_r| \leq N_r$, where N_r is the r-zooming number.

Fixing some $r_0 \in (0,1)$, note that in each rounds t when a ball of radius $< r_0$ was selected, regret is $\Delta(x_t, y_t) \leq O(r_0)$, so the total regret from all such rounds is at most $O(r_0\, T)$. Therefore, contextual regret can be written as follows:

$$\begin{aligned}
R(T) &= \textstyle\sum_{t=1}^{T} \Delta(x_t, y_t) \\
&= O(r_0\, T) + \textstyle\sum_{r=2^{-i}:\ r_0 \leq r \leq 1}\ \sum_{B \in \mathcal{F}_r} \sum_{t \in S_B} \Delta(x_t, y_t) \\
&\leq O(r_0\, T) + \textstyle\sum_{r=2^{-i}:\ r_0 \leq r \leq 1}\ \sum_{B \in \mathcal{F}_r} |S_B|\, O(r) \\
&\leq O\left(r_0 T \ + \textstyle\sum_{r=2^{-i}:\ r_0 \leq r \leq 1}\ \frac{1}{r}\, N_r \log(T) \right).
\end{aligned}$$

The N_r-type regret guarantee in Theorem 1 follows by taking inf on all $r_0 \in (0,1)$.

4.4 Improved Regret Bounds

Let us provide regret bounds that take into account "benign" context arrivals. The main difficulty here is to develop the corresponding definitions; the analysis then carries over without much modification. The added value is two-fold: first, we establish the intuition that benign context arrivals matter, and then the specific regret bound is used in Section 6.2 to match the result in Slivkins and Upfal (2008).

A crucial step in the proof of Theorem 1 is to bound the number of active radius-r balls by $N_r(\mathcal{P}_{\mu,r})$, which is accomplished by observing that their centers form an r-packing S of $\mathcal{P}_{\mu,r}$. We make this step more efficient, as follows. An active radius-r ball is called *full* if $\texttt{conf}_t(B) \leq r$ for some round t. Note that each active ball is either full or a child of some other ball that is full. The number of children of a given ball is bounded by the doubling constant of the similarity space. Thus, it suffices to consider the number of active radius-r balls that are full, which is at most $N_r(\mathcal{P}_{\mu,r})$, and potentially much smaller.

Consider active radius-r active balls that are full. Their centers form an r-packing S of $\mathcal{P}_{\mu,r}$ with an additional property: each point $p \in S$ is assigned at least $1/r^2$ context arrivals x_t so that $(x_t, y) \in B(p, r)$ for some arm y, and each context arrival is assigned to at most one point in S.[7] A set $S \subset \mathcal{P}$ with this property is called *r-consistent* (with context arrivals). The *adjusted r-packing number* of a set $\mathcal{P}' \subset \mathcal{P}$, denoted $N_r^{\text{adj}}(\mathcal{P}')$, is the maximal size of an r-consistent r-packing of $\mathcal{P}'$. It can be much smaller than the r-packing number of $\mathcal{P}'$ if most context arrivals fall into a small region of the similarity space.

We make one further optimization, tailored to the application in Section 6.2. Informally, we take advantage of context arrivals x_t such that expected payoff $\mu(x_t, y)$ is either optimal or very suboptimal. A point $(x, y) \in \mathcal{P}$ is called an *r-winner* if for each $(x', y') \in B((x, y),\, 2r)$ it holds that $\mu(x', y') = \mu^*(x')$. Let $\mathcal{W}_{\mu,r}$ be the set of all r-winners. It is easy to see that if B is a radius-r ball centered at an r-winner, and B or its child is se-

7. Each point $p \in S$ is assigned all contexts x_t such that the corresponding ball is chosen in round t.

lected in a given round, then this round does not contribute to contextual regret. Therefore, it suffices to consider (r-consistent) r-packings of $\mathcal{P}_{\mu,r} \setminus \mathcal{W}_{\mu,r}$.

Our final guarantee is in terms of $N^{\text{adj}}(\mathcal{P}_{\mu,r} \setminus \mathcal{W}_{\mu,r})$, which we term the *adjusted r-zooming number*.

Theorem 6 *Consider the contextual MAB problem with stochastic payoffs. The contextual regret $R(T)$ of the contextual zooming algorithm satisfies (5), where N_r is the adjusted r-zooming number and C_0 is the doubling constant of the similarity space times some absolute constant. Consequently, $R(T)$ satisfies the dimension-type guarantee (7), where d_c is the corresponding covering-type dimension.*

5. Lower Bounds

We match the upper bound in Theorem 1 up to $O(\log T)$ factors. Our lower bound is very general: it applies to an arbitrary product similarity space, and moreover for a given similarity space it matches, up to $O(\log T)$ factors, any fixed value of the upper bound (as explained below).

We construct a distribution $\mathcal{I}$ over problem instances on a given metric space, so that the lower bound is for a problem instance drawn from this distribution. A single problem instance would not suffice to establish a lower bound because a trivial algorithm that picks arm $y^*(x)$ for each context x will achieve regret 0.

The distribution $\mathcal{I}$ satisfies the following two properties: the upper bound in Theorem 1 is uniformly bounded from above by some number R, and any algorithm must incur regret at least $\Omega(R/\log T)$ in expectation over $\mathcal{I}$. Moreover, we constrict such $\mathcal{I}$ for every possible value of the upper bound in Theorem 1 on a given metric space, i.e., not just for problem instances that are "hard" for this metric space.

To formulate our result, let $R^{\text{UB}}_\mu(T)$ denote the upper bound in Theorem 1, i.e., is the right-hand side of (5) where $N_r = N_r(\mathcal{P}_{\mu,r})$ is the r-zooming number. Let $R^{\text{UB}}(T)$ denote the pessimistic version of this bound, namely right-hand side of (5) where $N_r = N_r(\mathcal{P})$ is the packing number of $\mathcal{P}$.

Theorem 7 *Consider the contextual MAB problem with stochastic payoffs, Let $(\mathcal{P}, \mathcal{D})$ be a product similarity space. Fix an arbitrary time horizon T and a positive number $R \leq R^{UB}(T)$. Then there exists a distribution $\mathcal{I}$ over problem instances on $(\mathcal{P}, \mathcal{D})$ with the following two properties:*

(a) $R^{UB}_\mu(T) \leq O(R)$ for each problem instance in $\texttt{support}(\mathcal{I})$.

(b) for any contextual bandit algorithm it holds that $\mathbb{E}_{\mathcal{I}}[R(T)] \geq \Omega(R/\log T)$,

To prove this theorem, we build on the lower-bounding technique from Auer et al. (2002b), and its extension to (context-free) bandits in metric spaces in Kleinberg (2004). In particular, we use the basic *needle-in-the-haystack* example from Auer et al. (2002b), where the "haystack" consists of several arms with expected payoff $\frac{1}{2}$, and the "needle" is an arm whose expected payoff is slightly higher.

5.1 The Lower-Bounding Construction

Our construction is parameterized by two numbers: $r \in (0, \frac{1}{2}]$ and $N \leq N_r(\mathcal{P})$, where $N_r(\mathcal{P})$ is the r-packing number of $\mathcal{P}$. Given these parameters, we construct a collection $\mathcal{I} = \mathcal{I}_{N,r}$ of $\Theta(N)$ problem instances as follows.

Let $N_{\mathrm{X},r}$ be the r-packing number of X in the context space, and let $N_{\mathrm{Y},r}$ be the r-packing number of Y in the arms space. Note that $N_r(\mathcal{P}) = N_{\mathrm{X},r} \times N_{\mathrm{Y},r}$. For simplicity, let us assume that $N = n_{\mathrm{X}}\, n_{\mathrm{Y}}$, where $1 \leq n_X \leq N_{\mathrm{X},r}$ and $2 \leq n_{\mathrm{Y}} \leq N_{\mathrm{Y},r}$.

An *r-net* is the set S of points in a metric space such that any two points in S are at distance $> r$ from each other, and each point in the metric space is within distance $\leq r$ from some point in S. Recall that any r-net on the context space has size at least $N_{\mathrm{X},r}$. Let S_{X} be an arbitrary set of n_{X} points from one such r-net. Similarly, let S_{Y} be an arbitrary set of n_{Y} points from some r-net on the arms space. The sequence $x_{(1..T)}$ of context arrivals is any fixed permutation over the points in S_{X}, repeated indefinitely.

All problem instances in $\mathcal{I}$ have 0-1 payoffs. For each $x \in S_{\mathrm{X}}$ we construct a needle-in-the-haystack example on the set S_{Y}. Namely, we pick one point $y^*(x) \in S_{\mathrm{Y}}$ to be the "needle", and define $\mu(x, y^*(x)) = \frac{1}{2} + \frac{r}{4}$, and $\mu(x, y) = \frac{1}{2} + \frac{r}{8}$ for each $y \in S_{\mathrm{Y}} \setminus \{y^*(x)\}$. We smoothen the expected payoffs so that far from $S_{\mathrm{X}} \times S_{\mathrm{Y}}$ expected payoffs are $\frac{1}{2}$ and the Lipschitz condition (1) holds:

$$\mu(x, y) \triangleq \max_{(x_0, y_0) \in S_{\mathrm{X}} \times S_{\mathrm{Y}}} \max\left(\, \tfrac{1}{2},\ \mu(x_0, y_0) - \mathcal{D}_{\mathrm{X}}(x, x_0) - \mathcal{D}_{\mathrm{Y}}(y, y_0)\ \right). \tag{19}$$

Note that we obtain a distinct problem instance for each function $y^*(\cdot) : S_{\mathrm{X}} \to S_{\mathrm{Y}}$. This completes our construction.

5.2 Analysis

The useful properties of the above construction are summarized in the following lemma:

Lemma 8 *Fix $r \in (0, \frac{1}{2}]$ and $N \leq N_r(\mathcal{P})$. Let $\mathcal{I} = \mathcal{I}_{N,r}$ and $T_0 = N\, r^{-2}$. Then:*

(i) for each problem instance in $\mathcal{I}$ it holds that $R_\mu^{UB}(T_0) \leq O(N/r)(\log T_0)$.

(ii) any contextual bandit algorithm has regret $\mathbb{E}_{\mathcal{I}}[R(T_0)] \geq \Omega(N/r)$ for a problem instance chosen uniformly at random from $\mathcal{I}$.

For the lower bound in Lemma 8, the idea is that in T rounds each context in S_{X} contributes $\Omega(|S_{\mathrm{Y}}|/r)$ to contextual regret, resulting in total contextual regret $\Omega(N/r)$.

Before we proceed to prove Lemma 8, let us use it to derive Theorem 7. Fix an arbitrary time horizon T and a positive number $R \leq R^{\mathrm{UB}}(T)$. Recall that since $N_r(\mathcal{P})$ is non-increasing in r, for some constant $C > 0$ it holds that

$$R^{\mathrm{UB}}(T) = C \times \inf_{r \in (0,1)} \left(rT + \tfrac{1}{r}\, N_r(\mathcal{P}) \log T \right). \tag{20}$$

Claim 9 *Let $r = \frac{R}{2CT(1+\log T)}$. Then $r \leq \frac{1}{2}$ and $Tr^2 \leq N_r(\mathcal{P})$.*

Proof Denote $k(r) = N_r(\mathcal{P})$ and consider function $f(r) \triangleq k(r)/r^2$. This function is non-increasing in r; $f(1) = 1$ and $f(r) \to \infty$ for $r \to 0$. Therefore there exists $r_0 \in (0, 1)$ such that $f(r_0) \leq T \leq f(r_0/2)$. Re-writing this, we obtain

$$k(r_0) \leq T\, r_0^2 \leq 4\, k(r_0/2).$$

It follows that

$$R \leq R^{\text{UB}}(T) \leq C(Tr_0 + \tfrac{1}{r_0}\, k(r_0) \log T) \leq C\, Tr_0(1 + \log T).$$

Thus $r \leq r_0/2$ and finally $T\, r^2 \leq T\, r_0^2/4 \leq k(r_0/2) \leq k(r) = N_r(\mathcal{P})$. ■

So, Lemma 8 with $r \triangleq \frac{R}{2C\, T(1+\log T)}$ and $N \triangleq T\, r^2$. implies Theorem 7.

5.3 Proof of Lemma 8

Claim 10 *Collection $\mathcal{I}$ consists of valid instances of contextual MAB problem with similarity space $(\mathcal{P}, \mathcal{D})$.*

Proof We need to prove that each problem instance in $\mathcal{P}$ satisfies the Lipschitz condition (1). Assume the Lipschitz condition (1) is violated for some points (x, y), $(x', y') \in X \times Y$. For brevity, let $p = (x, y)$, $p' = (x', y')$, and let us write $\mu(p) \triangleq \mu(x, y)$. Then $|\mu(p) - \mu(p')| > \mathcal{D}(p, p')$.

By (19), $\mu(\cdot) \in [\frac{1}{2}, \frac{1}{2} + \frac{r}{4}]$, so $\mathcal{D}(p, p') < \frac{r}{4}$.

Without loss of generality, $\mu(p) > \mu(p')$. In particular, $\mu(p) > \frac{1}{2}$. Therefore there exists $p_0 = (x_0, y_0) \in S_{\text{X}} \times S_{\text{Y}}$ such that $\mathcal{D}(p, p_0) < \frac{r}{4}$. Then $\mathcal{D}(p', p_0) < \frac{r}{2}$ by triangle inequality.

Now, for any other $p_0' \in S_{\text{X}} \times S_{\text{Y}}$ it holds that $\mathcal{D}(p_0, p_0') > r$, and thus by triangle inequality $\mathcal{D}(p, p_0') > \frac{3r}{4}$ and $\mathcal{D}(p', p_0') > \frac{r}{2}$. It follows that (19) can be simplified as follows:

$$\begin{cases} \mu(p) &= \max(\frac{1}{2},\, \mu(p_0) - \mathcal{D}(p, p_0)), \\ \mu(p') &= \max(\frac{1}{2},\, \mu(p_0) - \mathcal{D}(p', p_0)). \end{cases}$$

Therefore

$$\begin{aligned} |\mu(p) - \mu(p')| &= \mu(p) - \mu(p') \\ &= (\mu(p_0) - \mathcal{D}(p, p_0)) - \max(\tfrac{1}{2},\, \mu(p_0) - \mathcal{D}(p', p_0)) \\ &\leq (\mu(p_0) - \mathcal{D}(p, p_0)) - (\mu(p_0) - \mathcal{D}(p', p_0))) \\ &= \mathcal{D}(p', p_0) - \mathcal{D}(p, p_0) \leq \mathcal{D}(p, p'). \end{aligned}$$

So we have obtained a contradiction. ■

Claim 11 *For each instance in $\mathcal{P}$ and $T_0 = N\, r^{-2}$ it holds that $R_\mu^{UB}(T_0) \leq O(N/r)(\log T_0)$.*

Proof Recall that $R_\mu^{\text{UB}}(T_0)$ is the right-hand side of (5) with $N_r = N_r(\mathcal{P}_{\mu,r})$, where $\mathcal{P}_{\mu,r}$ is defined by (8).

Fix $r' > 0$. It is easy to see that

$$\mathcal{P}_{\mu,\, r'} \subset \cup_{p \in S_{\text{X}} \times S_{\text{Y}}}\, B(p, \tfrac{r}{4}).$$

It follows that $N_{r'}(\mathcal{P}_{\mu,r'}) \leq N$ whenever $r' \geq \frac{r}{4}$. Therefore, taking $r_0 = \frac{r}{4}$ in (5), we obtain

$$R_\mu^{\text{UB}}(T_0) \leq O(rT_0 + \tfrac{N}{r} \log T_0) = O(N/r)(\log T_0).$$

■

Claim 12 *Fix a contextual bandit algorithm $\mathcal{A}$. This algorithm has regret $\mathbb{E}_{\mathcal{I}}[R(T_0)] \geq \Omega(N/r)$ for a problem instance chosen uniformly at random from $\mathcal{I}$, where $T_0 = N\, r^{-2}$.*

Proof Let $R(x,T)$ be the contribution of each context $x \in S_{\mathrm{X}}$ to contextual regret:

$$R(x,T) = \sum_{t:\, x_t = x} \mu^*(x) - \mu(x, y_t),$$

where y_t is the arm chosen by the algorithm in round t. Our goal is to show that $R(x,T_0) \geq \Omega(r\, n_{\mathrm{Y}})$.

We will consider each context $x \in S_{\mathrm{X}}$ separately: the rounds when x arrives form an instance I_x of a context-free bandit problem that lasts for $T_0/n_{\mathrm{X}} = n_{\mathrm{Y}}\, r^{-2}$ rounds, where expected payoffs are given by $\mu(x,\cdot)$ as defined in (19). Let $\mathcal{I}_x$ be the family of all such instances I_x.

A uniform distribution over $\mathcal{I}$ can be reformulated as follows: for each $x \in S_{\mathrm{X}}$, pick the "needle" $y^*(x)$ independently and uniformly at random from S_{Y}. This induces a uniform distribution over instances in $\mathcal{I}_x$, for each context $x \in S_{\mathrm{X}}$. Informally, knowing full or partial information about $y^*(x)$ for some x reveals no information whatsoever about $y^*(x')$ for any $x' \neq x$.

Formally, the contextual bandit algorithm $\mathcal{A}$ induces a bandit algorithm $\mathcal{A}_x$ for I_x, for each context $x \in S_{\mathrm{X}}$: the $\mathcal{A}_x$ simulates the problem instance for $\mathcal{A}$ for all contexts $x' \neq x$ (starting from the "needles" $y^*(x')$ chosen independently and uniformly at random from S_{Y}). Then $\mathcal{A}_x$ has expected regret $R_x(T)$ which satisfies $\mathbb{E}[\, R(T)\,] = \mathbb{E}[\, R(x,T)\,]$, where the expectations on both sides are over the randomness in the respective algorithm and the random choice of the problem instance (resp., from $\mathcal{I}_x$ and from $\mathcal{I}$).

Thus, it remains to handle each $\mathcal{I}_x$ separately: i.e., to prove that the expected regret of any bandit algorithm on an instance drawn uniformly at random from $\mathcal{I}_x$ is at least $\Omega(r\, n_{\mathrm{Y}})$. We use the KL-divergence technique that originated in Auer et al. (2002b). If the set of arms were exactly S_{Y}, then the desired lower bound would follow from Auer et al. (2002b) directly. To handle the problem instances in $\mathcal{I}_x$, we use an extension of the technique from Auer et al. (2002b), which is implicit in Kleinberg (2004) and encapsulated as a stand-alone theorem in Kleinberg et al. (2013). We restate this theorem as Theorem 26 in Appendix A.

It is easy to check that the family $\mathcal{I}_x$ of problem instances satisfies the preconditions in Theorem 26. Fix $x \in S_{\mathrm{X}}$. For a given choice of the "needle" $y^* = y^*(x) \in S_{\mathrm{Y}}$, let $\mu(x, y \,|\, y^*)$. be the expected payoff of each arm y, and let $\nu_{y^*}(\cdot) = \mu(x, \cdot \,|\, y^*)$ be the corresponding payoff function for the bandit instance I_x. Then $\{\nu_{y^*}\}$, $y^* \in S_{\mathrm{Y}}$ is an "(ϵ, k)-ensemble" for $\epsilon = \frac{r}{8}$ and $k = |S_{\mathrm{Y}}|$. ■

6. Applications of Contextual Zooming

We describe several applications of contextual zooming: to MAB with slow adversarial change (Section 6.1), to MAB with stochastically evolving payoffs (Section 6.2), and to the "sleeping bandits" problem (Section 6.3). In particular, we recover some of the main results in Slivkins and Upfal (2008) and Kleinberg et al. (2008a). Also, in Section 6.4 we

discuss a recent application of contextual zooming to bandit learning-to-rank, which has been published in Slivkins et al. (2013).

6.1 MAB with Slow Adversarial Change

Consider the (context-free) adversarial MAB problem in which expected payoffs of each arm change over time *gradually*. Specifically, we assume that expected payoff of each arm y changes by at most σ_y in each round, for some a-priori known *volatilities* σ_y. The algorithm's goal here is continuously adapt to the changing environment, rather than converge to the best fixed mapping from contexts to arms. We call this setting the ***drifting MAB problem***.

Formally, our benchmark is a fictitious algorithm which in each round selects an arm that maximizes expected payoff for the current context. The difference in expected payoff between this benchmark and a given algorithm is called *dynamic regret* of this algorithm. It is easy to see that the worst-case dynamic regret of any algorithm cannot be sublinear in time.[8] We are primarily interested in algorithm's long-term performance, as quantified by *average* dynamic regret $\hat{R}(T) \triangleq R(T)/T$. Our goal is to bound the limit $\lim_{T\to\infty} \hat{R}(T)$ in terms of the parameters: the number of arms and the volatilities σ_y. (In general, such upper bound is non-trivial as long as it is smaller than 1, since all payoffs are at most 1.)

We restate this setting as a contextual MAB problem with stochastic payoffs in which the t-th context arrival is simply $x_t = t$. Then $\mu(t, y)$ is the expected payoff of arm y at time t, and dynamic regret coincides with contextual regret specialized to the case $x_t = t$. Each arm y satisfies a "temporal constraint":

$$|\mu(t, y) - \mu(t', y)| \leq \sigma_y \, |t - t'| \tag{21}$$

for some constant σ_y. To set up the corresponding similarity space $(\mathcal{P}, \mathcal{D})$, let $\mathcal{P} = [T] \times Y$, and

$$\mathcal{D}((t, y), (t', y')) = \min(1, \ \sigma_y \, |t - t'| + \mathbf{1}_{\{y \neq y'\}}). \tag{22}$$

Our solution for the drifting MAB problem is the contextual zooming algorithm parameterized by the similarity space $(\mathcal{P}, \mathcal{D})$. To obtain guarantees for the long-term performance, we run contextual zooming with a suitably chosen time horizon T_0, and restart it every T_0 rounds; we call this version *contextual zooming with period* T_0. Periodically restarting the algorithm is a simple way to prevent the change over time from becoming too large; it suffices to obtain strong provable guarantees.

The general provable guarantees are provided by Theorem 1 and Theorem 6. Below we work out some specific, tractable corollaries.

Corollary 13 *Consider the drifting MAB problem with k arms and volatilities $\sigma_y \equiv \sigma$. Contextual zooming with period T_0 has average dynamic regret $\hat{R}(T) = O(k\sigma \log T_0)^{1/3}$, whenever $T \geq T_0 \geq (\frac{k}{\sigma^2})^{1/3} \log \frac{k}{\sigma}$.*

8. For example, consider problem instances with two arms such that the payoff of each arm in each round is either $\frac{1}{2}$ or $\frac{1}{2} + \sigma$ (and can change from round to round). Over this family of problem instances, dynamic regret in T rounds is at least $\frac{1}{2}\sigma T$.

Proof It suffices to upper-bound regret in a single period. Indeed, if $R(T_0) \leq R$ for any problem instance, then $R(T) \leq R\lceil T/T_0 \rceil$ for any $T > T_0$. It follows that $\hat{R}(T) \leq 2\,\hat{R}(T_0)$. Therefore, from here on we can focus on analyzing contextual zooming itself, rather than contextual zooming with a period.

The main step is to derive the regret bound (5) with a specific upper bound on N_r. We will show that

$$\text{dynamic regret } R(\cdot) \text{ satisfies (5) with } N_r \leq k\,\lceil \tfrac{T\sigma}{r} \rceil. \tag{23}$$

Plugging $N_r \leq k\,(1 + \frac{T\sigma}{r})$ into (5) and taking $r_0 = (k\sigma \log T)^{1/3}$ we obtain[9]

$$R(T) \leq O(T)(k\sigma \log T)^{1/3} + O(\tfrac{k^2}{\sigma})^{1/3}(\log T) \qquad \forall T \geq 1.$$

Therefore, for any $T \geq (\frac{k}{\sigma^2})^{1/3} \log \frac{k}{\sigma}$ we have $\hat{R}(T) = O(k\sigma \log T)^{1/3}$.

It remains to prove (23). We use a pessimistic version of Theorem 1: (5) with $N_r = N_r(\mathcal{P})$, the r-packing number of $\mathcal{P}$. Fix $r \in (0, 1]$. For any r-packing S of $\mathcal{P}$ and each arm y, each time interval I of duration $\Delta_r \triangleq r/\sigma$ provides at most one point for S: there exists at most one time $t \in I$ such that $(t, y) \in S$. Since there are at most $\lceil T/\Delta_r \rceil$ such intervals I, it follows that $N_r(\mathcal{P}) \leq k\,\lceil T/\Delta_r \rceil \leq k\,(1 + T\frac{\sigma}{r})$. ■

The restriction $\sigma_y \equiv \sigma$ is non-essential: it is not hard to obtain the same bound with $\sigma = \frac{1}{k}\sum_y \sigma_y$. Modifying the construction in Section 5 (details omitted from this version) one can show that Corollary 13 is optimal up to $O(\log T)$ factors.

Drifting MAB with spatial constraints. The temporal version ($x_t = t$) of our contextual MAB setting with stochastic payoffs subsumes the drifting MAB problem and furthermore allows to combine the temporal constraints (21) described above (for each arm, across time) with "spatial constraints" (for each time, across arms). To the best of our knowledge, such MAB models are quite rare in the literature.[10] A clean example is

$$\mathcal{D}((t, y),\, (t', y')) = \min(1,\; \sigma\,|t - t'| + \mathcal{D}_{\mathrm{Y}}(y, y')), \tag{24}$$

where $(Y, \mathcal{D}_{\mathrm{Y}})$ is the arms space. For this example, we can obtain an analog of Corollary 13, where the regret bound depends on the covering dimension of the arms space $(Y, \mathcal{D}_{\mathrm{Y}})$.

Corollary 14 *Consider the drifting MAB problem with spatial constraints (24), where σ is the volatility. Let d be the covering dimension of the arms space, with multiplier k. Contextual zooming with period T_0 has average dynamic regret $\hat{R}(T) = O(k\,\sigma \log T_0)^{\frac{1}{d+3}}$, whenever $T \geq T_0 \geq k^{\frac{1}{d+3}}\,\sigma^{-\frac{d+2}{d+3}}\,\log \frac{k}{\sigma}$.*

Remark. We obtain Corollary 13 as a special case by setting $d = 0$.

9. This choice of r_0 minimizes the inf expression in (5) up to constant factors by equating the two summands.
10. The only other MAB model with this flavor that we are aware of, found in Hazan and Kale (2009), combines linear payoffs and bounded "total variation" (aggregate temporal change) of the cost functions.

Proof It suffices to bound $\hat{R}(T_0)$ for (non-periodic) contextual zooming. First we bound the r-covering number of the similarity space $(\mathcal{P}, \mathcal{D})$:

$$N_r(\mathcal{P}) = N_r^{\mathrm{X}}(X) \times N_r^{\mathrm{Y}}(Y) \leq \lceil \tfrac{T\sigma}{r} \rceil \, k \, r^{-d},$$

where $N_r^{\mathrm{X}}(\cdot)$ is the r-covering number in the context space, and $N_r^{\mathrm{Y}}(\cdot)$ is that in the arms space. We worked out the former for Corollary 13. Plugging this into (5) and taking $r_0 = (k \, \sigma \log T)^{1/(3+d)}$, we obtain

$$R(T) \leq O(T)(k\sigma \log T)^{\frac{1}{d+3}} + O\left(k^{\frac{2}{d+3}} \, \sigma^{\frac{d+1}{d+3}} \, \log T \right) \qquad \forall T \geq 1.$$

The desired bound on $\hat{R}(T_0)$ follows easily. ■

6.2 Bandits with Stochastically Evolving Payoffs

We consider a special case of drifting MAB problem in which expected payoffs of each arm evolve over time according to a stochastic process with a uniform stationary distribution. We obtain improved regret bounds for contextual zooming, taking advantage of the full power of our analysis in Section 4.

In particular, we address a version in which the stochastic process is a random walk with step $\pm\sigma$. This version has been previously studied in Slivkins and Upfal (2008) under the name "Dynamic MAB". For the main case ($\sigma_i \equiv \sigma$), our regret bound for Dynamic MAB matches that in Slivkins and Upfal (2008).

To improve the flow of the paper, the proofs are deferred to Appendix 7.

Uniform marginals. First we address the general version that we call *drifting MAB with uniform marginals*. Formally, we assume that expected payoffs $\mu(\cdot, y)$ of each arm y evolve over time according to some stochastic process Γ_y that satisfies (21). We assume that the processes Γ_y, $y \in Y$ are mutually independent, and moreover that the marginal distributions $\mu(t, y)$ are uniform on $[0, 1]$, for each time t and each arm y.[11] We are interested in $\mathbb{E}_\Gamma[\hat{R}(T)]$, average dynamic regret in expectation over the processes Γ_y.

We obtain a stronger version of (23) via Theorem 6. To use this theorem, we need to bound the adjusted r-zooming number, call it N_r. We show that

$$\mathbb{E}_\Gamma[N_r] = O(kr)\lceil \tfrac{T\sigma}{r} \rceil \text{ and } \left(r < \sigma^{1/3} \Rightarrow N_r = 0 \right). \tag{25}$$

Then we obtain a different bound on dynamic regret, which is stronger than Corollary 13 for $k < \sigma^{-1/2}$.

Corollary 15 *Consider drifting MAB with uniform marginals, with k arms and volatilities $\sigma_y \equiv \sigma$. Contextual zooming with period T_0 satisfies $\mathbb{E}_\Gamma[\hat{R}(T)] = O(k \, \sigma^{2/3} \, \log T_0)$, whenever $T \geq T_0 \geq \sigma^{-2/3} \log \frac{1}{\sigma}$.*

11. For example, this assumption is satisfied by any Markov Chain on $[0, 1]$ with stationary initial distribution.

The crux of the proof is to show (25). Interestingly, it involves using all three optimizations in Theorem 6: $N_r(\mathcal{P}_{\mu,r})$, $N_r(\mathcal{P}_{\mu,r} \setminus \mathcal{W}_{\mu,r})$ and $N_r^{\text{adj}}(\cdot)$, whereas any two of them do not seem to suffice. The rest is a straightforward computation similar to the one in Corollary 13.

Dynamic MAB. Let us consider the Dynamic MAB problem from Slivkins and Upfal (2008). Here for each arm y the stochastic process Γ_y is a random walk with step $\pm\sigma_y$. To ensure that the random walk stays within the interval $[0,1]$, we assume reflecting boundaries. Formally, we assume that $1/\sigma_y \in \mathbb{N}$, and once a boundary is reached, the next step is deterministically in the opposite direction.[12]

According to a well-known fact about random walks,[13]

$$\Pr\left[|\mu(t,y) - \mu(t',y)| \leq O(\sigma_y\,|t-t'|^{1/2}\log T_0)\right] \geq 1 - T_0^{-3} \quad \text{if } |t-t'| \leq T_0. \tag{26}$$

We use contextual zooming with period T_0, but we parameterize it by a different similarity space $(\mathcal{P}, \mathcal{D}_{T_0})$ that we define according to (26). Namely, we set

$$\mathcal{D}_{T_0}((t,y),\,(t',y')) = \min(1,\ \sigma_y\,|t-t'|^{1/2}\log T_0 + \mathbf{1}_{\{y\neq y'\}}). \tag{27}$$

The following corollary is proved using the same technique as Corollary 15:

Corollary 16 *Consider the Dynamic MAB problem with k arms and volatilities $\sigma_y \equiv \sigma$. Let $\texttt{ALG}_{T_0}$ denote the contextual zooming algorithm with period T_0 which is parameterized by the similarity space $(\mathcal{P}, \mathcal{D}_{T_0})$. Then $\texttt{ALG}_{T_0}$ satisfies $\mathbb{E}_\Gamma[\hat{R}(T)] = O(k\,\sigma\,\log^2 T_0)$, whenever $T \geq T_0 \geq \frac{1}{\sigma}\log\frac{1}{\sigma}$.*

6.3 Sleeping Bandits

The *sleeping bandits* problem Kleinberg et al. (2008a) is an extension of MAB where in each round some arms can be "asleep", i.e., not available in this round. One of the main results in Kleinberg et al. (2008a) is on sleeping bandits with stochastic payoffs. We recover this result using contextual zooming.

We model sleeping bandits as contextual MAB problem where each context arrival x_t corresponds to the set of arms that are "awake" in this round. More precisely, for every subset $S \subset Y$ of arms there is a distinct context x_S, and $\mathcal{P} = \{(x_S, y) : y \in S \subset Y\}$. is the set of feasible context-arm pairs. The similarity distance is simply $\mathcal{D}((x,y),\,(x',y')) = \mathbf{1}_{\{y\neq y'\}}$. Note that the Lipschitz condition (1) is satisfied.

For this setting, contextual zooming essentially reduces to the "highest awake index" algorithm in Kleinberg et al. (2008a). In fact, we can re-derive the result Kleinberg et al. (2008a) on sleeping MAB with stochastic payoffs as an easy corollary of Theorem 1.

Corollary 17 *Consider the sleeping MAB problem with stochastic payoffs. Order the arms so that their expected payoffs are $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_n$, where n is the number of arms. Let $\Delta_i = \mu_{i+1} - \mu_i$. Then*

$$R(T) \leq \inf_{r>0}\left(rT + \sum_{i:\,\Delta_i > r} \frac{O(\log T)}{\Delta_i}\right).$$

12. Slivkins and Upfal (2008) has a slightly more general setup which does not require $1/\sigma_y \in \mathbb{N}$.
13. For example, this follows as a simple application of Azuma-Hoeffding inequality.

Proof The r-zooming number $N_r(\mathcal{P}_{\mu,r})$ is equal to the number of distinct *arms* in $\mathcal{P}_{\mu,r}$, i.e., the number of arms $i \in Y$ such that $\Delta(x,i) \leq 12r$ for some context x. Note that for a given arm i, the quantity $\Delta(x,i)$ is minimized when the set of awake arms is $S = \{i, i+1\}$. Therefore, $N_r(\mathcal{P}_{\mu,r})$ is equal to the number of arms $i \in Y$ such that $\Delta_i \leq 12r$. It follows that

$$\begin{aligned}
N_{r>r_0}(\mathcal{P}_{\mu,r}) &= \textstyle\sum_{i=1}^{n} \mathbf{1}_{\{\Delta_i \leq 12r\}}. \\
\textstyle\sum_{r>r_0} \frac{1}{r} N_{r>r_0}(\mathcal{P}_{\mu,r}) &= \textstyle\sum_{r>r_0} \sum_{i=1}^{n} \frac{1}{r} \mathbf{1}_{\{\Delta_i \leq 12r\}} \\
&= \textstyle\sum_{i=1}^{n} \sum_{r>r_0} \frac{1}{r} \mathbf{1}_{\{\Delta_i \leq 12r\}} \\
&= \textstyle\sum_{i:\, \Delta_i > r_0} O(\frac{1}{\Delta_i}). \\
R(T) &\leq \inf_{r_0>0} \left(r_0\, T + O(\log T) \textstyle\sum_{r>r_0} \frac{1}{r} N_r(\mathcal{P}_{\mu,r}) \right) \\
&\leq \inf_{r_0>0} \left(r_0\, T + O(\log T) \textstyle\sum_{i:\, \Delta_i > r_0} O(\frac{1}{\Delta_i}) \right),
\end{aligned}$$

as required. (In the above equations, $\sum_{r>r_0}$ denotes the sum over all $r = 2^{-j} > r_0$ such that $j \in \mathbb{N}$.) ■

Moreover, the contextual MAB problem extends the sleeping bandits setting by incorporating similarity information on arms. The contextual zooming algorithm (and its analysis) applies, and is geared to exploit this additional similarity information.

6.4 Bandit Learning-to-Rank

Following a preliminary publication of this paper on `arxiv.org`, contextual zooming has been applied in Slivkins et al. (2013) to bandit learning-to-rank. Interestingly, the "contexts" studied in Slivkins et al. (2013) are very different from what we considered so far.

The basic setting, motivated by web search, was introduced in Radlinski et al. (2008). In each round a new user arrives. The algorithm selects a ranked list of k documents and presents it to the user who clicks on at most one document, namely on the first document that (s)he finds relevant. A user is specified by a binary vector over documents. The goal is to minimize *abandonment*: the number of rounds with no clicks.

Slivkins et al. (2013) study an extension in which metric similarity information is available. They consider a version with *stochastic payoffs*: in each round, the user vector is an independent sample from a fixed distribution, and assume a Lipschitz-style condition that connects expected clicks with the metric space. They run a separate bandit algorithm (e.g., contextual zooming) for each of the k "slots" in the ranking. Without loss of generality, in each round the documents are selected sequentially, in the top-down order. Since a document in slot i is clicked in a given round only if all higher ranked documents are not relevant, they treat the set of documents in the higher slots as a *context* for the i-th algorithm. The Lipschitz-style condition on expected clicks suffices to guarantee the corresponding Lipschitz-style condition on contexts.

7. Bandits with Stochastically Evolving Payoffs: Missing Proofs

We prove Corollary 15 and Corollary 16 which address the performance of contextual zooming for the stochastically evolving payoffs. In each corollary we bound from above the average dynamic regret $\hat{R}(T)$ of contextual zooming with period T_0, for any $T \geq T_0$. Since $\hat{R}(T) \leq 2\hat{R}(T_0)$, it suffices to bound $\hat{R}(T_0)$, which is the same as $\hat{R}(T_0)$ for (non-periodic) contextual zooming. Therefore, we can focus on analyzing the non-periodic algorithm.

We start with two simple auxiliary claims.

Claim 18 *Consider the contextual MAB problem with a product similarity space. Let $\Delta(x,y) \triangleq \mu^*(x) - \mu(x,y)$ be the "badness" of point (x,y) in the similarity space. Then*

$$|\Delta(x,y) - \Delta(x',y)| \leq 2\,\mathcal{D}_X(x,x') \qquad \forall x,x' \in X,\ y \in Y. \tag{28}$$

Proof First we show that the benchmark payoff $\mu(\cdot)$ satisfies a Lipschitz condition:

$$|\mu^*(x) - \mu^*(x')| \leq \mathcal{D}_{\mathrm{X}}(x,x') \qquad \forall x,x' \in X. \tag{29}$$

Indeed, it holds that $\mu^*(x) = \mu(x,y)$ and $\mu^*(x') = \mu(x,y')$ for some arms $y, y' \in Y$. Then

$$\mu^*(x) = \mu(x,y) \geq \mu(x,y') \geq \mu(x',y') - \mathcal{D}_{\mathrm{X}}(x,x') = \mu^*(x') - \mathcal{D}_{\mathrm{X}}(x,x'),$$

and likewise for the other direction. Now,

$$|\Delta(x,y) - \Delta(x',y)| \leq |\mu^*(x) - \mu^*(x')| + |\mu(x,y) - \mu(x',y)| \leq 2\,\mathcal{D}_{\mathrm{X}}(x,x').$$

■

Claim 19 *Let $Z_1, \ldots, Z_k$ be independent random variables distributed uniformly at random on $[0,1]$. Let $Z^* = \max_i Z_i$. Fix $r > 0$ and let $S = \{i : Z^* > Z_i \geq Z^* - r\}$. Then $\mathbb{E}[\,|S|\,] = kr$.*

This is a textbook result; we provide a proof for the sake of completeness.

Proof Conditional on Z^*, it holds that

$$\begin{aligned}
\mathbb{E}[\,|S|\,] &= \mathbb{E}\left[\textstyle\sum_i \mathbf{1}_{\{Z_i \in S\}}\right] = k\,\Pr[Z_i \in S] \\
&= k\,\Pr[Z_i \in S \,|\, Z_i < Z^*] \times \Pr[Z_i < Z^*] \\
&= k\,\tfrac{r}{Z^*}\,\tfrac{k-1}{k} = (k-1)r/Z^*.
\end{aligned}$$

Integrating over Z^*, and letting $F(z) \triangleq \Pr[Z^* \leq z] = z^k$, we obtain that

$$\begin{aligned}
\mathbb{E}[\,\tfrac{1}{Z^*}\,] &= \textstyle\int_0^1 \frac{1}{z}\,F'(z)dz = \frac{k}{k-1} \\
\mathbb{E}[\,|S|\,] &= (k-1)r\;\mathbb{E}[\,\tfrac{1}{Z^*}\,] = kr.
\end{aligned}$$

■

Proof of Corollary 15 It suffices to bound $\hat{R}(T_0)$ for (non-periodic) contextual zooming.

Let $\mathcal{D}_{\mathrm{X}}(t,t') \triangleq \sigma|t-t'|$ be the context distance implicit in the temporal constraint (21). For each $r>0$, pick a number T_r such that $\mathcal{D}_{\mathrm{X}}(t,t') \leq r \iff |t-t'| \leq T_r$. Clearly, $T_r \triangleq \frac{r}{\sigma}$.

The crux is to bound the adjusted r-zooming number, call it N_r, namely to show (25). For the sake of convenience, let us restate it here (and let us use the notation T_r):

$$\mathbb{E}_\Gamma[N_r] = O(kr)\lceil \tfrac{T}{T_r} \rceil \text{ and } \left(T_r < 1/r^2 \Rightarrow N_r = 0\right). \tag{30}$$

Recall that $N_r = N^{\mathrm{adj}}(\mathcal{P}_{\mu,r} \setminus \mathcal{W}_{\mu,r})$, where $\mathcal{W}_{\mu,r}$ is the set of all r-winners (see Section 4.4 for the definition). Fix $r \in (0,1]$ and let S be some r-packing of $\mathcal{P}_{\mu,r} \setminus \mathcal{W}_{\mu,r}$. Partition the time into $\lceil \frac{T}{T_r} \rceil$ intervals of duration T_r. Fix one such interval I. Let $S_I \triangleq \{(t,y) \in S : t \in I\}$, the set of points in S that correspond to times in I. Recall the notation $\Delta(x,y) \triangleq \mu^*(x) - \mu(x,y)$ and let

$$Y_I \triangleq \{y \in Y : \Delta(t_I, y) \leq 14\, r\}, \text{ where } t_I \triangleq \min(I). \tag{31}$$

All quantities in (31) refer to a fixed time t_I, which will allow us to use the uniform marginals property.

Note that Y_I contains at least one arm, namely the best arm $y^*(t_I)$. We claim that

$$|S_I| \leq 2\,|Y_I \setminus \{y^*(t_I)\}|. \tag{32}$$

Fix arm y. First, $\mathcal{D}_{\mathrm{X}}(t,t') \leq r$ for any $t,t' \in I$, so there exists at most one $t \in I$ such that $(t,y) \in S$. Second, suppose such t exists. Since $S \subset \mathcal{P}_{\mu,r}$, it follows that $\Delta(t,y) \leq 12\, r$. By Claim 18 it holds that

$$\Delta(t_I, y) \leq \Delta(t,y) + 2\,\mathcal{D}_{\mathrm{X}}(t,t') \leq 14\, r.$$

So $y \in Y_I$. It follows that $|S_I| \leq |Y_I|$.

To obtain (32), we show that $S_I = 0$ whenever $|Y_I| = 1$. Indeed, suppose $Y_I = \{y\}$ is a singleton set, and $|S_I| > 0$. Then $S_I = \{(t,y)\}$ for some $t \in I$. We will show that (t,y) is an r-winner, contradicting the definition of S. For any arm $y' \neq y$ and any time t' such that $\mathcal{D}_{\mathrm{X}}(t,t') \leq 2r$ it holds that

$$\begin{aligned}
\mu(t_I, y) &= \mu^*(t_I) > \mu(t_I, y') + 14r \\
\mu(t', y) &\geq \mu(t_I, y) - \mathcal{D}_{\mathrm{X}}(t', t_I) \geq \mu(t_I, y) - 3r \\
&> \mu(t_I, y') + 11r \\
&\geq \mu(t', y') - \mathcal{D}_{\mathrm{X}}(t', t_I) + 11r \\
&\geq \mu(t', y') + 8r.
\end{aligned}$$

and so $\mu(t',y) = \mu^*(t')$. Thus, (t,y) is an r-winner as claimed. This completes the proof of (32).

Now using (32) and Claim 19 we obtain that

$$\begin{aligned}
\mathbb{E}_\Gamma[\,|S_I|\,] &\leq 2\,\mathbb{E}_\Gamma[\,|Y_I \setminus \{y^*(t_I)\}|\,] \leq O(kr) \\
\mathbb{E}_\Gamma[\,|S|\,] &\leq \lceil \tfrac{T}{T_r} \rceil\, \mathbb{E}[\,|S_I|\,] \leq O(kr)\, \lceil \tfrac{T}{T_r} \rceil.
\end{aligned}$$

Taking the max over all possible S, we obtain $\mathbb{E}_\Gamma[\mathcal{P}_{\mu,r} \setminus \mathcal{W}_{\mu,r}] \leq O(kr)\ \lceil \frac{T}{T_r} \rceil$. To complete the proof of (30), we note that S cannot be r-consistent unless $|I| \geq 1/r^2$.

Now that we have (30), the rest is a simple computation. We use Theorem 6, namely we take (5) with $r_0 \to 0$, plug in (30), and recall that $T_r \geq 1/r^2 \iff r \geq \sigma^{1/3}$.

$$\begin{aligned} R(T) &\leq \textstyle\sum_{r=2^i \geq \sigma^{1/3}} \frac{1}{r}\ N_r\ O(\log T) \\ \mathbb{E}_\Gamma[R(T)] &\leq \textstyle\sum_{r=2^i \geq \sigma^{1/3}} O(k \log T)(\frac{T\sigma}{r} + 1) \\ &\leq O(k \log T)(T\sigma^{2/3} + \log \tfrac{1}{\sigma}). \end{aligned}$$

It follows that $\mathbb{E}_\Gamma[\hat{R}(T)] \leq O(k\,\sigma^{2/3} \log T)$ for any $T \geq \sigma^{-2/3} \log \frac{1}{\sigma}$. ■

Proof of Corollary 16 It suffices to bound $\hat{R}(T_0)$ for (non-periodic) contextual zooming.

Recall that expected payoffs satisfy the temporal constraint (26). Consider the high-probability event that

$$|\mu(t,y) - \mu(t',y)| \leq \sigma\,|t-t'|^{1/2} \log T_0 \qquad \forall t,t' \in [1,T_0],\ y \in Y. \tag{33}$$

Since expected regret due to the failure of (33) is negligible, from here on we will assume that (33) holds deterministically.

Let $\mathcal{D}_X(t,t') \triangleq \sigma\,|t-t'|^{1/2} \log T_0$ be the distance on contexts implicit in (33). For each $r > 0$, define $T_r \triangleq (\frac{r}{\sigma \log T_0})^2$. Then (30) follows exactly as in the proof of Corollary 15. We use Theorem 6 similarly: we take (5) with $r_0 \to 0$, plug in (30), and note that $T_r \geq 1/r^2 \iff r \geq (\sigma \log T_0)^{1/2}$. We obtain

$$\begin{aligned} \mathbb{E}_\Gamma[R(T_0)] &\leq \sum_{r=2^i \geq (\sigma \log T_0)^{1/2}} O(k \log T_0)(\tfrac{T_0}{T_r} + 1) \\ &\leq O(k \log^2 T_0)(T_0\,\sigma + \log \tfrac{1}{\sigma}). \end{aligned}$$

It follows that $\mathbb{E}_\Gamma[\hat{R}(T)] \leq O(k\,\sigma \log^2 T_0)$ as long as $T_0 \geq \frac{1}{\sigma} \log \frac{1}{\sigma}$. ■

8. Contextual Bandits with Adversarial Payoffs

In this section we consider the adversarial setting. We provide an algorithm which maintains an adaptive partition of the context space and thus takes advantage of "benign" context arrivals. It is in fact a *meta-algorithm*: given a bandit algorithm `Bandit`, we present a contextual bandit algorithm, called `ContextualBandit`, which calls `Bandit` as a subroutine.

8.1 Our Setting

Recall that in each round t, the context $x_t \in X$ is revealed, then the algorithm picks an arm $y_t \in Y$ and observes the payoff $\pi_t \in [0,1]$. Here X is the context set, and Y is the arms set. In this section, all context-arms pairs are feasible: $\mathcal{P} = X \times Y$.

Adversarial payoffs are defined as follows. For each round t, there is a payoff function $\hat{\pi}_t : X \times Y \to [0,1]$ such that $\pi_t = \hat{\pi}_t(x_t, y_t)$. The payoff function $\hat{\pi}_t$ is sampled independently

from a time-specific distribution Π_t over payoff functions. Distributions Π_t are fixed by the adversary in advance, before the first round, and not revealed to the algorithm. Denote $\mu_t(x,y) \triangleq \mathbb{E}[\Pi_t(x,y)]$.

Following Hazan and Megiddo (2007), we generalize the notion of regret for context-free adversarial MAB to contextual MAB. The context-specific best arm is

$$y^*(x) \in \text{argmax}_{y \in Y} \sum_{t=1}^{T} \mu_t(x,y), \tag{34}$$

where the ties are broken in an arbitrary but fixed way. We define *adversarial contextual regret* as

$$R(T) \triangleq \sum_{t=1}^{T} \mu_t(x_t,y_t) - \mu_t^*(x_t), \quad \text{where} \quad \mu_t^*(x) \triangleq \mu_t(x,y^*(x)). \tag{35}$$

Similarity information is given to an algorithm as a pair of metric spaces: a metric space $(X,\mathcal{D}_X)$ on contexts (the *context space*) and a metric space $(Y,\mathcal{D}_Y)$ on arms (the *arms space*), which form the product similarity space $(X\times Y, \mathcal{D}_X+\mathcal{D}_Y)$. We assume that for each round t functions μ_t and μ_t^* are Lipschitz on $(X\times Y,\mathcal{D}_X+\mathcal{D}_Y)$ and $(X,\mathcal{D}_X)$, respectively, both with Lipschitz constant 1 (see Footnote 1). We assume that the context space is compact, in order to ensure that the max in (34) is attained by some $y \in Y$. Without loss of generality, $\texttt{diameter}(X,\mathcal{D}_X) \leq 1$.

Formally, a problem instance consists of metric spaces $(X,\mathcal{D}_X)$ and $(Y,\mathcal{D}_Y)$, the sequence of context arrivals (denoted $x_{(1..T)}$), and a sequence of distributions $(\Pi_t)_{t\leq T}$. Note that for a fixed distribution $\Pi_t = \Pi$, this setting reduces to the stochastic setting, as defined in Introduction. For the fixed context case ($x_t = x$ for all t) this setting reduces to the (context-free) MAB problem with a randomized oblivious adversary.

8.2 Our Results

Our algorithm is parameterized by a regret guarantee for `Bandit` for the fixed context case, namely an upper bound on the convergence time.[14] For a more concrete theorem statement we will assume that the convergence time of `Bandit` is at most $T_0(r) \triangleq c_Y\, r^{-(2+d_Y)}\, \log(\frac{1}{r})$ for some constants c_Y and d_Y that are known to the algorithm. In particular, an algorithm in Kleinberg (2004) achieves this guarantee if d_Y is the c-covering dimension of the arms space and $c_Y = O(c^{2+d_Y})$.

This is a flexible formulation that can leverage prior work on adversarial bandits. For instance, if $Y \subset \mathbb{R}^d$ and for each fixed context $x \in X$ distributions Π_t randomize over linear functions $\hat{\pi}_t(x,\cdot) : Y \to \mathbb{R}$, then one could take `Bandit` from the line of work on adversarial bandits with linear payoffs. In particular, there exist algorithms with $d_Y = 0$ and $c_Y = \text{poly}(d)$ (Dani et al., 2007; Abernethy et al., 2008; Bubeck et al., 2012). Likewise, for convex payoffs there exist algorithms with $d_Y = 2$ and $c_Y = O(d)$ (Flaxman et al., 2005). For a bounded number of arms, algorithm EXP3 (Auer et al., 2002b) achieves $d_Y = 0$ and $c_Y = O(\sqrt{|Y|})$.

From here on, the context space $(X,\mathcal{D}_X)$ will be only metric space considered; balls and other notions will refer to the context space only.

14. The r-convergence time $T_0(r)$ is the smallest T_0 such that regret is $R(T) \leq rT$ for each $T \geq T_0$.

To quantify the "goodness" of context arrivals, our guarantees are in terms of the covering dimension of $x_{(1..T)}$ rather than that of the entire context space. (This is the improvement over the guarantee (3) for the uniform algorithm.) In fact, use a more refined notion which allows to disregard a limited number of "outliers" in $x_{(1..T)}$.

Definition 20 *Given a metric space and a multi-set S, the* (r,k)-covering number *of S is the r-covering number of the set $\{x \in S : |B(x,r) \cap S| \geq k\}$.*[15] *Given a constant c and a function $k : (0,1) \to \mathbb{N}$, the* ***relaxed covering dimension*** *of S with slack $k(\cdot)$ is the smallest $d > 0$ such that the $(r, k(r))$-covering number of S is at most $c\, r^{-d}$ for all $r > 0$.*

Our result is stated as follows:

Theorem 21 *Consider the contextual MAB problem with adversarial payoffs, and let* `Bandit` *be a bandit algorithm. Assume that the problem instance belongs to some class of problem instances such that for the fixed-context case, convergence time of* `Bandit` *is at most $T_0(r) \triangleq c_Y\, r^{-(2+d_Y)} \log(\frac{1}{r})$ for some constants c_Y and d_Y that are known to the algorithm. Then* `ContextualBandit` *achieves adversarial contextual regret $R(\cdot)$ such that for any time T and any constant $c_X > 0$ it holds that*

$$R(T) \leq O(c_{\text{DBL}}^2\, (c_X\, c_Y)^{1/(2+d_X+d_Y)})\ T^{1-1/(2+d_X+d_Y)}(\log T), \tag{36}$$

where d_X is the relaxed covering dimension of $x_{(1..T)}$ with multiplier c_X and slack $T_0(\cdot)$, and c_{DBL} is the doubling constant of $x_{(1..T)}$.

Remarks. For a version of (36) that is stated in terms of the "raw" (r, k_r)-covering numbers of $x_{(1..T)}$, see (38) in the analysis (page 2563).

8.3 Our Algorithm

The contextual bandit algorithm `ContextualBandit` is parameterized by a (context-free) bandit algorithm `Bandit`, which it uses as a subroutine, and a function $T_0(\cdot) : (0,1) \to \mathbb{N}$.

The algorithm maintains a finite collection $\mathcal{A}$ of balls, called *active balls*. Initially there is one active ball of radius 1. Ball B stays active once it is *activated*. Then a fresh instance $\texttt{ALG}_B$ of `Bandit` is created, whose set of "arms" is Y. $\texttt{ALG}_B$ can be parameterized by the time horizon $T_0(r)$, where r is the radius of B.

The algorithm proceeds as follows. In each round t the algorithm selects one active ball $B \in \mathcal{A}$ such that $x_t \in B$, calls $\texttt{ALG}_B$ to select an arm $y \in Y$ to be played, and reports the payoff π_t back to $\texttt{ALG}_B$. A given ball can be selected at most $T_0(r)$ times, after which it is called *full*. B is called *relevant* in round t if it contains x_t and is not full. The algorithm selects a relevant ball (breaking ties arbitrarily) if such ball exists. Otherwise, a new ball B' is activated and selected. Specifically, let B be the smallest-radius active ball containing x_t. Then $B' = B(x_t, \frac{r}{2})$, where r is the radius of B. B is then called the *parent* of B'. See Algorithm 2 for the pseudocode.

15. By abuse of notation, here $|B(x,r) \cap S|$ denotes the number of points $x \in S$, with multiplicities, that lie in $B(x,r)$.

Algorithm 2 Algorithm `ContextualBandit`.

Input:
  Context space $(X, \mathcal{D}_X)$ of diameter ≤ 1, set Y of arms.
  Bandit algorithm `Bandit` and a function $T_0(\cdot) : (0,1) \to \mathbb{N}$.
Data structures:
  A collection $\mathcal{A}$ of "active balls" in $(X, \mathcal{D}_X)$.
  $\forall B \in \mathcal{A}$: counter n_B, instance $\texttt{ALG}_B$ of `Bandit` on arms Y.
Initialization: $B \leftarrow B(x, 1)$; // center $x \in X$ is arbitrary
  $\mathcal{A} \leftarrow \{B\}$; $n_B \leftarrow 0$; initiate $\texttt{ALG}_B$.
  $\mathcal{A}^* \leftarrow \mathcal{A}$ // active balls that are not full
Main loop: for each round t
  Input context x_t.
  $\texttt{relevant} \leftarrow \{B \in \mathcal{A}^* : x_t \in B\}$.
  if $\texttt{relevant} \neq \emptyset$ **then**
    $B \leftarrow$ any $B \in \texttt{relevant}$.
  else // activate a new ball:
    $r \leftarrow \min_{B \in \mathcal{A}:\, x_t \in B} r_B$.
    $B \leftarrow B(x_t, r/2)$. // new ball to be added
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{B\}$; $\mathcal{A}^* \leftarrow \mathcal{A}^* \cup \{B\}$; $n_B \leftarrow 0$; initiate $\texttt{ALG}_B$.
  $y \leftarrow$ next arm selected by $\texttt{ALG}_B$.
  Play arm y, observe payoff π, report π to $\texttt{ALG}_B$.
  $n_B \leftarrow n_B + 1$.
  if $n_B = T_0(\texttt{radius}(B))$ **then** $\mathcal{A}^* \leftarrow \mathcal{A}^* \setminus \{B\}$. // ball B is full

8.4 Analysis: Proof of Theorem 21

First let us argue that algorithm `ContextualBandit` is well-defined. Specifically, we need to show that after the activation rule is called, there exists an active non-full ball containing x_t. Suppose not. Then the ball $B' = B(x_t, \frac{r}{2})$ activated by the activation rule must be full. In particular, B' must have been active before the activation rule was called, which contradicts the minimality in the choice of r. Claim proved.

We continue by listing several basic claims about the algorithm.

Claim 22 *The algorithm satisfies the following basic properties:*

(a) (Correctness) In each round t, exactly one active ball is selected.

(b) Each active ball of radius r is selected at most $T_0(r)$ times.

(c) (Separation) For any two active balls $B(x, r)$ and $B(x', r)$ we have $\mathcal{D}_X(x, x') > r$.

(d) Each active ball has at most c_{DBL}^2 children, where c_{DBL} is the doubling constant of $x_{(1..T)}$.

Proof Part (a) is immediate from the algorithm's specification. For (b), simply note that by the algorithms' specification a ball is selected only when it is not full.

To prove (c), suppose that $\mathcal{D}_X(x, x') \leq r$ and suppose $B(x', r)$ is activated in some round t while $B(x, r)$ is active. Then $B(x', r)$ was activated as a child of some ball B^* of radius $2r$. On the other hand, $x' = x_t \in B(x, r)$, so $B(x, r)$ must have been full in round t (else no ball would have been activated), and consequently the radius of B^* is at most r. Contradiction.

For (d), consider the children of a given active ball $B(x,r)$. Note that by the activation rule the centers of these children are points in $x_{(1..T)} \cap B(x,r)$, and by the separation property any two of these points lie within distance $> \frac{r}{2}$ from one another. By the doubling property, there can be at most c_{DBL}^2 such points. ■

Let us fix the time horizon T, and let $R(T)$ denote the contextual regret of `ContextualBandit`. Partition $R(T)$ into the contributions of active balls as follows. Let $\mathcal{B}$ be the set of all balls that are active after round T. For each $B \in \mathcal{B}$, let S_B be the set of all rounds t when B has been selected. Then

$$R(T) = \textstyle\sum_{B\in\mathcal{B}} R_B(T), \quad \text{where} \quad R_B(T) \triangleq \sum_{t\in S_B} \mu_t^*(x_t) - \mu_t(x_t,\, y_t).$$

Claim 23 *For each ball $B = B(x,r) \in \mathcal{B}$, we have $R_B \leq 3\, r\, T_0(r)$.*

Proof By the Lipschitz conditions on μ_t and μ_t^*, for each round $t \in S_B$ it is the case that

$$\mu_t^*(x_t) \leq r + \mu_t^*(x) = r + \mu_t(x, y^*(x)) \leq 2rn + \mu_t(x_t,\, y^*(x)).$$

The t-round regret of `Bandit` is at most $R_0(t) \triangleq t\, T_0^{-1}(t)$. Therefore, letting $n = |S_B|$ be the number of times algorithm $\texttt{ALG}_B$ has been invoked, we have that

$$R_0(n) + \textstyle\sum_{t\in S_B} \mu_t(x_t,\, y_t) \geq \sum_{t\in S_B} \mu_t(x_t, y^*(x)) \geq \sum_{t\in S_B} \mu_t^*(x_t) - 2rn.$$

Therefore $R_B(T) \leq R_0(n) + 2rn$. Recall that by Claim 22(b) we have $n \leq T_0(r)$. Thus, by definition of convergence time $R_0(n) \leq R_0(T_0(r)) \leq r\, T_0(r)$, and therefore $R_B(T) \leq 3\, r\, T_0(r)$. ■

Let $\mathcal{F}_r$ be the collection of all full balls of radius r. Let us bound $|\mathcal{F}_r|$ in terms the (r,k)-covering number of $x_{(1..T)}$ in the context space, which we denote $N(r,k)$.

Claim 24 *There are at most $N(r,\, T_0(r))$ full balls of radius r.*

Proof Fix r and let $k = T_0(r)$. Let us say that a point $x \in x_{(1..T)}$ is *heavy* if $B(x,r)$ contains at least k points of $x_{(1..T)}$, counting multiplicities. Clearly, $B(x,r)$ is full only if its center is heavy. By definition of the (r,k)-covering number, there exists a family $\mathcal{S}$ of $N(r,k)$ sets of diameter $\leq r$ that cover all heavy points in $x_{(1..T)}$. For each full ball $B = B(x,r)$, let S_B be some set in $\mathcal{S}$ that contains x. By Claim 22(c), the sets S_B, $B \in \mathcal{F}_r$ are all distinct. Thus, $|\mathcal{F}_r| \leq |\mathcal{S}| \leq N(r,k)$. ■

Let $\mathcal{B}_r$ be the set of all balls of radius r that are active after round T. By the algorithm's specification, each ball in $\mathcal{F}_r$ has been selected $T_0(r)$ times, so $|\mathcal{F}_r| \leq T/T_0(r)$. Then using Claim 22(b) and Claim 24, we have

$$\begin{aligned} |\mathcal{B}_{r/2}| &\leq c_{\text{DBL}}^2\, |\mathcal{F}_r| \leq c_{\text{DBL}}^2\ \min(T/T_0(r),\ N(r,\, T_0(r))) \\ \textstyle\sum_{B\in\mathcal{B}_{r/2}} R_B &\leq O(r)\, T_0(r)\, |\mathcal{B}_{r/2}| \leq O(c_{\text{DBL}}^2)\ \min(rT,\ r\, T_0(r)\, N(r, T_0(r))). \end{aligned} \tag{37}$$

Trivially, for any full ball of radius r we have $T_0(r) \leq T$. Thus, summing (37) over all such r, we obtain

$$R(T) \leq O(c_{\text{DBL}}^2) \sum_{r=2^{-i}:\, i\in\mathbb{N} \text{ and } T_0(r)\leq T} \min(rT,\ r\,T_0(r)\,N(r,T_0(r))). \tag{38}$$

Note that (38) makes no assumptions on $N(r,T_0(r))$. Now, plugging in $T_0(r) = c_{\text{Y}}\, r^{-(2+d_{\text{Y}})}$ and $N(r,T_0(r)) \leq c_{\text{X}}\, r^{-d_{\text{X}}}$ into (38) and optimizing it for r it is easy to derive the desired bound (36).

9. Conclusions

We consider a general setting for contextual bandit problems where the algorithm is given information on similarity between the context-arm pairs. The similarity information is modeled as a metric space with respect to which expected payoffs are Lipschitz-continuous. Our key contribution is an algorithm which maintains a partition of the metric space and adaptively refines this partition over time. Due to this "adaptive partition" technique, one can take advantage of "benign" problem instances without sacrificing the worst-case performance; here "benign-ness" refers to both expected payoffs and context arrivals. We essentially resolve the setting where expected payoff from every given context-arm pair either does not change over time, or changes slowly. In particular, we obtain nearly matching lower bounds (for time-invariant expected payoffs and for an important special case of slow change).

We also consider the setting of adversarial payoffs. For this setting, we design a different algorithm that maintains a partition of contexts and adaptively refines it so as to take advantage of "benign" context arrivals (but not "benign" expected payoffs), without sacrificing the worst-case performance. Our algorithm can work with, essentially, any given off-the-shelf algorithm for standard (non-contextual) bandits, the choice of which can then be tailored to the setting at hand.

The main open questions concern relaxing the requirements on the quality of similarity information that are needed for the provable guarantees. First, it would be desirable to obtain similar results under weaker versions of the Lipschitz condition. Prior work (Kleinberg et al., 2008b; Bubeck et al., 2011a) obtained several such results for the non-contextual version of the problem, mainly because their main results do not require the full power of the Lipschitz condition. However, the analysis in this paper appears to make a heavier use of the Lipschitz condition; it is not clear whether a meaningful relaxation would suffice. Second, in some settings the available similarity information might not include any numeric upper bounds on the difference in expected payoffs; e.g., it could be given as a tree-based taxonomy on context-arm pairs, without any explicit numbers. Yet, one wants to recover the same provable guarantees *as if* the numerical information were explicitly given. For the non-contextual version, this direction has been explored in (Bubeck et al., 2011b; Slivkins, 2011).[16]

Another open question concerns our results for adversarial payoffs. Here it is desirable to extend our "adaptive partitions" technique to also take advantage of "benign" expected

16. (Bubeck et al., 2011b; Slivkins, 2011) have been published after the preliminary publication of this paper on `arxiv.org`.

payoffs (in addition to "benign" context arrivals). However, to the best of our knowledge such results are not even known for the non-contextual version of the problem.

Acknowledgments

The author is grateful to Ittai Abraham, Bobby Kleinberg and Eli Upfal for many conversations about multi-armed bandits, and to Sebastien Bubeck for help with the manuscript. Also, comments from anonymous COLT reviewers and JMLR referees have been tremendously useful in improving the presentation.

Appendix A. The KL-divergence Technique, Encapsulated

To analyze the lower-bounding construction in Section 5, we use an extension of the KL-divergence technique from Auer et al. (2002b), which is implicit in Kleinberg (2004) and encapsulated as a stand-alone theorem in Kleinberg et al. (2013). To make the paper self-contained, we state the theorem from Kleinberg et al. (2013), along with the relevant definitions. The remainder of this section is copied from Kleinberg et al. (2013), with minor modifications.

Consider a very general MAB setting where the algorithm is given a strategy set X and a collection $\mathcal{F}$ of feasible payoff functions; we call it the *feasible MAB problem* on $(X, \mathcal{F})$. For example, $\mathcal{F}$ can consist of all functions $\mu : X \to [0, 1]$ that are Lipschitz with respect to a given metric space. The lower bound relies on the existence of a collection of subsets of $\mathcal{F}$ with certain properties, as defined below. These subsets correspond to children of a given tree node in the ball-tree

Definition 25 *Let X be the strategy set and $\mathcal{F}$ be the set of all feasible payoff functions. An* (ϵ, k)-ensemble *is a collection of subsets $\mathcal{F}_1, \ldots, \mathcal{F}_k \subset \mathcal{F}$ such that there exist mutually disjoint subsets $S_1, \ldots, S_k \subset X$ and a number $\mu_0 \in [\frac{1}{3}, \frac{2}{3}]$ which satisfy the following. Let $S = \cup_{i=1}^{k} S_i$. Then*

- *on $X \setminus S$, any two functions in $\cup_i \mathcal{F}_i$ coincide, and are bounded from above by μ_0.*
- *for each i and each function $\mu \in \mathcal{F}_i$ it holds that $\mu = \mu_0$ on $S \setminus S_i$ and $\sup(\mu_i, S_i) = \mu_0 + \epsilon$.*

Assume the payoff function μ lies in $\cup_i \mathcal{F}_i$. The idea is that an algorithm needs to play arms in S_i for at least $\Omega(\epsilon^{-2})$ rounds in order to determine whether $\mu \in \mathcal{F}_i$, and each such step incurs ϵ regret if $\mu \notin \mathcal{F}_i$. In our application, subsets $S_1, \ldots, S_k$ correspond to children $u_1, \ldots, u_k$ of a given tree node in the ball-tree, and each $\mathcal{F}_i$ consists of payoff functions induced by the ends in the subtree rooted at u_i.

Theorem 26 (Theorem 5.6 in Kleinberg et al. (2013)) *Consider the feasible MAB problem with 0-1 payoffs. Let $\mathcal{F}_1, \ldots, \mathcal{F}_k$ be an (ϵ, k)-ensemble, where $k \geq 2$ and $\epsilon \in (0, \frac{1}{12})$. Then for any $t \leq \frac{1}{32} k \epsilon^{-2}$ and any bandit algorithm there exist at least $k/2$ distinct i's such that the regret of this algorithm on any payoff function from $\mathcal{F}_i$ is at least $\frac{1}{60} \epsilon t$.*

In Auer et al. (2002b), the authors analyzed a special case of an (ϵ, k)-ensemble in which there are k arms $u_1, \ldots, u_k$, and each $\mathcal{F}_i$ consists of a single payoff function that assigns expected payoff $\frac{1}{2} + \epsilon$ to arm u_i, and $\frac{1}{2}$ to all other arms.

References

Jacob Abernethy, Elad Hazan, and Alexander Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *21th Conf. on Learning Theory (COLT)*, pages 263–274, 2008.

Rajeev Agrawal. The continuum-armed bandit problem. *SIAM J. Control and Optimization*, 33(6):1926–1951, 1995.

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *J. of Machine Learning Research (JMLR)*, 3:397–422, 2002. Preliminary version in *41st IEEE FOCS*, 2000.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002a. Preliminary version in *15th ICML*, 1998.

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002b. Preliminary version in *36th IEEE FOCS*, 1995.

Peter Auer, Ronald Ortner, and Csaba Szepesvári. Improved rates for the stochastic continuum-armed bandit problem. In *20th Conf. on Learning Theory (COLT)*, pages 454–468, 2007.

Baruch Awerbuch and Robert Kleinberg. Online linear optimization and adaptive routing. *J. of Computer and System Sciences*, 74(1):97–114, February 2008. Preliminary version in *36th ACM STOC*, 2004.

Jeffrey Banks and Rangarajan Sundaram. Denumerable-armed bandits. *Econometrica*, 60 (5):1071–1096, 1992.

Donald A. Berry, Robert W. Chen, Alan Zame, David C. Heath, and Larry A. Shepp. Bandit problems with infinitely many arms. *Annals of Statistics*, 25(5):2103–2116, 1997.

Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

Sébastien Bubeck and Rémi Munos. Open loop optimistic planning. In *23rd Conf. on Learning Theory (COLT)*, pages 477–489, 2010.

Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvari. Online optimization in X-armed bandits. *J. of Machine Learning Research (JMLR)*, 12:1587–1627, 2011a. Preliminary version in *NIPS 2008*.

Sébastien Bubeck, Gilles Stoltz, and Jia Yuan Yu. Lipschitz bandits without the Lipschitz constant. In *22nd Intl. Conf. on Algorithmic Learning Theory (ALT)*, pages 144–158, 2011b.

Sébastien Bubeck, Nicolò Cesa-Bianchi, and Sham M. Kakade. Towards minimax policies for online linear optimization with bandit feedback. In *25th Conf. on Learning Theory (COLT)*, 2012.

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge Univ. Press, 2006.

Wei Chu, Lihong Li, Lev Reyzin, and Robert E. Schapire. Contextual bandits with linear payoff functions. In *14th Intl. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2011.

Richard Cole and Lee-Ad Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *38th ACM Symp. on Theory of Computing (STOC)*, pages 574–583, 2006.

Varsha Dani, Thomas P. Hayes, and Sham Kakade. The price of bandit information for online optimization. In *20th Advances in Neural Information Processing Systems (NIPS)*, 2007.

Abraham Flaxman, Adam Kalai, and Brendan McMahan. Online convex optimization in the bandit setting: Gradient descent without a gradient. In *16th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 385–394, 2005.

John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-Armed Bandit Allocation Indices*. John Wiley & Sons, 2011.

Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low–distortion embeddings. In *44th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 534–543, 2003.

Elad Hazan and Satyen Kale. Better algorithms for benign bandits. In *20th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 38–47, 2009.

Elad Hazan and Nimrod Megiddo. Online learning with prior information. In *20th Conf. on Learning Theory (COLT)*, pages 499–513, 2007.

Juha Heinonen. *Lectures on Analysis on Metric Spaces*. Universitext. Springer-Verlag, New York, 2001.

Jon Kleinberg, Aleksandrs Slivkins, and Tom Wexler. Triangulation and embedding using small sets of beacons. In *45th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 444–453, 2004.

Jon Kleinberg, Aleksandrs Slivkins, and Tom Wexler. Triangulation and embedding using small sets of beacons. *J. of the ACM*, 56(6), September 2009. Subsumes conference papers in *IEEE FOCS 2004* Kleinberg et al. (2004) and *ACM-SIAM SODA 2005* Slivkins (2005).

Robert Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. In *18th Advances in Neural Information Processing Systems (NIPS)*, 2004.

Robert Kleinberg. *Online Decision Problems with Large Strategy Sets*. PhD thesis, MIT, 2005.

Robert Kleinberg and Aleksandrs Slivkins. Sharp dichotomies for regret minimization in metric spaces. In *21st ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2010.

Robert Kleinberg, Alexandru Niculescu-Mizil, and Yogeshwer Sharma. Regret bounds for sleeping experts and bandits. In *21st Conf. on Learning Theory (COLT)*, pages 425–436, 2008a.

Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *40th ACM Symp. on Theory of Computing (STOC)*, pages 681–690, 2008b.

Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Bandits and experts in metric spaces. Technical report `http://arxiv.org/abs/1312.1277`. Merged and revised version of conference papers in *ACM STOC 2008* and *ACM-SIAM SODA 2010*, Dec 2013.

Levente Kocsis and Csaba Szepesvari. Bandit based Monte-Carlo planning. In *17th European Conf. on Machine Learning (ECML)*, pages 282–293, 2006.

Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

John Langford and Tong Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *21st Advances in Neural Information Processing Systems (NIPS)*, 2007.

Alessandro Lazaric and Rémi Munos. Hybrid stochastic-adversarial on-line learning. In *22nd Conf. on Learning Theory (COLT)*, 2009.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *19th Intl. World Wide Web Conf. (WWW)*, 2010.

Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *4th ACM Intl. Conf. on Web Search and Data Mining (WSDM)*, 2011.

Tyler Lu, Dávid Pál, and Martin Pál. Showing relevant ads via Lipschitz context multi-armed bandits. In *14th Intl. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2010.

Odalric-Ambrym Maillard and Rémi Munos. Online learning in adversarial Lipschitz environments. In *European Conf. on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pages 305–320, 2010.

H. Brendan McMahan and Matthew Streeter. Tighter bounds for multi-armed bandits with expert advice. In *22nd Conf. on Learning Theory (COLT)*, 2009.

Rémi Munos and Pierre-Arnaud Coquelin. Bandit algorithms for tree search. In *23rd Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2007.

Sandeep Pandey, Deepak Agarwal, Deepayan Chakrabarti, and Vanja Josifovski. Bandits for taxonomies: A model-based approach. In *SIAM Intl. Conf. on Data Mining (SDM)*, 2007.

Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. In *25th Intl. Conf. on Machine Learning (ICML)*, pages 784–791, 2008.

Philippe Rigollet and Assaf Zeevi. Nonparametric bandits with covariates. In *23rd Conf. on Learning Theory (COLT)*, pages 54–66, 2010.

Herbert Robbins. Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc.*, 58:527–535, 1952.

Aleksandrs Slivkins. Distributed approaches to triangulation and embedding. In *16th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 640–649, 2005.

Aleksandrs Slivkins. Multi-armed bandits on implicit metric spaces. In *25th Advances in Neural Information Processing Systems (NIPS)*, 2011.

Aleksandrs Slivkins and Eli Upfal. Adapting to a changing environment: the Brownian restless bandits. In *21st Conf. on Learning Theory (COLT)*, pages 343–354, 2008.

Aleksandrs Slivkins, Filip Radlinski, and Sreenivas Gollapudi. Learning optimally diverse rankings over large document collections. *J. of Machine Learning Research (JMLR)*, 14 (Feb):399–436, 2013. Preliminary version in *27th ICML*, 2010.

Kunal Talwar. Bypassing the embedding: Algorithms for low-dimensional metrics. In *36th ACM Symp. on Theory of Computing (STOC)*, pages 281–290, 2004.

Chih-Chun Wang, Sanjeev R. Kulkarni, and H. Vincent Poor. Bandit problems with side observations. *IEEE Trans. on Automatic Control*, 50(3):338355, 2005.

Yizao Wang, Jean-Yves Audibert, and Rémi Munos. Algorithms for infinitely many-armed bandits. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1729–1736, 2008.

Michael Woodroofe. A one-armed bandit problem with a concomitant variable. *J. Amer. Statist. Assoc.*, 74(368), 1979.

Journal of Machine Learning Research 15 (2014) 2569-2605 Submitted 3/12; Revised 10/13; Published 7/14

Boosting Algorithms for Detector Cascade Learning

Mohammad Saberian SABERIAN@UCSD.EDU
Nuno Vasconcelos NUNO@UCSD.EDU
Statistical Visual Computing Laboratory,
University of California, San Diego
La Jolla, CA 92039, USA

Editor: Yoram Singer

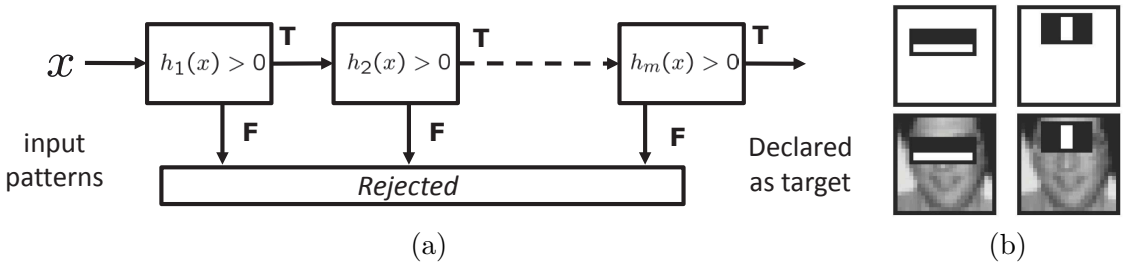
Abstract

The problem of learning classifier cascades is considered. A new cascade boosting algorithm, *fast cascade boosting* (FCBoost), is proposed. FCBoost is shown to have a number of interesting properties, namely that it 1) minimizes a Lagrangian risk that jointly accounts for classification accuracy and speed, 2) generalizes adaboost, 3) can be made cost-sensitive to support the design of high detection rate cascades, and 4) is compatible with many predictor structures suitable for sequential decision making. It is shown that a rich family of such structures can be derived recursively from cascade predictors of two stages, denoted *cascade generators.* Generators are then proposed for two new cascade families, *last-stage* and *multiplicative* cascades, that generalize the two most popular cascade architectures in the literature. The concept of *neutral predictors* is finally introduced, enabling FCBoost to automatically determine the cascade configuration, i.e., number of stages and number of weak learners per stage, for the learned cascades. Experiments on face and pedestrian detection show that the resulting cascades outperform current state-of-the-art methods in both detection accuracy and speed.

Keywords: complexity-constrained learning, detector cascades, sequential decision-making, boosting, ensemble methods, cost-sensitive learning, real-time object detection

1. Introduction

There are many applications where a classifier must be designed under computational constraints. A prime example is object detection, in computer vision, where a classifier must process hundreds of thousands of sub-windows per image, extracted from all possible image locations and scales, at a rate of several images per second. One possibility to deal with this problem is to adopt sophisticated search strategies, such as branch-and-bound or divide-and-conquer, to reduce the number of sub-windows to classify (Lampert et al., 2009; Vijayanarasimhan and Grauman, 2011; Lampert, 2010). While these methods are compatible with popular classification architectures, e.g., the combination of a support vector machine (SVM) and the bag-of-words image representation, they do not speed up the classifier itself. An alternative solution is to examine all sub-windows but adapt the complexity of the classifier to the difficulty of their classification. This strategy has been the focus of substantial attention since the introduction of the detector cascade architecture (Viola and Jones, 2001). As illustrated in Figure 1 a) this architecture is implemented as a sequence of binary classifiers $h_1(x), \ldots h_m(x)$, known as the *cascade stages*. These stages



(a) (b)

Figure 1: (a) detector cascade and (b) examples of weak learners used for face detection (Viola and Jones, 2001).

have increasing complexity, ranging from a few machine operations for $h_1(x)$ to extensive computation for $h_m(x)$. An example x is declared a target by the cascade if and only if it is declared a target by all its stages. Since the overwhelming majority of sub-windows in an image do not contain the target object, a very large portion of the image is usually rejected by the early cascade stages. This makes the average detection complexity quite low. However, because the later stages can be arbitrarily complex, the cascade can have very good classification accuracy. This was convincingly demonstrated by using the cascade architecture to design the first real-time face detector with state-of-the-art classification accuracy (Viola and Jones, 2001). This detector has since found remarkable practical success, and is today popular in applications of face detection involving low-complexity processors, such as digital cameras or cell phones.

In the method of Viola and Jones (2001), cascade stages are designed sequentially, by simply training each detector on the examples rejected by its predecessors. Each stage is designed by boosting decision stumps that operate on a space of Haar wavelet features, such as those shown in Figure 1-b). Hence, each stage is a linear combination of weak learners, each consisting of a Haar wavelet and a threshold. This has two appealing properties. First, because it is possible to evaluate each Haar wavelet with a few machine operations, cascade stages can be very efficient. Second, it is possible to control the complexity of each stage by controlling its number of weak learners. However, while fast and accurate, this detector is not optimal under any sensible definition of cascade optimality. For example, it does not address the problems of 1) how to automatically determine the optimal cascade configuration, e.g., the numbers of cascade stages and weak learners per stage, 2) how to design individual stages so as to guarantee optimality of the cascade as a whole, or 3) how to factor detection speed as an explicit variable of the optimization process. These limitations have motivated many enhancements to the various components of cascade design, including 1) new features (Lienhart and Maydt, 2002; Dalal and Triggs, 2005; Pham et al., 2010; Dollár et al., 2009), 2) faster feature selection procedures (Wu et al., 2008; Pham and Cham, 2007), 3) post-processing procedures to optimize cascade performance (Lienhart and Maydt, 2002; Luo, 2005; Sun et al., 2004), 4) extensions of adaboost for improved design of the cascade stages (Viola and Jones, 2002; Masnadi-Shirazi and Vasconcelos, 2007; Sochman and Matas, 2005; Schneiderman, 2004; Li and Zhang, 2004; Tuzel et al., 2008), 5) alternative cascade structures (Xiao et al., 2003; Bourdev and Brandt, 2005; Xiao et al., 2007; Sochman and

Matas, 2005), and 6) joint, rather than sequential stage design (Dundar and Bi, 2007; Lefakis and Fleuret, 2010; Sochman and Matas, 2005; Bourdev and Brandt, 2005). While these advances improved the performance, the optimal design of a whole cascade is still an open problem. Most existing solutions rely on assumptions, such as the independence of cascade stages, that do not hold in practice.

In this work, we address the problem of automatically learning both the configuration and the stages of a high detection rate detector cascade, under a definition of optimality that accounts for both classification accuracy and speed. This is accomplished with the *fast cascade boosting* (FCBoost) algorithm, an extension of adaboost derived from a Lagrangian risk that trades-off detection performance and speed. FCBoost optimizes this risk with respect to a predictor that complies with the sequential decision making structure of the cascade architecture. These predictors are called *cascade predictors*, and it is shown that a rich family of such predictors can be derived recursively from a set of *cascade generator* functions, which are cascade predictors of two stages. Boosting algorithms are derived for two elements of this family, *last-stage* and *multiplicative* cascades. These are shown to generalize the cascades of embedded (Xiao et al., 2003; Bourdev and Brandt, 2005; Xiao et al., 2007; Sochman and Matas, 2005; Masnadi-Shirazi and Vasconcelos, 2007; Pham et al., 2008) or independent (Viola and Jones, 2001; Schneiderman, 2004; Brubaker et al., 2008; Wu et al., 2008; Shen et al., 2011, 2010) stages commonly used in the literature. The search for the cascade configuration is naturally integrated in FCBoost by the introduction of *neutral predictors*. This allows FCBoost to automatically determine 1) number of cascade stages and 2) number of weak learners per stage, by simple minimization of the Lagrangian risk. The procedure is compatible with existing cost-sensitive extensions of boosting (Viola and Jones, 2002; Masnadi-Shirazi and Vasconcelos, 2007; Pham et al., 2008; Masnadi-Shirazi and Vasconcelos, 2010) that guarantee cascades of high detection rate, and generalizes adaboost in a number of interesting ways. A detailed experimental evaluation on face and pedestrian detection shows that the resulting cascades outperform current state-of-the-art methods in both detection accuracy and speed.

The paper is organized as follows. Section 2 reviews the challenges of cascade learning and previously proposed solutions. Section 3 briefly reviews adaboost, the most popular stage learning algorithm, and proposes its generalization for the learning of detector cascades. Section 4 studies the structure of cascade predictors, introducing the concept of cascade generators. Two generators are then proposed, from which two cascade families (last-stage and multiplicative) are derived. The search for the cascade configuration is then studied in Section 5. In this section the Lagrangian extension of the cascade boosting algorithm is introduced, so as to account for detector complexity in the cascade optimization, and a procedure for the automatic addition of cascade stages during boosting is developed, using neutral predictors. All these contributions are consolidated into the FCBoost algorithm in Section 6, whose specialization to last-stage and multiplicative cascades is shown to generalize the two main previous approaches to cascade design. A number of interesting properties of the algorithm are also discussed, and a cost-sensitive extension is derived. Finally, an experimental evaluation is presented in Section 7 and some conclusions drawn in Section 8. An early version of this work was presented in NIPS (Saberian and Vasconcelos, 2010).

2. Prior Work

A large literature on detector cascade learning has emerged over the past decade. In this section, we briefly review the main problems in this area and their current solutions.

2.1 The Problems of Cascade Learning

As illustrated in Figure 1, a cascaded detector is a sequence of detector *stages*. The aim is to detect instances from a *target* class. Examples from this class are denoted *positives* while all others are denoted *negatives*. An example rejected, i.e., declared a negative, by any stage is rejected by the cascade. Examples classified as positives are propagated to subsequent stages. To be computationally efficient, the cascade must use simple classifiers in the early stages and complex ones later on. Under the procedure proposed by Viola and Jones (2001), the cascade designer must first select a number of stages and the target detection/false-positive rate for each stage. A high detection rate is critical, since improperly rejected positives cannot be recovered. The false-positive rate is less critical, since the cascade false-positive rate can be decreased by addition of stages, although at the price of extra computation. The stages are designed with adaboost. The target detection rate is met by manipulating the stage threshold, and the target false-positive rate by increasing the number of weak learners. This frequently leads to an exceedingly complex learning procedure. One difficulty is that the optimal cascade configuration (number of stages and stage target rates) is unknown. We refer to this as the *cascade configuration problem*. While some configurations have evolved by default, e.g., 20 stages, with a detection rate of 99.5% and a false-positive rate of 50%, there is nothing special about these values. This problem is compounded by the fact that, for late stages where negative examples are close to the classification boundary, it may be impossible to meet the target rates. In this case, the designer must backtrack (redesign some of the previous stages). Frequently, various iterations of parameter tuning are needed to reach a satisfactory cascade. Since each iteration requires boosting over a large set of examples and features, the process can be tedious and time consuming. We refer to this as the *design complexity* problem.

Even when a cascade is successfully designed, the process has no guarantees of optimal classification performance. One problem is that, while computationally efficient, the Haar wavelet features lacks discriminant power for many applications. This is the *feature design* problem. This problem is frequently compounded by lack of convergence of adaboost. Note that while adaboost is consistent (Bartlett and Traskin, 2007), there are no guarantees that a classifier with small number of boosting iterations, e.g., early stages of a cascade, will produce classifiers that generalize well. We refer to this as the *convergence problem*. This problem is magnified by the mismatch between the adaboost risk, which penalizes misses/false-positives equally, and the asymmetry of the target detection and false-positive rates used in practice. Although a stage can always meet the target detection rate by threshold manipulation, the resulting false-positive rate can be strongly sub-optimal (Masnadi-Shirazi and Vasconcelos, 2010). In general, better performance is obtained with *asymmetric* learning algorithms, that optimize the detector explicitly for the target detection rate. This is the *cost-sensitive* learning problem. Besides classification optimality, the learned cascade is rarely the fastest possible. This is not surprising, since speed is not an explicit variable of the cascade optimization process. While the specification

of stage false-positive rates can be used to shuffle computation between stages, there is no way to predict the amount of computation corresponding to a particular rate. This is the *complexity optimization* problem.

2.2 Previous Solutions

Over the last ten years, significant research has been devoted to all of the above problems.

Feature design: Viola and Jones introduced a very efficient set of Haar wavelets (Viola and Jones, 2001). They showed that these features could be extracted, with a few operations, from an integral image (cumulative image sum). While all features in the original Haar set were axis-aligned, it is possible to extent it for 45° rectangles, (Lienhart and Maydt, 2002). Similarly, several authors pursued extensions to other orientations (Carneiro et al., 2008; Du et al., 2006; Messom and Barczak, 2006). More recently, this has been extended to compute integral images over arbitrary polygonal regions (Pham et al., 2010). Beyond these features, integral images can also be used to efficiently compute histograms (Porikli, 2005). This reduces to quantizing the image into a set of *channels* (associated with the histogram bins) and computing an integral image per channel. For example, a computationally efficient version of the HOG descriptor (Dalal and Triggs, 2005) was then developed and used to design a real-time pedestrian detector cascade (Zhu et al., 2006). More recently, this idea has been extended to multiple other channels (Dollár et al., 2009). Finally, extensions have been developed for more general statistical descriptors, e.g., the covariance features (Tuzel et al., 2008). While the algorithms proposed in this work support any of these features, we adopt the Haar set (Viola and Jones, 2001). This is mostly for consistency with the cascade learning literature, where Haar wavelets are predominant.

Design of stage classifiers: A number of enhancements to the stage learning method of Viola and Jones detector have been proposed specifically to address the problems of convergence rate, cost-sensitive learning, and training complexity. One potential solution to the convergence problem is to adopt recent extensions of adaboost, which converge with smaller numbers of weak learners. Since adaboost is a greedy feature selection algorithm, the effective number of weak learners can be reduced by using forward-backward feature selection procedures (Zhang, 2011) or reweighing weak learners by introduction of sparsity constraints in the optimization (Collins et al., 2002; Duchi and Singer, 2009). This results in more accurate classification with less weak learners, i.e., a faster classifier. While these algorithms have not been used in the cascade learning literature, several authors have used similar ideas to improve stage classifiers. For example, augmenting adaboost with a floating search that eliminates weak learners of small contribution to classifier performance (Li and Zhang, 2004) or by using linear discriminant analysis (LDA) (Shen et al., 2011, 2010). Moreover, by interpreting the boosted classifier as a hyperplane in the space of weak learner outputs, several authors have shown how to refine the hyperplane normal so as to maximize class discrimination. Procedures that recompute the weight of each weak learner have been implemented with SVMs (Xiao et al., 2003), variants of LDA (Wu et al., 2008; Shen et al., 2011, 2010), and non-linear feature transformations (Schneiderman, 2004). The hyperplane refinement usually optimizes classification error directly, rather than the exponential loss of adaboost, further improving the match between learning objective and classification performance. Finally, faster convergence is usually possible with different weak

learners, e.g., linear SVMs (Zhu et al., 2006) or decision trees of depth two (Dollár et al., 2009), and boosting algorithms such as realboost or logitboost (Sochman and Matas, 2005; Schneiderman, 2004; Li and Zhang, 2004; Tuzel et al., 2008).

Beyond classification performance, some attention has been devoted to design complexity. Since the bulk of the learning time is spent on weak learner selection, low-complexity methods have been proposed for this. For example, it is possible to trade off memory for computational efficiency (Wu et al., 2008) or to model Haar wavelet responses as Gaussian variables, whose statistics can be computed efficiently (Pham and Cham, 2007). While speeding up the design of each stage, these methods do not eliminate all aspects of threshold tuning, stage backtracking, etc. It could be argued that this is the worst component of design complexity, since these operations require manual supervision. A number of enhancements have been proposed in this area. While Viola and Jones proposed stage-specific threshold adjustments (Viola and Jones, 2001), it is possible to formulate threshold adjustments as an a-posteriori optimization of the whole cascade (Luo, 2005; Sun et al., 2004). These methods are hampered by the limited effectiveness of threshold adjustments when stage detectors have poor ROC performance (Masnadi-Shirazi and Vasconcelos, 2010). Better performance is usually achieved with cost-sensitive extensions of boosting, which optimize a cost-sensitive risk directly (Viola and Jones, 2002; Masnadi-Shirazi and Vasconcelos, 2007; Pham et al., 2008). More recently, Masnadi-Shirazi *et al.* proposed Bayes consistent cost-sensitive extensions of adaboost, logitboost, and realboost (Masnadi-Shirazi and Vasconcelos, 2010). These algorithms were shown to substantially improve the false-positive performance of cascades of high detection rate (Masnadi-Shirazi and Vasconcelos, 2007). These could be combined with the methods which devise a predictor of the optimal false positive and detection rate for each stage, from statistics of the previous stages, so as to design a cascade of cost-sensitive stages automatically (Brubaker et al., 2008; Dundar and Bi, 2007).

Cascade configuration: Most of the above enhancements assume a known cascade configuration and sequential stage learning. This is a suboptimal design strategy and the assumed cascade configuration may not be attainable in practice. An alternative is to adopt cascades of *embedded stages* where each stage is the starting point for the design of the next (Xiao et al., 2003; Bourdev and Brandt, 2005; Xiao et al., 2007; Sochman and Matas, 2005; Masnadi-Shirazi and Vasconcelos, 2007; Pham et al., 2008). The main advantage of this structure is that the whole cascade can be designed with a single boosting run, and adding exit points to a standard classifier ensemble. This also minimizes the convergence rate problems of individual stage design. Using Wald's theory of sequential decision making, it is possible to derive a method for learning embedded stages (Sochman and Matas, 2005). While attempting to optimize the whole cascade, these approaches do not fully address the configuration problem. Some simply add an exit point per weak learner (Masnadi-Shirazi and Vasconcelos, 2007; Xiao et al., 2007; Sochman and Matas, 2005), while others use post-processing (Bourdev and Brandt, 2005; Xiao et al., 2003) or pre-specified detection and false-positive rates (Pham et al., 2008) to determine exit point locations. More recently, it is proposed to learn all stages simultaneously, by modeling a cascade as the product, or logical "AND", of its stages (Lefakis and Fleuret, 2010; Raykar et al., 2010).

Overall, despite substantial progress, no method addresses all problems of cascade learning. Since few approaches explicitly optimize the cascade configuration, fewer among these rely on cost-sensitive learning, and no method optimizes detection speed explicitly, cas-

cade learning can require extensive trial and error. This can be quite expensive from a computational point of view and leads to a tedious design procedure, which can produce sub-optimal cascades. In the following sections we propose an alternative framework, which is fully automated and jointly determines 1) the number of cascade stages, 2) the number of weak learners per stage, and 3) the predictor of each stage, by minimizing a Lagrangian risk that is cost-sensitive and explicitly accounts for detection speed.

3. An Extension of Adaboost for the Design of Classifier Cascades

We start with a brief review of boosting.

3.1 Boosting

A binary classifier $h : \mathcal{X} \to \{-1, 1\}$ maps an example x into a class label $y(x)$. A learning algorithm seeks the classifier of minimum probability of error, $P_X(h(x) \neq y(x))$, in the space of binary mappings

$$\mathbf{H} = \{h | h : \mathcal{X} \to \{-1, 1\}\}.$$

Since $\mathbf{H}$ is not convex and $h \in \mathbf{H}$ not necessarily differentiable, this is usually done by restricting the search to mappings of the form

$$h(x) = sign[f(x)],$$

where $f : \mathcal{X} \to \mathbb{R}$, is a *predictor*. The goal is then to learn the optimal $f(x)$ in a set of predictors

$$\mathbf{F} = \{f | f : \mathcal{X} \to \mathbb{R}\}.$$

This is the predictor which minimizes the classification risk, $\mathcal{R}_E : \mathbf{F} \to \mathbb{R}$,

$$\mathcal{R}_E[f] = E_{X,Y}\{L(y(x), f(x))\} \simeq \frac{1}{|S_t|} \sum_i L(y_i, f(x_i)), \tag{1}$$

where $L : \{+1, -1\} \times \mathbb{R} \to \mathbb{R}$ is a loss function, and $S_t = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is a set of training examples x_i of labels y_i.

Boosting algorithms are iterative procedures that learn f as a combination of simple predictors, known as *weak learners,* from a set $\mathbf{G} = \{g_1(x), \ldots, g_n(x)\} \subset \mathbf{F}$. The optimal combination is the solution of

$$\begin{cases} \min_{f(x)} & \mathcal{R}_E[f] \\ s.t: & f(x) \in span(\mathbf{G}). \end{cases} \tag{2}$$

Each boosting iteration reweights the training set and adds the weak learner of lowest weighted error rate to the weak learner ensemble. When $\mathbf{G}$ is rich enough, i.e., contains a predictor with better than chance-level weighted error rate for any distribution over training examples, the boosted classifier can be arbitrarily close to the minimum probability of error classifier (Freund and Schapire, 1997). For most problems of practical interest, $\mathbf{G}$ is an overcomplete set and the solution of (2) can have many decompositions in $span(\mathbf{G})$. In this case, sparser decompositions are likely to have better performance, i.e., faster computation

and better generalization. Boosting can be interpreted as a greedy forward feature selection procedure to find such sparse solutions.

Although the ideas proposed in this work can be combined with most boosting algorithms, we limit the discussion to adaboost (Freund and Schapire, 1997). This is an algorithm that learns a predictor f by minimizing the risk of (1) when L is the negative exponential of the *margin* $y(x)f(x)$

$$L(y(x), f(x)) = e^{-y(x)f(x)}. \tag{3}$$

This is known as the exponential loss function (Schapire and Singer, 1999).

The boosting algorithms proposed in this paper are inspired by the statistical view of adaboost (Mason et al., 2000; Friedman, 1999). Under this view, each iteration of boosting computes the functional derivatives of the risk along the directions of the weak learners $g_k(x)$, at the current solution $f(x)$. This can be written as

$$\begin{aligned} < \delta\mathcal{R}_E[f], g > &= \frac{d}{d\epsilon}\mathcal{R}_E[f + \epsilon g]\Big|_{\epsilon=0} \\ &= \frac{1}{|S_t|}\sum_i \left[\frac{d}{d\epsilon}e^{-y_i(f(x_i)+\epsilon g(x_i))}\right]_{\epsilon=0} \\ &= -\frac{1}{|S_t|}\sum_i y_i w_i g(x_i), \end{aligned} \tag{4}$$

where $y_i = y(x_i)$ and

$$w_i = w(x_i) = e^{-y_i f(x_i)}, \tag{5}$$

is the weight of example x_i. The latter measures how well x_i is classified by the current predictor $f(x)$. The predictor is then updated by selecting the direction (weak learner) of steepest descent

$$\begin{aligned} g^*(x) &= \arg\max_{g\in\mathbf{G}} < -\delta\mathcal{R}_E[f], g > \\ &= \arg\max_{g\in\mathbf{G}} \frac{1}{|S_t|}\sum_i y_i w_i g(x_i), \end{aligned} \tag{6}$$

and computing the optimal step size along this direction

$$\alpha^* = \arg\min_{\alpha\in\mathbb{R}} \mathcal{R}_E[f + \alpha g^*]. \tag{7}$$

While the optimal step size has a closed form for adaboost (Freund and Schapire, 1997), it can also be found by a line search. The predictor is finally updated according to

$$f(x) = f(x) + \alpha^* g^*(x), \tag{8}$$

and the procedure iterated, as summarized in Algorithm 1.

Algorithm 1 adaboost

Input: Training set $S_t = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where $y_i \in \{1, -1\}$ is the class label of example $\mathbf{x}_i$, and number of iterations N.
Initialization: Set $f(x) = 0$.
for $t = 1$ to N **do**
  Compute $< -\delta R_E[f], g >$ for all weak learners using (4).
  Select the best weak learner $g^*(x)$ using (6).
  Find the optimal step size α^* along $g^*(x)$ using (7).
  Update $f(x) = f(x) + \alpha^* g^*(x)$.
end for
Output: decision rule: $sign[f(x)]$

3.2 Cascade Boosting

In this work, we consider the question of whether boosting can be extended to learn a detector cascade. We start by introducing some notation. As shown in Figure 1-a), a classifier cascade is a binary classifier $H(x) \in \mathbf{H}$ implemented as a sequence of classifiers

$$h_i(x) = sgn[f_i(x)] \quad i = 1, \ldots, m, \tag{9}$$

where the predictors $f_i(x)$ can be any real functions, e.g., linear combinations of weak learners. The cascade implements the mapping $H : \mathcal{X} \to \{-1, 1\}$ where

$$H(x) = \mathcal{H}^m[h_1, \ldots, h_m](x) \quad = \quad \begin{cases} -1 & \text{if } \exists\ k : h_k(x) < 0 \\ +1 & \text{otherwise,} \end{cases} \tag{10}$$

and $\mathcal{H}^m[h_1, \ldots, h_m]$ is a *classifier cascading (CC) operator,* i.e., a functional mapping $\mathcal{H}^m : \mathbf{H}^m \to \mathbf{H}$ of the stage classifiers $h_1, \ldots, h_m$ into the cascaded classifier H.[1]

Similarly, it is possible to define a cascade predictor $F(x)$ for $H(x)$, i.e., a mapping $F : \mathcal{X} \to \mathbb{R}$ such that

$$H(x) = sign[F(x)], \tag{11}$$

where

$$F(x) = \mathcal{F}^m[f_1, \ldots, f_m](x), \tag{12}$$

and $\mathcal{F}^m : \mathbf{F}^m \to \mathbf{F}$ is a *predictor cascading (PC) operator*, i.e., a functional mapping of the stage predictors $f_1, \ldots, f_m$ into the cascade predictor F. We will study the structure of this operator in Section 4. For now, we consider the problem of learning a cascade, given that the operator $\mathcal{F}^m$ is known.

To generalize adaboost to this problem it suffices to use the predictor $F(x)$ in the exponential loss of (3) and solve the optimization problem

$$\begin{cases} \min_{m, f_1, \ldots f_m} & \mathcal{R}_E[F] = \frac{1}{|S_t|} \sum_i e^{-y_i F(x_i)} \\ s.t: & F(x) = \mathcal{F}^m[f_1, \ldots, f_m](x) \\ & \forall i \quad f_i(x) \in span(\mathbf{G}) \end{cases} \tag{13}$$

1. The notation $\mathcal{H}^m[h_1, \ldots, h_m](x)$ should be read as: the value at x of the image of $(h_1, \ldots, h_m)$ under operator $\mathcal{H}^m$.

by gradient descent in $span(\mathbf{G})$. The main difference with respect to adaboost is that, since any of the cascade stages can be updated, multiple gradient steps are possible per iteration. The directional gradient for updating the predictor of the k^{th} stage is

$$\begin{aligned} < \delta\mathcal{R}_E[F], g >_k &= \frac{d}{d\epsilon}\mathcal{R}_E[\mathcal{F}^m[f_1, \dots f_k + \epsilon g, \dots f_m]]\Big|_{\epsilon=0} \\ &= \frac{1}{|S_t|}\sum_i \left[\frac{d}{d\epsilon} e^{-y_i \mathcal{F}^m[f_1,\dots f_k+\epsilon g,\dots f_m](x_i)}\right]_{\epsilon=0} \\ &= \frac{1}{|S_t|}\sum_i \left\{(-y_i)e^{-y_i \mathcal{F}^m[f_1,\dots f_m](x_i)} \left[\frac{d}{d\epsilon}\mathcal{F}^m[f_1, \dots f_k + \epsilon g, \dots f_m]\right]_{\epsilon=0}(x_i)\right\} \\ &= -\frac{1}{|S_t|}\sum_i y_i w(x_i) b_k(x_i) g(x_i), \end{aligned} \tag{14}$$

with

$$w(x_i) = e^{-y_i \mathcal{F}^m[f_1,\dots f_m](x_i)} = e^{-y_i F(x_i)} \tag{15}$$

$$b_k(x_i) = \frac{d}{d\epsilon}\mathcal{F}^m[f_1, \dots f_k + \epsilon g, \dots f_m]\Big|_{\epsilon=0}(x_i). \tag{16}$$

The optimal descent direction for the k^{th} stage is then

$$\begin{aligned} g_k^* &= \arg\max_{g\in\mathbf{G}} < -\delta\mathcal{R}_E[F], g >_k \\ &= \arg\max_{g\in\mathbf{G}} \frac{1}{|S_t|}\sum_i y_i w(x_i) b_k(x_i) g(x_i), \end{aligned} \tag{17}$$

the optimal step size along this direction is

$$\alpha_k^* = \arg\min_{\alpha\in\mathbb{R}} \mathcal{R}_E[\mathcal{F}^m[f_1, .., f_k + \alpha g_k^*, ..f_m]], \tag{18}$$

and the optimal stage update is

$$f_k(x) = f_k(x) + \alpha^* g^*(x). \tag{19}$$

The steps of (15), (17), and (19) constitute a functional gradient descent algorithm for learning a detector cascade, which generalizes adaboost. In particular, the weight of (15) generalizes that of (5), reweighing examples by how well the current cascade classifies them. The weak learner selection rule of (17) differs from that of (6) only in that this weight is multiplied by coefficient $b_k(x_i)$. Finally, (19) is an additive update, similar to that of (8). If the structure of the optimal cascade were known, namely how many stages it contains, these steps could be used to generalize Algorithm 1. It would suffice to, at each iteration t, select the stage k such that g_k^* achieves the smallest risk in (18) and update the predictor of that stage. This only has a fundamental difference with respect to adaboost: the introduction of the coefficients $b_k(x_i)$ in the weak learner selection. We will see that the procedure above can also be extended into an algorithm that learns the cascade configuration. Since these extensions depend on the PC operator $\mathcal{F}$ of (12), we start by studying its structure.

4. The Structure of Cascade Predictors

In this section, we derive a general form for $\mathcal{F}^m$. We show that any cascade is compatible with an infinite set of predictors and that these can be computed recursively. This turns out to be important for the efficient implementation of the learning algorithm of the previous section. We next consider a class of PC operators synthesized by recursive application of a two-stage PC operator, denoted the *generator* of the cascade. Two generators are then proposed, from which we derive two new cascade predictor families that generalize the two most common cascade structures in the literature.

4.1 Cascade Predictors

From (10), a classifier cascade implements the logical-AND of the outputs of its stage classifiers, i.e., $\mathcal{H}^m$ is the pointwise logical-AND of $h_1, \ldots, h_m$,

$$\mathcal{H}^m[h_1, \ldots, h_m](x) = h_1(x) \wedge \ldots \wedge h_m(x), \tag{20}$$

where $\wedge$ is the logical-AND operation. Since, from (10)-(12),

$$\mathcal{H}^m[h_1, \ldots, h_m](x) = sgn[\mathcal{F}^m[f_1, \ldots, f_m](x)], \tag{21}$$

it follows from (9) that

$$sign[\mathcal{F}^m[f_1, \ldots, f_m](x)] = sgn[f_1(x)] \wedge \ldots \wedge sgn[f_m(x)]. \tag{22}$$

This holds if and only if

$$\begin{cases} \mathcal{F}^m[f_1, \ldots, f_m](x) < 0 & \text{if } \exists\, k : f_k(x) < 0 \\ \mathcal{F}^m[f_1, \ldots, f_m](x) > 0 & \text{otherwise.} \end{cases} \tag{23}$$

Since (22) holds for any operator with this property, any such $\mathcal{F}^m$ is denoted a pointwise *soft-AND* of its arguments. In summary, while a cascade implements the logical-AND of its stage decisions, the cascade predictor implements a soft-AND of the corresponding stage predictions. Note that there is an infinite number of soft-AND operators which will implement the same logical-AND operator, once thresholded according to (21). This makes the set of cascade predictors much richer than that of cascades.

4.2 Recursive Implementation

For any m, it follows from (20) and the associative property of the logical-AND that

$$\mathcal{H}^m[h_1, \ldots, h_m] = \begin{cases} \mathcal{H}^2[h_1, h_2], & m = 2 \\ \mathcal{H}^2\left[h_1, \mathcal{H}^{m-1}[h_2, \ldots, h_m]\right] & m > 2. \end{cases} \tag{24}$$

A similar decomposition holds for the soft-AND operator of (23), since

$$sgn\left[\mathcal{F}^m[f_1, \ldots, f_m](x)\right] = \begin{cases} sgn\left[\mathcal{F}^2[f_1, f_2](x)\right], & m = 2 \\ sgn\left[\mathcal{F}^2\left[f_1, \mathcal{F}^{m-1}[f_2, \ldots, f_m]\right](x)\right] & m > 2. \end{cases} \tag{25}$$

The main difference between the two recursions is that, while there is only one logical-AND $\mathcal{H}^2[f_1, f_2]$, an infinite set of soft-AND operators $\mathcal{F}^2[f_1, f_2]$ can be used in (25). In fact, it is

possible to use a different operator $\mathcal{F}^2$ at each level of the recursion, i.e., replace $\mathcal{F}^2$ by $\mathcal{F}^2_m$, to synthesize all possible sequences of soft-AND operators $\{\mathcal{F}^i\}_{i=2}^m$ for which the left-hand side of (25) is the same. For simplicity, we only consider soft-AND operators of the form of (25) in this work.

The recursions above make it possible to derive a recursive decomposition of both the cascade and the sign of its predictor. In particular, defining

$$H_k(x) = \mathcal{H}^{m-k+1}[h_k, \ldots, h_m](x),$$

(24) leads to the *cascade recursion*

$$H_k(x) = \begin{cases} h_m(x), & k = m \\ \mathcal{H}^2\left[h_k, H_{k+1}\right](x), & 1 \leq k < m, \end{cases}$$

with $H_1(x) = H(x)$. Similarly, for any sequence of soft-AND operators $\{\mathcal{F}^i\}_{i=2}^m$ compatible with (25), defining

$$F_k(x) = \mathcal{F}^{m-k+1}[f_k, \ldots, f_m](x),$$

leads to the *predictor recursion*

$$sgn[F_k(x)] = \begin{cases} sgn[f_m(x)], & k = m \\ sgn\left[\mathcal{F}^2\left[f_k, F_{k+1}\right](x)\right], & 1 \leq k < m, \end{cases} \tag{26}$$

with $sgn[F_1(x)] = sgn[F(x)]$. Simplifying (26), in the remainder of this work we consider predictors of the form

$$F_k(x) = \begin{cases} f_m(x), & k = m \\ \mathcal{F}^2\left[f_k, F_{k+1}\right](x), & 1 \leq k < m. \end{cases} \tag{27}$$

Since the core of this recursion is the two-stage predictor

$$\mathcal{G}[f_1, f_2] = \mathcal{F}^2[f_1, f_2], \tag{28}$$

this is denoted the *generator* of the cascade. We will show that the two most popular cascade architectures can be derived from two such generators. For each, we will then derive the cascade predictors $F_k(x)$, the cascade boosting weights $w(x_i)$ of (15), and the coefficients $b_k(x_i)$ of (16). We start by defining some notation to be used in these derivations.

4.3 Some Definitions

Some of the computations of the following sections involve derivatives of Heaviside step functions $u(.)$, which are not differentiable. As is common in the neural network literature, this problem is addressed with the sigmoidal approximation

$$u(x) \approx \sigma(x) = \frac{1}{2}(\tanh(\mu x) + 1). \tag{29}$$

The parameter μ controls the sharpness of the sigmoid. This approximation is well known to have the symmetry $\sigma(-x) = 1 - \sigma(x)$ and derivative $\sigma'(x) = 2\mu\sigma(x)\sigma(-x)$. We also introduce the sequence of *cascaded Heaviside functions*

$$\gamma_k(x) = \begin{cases} 1, & k = 1 \\ \prod_{j<k} u[f_j(x)], & k > 1, \end{cases} \tag{30}$$

and *cascaded rectification functions*

$$\xi_k(x) = \begin{cases} 1, & k = 1 \\ \prod_{j<k} f_j(x) u[f_j(x)], & k > 1, \end{cases} \tag{31}$$

where $u(.)$ is the Heaviside step. The former generalize the Heaviside step, in the sense that $\gamma_k(x) = 1$ if $f_j(x) > 0$ for all $j < k$ and $\gamma_k(x) = 0$ otherwise. The latter generalize the half-wave rectifier, in the sense that $\gamma_k(x) = \prod_{j<k} f_j(x)$ if $f_j(x) > 0$ for all $j < k$ and $\gamma_k(x) = 0$ otherwise.

4.4 Last Stage Cascades

The first family of cascade predictors that we consider is derived from the generator

$$\begin{aligned} \mathcal{G}_1[f_1, f_2](x) &= f_1(x) u[-f_1(x)] + u[f_1(x)] f_2(x) \\ &= \begin{cases} f_1(x) & \text{if } f_1(x) < 0 \\ f_2(x) & \text{if } f_1(x) \geq 0, \end{cases} \end{aligned} \tag{32}$$

Using (27), the associated predictor recursion is

$$F_k(x) = \begin{cases} f_m(x), & k = m \\ f_k(x) u[-f_k(x)] + u[f_k(x)] F_{k+1}(x), & 1 \leq k < m. \end{cases} \tag{33}$$

The k^{th} stage of the associated cascade passes example x to stage $k+1$ if $f_k(x) \geq 0$. Otherwise, the example is rejected with prediction $f_k(x)$. Hence,

$$\mathcal{F}^m[f_1, \ldots, f_m](x) = \begin{cases} f_j(x) & \text{if } f_j(x) < 0 \text{ and} \\ & f_i(x) \geq 0 \;\; i = 1, \ldots, j-1 \\ f_m(x) & \text{if } f_i(x) \geq 0 \;\; i = 1 \ldots, m-1, \end{cases}$$

i.e., the cascade prediction is that of the last stage visited by the example. For this reason, the cascade is denoted a *last-stage cascade*.

This property makes it trivial to compute the weights $w(x)$ of the cascade boosting algorithm, using (15). It suffices to evaluate

$$w(x_i) = e^{-y_i f_{j^*}(x_i)}, \tag{34}$$

where j^* is the smallest k for which $f_k(x_i)$ is negative and $j^* = m$ if there is no such k. The computation of $b_k(x)$ with (16) requires a differentiable form of $\mathcal{F}^m[f_1, \ldots, f_m]$ with

respect to f_k. This can be obtained by recursive application of (33), since

$$\begin{aligned}
\mathcal{F}^m[f_1,\ldots,f_m](x) &= F_1(x) \\
&= f_1(x)u[-f_1(x)] + u[f_1(x)]F_2(x) \\
&= f_1(x)u[-f_1(x)] + u[f_1(x)]\left\{f_2(x)u[-f_2(x)] + u[f_2(x)]F_3(x)\right\} \\
&= \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\prod_{j<i} u[f_j(x)]\right] + F_k(x)\prod_{j<k} u[f_j(x)] \\
&= \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\gamma_i(x)\right] + F_k(x)\gamma_k(x) \qquad k = 1\ldots m \\
&= \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\gamma_i(x)\right] + \gamma_k(x)\left\{f_k(x)u[-f_k(x)] + u[f_k(x)]F_{k+1}(x)\right\} \quad k < m \\
&= \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\gamma_i(x)\right] + \gamma_k(x)\left\{f_k(x) + u[f_k(x)][F_{k+1}(x) - f_k(x)]\right\} \\
&\approx \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\gamma_i(x)\right] + \gamma_k(x)f_k(x) + \gamma_k(x)\sigma[f_k(x)][F_{k+1}(x) - f_k(x)] \qquad (35)
\end{aligned}$$

where $\gamma_k(x)$ are the cascaded Heaviside functions of (30) and we used the differentiable approximation of (29) in (35). Note that neither the first term on the right-hand side of (35) nor γ_k or F_{k+1} depend on f_k. It follows from (16) that

$$b_k(x) = \begin{cases} \gamma_k(x), & k = m \\ \gamma_k(x)\{1 + 2\mu\sigma[f_k(x)][F_{k+1}(x) - f_k(x)]\}\sigma[-f_k(x)] & 1 \le k < m, \end{cases} \qquad (36)$$

where $\sigma(.)$ is defined in (29). Given x, all these quantities can be computed with a sequence of a forward, a backward, and a forward pass through the cascade. The initial forward pass computes $\gamma_k(x)$ for all k according to (30). The backward pass then computes $F_{k+1}(x)$ using (33). The final forward pass computes the weight $w(x)$ and coefficients $b_k(x)$ using (34) and (36). These steps are summarized in Algorithm 2. The procedure resembles the back-propagation algorithm for neural network training (Rumelhart et al., 1968).

4.5 Multiplicative Cascades

The second family of cascade predictors has generator

$$\begin{aligned}
\mathcal{G}_2[f_1,f_2](x) &= f_1(x)u[-f_1(x)] + u[f_1(x)]f_1(x)f_2(x) \\
&= \begin{cases} f_1(x) & \text{if } f_1(x) < 0 \\ f_1(x)f_2(x) & \text{if } f_1(x) \ge 0. \end{cases} \qquad (37)
\end{aligned}$$

Using (27), the associated predictor recursion is

$$F_k(x) = \begin{cases} f_m(x), & k = m \\ f_k(x)u[-f_k(x)] + u[f_k(x)]f_k(x)F_{k+1}(x), & 1 \le k < m \end{cases} \qquad (38)$$

Algorithm 2 Last-stage cascade

Input: Training example (x, y), stage predictors $f_k(x), k = 1, \ldots, m$, sigmoid parameter μ.
Evaluation:
Set $\gamma_1(x) = 1$.
for $k = 2$ to m **do**
  Set $\gamma_k(x) = \gamma_{k-1}(x)u[f_k(x)]$.
end for
Set $F_m(x) = f_m(x)$.
for $k = m - 1$ to 1 **do**
  Set $F_k(x) = f_k(x)u[-f_k(x)] + u[f_k(x)]F_{k+1}(x)$.
end for
Learning:
Set $w(x) = e^{-yf_{j^*}(x)}$ where j^* is the smallest k for which $f_k(x_i) < 0$ and $j^* = m$ if there is no such k.
for $k = 1$ to $m - 1$ **do**
  Set $b_k(x) = \gamma_k(x)\{1 + 2\mu\sigma[f_k(x)][F_{k+1}(x) - f_k(x)]\}\sigma[-f_k(x)]$.
end for
Set $b_k(x) = \gamma_m(x)$.
Output: $w(x), \{F_k(x), b_k(x)\}_{k=1}^m$.

and

$$\mathcal{F}^m[f_1, \ldots, f_m](x) = \begin{cases} \prod_{i \le j} f_i(x) & \text{if } \; f_j(x) < 0 \text{ and} \\ & f_i(x) \ge 0 \;\; i = 1..j-1 \\ \prod_{i=1}^m f_i(x) & \text{if } \; f_i(x) \ge 0 \;\; i = 1..m-1. \end{cases}$$

Hence, the cascade predictor is the product of all stage predictions up-to and including that where the example is rejected. This is denoted a *multiplicative cascade*.

The weights $w(x)$ of the cascade boosting algorithm are

$$w(x_i) = e^{-y_i \prod_{k \le j^*} f_k(x_i)},$$

where j^* is the smallest k for which $f_k(x_i)$ is negative and $j^* = m$ if there is no such k. The computation of $b_k(x)$ with (16) requires a differentiable form of $\mathcal{F}^m[f_1, \ldots, f_m]$ with

respect to f_k. This can be obtained by recursive application of (38), since

$$\begin{aligned}
&\mathcal{F}^m[f_1,\ldots,f_m](x) = F_1(x)\\
&= f_1(x)u[-f_1(x)] + u[f_1(x)]f_1(x)F_2(x)\\
&= f_1(x)u[-f_1(x)] + u[f_1(x)]f_1(x)\{f_2(x)u[-f_2(x)] + u[f_2(x)]f_2(x)F_3(x)\}\\
&= \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\prod_{j<i} f_j(x)u[f_j(x)]\right] + F_k(x)\prod_{j<k} f_j(x)u[f_j(x)]\\
&= \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\xi_i(x)\right] + F_k(x)\xi_k(x) \qquad k = 1\ldots m\\
&= \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\xi_i(x)\right] + \xi_k(x)\left\{f_k(x)u[-f_k(x)] + u[f_k(x)]f_k(x)F_{k+1}(x)\right\} \quad k < m\\
&= \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\xi_i(x)\right] + \xi_k(x)f_k(x)\left\{1 + u[f_k(x)][F_{k+1}(x) - 1]\right\}\\
&\approx \left[\sum_{i=1}^{k-1} f_i(x)u[-f_i(x)]\xi_i(x)\right] + \xi_k(x)f_k(x)\left\{1 + \sigma[f_k(x)][F_{k+1}(x) - 1]\right\}, \qquad (39)
\end{aligned}$$

where $\xi_i(x)$ are the rectification functions of (31) and we used (38) and the differentiable approximation of (29) in (39). Since neither the first term on the right hand side, ξ_k, or F_{k+1} depend on f_k, it follows from (16) that

$$b_k(x) = \begin{cases} \xi_m(x), & k = m\\ \xi_k(x)\{1 + \sigma[f_k(x)][F_{k+1}(x) - 1]\}\{1 + 2\mu f_k(x)\sigma[-f_k(x)]\} & 1 \leq k < m, \end{cases} \qquad (40)$$

where $\sigma(.)$ is defined in (29). Again, these coefficients can be computed with a forward, a backward, and a forward pass through the cascade, which resembles back-propagation, as summarized in Algorithm 3.

5. Learning the Cascade Configuration

Given a cascade configuration, Algorithms 2 or 3, could be combined with the algorithm of Section 3.2 to extend adaboost to the design of last-stage or multiplicative cascades, respectively. However, the cascade configuration is usually not known and must be learned. This consists of determining the number of cascade stages and the number of weak learners per stage.

5.1 Complexity Loss

We start by assuming that the number of cascade stages is known and concentrate on the composition of these stages. So far, we have proposed to simply update, at each boosting iteration, the stage k with the weak learner g_k^* that achieves the smallest risk in (18). While this will produce cascades with good detection accuracy, there is no incentive for the cascade configuration to be efficient, i.e., achieve an optimal trade-off between detection accuracy

Algorithm 3 multiplicative cascade

Input: Training example (x, y), stage predictors $f_k(x), k = 1, \ldots, m$, sigmoid parameter μ.
Evaluation:
Set $\xi_1 = 1$.
for $k = 2$ to m **do**
  Set $\xi_k(x) = \xi_{k-1}(x) f_k(x) u[f_k(x)]$.
end for
Set $F_m(x) = f_m(x)$.
for $k = m - 1$ to 1 **do**
  Set $F_k(x) = f_k(x)u[-f_k(x)] + u[f_k(x)]f_k(x)\ F_{k+1}(x)$.
end for
Learning:
Set $w(x) = e^{-y \prod_{k \leq j^*} f_k(x)}$ where j^* is the smallest k for which $f_k(x_i) < 0$ and $j^* = m$ if there is no such k.
for $k = 1$ to $m - 1$ **do**
  Set $b_k(x) = \xi_k(x)\{1 + \sigma[f_k(x)][F_{k+1}(x) - 1]\}\{1 + 2\mu f_k(x)\sigma[-f_k(x)]\}$.
end for
Set $b_m(x) = \xi_m(x)$.
Output: $w(x), \{F_k(x), b_k(x)\}_{k=1}^m$.

and classification speed. To guarantee such a trade-off it is necessary to search for the most accurate detector under a complexity constraint. This can be done by minimizing the Lagrangian

$$\mathcal{L}[F] \quad = \quad \mathcal{R}_E[F] + \eta \mathcal{R}_C[F], \tag{41}$$

where $F(x)$ and $R_E[F]$ are the cascade predictor and classification risk of (13), respectively,

$$\mathcal{R}_C[F] \quad = \quad E_{X|Y}\{L_C(F, x)|y(x) = -1\} \simeq \frac{1}{|S_t^-|} \sum_{x_i \in S_t^-} L_C(F, x_i),$$

is a complexity risk and η a Lagrange multiplier that determines the trade-off between accuracy and computational complexity. $\mathcal{R}_C[F]$ is the empirical average of a computational loss $L_C(F, x)$, which reflects the number of machine operations required to evaluate $F(x) = \mathcal{F}^m[f_1, \ldots, f_m](x)$, over the set S_t^- of negatives in S_t. The restriction to negative examples is not necessary but common in the classifier cascade literature, where computational complexity is usually defined as the average computation required to reject negative examples. This is mostly because positives are rare and contribute little to the overall computation.

As is the case for the classification risk, where the loss of (3) is an upper bound on the margin and not the margin itself, the computational loss $L_C[F]$ is a surrogate for the computational cost $\mathcal{C}(F, x)$ of evaluating the cascade prediction $F(x)$ for example x. Using the predictor recursions of Section 4.2, this cost can itself be computed recursively. Since, by definition of cascade, example x is either rejected by the predictor f_k of stage k or passed

to the remaining stages,

$$\mathcal{C}(F_k, x) = \begin{cases} \Omega(f_k) + u[f_k(x)]\mathcal{C}(F_{k+1}, x), & k < m \\ \Omega(f_m), & k = m, \end{cases} \quad (42)$$

where $F_k(x)$ is as defined in (27) and $\Omega(f_k)$ is the computational cost of evaluating stage k. Defining $\mathcal{C}(F_{m+1}, x) = 0$, it follows that

$$\begin{aligned} \mathcal{C}(F, x) &= \Omega(f_1) + u[f_1(x)]\mathcal{C}(F_2, x) \\ &= \Omega(f_1) + u[f_1(x)][\Omega(f_2) + u[f_2(x)]\mathcal{C}(F_3, x)] \\ &= \left[\sum_{i=1}^{k-1} \Omega(f_i) \prod_{j<i} u[f_j(x)]\right] + \mathcal{C}(F_k, x) \prod_{j<k} u[f_j(x)] \\ &= \left[\sum_{i=1}^{k-1} \Omega(f_i)\gamma_i(x)\right] + \Omega(f_k)\gamma_k(x) + u[f_k(x)]\mathcal{C}(F_{k+1}, x)\gamma_k(x) \\ &= \delta_k(x) + \Omega(f_k)\gamma_k(x) + \theta_k(x)u[f_k(x)], \end{aligned} \quad (43)$$

where $\gamma_i(x)$ are the cascaded Heaviside functions of (30) and

$$\begin{aligned} \delta_k(x) &= \sum_{i=1}^{k-1} \Omega(f_i)\gamma_i(x), \\ \theta_k(x) &= \mathcal{C}(F_{k+1}, x)\gamma_k(x). \end{aligned} \quad (44)$$

This relates the cascade complexity to the complexity of the k^{th} stage, $\Omega(f_k)$. The surrogate computational loss $L_C[F, x]$ is inspired by the surrogate classification loss of adaboost, which upper bounds the zero-one loss $u[-yf(x)]$ by the exponential $e^{-yf(x)}$. Using the bound $u[f(x)] \leq e^{f(x)}$ on (43) leads to

$$L_C[F, x] = \delta_k(x) + \Omega(f_k)\gamma_k(x) + \theta_k(x)e^{f_k(x)},$$

and the computational risk

$$R_C[F] = \frac{1}{|S_t^-|} \sum_{x_i \in S_t^-} \delta_k(x_i) + \Omega(f_k)\gamma_k(x_i) + \theta_k(x_i)e^{f_k(x_i)}. \quad (45)$$

To evaluate this risk, it remains to determine the computational cost $\Omega(f_k)$ of the predictor of the k^{th} cascade stage. Since $f_k(x) = \sum_l \alpha_l g_l(x)$, $g_l \in \mathbf{G}$, is a linear combination of weak learners, we define

$$\Omega(f_k) = \sum_l \Omega(g_l). \quad (46)$$

Let $\mathcal{W}(f_k) \subset \mathbf{G}$ be the set of weak learners, g_l, that appear in (46). In this work, we restrict our attention to the case where all g_l have the same complexity and $\Omega(f_k)$ is proportional to $|\mathcal{W}(f_k)|$. This is the most common scenario in computer vision problems, such as face detection, where all weak learners are thresholded Haar wavelet features (Viola and Jones, 2001) and have similar computational cost. We will, however, account for the fact that

there is no cost in the repeated evaluation of a weak learner. For this, $\mathcal{W}(f_k)$ is split into two sets. The first, $\mathcal{O}(f_k)$, contains the weak learners used in some earlier cascade stage $f_j, j \leq k$. Since the outputs of these learners can be kept in memory, they require minimal computation (multiplication by α_l and addition to cumulative sum). The second is the set $\mathcal{N}(f_k)$ of weak learners unused in prior stages. The computational cost of f_k is then

$$\Omega(f_k) = |\mathcal{N}(f_k)| + \lambda|\mathcal{O}(f_k)|, \tag{47}$$

where $\lambda < 1$ is the ratio of computation required to evaluate a used vs. new weak learner. This implies that when updating the k^{th} stage predictor

$$\Omega(f_k + \epsilon g) = \Omega(f_k) + \rho(g, f_k),$$

with

$$\rho(g, f_k) = \begin{cases} \lambda & \text{if } g \in \mathcal{O}(f_k) \\ 1 & \text{if } g \in \mathcal{N}(f_k). \end{cases} \tag{48}$$

5.2 Boosting with Complexity Constraints

Given the computational risk of (45), it is possible to derive a boosting algorithm that accounts for cascade complexity. We start by deriving the steepest descent direction of the Lagrangian of (41), with respect to stage k

$$\begin{aligned} < -\delta\mathcal{L}[F], g >_k &= < -\delta \left(R_E[F] + \eta R_C[F] \right), g >_k \\ &= < -\delta R_E[F], g >_k + \eta < -\delta R_C[F], g >_k . \end{aligned}$$

The first term is given by (14), the second requires the descent direction with respect to the complexity risk $R_C[F]$. Using (45),

$$\begin{aligned} < \delta R_C[F], g >_k &= \left. \frac{d}{d\epsilon} R_C(\mathcal{F}^m[f_1, .., f_k + \epsilon g, ..f_m]) \right|_{\epsilon=0} \\ &= \frac{1}{|S_t^-|} \sum_i y_i^s \left. \frac{d}{d\epsilon} L_C \left[\mathcal{F}^m[f_1, .., f_k + \epsilon g, ..f_m], x_i \right] \right|_{\epsilon=0} \\ &= \frac{1}{|S_t^-|} \sum_i y_i^s \left. \frac{d}{d\epsilon} \left[\delta_k(x_i) + [\Omega(f_k) + \rho(f_k, g)]\gamma_k(x_i) + \theta_k(x_i) e^{f_k(x_i) + \epsilon g(x_i)} \right] \right|_{\epsilon=0} \\ &= \frac{1}{|S_t^-|} \sum_i y_i^s \psi_k(x_i)\theta_k(x_i) g(x_i), \end{aligned} \tag{49}$$

where $y_i^s = I(y_i = -1)$, $I(x)$ is the indicator function, $\theta_k(x_i)$ as in (44) and

$$\psi_k(x_i) = e^{f_k(x_i)}. \tag{50}$$

Finally, combining (14) and (49),

$$< -\delta\mathcal{L}[F], g >_k = \sum_i \left(\frac{y_i w(x_i) b_k(x_i)}{|S_t|} - \eta \frac{y_i^s \psi_k(x_i)\theta_k(x_i)}{|S_t^-|} \right) g(x_i), \tag{51}$$

Algorithm 4 BestStageUpdate

Input: Training set S_t, trade-off parameter η, cascade $[f_1, \ldots, f_m]$, index k of the stage to update, sigmoid parameter μ.
for each pair (x_i, y_i) in S_t **do**
Compute $w(x_i)$, $b_k(x_i)$, $F_k(x_i)$ e.g., using Algorithm 2 for last-stage or Algorithm 3 for multiplicative cascades.
Compute $\theta_k(x_i)$, $\psi_k(x_i)$ with (44) and (50).
end for
Find the best update $(\alpha_k^*, g_k^*(x))$ for the k^{th} stage using (51)-(54).
Output: $\alpha_k^*, g_k^*(x)$

where $w(x_i) = e^{-y_i F(x_i)}$ and $b_k(x_i)$ is given by (36) for last-stage and by (40) for multiplicative cascades.

It should be noted that, although (51) does not depend on $\rho(f_k, g)$, the complexity of the optimal weak learner g^* affects the computational risk in (45) and thus the magnitude of the steepest descent step. To account for this, we find the best update for f_k in two steps. The first step searches for the best update within $\mathcal{O}(f_k)$ and $\mathcal{N}(f_k)$

$$g_{1,k}^* = \arg \max_{g \in \mathcal{O}(f_k)} < -\delta \mathcal{L}[F], g >_k \quad (52)$$

$$g_{2,k}^* = \arg \max_{g \in \mathcal{N}(f_k)} < -\delta \mathcal{L}[F], g >_k, \quad (53)$$

and computes the corresponding optimal steps sizes

$$\alpha_{j,k}^* = \arg \min_{\alpha \in \mathbb{R}} \mathcal{L}[\mathcal{F}^m[f_1, ..f_k + \alpha g_{j,k}, ..f_m]], \quad (54)$$

for $j = 1, 2$. The second step chooses the update that most reduces $\mathcal{L}[F]$ as the best update for the k^{th} stage. The overall procedure is summarized in Algorithm 4. Using this procedure to cycle through all cascade stage updates within each iteration of the algorithm of Section 3.2 and selecting the one that most reduces $\mathcal{L}[F]$ produces an extension of adaboost for cascade learning that optimizes the trade-off between detection accuracy and complexity.

5.3 Growing a Detector Cascade

So far, we have assumed that the number of cascade stages is known. Since this is usually not the case, there is a need for a procedure that learns this component of the cascade configuration. In this work, we adopt a greedy strategy, where cascade stages are added by the boosting algorithm itself, whenever this leads to a reduction of the risk. It is assumed that a new stage, or predictor g, can only be added at the end of the existing cascade, i.e., transforming a m-stage predictor $\mathcal{F}^m[f_1, \ldots, f_m](x)$ into a $m+1$-stage predictor $\mathcal{F}^{m+1}[f_1, \ldots, f_m, g](x)$. This is consistent with current cascade design practices, where stages are appended to the cascade when certain heuristics are met.

The challenge of a risk-minimizing formulation of this process is to pose the addition of a new stage as a possible gradient step. Recall that, at each iteration of a gradient descent algorithm, the current solution, v^t, is updated by

$$v^{t+1} = v^t + \alpha \overline{v},$$

where $\overline{v}$ is the gradient update and α is step size found by a line search. An immediate consequence is that, if no update is taken in an iteration, i.e., $\alpha = 0$ or $\overline{v} = 0$, the value of the objective function should remain unaltered. For the proposed cascade boosting algorithms this condition is not trivial to guarantee when a new stage is appended to the current cascade. For example, choosing $g(x) = 0$ may change the current solution since, in general,

$$\mathcal{F}^{m+1}[f_1, \ldots, f_m, 0](x) \neq \mathcal{F}^m[f_1, \ldots, f_m](x).$$

To address this problem, we introduce the concept of *neutral predictors*. A stage predictor $n(x) : \mathcal{X} \to \mathbb{R}$ is neutral for a cascade of predictor $\mathcal{F}^m[f_1, \ldots, f_m]$ if and only if

$$\mathcal{F}^{m+1}[f_1, \ldots, f_m, n](x) = \mathcal{F}^m[f_1, \ldots, f_m](x). \tag{55}$$

If such a neutral predictor exists, then it is possible to grow a cascade by defining the new stage as

$$f_{m+1}(x) = n(x) + g(x),$$

where $g(x)$ is the best update found by gradient descent. In this case, it follows from (55) that a step of $g(x) = 0$ will leave the cascade risk unaltered. Given a cascade generator, a predictor n that satisfies (55) can usually be found with (28), i.e., it suffices that n satisfies

$$f_m(x) = \mathcal{G}[f_m, n](x), \tag{56}$$

where $\mathcal{G}$ is the generator that defines the PC operator $\mathcal{F}^m$. For example, from (32), the neutral predictor of a last-stage cascade must satisfy

$$f_m(x) = f_m(x)u[-f_m(x)] + u[f_m(x)]n(x),$$

a condition met by

$$n(x) = f_m(x). \tag{57}$$

Similarly, from (37), the neutral predictor of a multiplicative cascade must satisfy

$$f_m(x) = f_m(x)u[-f_m(x)] + u[f_m(x)]f_m(x)n(x),$$

which is met by

$$n(x) = 1. \tag{58}$$

These neutral predictors are also computationally efficient. In fact, (57) and (58) add no computation to the evaluation of predictor $f_{m+1}(x)$, i.e., to the computation of $g(x)$ itself. This is obvious for (58) which is a constant, and follows from the fact that $f_m(x)$ has already been computed in stage m for (57). This computation can simply be reused at stage $m + 1$ with no additional cost. Hence, for both models

$$\mathcal{C}(\mathcal{F}^{m+1}[f_1, \ldots, f_m, n], x) = \mathcal{C}(\mathcal{F}^m[f_1, \ldots, f_m], x),$$

and

$$\mathcal{L}[\mathcal{F}^{m+1}[f_1, \ldots, f_m, n]] = \mathcal{L}[\mathcal{F}^m[f_1, \ldots, f_m]].$$

In summary, the addition of stages *does not* require special treatment in the proposed cascade learning framework. It suffices to append a neutral predictor to the cascade and find the best update for this new stage. If this reduces the objective function of (41) further than updating other stages, the new stage is *automatically* created and appended to the cascade. In this way, the cascade grows organically, as a side effect of the risk optimization, and there is no need for heuristics.

Algorithm 5 FCBoost

Input: Training set $S = \{(\mathbf{x}_1, y_1) \ldots, (\mathbf{x}_n, y_n)\}$, trade-off parameter η, sigmoid parameter μ, and number of iterations N.
Initialization: Set $m = 0$ and $f_1(x) = n(x)$, e.g., using (57) for last-stage and (58) for multiplicative cascade.
for $t = 1$ to N **do**
  for $k = 1$ to m **do**
    $(\alpha_k^*, g_k^*) = \text{BestStageUpdate}(S, \eta, [f_1, ...f_m], k, \mu)$.
  end for
  $(\alpha_{m+1}^*, g_{m+1}^*) = \text{BestStageUpdate}(S, \eta, [f_1, ...f_{m+1}], m+1, \mu)$.
  for $k = 1$ to m **do**
    Set $\hat{\mathcal{L}}(k) = \mathcal{L}\left[\mathcal{F}^m(f_1, .., f_k + \alpha_k^* g_k^*, .., f_m)\right]$ using (41).
  end for
  Set $\hat{\mathcal{L}}(m+1) = \mathcal{L}\left[\mathcal{F}^{m+1}(f_1, .., f_m, f_{m+1} + \alpha_{m+1}^* g_{m+1}^*(x))\right]$ using (41).
  Find $k^* = \arg\min_{k \in \{1,\ldots,m+1\}} \hat{\mathcal{L}}(k)$.
  Set $f_{k^*} = f_{k^*} + \alpha_{k^*}^* g_{k^*}^*$.
  if $k^* = m + 1$ **then**
    Set $m = m + 1$.
    Set $f_{m+1}(x) = n(x)$.
  end if
end for
Output: decision rule: $sgn[\mathcal{F}^m(f_1, \ldots, f_m)]$.

6. The FCBoost Cascade Learning Algorithm

In this section, we combine the contributions from the previous sections into the *Fast Cascade Boosting* (FCBoost) algorithm, discuss its connections with the previous literature and some interesting properties.

6.1 FCBoost

FCBoost is initialized with a neutral predictor. At each iteration, it finds the best update $g_k^*(x)$ for each of the cascade stages and the best stage to add at the end of the cascade. It then selects the stage k^* whose update $g_{k^*}^*(x)$ most reduces the Lagrangian $\mathcal{L}[F]$. If k^* is the newly added stage, a new stage is created and appended to the cascade. The procedure is summarized in Algorithm 5. Note that the only parameters are the multiplier η of (41), which encodes the relative importance of cascade speed vs. accuracy for the cascade designer, and the sigmoid parameter μ that controls the smoothness of the Heaviside approximation. In our implementation we always use $\mu = 5$. Given these parameters, FCBoost will automatically determine both the cascade configuration (number of stages and number of weak learners per stage) and the predictor of each stage, so as to optimize the trade-off between detection accuracy and complexity which is specified through η.

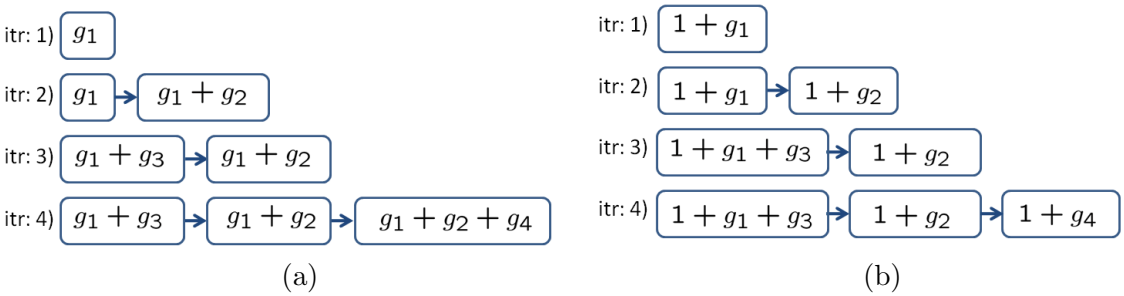


Figure 2: Illustration of the different configurations produced by identical steps of (a) last-stage and (b) multiplicative cascade learning.

6.2 Connections to the Previous Cascade Learning Literature

FCBoost supports a large variety of cascade structures. The cascade structure is defined by the generator $\mathcal{G}$ of (28), since this determines the neutral predictor $n(x)$, according to (56), and consequently how the cascade grows as boosting progresses. The two cascade predictors used in this work, last-stage and multiplicative, cover the two predominant cascade structures in the literature. The first is the *independent stage* (IS) structure, (Viola and Jones, 2001). In this structure stage predictors are designed independently,[2] in the sense that the learning of f_k starts from an empty predictor which is irrespective of the composition of the previous stages, $f_j, j < k$. The second structure is the *embedded stage* (ES) structure where predictors of consecutive stages are related by

$$f_{k+1}(x) = f_k(x) + \mathbf{w}(x),$$

and $\mathbf{w}(x)$ is a single or linear combination of weak learners (Xiao et al., 2003). Under this structure, each stage predictor contains the predictor of the previous stage, which is augmented with some weak learners.

The connection between these structures and the models proposed in this paper can be understood by considering the neutral predictors of the latter. For multiplicative cascades, it follows from (58) that

$$f_{m+1}(x) = 1 + \alpha g(x),$$

and there is no dependence between consecutive stages. Hence, multiplicative cascades have the IS structure. For last stage cascades, it follows from (57) that

$$f_{m+1}(x) = f_m(x) + \alpha g(x).$$

If FCBoost always updates the last two stages, this produces a cascade with the ES structure. Since FCBoost is free to update any stage, it can produce more general cascades, i.e., a superset of the set of cascades with the ES structure.

It is interesting that two predictors with the very similar generators of (32) and (37) produce very different cascade structures. This is illustrated in Figure 2, where we consider the cascades resulting from the following sequence of operations:

2. Note that the predictors are always *statistically* dependent, since the role of h_{i+1} is to classify examples not rejected by h_i.

- **iteration 1:** start form an empty classifier, create first stage.
- **iteration 2:** add a new stage.
- **iteration 3:** update first stage.
- **iteration 4:** add a new stage.

Note that while the last-stage cascade of a) has substantial weak learner sharing across stages, this is not true for the multiplicative cascade of b), which is similar to the Viola and Jones cascade (Viola and Jones, 2001).

6.3 Properties

Beyond these connections to the literature, FCBoost has various interesting properties as a cascade boosting algorithm. First, its example weighing is very similar to that of adaboost (Freund and Schapire, 1997). A comparison of (5) and (15) shows that FCBoost reweights examples by how well they are classified by the current cascade. As in adaboost, this is measured by the classification margin, but now with respect to the cascade predictor, F, (margin yF) rather than a simple predictor f (margin yf). Second, the weak learner selection rule of FCBoost is very similar to that of adaboost. While in (6) adaboost selects the weak learner g that maximizes

$$\frac{1}{|S_t|}\sum_i y_i w_i g(x_i),$$

in (52)-(53) FCBoost selects the stage k and weak learner g that maximize

$$\sum_i \left(\frac{y_i w(x_i) b_k(x_i)}{|S_t|} - \eta \frac{y_i^s \psi_k(x_i)\theta_k(x_i)}{|S_t^-|} \right) g(x_i). \tag{59}$$

When $\eta = 0$, the only significant difference is the inclusion of $b_k(x_i)$ in (59). To understand the role of this term note that, from (36) and (40), $b_k(x_i) = 0$ whenever $\gamma_k(x_i) = 0$ in (30), and $\xi_k(x_i) = 0$ in (31). This implies that there is at least one stage $j < k$ such that $f_j(x_i) < 0$, i.e., where x_i is rejected. When this holds, $b_k(x_i) = 0$ prevents x_i from influencing the update of $f_k(x)$. This is sensible: since x_i will not reach the k^{th} stage, it should not affect its learning. Hence, the coefficients $b_k(x_i)$ can be seen as *gating coefficients*, which prevent examples rejected by earlier stages from affecting the learning of stage k. If $\eta \neq 0$, a similar role is played by $\theta_k(x_i)$ in the second term of (59) since, from (44), $\theta_k(x_i) = 0$ whenever $\gamma_k(x_i) = 0$. Thus, if x_i is rejected by a stage $j < k$, its processing complexity is not considered for any stage posterior to j. Due to the gating coefficients $b_k(x_i)$ and $\theta_k(x_i)$, FCBoost emulates the bootstrapping procedure commonly used in cascade design. This is a procedure that eliminates the examples rejected by each stage from the training set of subsequent stages. These examples are replaced with new false positives (Viola and Jones, 2001; Sung and Poggio, 1998). While FCBoost emulates "example discarding" with the gating coefficients $b_k(x_i)$ and $\theta_k(x_i)$, it does not seek new false positives. This still requires the "training set augmentation" of bootstrapping.

A third interesting property of FCBoost is the complexity penalty (second term) of (59). From (44) and (50) this is, up to constants,

$$-y_i^s \gamma_k(x_i) e^{f_k(x_i)} \mathcal{C}(F_{k+1}, x_i) g(x_i).$$

Given example x_i and cascade stage k, all factors in this product have a meaningful interpretation. First, since $y_i^s \gamma_k(x_i)$ is non-zero only for negative examples which have not been rejected by earlier cascade stages ($j < k$), it acts as a selector of the false-positives that reach stage k. Second, since $f_k(x_i)$ measures how deeply x_i penetrates the positive side of the stage k classification boundary, $e^{f_k(x_i)}$ is large for the false-positives that stage k confidently assigns to the positive class. Third, since $\mathcal{C}(F_{k+1}, x_i)$ is the complexity of processing x_i by the stages beyond k, it measures how deeply x_i penetrates the cascade, if not rejected by stage k. Finally, $g(x_i)$ is the label given to x_i by weak learner $g(x)$. Since only $g(x_i)$ can be negative, the product is maximized when $g(x_i) = -1$, $\gamma_k(x_i) = 1$ and $f_k(x_i)$ and $\mathcal{C}(F_{k+1}, x_i)$ are as large as possible. Hence, the best weak learner is that which, on average, declares as negatives the examples which 1) are false-positives of the earlier stages, 2) are most confidently accepted as false-positives by the current stage, and 3) penetrate the cascade most deeply beyond this stage. This is intuitive, in the sense that it encourages the selection of the weak learner that most contradicts the current cascade on its most costly mistakes.

In summary, FCBoost is a generalization of adaboost with similar example weighting, gating coefficients that guarantee consistency with the cascade structure, and a cost function that accounts for classifier complexity. This encourages the selection of weak learners that correct the false-positives of greatest computational cost. It should be mentioned that while we have used adaboost to derive FCBoost, similar algorithms could be derived from other forms of boosting, e.g., logitboost, gentle boost (Friedman et al., 1998), KLBoost (Liu and Shum, 2003) or float boost (Li and Zhang, 2004). This would amount to replacing the exponential loss, (3), with other loss functions. While the resulting algorithms would be different, the fundamental properties (example reweighing, additive updates, gating coefficients) would not. We next exploit this to develop a cost-sensitive extension of FCBoost.

6.4 Cost-Sensitive FCBoost

While positive examples rejected by a cascade stage cannot be recovered by subsequent stages, the cascade false positive rate can always be reduced through addition of stages. Hence, in cascade learning, maintaining a high detection rate across stages is more critical than maintaining a low false positive rate. This is difficult to guarantee with the risk of (1), which is an upper bound on the error rate, treating misses and false positives equally. Several approaches have been proposed to enforce asymmetry during cascade learning. One possibility is to manipulate the thresholds of the various detector stages to guarantee the desired detection rate (Viola and Jones, 2001; Sochman and Matas, 2005; Luo, 2005). This is usually sub-optimal, since boosting predictors are not well calibrated outside a small neighborhood of the classification boundary (Mease and Wyner, 2008). Threshold tuning merely changes the location of the boundary and can perform poorly (Masnadi-Shirazi and Vasconcelos, 2010). An alternative is to use cost sensitive boosting algorithms (Viola and Jones, 2002; Masnadi-Shirazi and Vasconcelos, 2007), derived from asymmetric losses

that weigh miss-detections more than false-positives, optimizing the cost-sensitive boundary directly. This usually outperforms threshold tuning.

In this work we adopt the cost sensitive risk of

$$\begin{aligned} R_E^c(f) &= \frac{C}{|S_t^+|}\sum_{x_i\in S_t^+} e^{-y_i f(x_i)} + \frac{1-C}{|S_t^-|}\sum_{x_i\in S_t^-} e^{-y_i f(x_i)} \\ &= \sum_{x_i\in S_t} y_i^c e^{-y_i f(x_i)}, \end{aligned} \tag{60}$$

where $C \in [0,1]$ is a cost factor,

$$y_i^c = \frac{C}{|S_t^+|} I(y_i = 1) + \frac{1-C}{|S_t^-|} I(y_i = -1),$$

$I(.)$ the indicator function, and the relative importance of positive vs. negative examples is determined by the ratio $\frac{C}{1-C}$ (Viola and Jones, 2002). This leads to the cost-sensitive Lagrangian

$$\mathcal{L}^c[F] = R_E^c[F] + R_C[F]. \tag{61}$$

A derivation similar to that of (14) can be used to show that

$$< \delta R_E^c[F], g >_k = -\sum_i y_i y_i^c w(x_i) b_k(x_i) g(x_i), \tag{62}$$

where $w(x_i) = e^{-y_i F(x_i)}$ and $b_k(x_i)$ is given by (36) for last-stage and by (40) for multiplicative cascades. Finally, combining (61), (62), and (49),

$$< -\delta\mathcal{L}^c[F], g >_k = \sum_i \left(y_i y_i^c w(x_i) b_k(x_i) - \eta \frac{y_i^s \psi_k(x_i)\theta_k(x_i)}{|S_t^-|} \right) g(x_i). \tag{63}$$

The cost-sensitive version of FCBoost replaces (51) with (63) in (52)-(53) and $\mathcal{L}$ by $\mathcal{L}^c$ in (54).

6.5 Open Issues

One subtle difference between adaboost and FCBoost, with $\eta = 0$, is the feasible set of the underlying optimization problems. Rewriting the FCBoost problem of (13) as

$$\begin{cases} \min_f & R_E[f] \\ s.t: & f \in \Omega_{\mathbf{G}}, \end{cases} \tag{64}$$

where

$$\Omega_{\mathbf{G}} = \{f | \exists f_1, ...f_m \in \mathbf{G} \quad \text{such that} \quad f(x) = \mathcal{F}^m[f_1, \ldots, f_m](x) \quad \forall x\}.$$

and comparing (64) to (2), the two problems differ in their feasible sets, $span(\mathbf{G})$ for adaboost vs. $\Omega_{\mathbf{G}}$ for FCBoost. Since any $\hat{f} \in span(\mathbf{G})$ is equivalent to a one-stage cascaded predictor, it follows that $\hat{f} \in \Omega_{\mathbf{G}}$ and

$$span(\mathbf{G}) \subset \Omega_{\mathbf{G}}.$$

Hence, the feasible set of FCBoost is larger than that of adaboost, and FCBoost can, in principle, find detectors of lower risk. Hence, all generalization guarantees of adaboost hold, in principle, for cascades learned with FCBoost. There is, however, one significant difference. Since $span(\mathbf{G})$ is a convex set, the optimization problem of (2) is convex whenever $R_E(f)$ is a convex function of f. This is the case for the adaboost risk, and adaboost is thus guaranteed to converge to a global minimum. However, since $\Omega_{\mathbf{G}}$ can be a non-convex set, no such guarantees exist for FCBoost. Hence, FCBoost can converge to a local minimum. We illustrate this with an example in Section 7.1. In general, the convexity of $\Omega_{\mathbf{G}}$ depends on the PC operator $\mathcal{F}^m$ and the set of weak learners $\mathbf{G}$. There is currently little understanding on what conditions are necessary to guarantee convexity.

7. Evaluation

In this section, we report on several experiments conducted to evaluate FCBoost. We start with a set of experiments designed to illustrate the properties of the algorithm. We then report results on its use to build face and pedestrian detectors with state-of-the-art performance in terms of detection accuracy and complexity. In all cases, the training set for face detection contained $4,500$ faces (along with their flipped replicas) and $9,000$ negative examples, of size 24×24 pixels, while pedestrian detection relied on a training set of $2,347$ positive and $2,000$ negatives examples, of size 72×30, from the Caltech Pedestrian data set (Dollár et al., 2012). All weak learners were decision-stumps on Haar wavelets (Viola and Jones, 2001).

7.1 Effect of η

We started by studying the impact of the Lagrange multiplier η, of (41), on the accuracy vs. complexity performance of FCBoost cascades. The test set consisted of 832 faces (along with their flipped replicas) and $1,664$ negatives. All detectors were trained for 50 iterations. The unit computational cost was set to the cost of evaluating a new Haar feature. This resulted in a cost of $\frac{1}{5}$ units for feature recycling, i.e., $\lambda = \frac{1}{5}$ in (47). Figure 3 quantifies the structure of the cascades learned by FCBoost with $\eta = 0$ and $\eta = 0.04$: multiplicative in a) and last-stage in b). The top plots summarize the number of features assigned to each cascade stage, and those at the bottom the computational cost per stage. Note that since, from (57), the neutral predictor of the last-stage cascade is its last stage, each of the last-stage cascade stages benefits from the features evaluated in the previous stages. Hence, as shown in the top plot of Figure 3-b, the number of weak learners per stage is monotonically increasing. However, because most features are recycled, the cost is still dominated by the early stages, when $\eta = 0$. With respect to the impact of η, its is clear that, for both structures, a small η produces short cascades whose early stages contain many weak learners. On the other hand, a large η leads to much deeper cascades, and a more uniform distribution of weak learners and computation. This is sensible, since larger η place more emphasis on computational efficiency and this requires that the early stages, which tend to be evaluated for most examples, be very efficient. Hence, long cascades with a few weak learners per stage tend to be computationally more efficient than short cascades with many learners per stage.

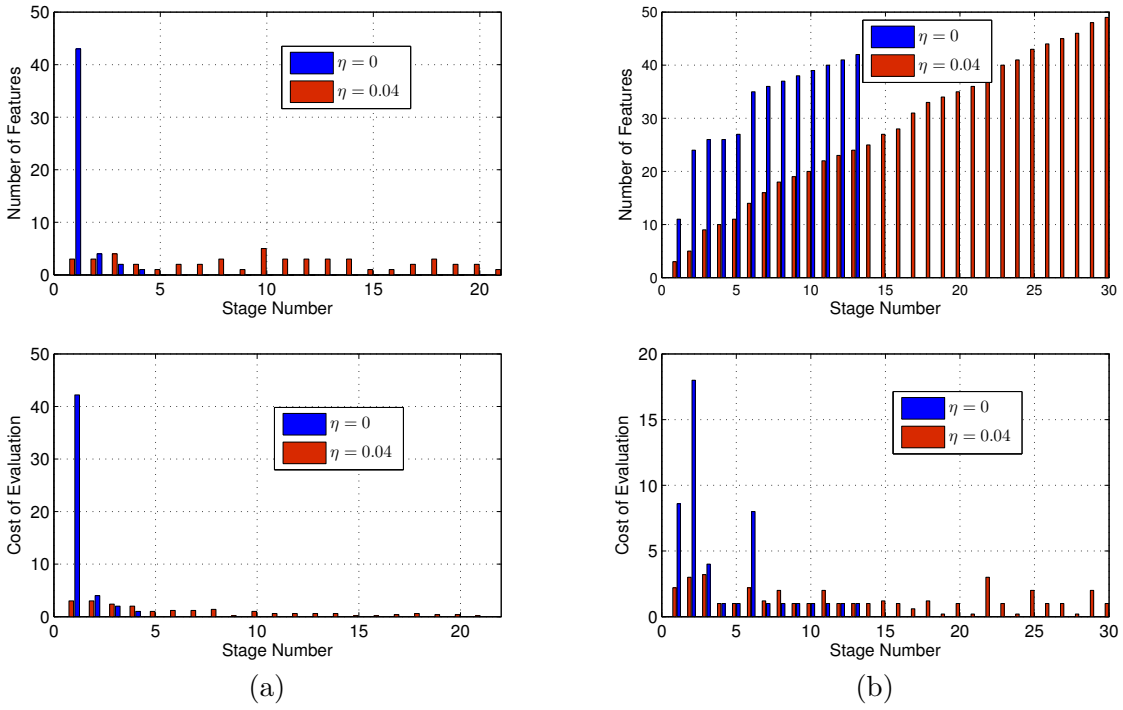


Figure 3: Number of features (top) and computational cost (bottom) per stage of an FCBoost cascade: (a) multiplicative, (b) last-stage.

The accuracy vs. complexity trade-off of these cascades was compared to those of a non-cascaded adaboost detector and a cascade of embedded stages derived from this detector (Masnadi-Shirazi and Vasconcelos, 2007). This converts the detector into a cascade by inserting a rejection point per weak learner. The resulting cascade has embedded stages which add a single weak learner to their predecessors and is equivalent to the chain boost cascade (Xiao et al., 2003). Figure 4 depicts the trade-off between computation and accuracy of adaboost, chain boost, and FCBoost cascades with $\eta \in [0, 0.04]$. The left-most (right-most) point on the FCBoost curves corresponds to $\eta = 0$ ($\eta = 0.04$). adaboost and ChainBoost points were obtained by limiting the number of weak learners, with a single weak learner (full detector) for the right-most (left-most) point. Several observations can be made from the figure. First, as expected, increasing the trade-off parameter η produces FCBoost cascades with less computation and higher error. Second, FCBoost has a better trade-off between complexity and accuracy (curves closer to the origin). Third, among FCBoost models, last-stage cascades have uniformly better trade-off than their multiplicative counterparts. Since last-stage are generalized embedded cascades, this confirms previous reports on the advantages of embedded over independent stages (Pham et al., 2008; Xiao et al., 2003). Finally, it is interesting to note that, when $\eta = 0$, the Lagrangian of (41) is equivalent to the adaboost risk, i.e., FCBoost and adaboost minimize the same objective. However, due to their different feasible sets, they can learn very different detectors

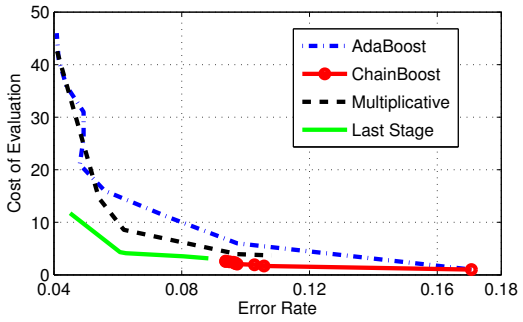


Figure 4: Computational cost vs. error rate of the detectors learned with adaboost, chain boost, and FCBoost with the last-stage and multiplicative structures.

	adaboost	FCBoost+last-stage	FCBoost+multiplicative
Err. rate	4.03%	4.51%	4.15%
Eval. cost	50	11.74	42.54

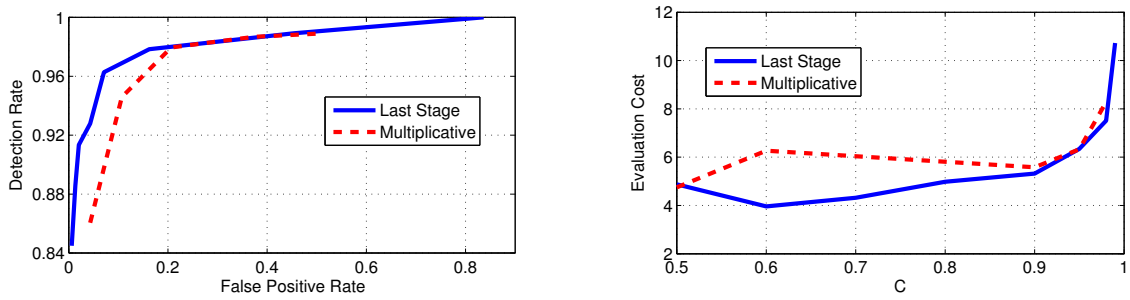
Table 1: Performance comparison between adaboost and FCBoost, for $\eta = 0$.

(see Section 6.5). While the larger feasible set of FCBoost suggests that it should produce detectors of smaller risk than adaboost, this did not happen in our experiments.

Table 1 summarizes the error and cost of adaboost and the two FCBoost methods for $\eta = 0$. Note that the adaboost detector has a slightly lower error. The weaker accuracy of the FCBoost detectors suggests that the latter does get trapped in local minima. This is, in fact, intuitive as the decision to add a cascade stage makes it impossible for the gradient descent procedure to revert back to a non-cascaded detector. By making such a decision, FCBoost can compromise the global optimality of its solution, if the global optimum is a non-cascaded detector. Interestingly, FCBoost sometimes decides to add stages even when $\eta = 0$ (see Figure 3). As shown in Table 1, this leads to a slightly more error-prone but much more efficient detector than adaboost. In summary, even without pressure to minimize complexity ($\eta = 0$), FCBoost may trade-off error for complexity. This may be desirable or not, depending on the application. In the experiment of Table 1, FCBoost seems to make sensible choices. For the last-stage structure, it trades a small increase in error (0.48%) for a large decrease in computation (76.5%). For the multiplicative structure, it trades-off a very small increase in error (0.12%) for a moderate (16%) decrease in computation.

7.2 Cost-Sensitive FCBoost

We next consider the combination of FCBoost and the cost sensitive risk of (60). Since the advantages of cost-sensitive boosting over threshold tuning are now well established (Viola and Jones, 2002; Masnadi-Shirazi and Vasconcelos, 2010), we limit the discussion to the effect of the cost factor C on the behavior of FCBoost cascades. Cascaded face detectors were learned for cost factors $C \in \{0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99\}$. Figure 5 a) presents the trade-off between detection and false positive rate for last-stage and multiplicative



(a) (b)

Figure 5: Performance of cascades learned with cost-sensitive FCBoost, using different cost factors C. (a) ROC curves, (b) computational complexity.

cascades. In both cases, the leftmost (rightmost) point corresponds to $C = 0.5$ ($C = 0.99$). Figure 5 b) presents the equivalent plot for computational cost. Several observations can be made. First, as expected, larger cost factors C produce detectors of higher detection and higher false-positive rate. Second, they lead to cascades of higher complexity. This is intuitive since, for large cost factors, FCBoost aims for a high detection rate and is very conservative about rejecting examples. Hence, many negatives penetrate deep into the cascade, and computation increases. Third, comparing the curves of the last-stage and multiplicative cascades, the former again has better performance. In particular, last-stage cascades combine higher ROC curves in Figure 5 a) with lower computational cost in Figure 5 b).

7.3 Face and Pedestrian Detection

Over the last decade, there has been significant interest in the problem of real-time object detection from video streams. In particular, the sub-problems of face and pedestrian detection have been the focus of extensive research, due to the demand for face detection in low-power consumer electronics (e.g., cameras or smart-phones) and pedestrian detection in intelligent vehicles. In this section, we compare the performance of FCBoost cascades with those learned by several state of the art methods in the face and pedestrian detection literatures.

We start with face detection, where cascaded detectors have become predominant, comparing FCBoost to the method of Viola and Jones (VJ) (Viola and Jones, 2001), Wald boost (Sochman and Matas, 2005) and multi-exit (Pham et al., 2008). Since extensive results on these and other methods are available on the MIT-CMU test set, all detectors were evaluated on this data set. The methods above have been shown to outperform a number of other cascade learning algorithms (Pham et al., 2008) and, to the best of our knowledge, hold the best results in this data set. In all cases, the target detection rate was set to $D_T = 95\%$. For Wald boost, multi-exit, and VJ, the training set was bootstrapped when a new stage was added to the cascade, for FCBoost when the false positive rate dropped below 95%. For VJ and multi-exit cascades, which require the specification of the number of cascade stages and a target false-positive and detection rate per stage, we used 20 stages,

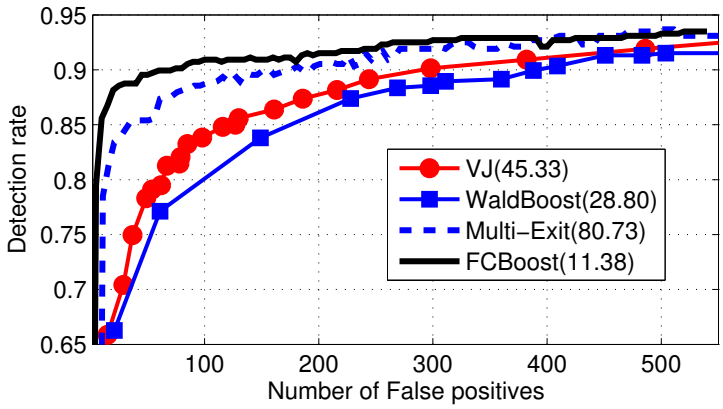


Figure 6: ROCs of various face detectors on MIT-CMU. The number in the legend is the average evaluation cost, i.e., average number of features evaluated per sub-window.

and the popular strategy of setting the false positive rate to 50% and the detection rate to $D_T^{\frac{1}{20}}$. For FCBoost we used a last-stage cascade, since this structure achieved the best balance between accuracy and speed in the previous experiment. We did not attempt to optimize η, simply using $\eta = 0.02$. The cost factor C was initialized with $C = 0.99$. If after a boosting update the cascade did not meet the detection rate, C was increased to

$$C_{new} = \frac{C_{old} + 1}{2}. \tag{65}$$

This placed more emphasis on avoiding misses than false positives, and was repeated until the updated cascade satisfied the rate constraint. The final value of C was used as the initial value for the next boosting update.

Figure 6 show the ROCs of all detectors. The average evaluation cost, i.e., average number of features evaluated per sub-window, is shown in the legend for each method. Note that the FCBoost cascade is simultaneously more accurate and faster than those of all other methods. For example, at 100 false positives, FCBoost has a detection rate of 91% as opposed to 88% for multi-exit, 83% for VJ, and 80% for Wald boost. With regards to computation, FCBoost is 7.1, 4, and 2.5 times faster than multi-exit, VJ, and Wald boost, respectively. Overall, when compared to the FCBoost cascade, the closest cascade in terms of detection rate (multi-exit, 3% drop) is significantly slower (7 times) and the closest cascade in terms of detection speed (Wald boost, 2.5 times slower) has a very poor detection rate (11% smaller).

We next considered the problem of pedestrian detection, comparing results to a large set of state-of-the-art pedestrian detectors on the Caltech Pedestrian data set (Dollár et al., 2012). In this literature, it is well known that a good representation for pedestrians must account for both edge orientation and color (Dalal and Triggs, 2005; Dollár et al., 2009). Similarly to Dollar *et al.* (Dollár et al., 2009), we adopted an image representation based on a 10 channel decomposition. This included 3 color channels (YUV color space), 6 gradient orientation channels, and a gradient magnitude channel. In all other aspects, the cascade

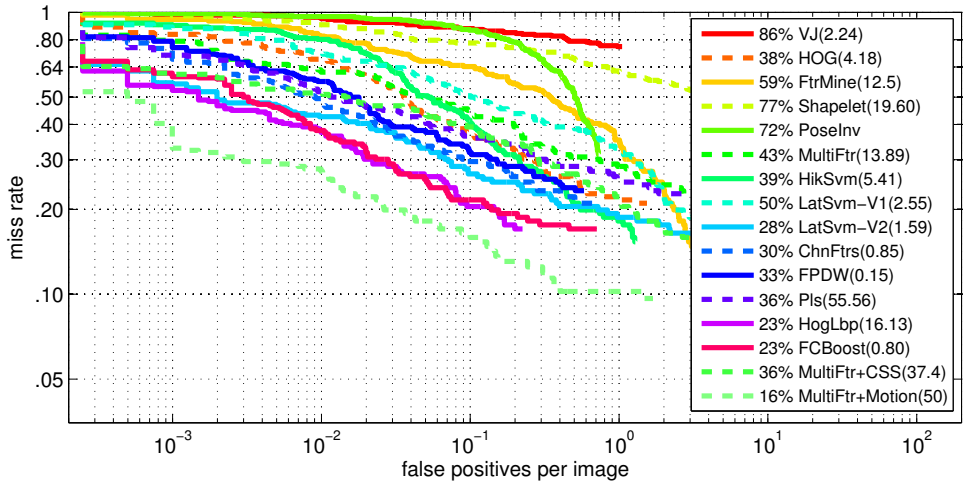


Figure 7: Accuracy curves and complexity of various pedestrian detectors on the Caltech data set. Legend: (left) miss rates at 0.1 FPPI, (right) average time, in seconds, required to process 480×640 frame.

architecture was as before, e.g., using Haar wavelet features and decision stumps as weak learners, the previously used values for parameters D_T, and η, etc. When compared to the face detection experiments, the only difference is that the set of weak learners was replicated for each channel. At each iteration, FCBoost chose the best weak learner and the best channel to add to the cascade predictor. The performance of the FCBoost cascade was evaluated with the toolbox of (Dollár et al., 2012). Figure 7 compares its complexity and curve of miss-detection rate vs number of false positives per image (FPPI) to those of a number of recent pedestrian detectors. The comparison was restricted to the popular near scale-large setting, which evaluates the detection of pedestrians with more than 100 pixels in height. The numbers shown in the left of the legend summarize the detection performance by the miss rate at 0.1 FPPI. The numbers shown in the right indicated the average time, in seconds, required for processing a 480×640 video frame. Note that the evaluation is *not* restricted to fast detectors, including the most popular architectures for object detection in computer vision, such as the HOG detector (Dalal and Triggs, 2005) or the latent SVM (Felzenszwalb et al., 2010). For more information on the curves and other methods the reader is referred to Dollár et al. (2012).

Two sets of conclusions can be derived from these results. First, they confirm the observation that the FCBoost cascade significantly outperforms previous cascaded detectors. A direct comparison is in fact possible against the ChnFtrs method (Dollár et al., 2009). This work introduced the multi channel features that we adopt but uses the SoftCascade algorithm (Bourdev and Brandt, 2005) for cascade learning. The resulting detector is among the top methods on this data set, missing 30% of the pedestrians at 0.1 FPPI and using 0.85 seconds to process a frame. Nevertheless, the FCBoost cascade has substantially better accuracy, missing only 23% of the pedestrians at 0.1 FPPI, and requires less time (a 6% speed up). Second, the results of Figure 7 show that the FCBoost cascade is one of the most accurate pedestrian detectors in the literature, and significantly faster than the detectors of

comparable accuracy. In fact, only two detectors have been reported to achieve equivalent or lower miss rates. The Hog-Lbp detector (Wang et al., 2009) has the same miss rate (23% at 0.1 FPPI) but is 20 times slower. The MultiFtr+Motion (Walk et al., 2010) detector has a smaller miss rate of 16% (at 0.1 FPPI) but is 62 times slower (almost 1 minute per frame). The inclusion of this method in Figure 7 is somewhat unfair, since it is the only approach that exploits motion features. All other detectors, including the FCBoost cascade, operate on single-frames. We did not investigate the impact of adding motion features to FCBoost. Finally, it should be noted that the FCBoost cascade could be enhanced with various computational speed ups proposed in the design of the FPDW detector (Dollár et al., 2010). This is basically a fast version of the ChnFtrs detector, using several image processing speed-ups to reduce the time necessary to produce the image channels on which the classifier operates. These speed-ups lead to a significant increase in speed (0.15 vs 0.85 seconds) at a marginal cost in terms of detection accuracy (33% vs. 30% miss rate at 0.1 FPPI). Since these enhancements are due to image processing, not better cascade design, we have not considered them in our implementation. We would expect, however, to see similar computational gains in result of their application to the FCBoost cascade.

8. Conclusions

In this work we have addressed the problem of detector cascade learning by introducing the FCBoost algorithm. This algorithm optimizes a Lagrangian risk that accounts for both detector speed and accuracy with respect to a predictor that complies with the sequential decision making structure of the cascade architecture. By exploiting recursive properties of the latter, it was shown that many cascade predictors can be derived from generator functions, which are cascade predictors of two stages. Variants of FCBoost were derived for two members of this family, last-stage and multiplicative cascades, which were shown to generalize the popular independent and embedded stage cascade architectures. The concept of neutral predictors was exploited to integrate the search for cascade configuration into the boosting algorithm. In result, FCBoost can automatically determine 1) the number of cascade stages and 2) the number of weak learners per stage, by minimizing the Lagrangian risk. It was also shown that FCBoost generalizes adaboost, and is compatible with existing cost-sensitive extensions of boosting. Hence, it can be used to learn cascades of high detection rate. Experimental evaluation has shown that the resulting cascades outperform current state-of-the-art methods in both detection accuracy and speed.

Acknowledgments

This work was supported by NSF grant (NSF IIS-1208522) and the Technology Development Program for Commercializing System Semiconductor funded By the Ministry of Trade, industry & Energy(MOTIE, Korea), [No. 10041126, Title: International Collaborative R&BD Project for System Semiconductor].

References

P. Bartlett and M. Traskin. Adaboost is consistent. *Journal of Machine Learning Research*, 8:2347–2368, December 2007.

L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 236–243, 2005.

S. Brubaker, M. Mullin, and J. Rehg. On the design of cascades of boosted ensembles for face detection. *International Journal of Computer Vision*, 77:65–86, 2008.

G. Carneiro, B. Georgescu, S. Good, and D. Comaniciu. Detection and measurement of fetal anatomies from ultrasound images using a constrained probabilistic boosting tree. *IEEE Transactions on Medical Imaging*, 27(9):1342 –1355, sept. 2008.

M. Collins, R. Schapire, and Y. Singer. Logistic regression, adaboost and Bregman distances. *Machine Learning*, 48(1-3):253–285, 2002.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *Proceedings of British Machine Vision Conference*, 2009.

P. Dollár, S. Belongie, and P. Perona. The fastest pedestrian detector in the west. In *Proceedings of British Machine Vision Conference*, 2010.

P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34 (4):743–761, 2012.

S. Du, N. Zheng, Q. You, Y. Wu, M. Yuan, and J. Wu. Rotated Haar-like features for face detection with in-plane rotation. In *Proceedings of international conference on Interactive Technologies and Sociotechnical Systems*, pages 128–137, 2006.

J. Duchi and Y. Singer. Boosting with structural sparsity. *Proceedings of the International Conference on Machine Learning*, pages 297–304, 2009.

M. Dundar and J. Bi. Joint optimization of cascaded classifiers for computer aided detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Comp. and Sys. Science*, 1997.

J. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 1999.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28, 1998.

C. Lampert. An efficient divide-and-conquer cascade for nonlinear object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1022 –1029, 2010.

C. Lampert, M. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 2129–2142, 2009.

L. Lefakis and F. Fleuret. Joint cascade optimization using a product of boosted classifiers. In *Proceedings of the Neural Information Processing Systems Conference*, 2010.

S. Li and Z. Zhang. Floatboost learning and statistical face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1112–1123, 2004.

R. Lienhart and J. Maydt. An extended set of Haar-like features for rapid object detection. In *Proceedings of International Conference on Image Processing*, pages I–900 – I–903 vol.1, 2002.

C. Liu and H. Shum. Kullback-Leibler boosting. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 587–594, 2003.

H. Luo. Optimization design of cascaded classifiers. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 480–485, 2005.

H. Masnadi-Shirazi and N. Vasconcelos. High detection-rate cascades for real-time object detection. In *Proceedings of International Conference on Computer Vision*, volume 2, pages 1–6, 2007.

H. Masnadi-Shirazi and N. Vasconcelos. Cost-sensitive boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99, 2010.

L. Mason, J. Baxter, P. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. *Advances in Large Margin Classifiers*, pages 1221–246, 2000.

D. Mease and A. Wyner. Evidence contrary to the statistical view of boosting. *Journal of Machine Learning Research*, 9:131–156, 2008.

C. Messom and A. Barczak. Fast and efficient rotated haar-like features using rotated integral images. In *Proceedings of the Australasian Conference on Robotics and Automation*, 2006.

M. Pham and T. Cham. Fast training and selection of Haar features using statistics in boosting-based face detection. In *Proceedings of IEEE International Conference on Computer Vision*, pages 1–7, 2007.

M. Pham, V. Hoang, and T. Cham. Detection with multi-exit asymmetric boosting. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1 – 8, 2008.

M. Pham, Y. Gao, V. Hoang, and T. Cham. Fast polygonal integration and its application in extending Haar-like features to improve object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 942 –949, 2010.

F. Porikli. Integral histogram: A fast way to extract histograms in Cartesian spaces. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 829 – 836, 2005.

V. Raykar, B. Krishnapuram, and S. Yu. Designing efficient cascaded classifiers: Tradeoff between accuracy and cost. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 853–860, 2010.

D. Rumelhart, G. Hinton, and R. Williams. Example based learning for view-based human face detection. *Nature*, pages 533–536, 1968.

M. Saberian and N. Vasconcelos. Boosting classifer cascades. In *Proceedings of the Neural Information Processing Systems Conference*, 2010.

R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

H. Schneiderman. Feature-centric evaluation for efficient cascaded object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 29–36, 2004.

C. Shen, P. Wang, and H. Li. Lacboost and fisherboost: optimally building cascade classifiers. In *Proceedings of European Conference on Computer Vision*, pages 608–621, 2010.

C. Shen, S. Paisitkriangkrai, and J. Zhang. Efficiently learning a detection cascade with sparse eigenvectors. *IEEE Transactions on Image Processing*, 20(1):22 –35, jan. 2011.

J. Sochman and J. Matas. Waldboost - learning for time constrained sequential detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 150–157, 2005.

J. Sun, J. Rehg, and A. Bobick. Automatic cascade training with perturbation bias. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 276–283, 2004.

K. Kay Sung and T. Poggio. Example based learning for view-based human face detection. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 20:39–51, 1998.

O. Tuzel, F. Porikli, and P. Meer. Pedestrian detection via classification on Riemannian manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1713 –1727, 2008.

S. Vijayanarasimhan and K. Grauman. Efficient region search for object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1401 –1408, 2011.

P. Viola and M. Jones. Robust real-time object detection. *Workshop on Statistical and Computational Theories of Vision*, 2001.

P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In *Proceedings of the Neural Information Processing Systems Conference*, pages 1311–1318, 2002.

S. Walk, N. Majer, K. Schindler, and B. Schiele. New features and insights for pedestrian detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1030–1037, 2010.

X. Wang, T. Han, and S. Yan. An HOG-LBP human detector with partial occlusion handling. In *Proceedings of IEEE International Conference on Computer Vision*, pages 32–39, 2009.

J. Wu, S. Brubaker, M. Mullin, and J. Rehg. Fast asymmetric learning for cascade face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3:369–382, 2008.

R. Xiao, L. Zhu, and H. Zhang. Boosting chain learning for object detection. In *Proceedings of International Conference on Computer Vision*, pages 709 –71, 2003.

R. Xiao, H. Zhu, H. Sun, and X. Tang. Dynamic cascades for face detection. In *Proceedings of International Conference on Computer Vision*, pages 1 – 8, 2007.

T. Zhang. Adaptive forward-backward greedy algorithm for learning sparse representations. *IEEE Transactions on Information Theory*, 57(7):4689 –4708, july 2011.

Q. Zhu, M. Yeh, K. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1491 – 1498, 2006.

Journal of Machine Learning Research 15 (2014) 2607-2627 Submitted 1/13; Revised 3/14; Published 7/14

Efficient and Accurate Methods for Updating Generalized Linear Models with Multiple Feature Additions

Amit Dhurandhar ADHURAN@US.IBM.COM
Marek Petrik MPETRIK@US.IBM.COM
IBM TJ Watson,
1101 Kitchawan Road, Yorktown Heights, NY 10598 USA

Editor: Charles Elkan

Abstract

In this paper, we propose an approach for learning regression models efficiently in an environment where multiple features and data-points are added incrementally in a multi-step process. At each step, any finite number of features maybe added and hence, the setting is not amenable to low rank updates. We show that our approach is not only efficient and optimal for ordinary least squares, weighted least squares, generalized least squares and ridge regression, but also more generally for generalized linear models and lasso regression that use iterated re-weighted least squares for maximum likelihood estimation. Our approach instantiated to linear settings has close relations to the partitioned matrix inversion mechanism based on Schur's complement. For arbitrary regression methods, even a relaxation of the approach is no worse than using the model from the previous step or using a model that learns on the additional features and optimizes the residual of the model at the previous step. Such problems are commonplace in complex manufacturing operations consisting of hundreds of steps, where multiple measurements are taken at each step to monitor the quality of the final product. Accurately predicting if the finished product will meet specifications at each or, at least, important intermediate steps can be extremely useful in enhancing productivity. We further validate our claims through experiments on synthetic and real industrial data sets.

Keywords: linear regression, logistic regressions, lasso, group lasso, feature selection, manufacturing

1. Introduction

Complex manufacturing is a multi-billion dollar industry, which encompasses diverse domains ranging from semiconductor to pharmaceutical to consumer and snack products. Each of these industries undertake complex operations that consist of hundreds of steps from the beginning to the very end when the final product is produced. In each of these steps, multiple measurements are made to monitor the process. Thus, every product has an incremental history of measurements accumulated over time. The hope is that the final finished product will meet the necessary specifications. However, in each of these industries, there are millions (to even billions) of dollars of losses every year because of out-of-spec products. It would thus be extremely useful if one could efficiently utilize the accrued intermediate measurements or features to accurately predict the quality of the final product. For example, in the semiconductor industry, wafers—which are a collection of chips—travel

together in groups called lots through hundreds of processing steps with thousands of measurements being accrued as the wafers reach the end. At this point, the quality of the wafers is determined by either checking their speed or power. It can be extremely useful if one could, at least at critical intermediate steps, provide an accurate estimate of the final speed so as to possibly take corrective actions or avoid further processing.

In particular, the data flows through K steps as groups of data-points or a batch with the target becoming available almost simultaneously for each member of the group at the very end. We want to efficiently build a model at each step based on this data so as to estimate the performance of forthcoming batches at these steps. There are multiple such batches that flow through the processing steps and hence, we also want to update our model at each step based on recently acquired data. Therefore, in our problem, both features and data-points are added as batches move through the process and as new batches are created. The notation used throughout the paper is given in Table 1 and an illustration of our problem setting is shown in Figure 1.

The methods we propose for incremental updates with new feature additions are by no means constrained only to manufacturing. They also are applicable to feature selection algorithms such as least angle regression (Efron et al., 2004), the homotopy method for lasso (Tibshirani and Taylor, 2011), or orthogonal matching pursuit (Hastie et al., 2009). These feature selection methods must efficiently update the model whenever a new feature is added. The most common approach is to keep a QR factorization of the regression matrix. This factorization can be updated with every new feature added using either Givens rotations or Householder reflections, which are both computationally efficient and numerically stable (Golub and Loan, 1996).

The problem we study differs from the typical feature selection setting in two important aspects. First, while most feature selection methods add one feature at a time, our method is more suitable when many features are to be added at a time. Second, the QR factorization can only be used with linear regressors, while our method also works with generalized linear models. As a result, our methods can also be applied in more sophisticated settings, such as to efficiently update regression models in non-linear group variable selection techniques (Lozano et al., 2009, 2011).

Our method when instantiated to the linear regression case, where we minimize a least squares objective, is closely related to the partitioned matrix inverse mechanism based on Schur's complement (Boyd and Vandenberghe, 2004). Although both absolve the need for computing matrix inverses over the entire currently available feature set, our procedure has an intuitive explanation. Moreover, our method is a meta-technique applicable to any base regression method, and as you will see later, is an essential component in optimally updating generalized linear models such as logistic regression, which as we know are non-linear.

In the rest of the paper, we first formally define our problem. We then propose a novel algorithm, which we show to be optimal for generalized least squares and ridge regression. We discuss its relation to the partitioned matrix inversion mechanism. We provide an extension of our algorithm, which is efficient and optimal for generalized linear models. We provide a relaxation of our algorithm, which can be used to efficiently update other regression techniques and whose performance is no worse than using the model from the previous step or using a model that learns on the additional features and optimizes the residual of the model at the previous step.

Term/Symbol	Description
Step/Epoch	The instant when new features become available.
Batch	A set of data points that move through all the steps.
d_i	Number of features accumulated till (and including) step i.
X_i^b,x_i^b	Input data from batch b with d_i and $(d_i - d_{i-1})$ features respectively.
M_i^b,m_i^b	Model built on the first d_i features with all the data till batch b and only data in batch b respectively.
Y^b	Outputs corresponding to batch b.
$\hat{Y}_q^p$,R_q^p	Matrix of predictions and the matrix of residuals respectively, obtained by regressing inputs q on columns of p.

Table 1: The notation used in the paper. In much of the paper we drop the batch b from the superscript since, we propose methods for efficient updates to models with feature additions that are applicable to any batch.

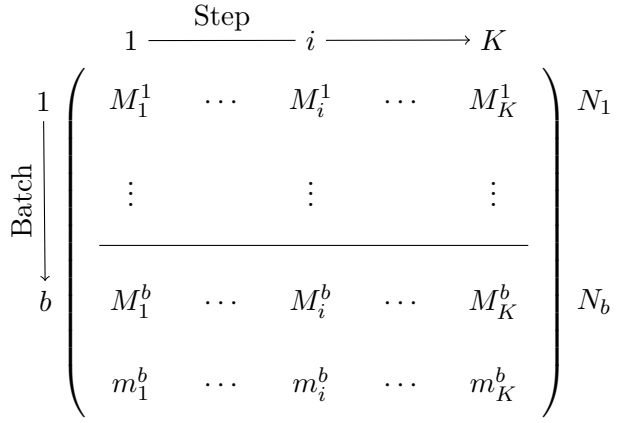


Figure 1: Process where features and data-points are added, with new models being learned at each step.

2. Problem Statement

Before describing the problem statement, we will introduce some notation. Let K denote the number of steps in the process. Let d_i denote the number of features $F_i = \{f_1, ..., f_{d_i}\}$ present at step i. It is important to note that $F_i \supset F_{i-1}, \forall i \in \{2, ..., K\}$ which implies $d_i > d_{i-1}, \forall i \in \{2, ..., K\}$. There are multiple batches that flow through the K steps and we denote the size of one such batch b, by N_b. Based on a particular learning algorithm, we denote the model learned at step i trained on data from batch b, by m_i^b (local model). Let M_i^b (global model) denote the model based on all the data accumulated till batch b in step i. For efficiency reasons, though we may not learn from scratch in step i using all the available data, M_i^b is obtained by potentially learning over a sample of size $\sum_{j=1}^{b} N_j$. Thus,

m_i^b is a local model learned over just recent data, while M_i^b is the model we will use to predict the outcomes of the data points in the $(b+1)^{th}$ batch when they reach step i. Note that for batch 1, the local and global models are the same, i.e., $m_i^1 = M_i^1, \forall i \in \{1, ..., K\}$. A pictorial representation of the process with notation is shown in Figure 1.

Let X_i^b ($N_b \times d_i$ matrix) and x_i^b ($N_b \times (d_i - d_{i-1})$ matrix) denote all the input data available[1] in step i batch b and the input data for only the additional features in step i batch b, i.e., $\{f_{d_{i-1}+1}, ..., f_{d_i}\}$ respectively. Let Y^b denote the final outcomes or target in batch b. Let $\hat{Y}_q^p$ and R_q^p denote the matrix of predictions and the matrix of residuals respectively, obtained by regressing inputs q on columns of p based on a chosen regression technique. Thus, if p is a matrix, each of its columns is considered a target and q is regressed separately on each of them. Consequently, $\hat{Y}_{X_i^b}^Y$ and $R_{X_i^b}^Y$ are the predictions and residuals of the model m_i^b respectively. Let I_j denote an identity matrix of rank j.

Given our dual goal of efficiently learning with i) new features and ii) with new data points being added, we address each of these issues in isolation. With this, we have the following two problems that we need to tackle:

I During any batch b of the K step process, given a model at step $i \in \{1, ..., K-1\}$ learned over d_i features $\{f_1, ..., f_{d_i}\}$ and a sample size N_b, how do we update this model at step $i+1$ with d_{i+1} features $\{f_1, ..., f_{d_i}, ..., f_{d_{i+1}}\}$ such that the resultant model is (a) no less accurate than the model at step i? (b) no less accurate than the composite model which consists of the model learned using only the additional features $\{f_{d_i+1}, ..., f_{d_{i+1}}\}$ on the residuals of the model at step i? and (c) more efficient to learn than learning from scratch over all the features available at step $i+1$ i.e., $\{f_1, ..., f_{d_{i+1}}\}$?

II At any step i in the K step process, given the model M_i^b and the model m_i^{b+1}, how do we efficiently obtain the model M_i^{b+1}?

Though not completely solved, there has been extensive work to handle Question II (Blum, 1996; Smale and Yao, 2005; Bottou and Cun, 2003). We thus focus our attention on question I. Given this and for simplicity of notation, from here on, we do not refer to the batch any more; that is, we drop b from the notation, since the methods we describe are applicable to any batch in the process.

3. Methodology

In this section, we first suggest a meta-algorithm to successfully tackle question I. We then show that not only can this algorithm be realized efficiently for ordinary least squares, weighted least squares, generalized least squares and ridge regression, but it is also optimal for these techniques. We also show that the algorithm can be used as a core function to efficiently and optimally solve iterated re-weighted least squares procedures, which are used to find the maximum likelihood estimates for Generalized Linear Models (McCullagh and Nelder, 1990) and sometimes even Lasso (Tibshirani, 1994). For arbitrary regression algorithms, it can be shown that even a relaxation of our technique results in a positive response to questions I(a), I(b) and I(c). We refer to the optimal model in any step that

1. This includes the constant term.

Algorithm 1 Proposed meta-algorithm—which can be embedded as a core function in other algorithms—to update model built in step i during the next step $i+1$. ζ_i are other regression algorithm specific parameters.

Input: m_i, $R^Y_{X_i}$, X_{i+1}, Y and ζ_i.
Output: m_{i+1}, $R^Y_{X_{i+1}}$ and ζ_{i+1}
Compute $R^{x_{i+1}}_{X_i}$ {Use X_i to predict columns of x_{i+1} and compute the residual matrix.}
Regress $R^{x_{i+1}}_{X_i}$ on $R^Y_{X_i} \rightarrow \bar{m}_{i+1}$ {Regressing the residuals outputs the model on the right.}
Regress X_i on $Y - \hat{Y}^{R^Y_{X_i}}_{R^{x_{i+1}}_{X_i}} \rightarrow \hat{m}_i$ {Regressing the input of step i with the previous residual yields the model on the right.}
$m_{i+1} = (\hat{m}_i; \bar{m}_{i+1})$ {Composing these models, i.e., for example concatenating their parameter vectors, gives the final model.}
if $i+1 == K$ **then**
  {If $i+1$ is the last step.}
  Return m_{i+1}
else
  Return m_{i+1}, $R^Y_{X_{i+1}}$ and ζ_{i+1}
end if

is learned from scratch by method Standard. We refer to the model that learns on the **a**dditional **f**eatures and **o**ptimizes the **r**esidue of a model at the previous step by method AFOR.

Algorithm 1 can be described as follows: First, we find the portion of the target that was not modeled by the regression algorithm (residuals) in the previous step. We then try to find the additional information that the new features in the current step contain. Using this additional information we fit to the residuals in the previous step. The residuals from this model are subtracted from the original target and the features in the previous step are fit to this modified target. The final model is a composition of these two latter models.

Loosely speaking, the intuition behind Algorithm 1 is to find what additional benefit the new features bring us in predicting the portion of the target that was not modeled by the previously available set of features, and after the removal of their effect, we focus on modeling this new modified target using the previously available set of features. This procedure thus uses the old and new set of features to model, as much as possible, the parts of the target that are not explained by the other, resulting in significantly reduced redundancy in the final model.

We now see how effectively this method can be applied to different regression algorithms. A comparison of the different approaches using different regression methods is shown in Table 2. The computational complexity of each algorithm depends on the method used to invert the matrices. The most suitable method depends on the size and sparsity of the matrix (Golub and Loan, 1996; Saad, 1997). The fastest known algorithms have a time complexity of $O(d^{2.3})$ but are generally not useful in practice because of a large multiplicative constant (Coppersmith and Winograd, 1990).

	OLS	WLS	GLS	Ridge	IRLS (for GLM and Lasso)
	Efficiency, Optimality				
Our Method	$(d_{i+1} - d_i) \times (d_{i+1} - d_i)$ inverse, Optimal				Smaller problem size, Optimal
Standard	$d_{i+1} \times d_{i+1}$ inverse, Optimal				Optimal
AFOR	$(d_{i+1} - d_i) \times (d_{i+1} - d_i)$ inverse, Sub-optimal				Sub-optimal

Table 2: Comparison of three meta-approaches across different regression algorithms in step $i+1$. Standard denotes learning from scratch.

3.1 Generalized Least Squares and Ridge Regression

In least squares regression (Hastie et al., 2009), we minimize a quadratic loss function wherein each datapoint may be weighted or unweighted (i.e., equi-weighted). Based on the weighting scheme, we have the following three variants with each successive variant being a generalization of the ones before. Ordinary Least Squares (OLS) is the unweighted variant. Weighted Least Squares (WLS) is the variant, where the weight matrix is diagonal. Generalized Least Squares (GLS) is the most general variant, wherein we have a full weight matrix W. Ridge regression on the other hand is regularized OLS with a quadratic penalty (Hoerl and Kennard, 1970).

We now instantiate Algorithm 1 for GLS with a quadratic penalty. We could refer to this method as Generalized Ridge (GR) regression. Eliminating the penalty term or setting W to the identity matrix in GR would give us instantiations for GLS and ridge regression respectively.

The objective function minimized in step i for GR is

$$Q_{gr} = (Y - X_i\beta_i)^T W (Y - X_i\beta_i) + \lambda {\beta_i}^T \beta_i,$$

where β_i is the parameter vector we wish to estimate[2] and $\lambda > 0$ is the regularization parameter. The optimal β_i is given by $m_i = ({X_i}^T W X_i + \lambda I_{d_i})^{-1} {X_i}^T W Y$, where I_{d_i} is a $d_i \times d_i$ identity matrix. To estimate β_{i+1} we use Algorithm 1 as follows:

- $R_{X_i}^{x_{i+1}} = (I_N - H_i) x_{i+1}$. Here $\zeta_i = H_i = X_i({X_i}^T W X_i + \lambda I_{d_i})^{-1} {X_i}^T W$.
- $\bar{m}_{i+1} = ({x_{i+1}}^T W (I_N - H_i) x_{i+1} + \lambda I_{(d_{i+1} - d_i)})^{-1} {x_{i+1}}^T (I_N - H_i) Y$. This is a consequence of the fact that H_i is idempotent.
- $\hat{m}_i = ({X_i}^T W X_i + \lambda I_{d_i})^{-1} {X_i}^T W (Y - x_{i+1} \bar{m}_{i+1}) = m_i - \alpha_i \bar{m}_{i+1}$, where $\alpha_i = ({X_i}^T W X_i + \lambda I_{d_i})^{-1} {X_i}^T W x_{i+1}$.
- $m_{i+1} = (\hat{m}_i; \bar{m}_{i+1})$. Now m_{i+1} can be immediately used to obtain predictions for new batches whose target is currently unknown and have reached step $i+1$.
- If $i + 1 < K$, then return m_{i+1}, $R_{X_{i+1}}^Y$ and H_{i+1}. Here H_{i+1}—which also determines $({X_{i+1}}^T W X_{i+1} + \lambda I_{d_{i+1}})^{-1}$—can be computed from $({X_i}^T W X_i + \lambda I_{d_i})^{-1}$ and

2. For simplicity of notation assume the data has zero intercept or the constant term is included in X_i.

$({x_{i+1}}^T W(I_N - H_i)x_{i+1} + \lambda I_{(d_{i+1}-d_i)})^{-1}$ using standard partitioned matrix inversion property with only matrix multiplications but no more inversions (Boyd and Vandenberghe, 2004).

Let us now see why this method is more efficient than performing regression from scratch in step i+1. If we performed regression from scratch we would have had to invert a $d_{i+1}\times d_{i+1}$ matrix. In our case however, we have to invert only a $(d_{i+1} - d_i) \times (d_{i+1} - d_i)$ matrix in step $i + 1$. The reason for this is that $({X_i}^T W X_i + \lambda I_{d_i})^{-1}$, which is a $d_i \times d_i$ matrix, is already available from step i. Moreover, our method is also more efficient at obtaining the current step model than applying the partitioned matrix inversion property based on Schur's complement directly, because although no more inversions are required by it there are many more matrix multiplications that have to be performed.

For instance, based on Schur's complement and the partitioned matrix inversion mechanism we would compute the optimal estimate for β_{i+1} denoted by β_{i+1}^{opt} as follows

$$\begin{aligned}\beta_{i+1}^{opt} &= \begin{bmatrix} {X_i}^T W X_i + \lambda I_{d_i} & {X_i}^T W x_{i+1} \\ x_{i+1}^T W X_i & x_{i+1}^T W x_{i+1} + \lambda I_{(d_{i+1}-d_i)} \end{bmatrix}^{-1} \begin{pmatrix} {X_i}^T \\ x_{i+1}^T \end{pmatrix} Y \\ &= \begin{bmatrix} A^{-1} + A^{-1}UC^{-1}VA^{-1} & -A^{-1}UC^{-1} \\ -C^{-1}VA^{-1} & C^{-1} \end{bmatrix} \begin{pmatrix} {X_i}^T \\ x_{i+1}^T \end{pmatrix} Y \end{aligned}, \tag{1}$$

where $A = {X_i}^T W X_i + \lambda I_{d_i}$, $U = {X_i}^T W x_{i+1}$, $V = x_{i+1}^T W X_i$ and $C = x_{i+1}^T W(I_N - H_i)x_{i+1} + \lambda I_{(d_{i+1}-d_i)}$. Here C is the Schur's complement.

It turns out that our estimate $m_{i+1} = \beta_{i+1}^{opt}$ and is thus optimal for GR regression as seen below.

Theorem 1 *If we use GR regression in our process, then in any step $i + 1$, where $i \in 1, ..., K - 1$, m_{i+1} is the optimal GR estimator.*

Proof Equation (1) can be expanded and rewritten as follows

$$\begin{aligned}\beta_{i+1}^{opt} &= \begin{bmatrix} A^{-1} + A^{-1}{X_i}^T W x_{i+1} C^{-1} x_{i+1}^T W X_i A^{-1} & -A^{-1}{X_i}^T W x_{i+1} C^{-1} \\ -C^{-1} x_{i+1}^T W X_i A^{-1} & C^{-1} \end{bmatrix} \begin{pmatrix} {X_i}^T \\ x_{i+1}^T \end{pmatrix} Y \\ &= \begin{pmatrix} A^{-1}{X_i}^T W(Y - A^{-1}{X_i}^T W {x_{i+1}}^T C^{-1} {x_{i+1}}^T (I_N - H_i)Y) \\ C^{-1}{x_{i+1}}^T (I_N - H_i)Y \end{pmatrix} \\ &= \begin{pmatrix} ({X_i}^T W X_i + \lambda I_{d_i})^{-1}{X_i}^T W(Y - x_{i+1}\bar{m}_{i+1}) \\ ({x_{i+1}}^T W(I_N - H_i)x_{i+1} + \lambda I_{(d_{i+1}-d_i)})^{-1}{x_{i+1}}^T (I_N - H_i)Y \end{pmatrix} \\ &= \begin{pmatrix} \hat{m}_i \\ \bar{m}_{i+1} \end{pmatrix} \\ &= m_{i+1}\end{aligned}.$$

■

Remark 1 *The result in Theorem 1 implies that using Algorithm 1, we get efficient and optimal estimates for OLS, WLS, GLS and ridge regression in step $i + 1$.*

Algorithm 2 Method to update model built using IRLS in step i during the next step $i+1$, using Algorithm 1.

Input: m_i or β_i, $R_{X_i}^{z_i^{t_i}}$, X_{i+1}, Y, $z_i^{t_i}$, $H_i^{t_i}$, $W_i^{t_i}$, $\epsilon_{\mathcal{D}} \in (0,1]$ and $\epsilon_\beta \geq 0$.

Output: m_{i+1} or β_{i+1}, $R_{X_{i+1}}^{z_{i+1}^{t_{i+1}}}$ and $H_{i+1}^{t_{i+1}}$

Run Algorithm 1 with inputs m_i, $R_{X_i}^{z_i^{t_i}}$, X_{i+1}, $z_i^{t_i}$, $H_i^{t_i}$ and $W_i^{t_i}$
if $P[\mathcal{D}_{i+1} > -2(\mathcal{L}(m_{i+1}|X_{i+1},Y) - \mathcal{L})] \leq \epsilon_{\mathcal{D}}$ **then**
  {Checking deviance, where $\mathcal{L}$ is the max possible log-likelihood value.}
  $z_{i+1}^{t_{i+1}} = z_{i+1}^{t_i}$, $W_{i+1}^{t_{i+1}} = W_i^{t_i}$
else if $max_{j\in\{1,\ldots,d_i\}}|m_{i+1}(j) - m_i(j)| \leq \epsilon_\beta$ **then**
  Let $\hat{m}_{i+1} = (m_{i+1}(1), ..., m_{i+1}(d_i))^T$
  Run IRLS only on x_{i+1} from this point onward with $\hat{m}_{i+1}$ fixed
  Let $\bar{m}_{i+1}$ be the optimal solution of the above IRLS run
  $m_{i+1} = (\hat{m}_{i+1}; \bar{m}_{i+1})$
else
  Run IRLS using output from Algorithm 1 i.e., m_{i+1}, $R_{X_{i+1}}^{z_{i+1}^{t_i}}$ and $H_{i+1}^{t_i}$
end if
Return m_{i+1}, $R_{X_{i+1}}^{z_{i+1}^{t_{i+1}}}$ and $H_{i+1}^{t_{i+1}}$

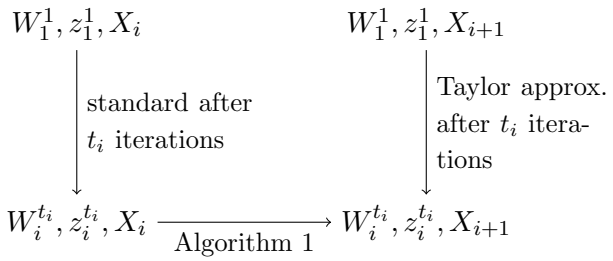


Figure 2: The figure shows where Algorithm 1 can be used to efficiently transform the optimal solution in step i to a solution in step $i+1$.

3.2 Generalized Linear Models and Lasso using IRLS

Generalized Linear Models (GLMs) assume that the target is generated from certain specific distributions belonging to the exponential family. Normal, Poisson, binomial, gamma and exponential are some of the distributions that are considered. The target is related to a linear combination of the inputs through a linear or non-linear function called the link function. Formally, if we learned a GLM in step i, we would have the following relationship

$$E[Y] = \hat{Y}_{X_i}^Y = g^{-1}(X_i\beta_i),$$

where $g(.)$ is the link function. For a normal distribution $g(.)$ is identity and the resultant regression method is just OLS. For a binomial $g(.)$ is the logit function and if values of the target lie in $[0,1]$, then the resultant regression method is logistic regression.

Lasso (Tibshirani, 1994) is L1 regularized OLS. It is usually used to solve under-determined systems of equations, where we have more features than data points. The idea here is to avoid over-fitting and choose features that are truly predictive. Formally, a lasso in step i would minimize the following objective

$$Q_L = (Y - X_i\beta_i)^T(Y - X_i\beta_i) + \lambda|\beta_i|^1.$$

3.2.1 Iteratively Re-weighted Least Squares

Both of the above classes of regression methods can be solved using iterative procedures. Iteratively Re-weighted Least Squares (IRLS) is a popular technique used to find the maximum likelihood estimates (MLE) for GLMs. Although there are other preferred methods to solve lasso, IRLS is still an effective method in this context. IRLS usually uses Newton Raphson updates, where the updated predictions at each iteration determine the weight matrix and the target in the next iteration. In particular, if $W_i^{t_j}$ and $z_i^{t_j}$ are the weight matrix and target in step i, and in the t_j^{th} iteration respectively, then the weight matrix is a $N \times N$ diagonal matrix whose diagonal elements correspond to the reciprocal of the variances computed from predictions in the previous iteration $(t_j - 1)$. $z_i^{t_j}$ on the other hand is given by $X_i\beta_i^{t_j-1} + W_i^{t_j}(Y - \hat{Y}_{X_i}^{z_i^{t_j-1}})$.

On the left hand portion of Figure 2, we see that in step i if we run IRLS, we get the optimal solution after t_i iterations. This optimal solution corresponds to an optimal weight matrix $W_i^{t_i}$ and an optimal target $z_i^{t_j}$. Using this weight matrix and target we can efficiently obtain the corresponding WLS solution in step $i+1$ using Algorithm 1. This solution relates to performing IRLS in step $i+1$, where at each iteration we approximate (zeroth order) the predictions around $R_{X_i}^{x_{i+1}}x_{i+1}$ using Taylor expansion

$$\hat{Y}_{X_{i+1}}^{z_{i+1}^{t_j}} = \hat{Y}_{X_i}^{z_i^{t_j}} + R_{X_i}^{x_{i+1}}x_{i+1}s(X_{i+1}, z_{i+1}^{t_j}),$$

where $s(.)$ is a function denoting higher order terms. Hence, at each iteration if we take the zeroth order approximation, then the WLS problem solved has the same weight matrix and target as that solved in step i at the same iteration. This is depicted on the right hand side of Figure 2.

3.2.2 Understanding Algorithm 2

In Algorithm 2, we first use Algorithm 1 to find the optimal solution in the current step $i+1$ of the WLS problem at iteration t_i in step i. The weight matrix and target in this WLS problem correspond to the optimal IRLS solution in step i. We then check to see if the probability of deviance $\mathcal{D}_{i+1}$ (McCullagh and Nelder, 1990) being greater than twice the difference between the max possible log-likelihood value (i.e., if $\hat{Y}_{X_{i+1}}^{Y} = Y$) and the log-likelihood of the current solution is less than a small constant $\epsilon_{\mathcal{D}}$. Deviance is a statistic that measures goodness of fit. For any step i, $\mathcal{D}_i = -2(\mathcal{L}(m_i|X_i, Y) - \mathcal{L})$, where m_i denotes a corresponding optimal model. Asymptotically, $\mathcal{D}_i$ has a χ^2 distribution with $N - d_i$ degrees of freedom. Using this fact we can compute the required probability in Algorithm 2 and check for the specified condition. A small value of the probability indicates that we are already near an optimal solution and hence, we have a satisfactory solution.

If this condition is not satisfied, we then check to see what is the maximum change in the d_i parameter values going from the optimal IRLS solution in step i to our current solution in step $i+1$. If the maximum change is small, i.e., $\leq \epsilon_\beta$, we fix these parameters and only run IRLS on the remaining set. A small change in the previous step parameters indicates that the new set of features in step $i+1$ are practically orthogonal to step i features and hence, the previous step parameters will change little as we approach the optimal solution in the current step.

If neither of the above conditions are satisfied we simply run IRLS, starting at the current solution.

3.2.3 Analysis

It is easy to see that if the first condition in Algorithm 2 is satisfied, then we have reached our desired solution and hence, this is definitely more efficient than starting from the beginning. The question is, are we doing better in the other two cases, namely, when only the second condition is satisfied or when neither condition is satisfied? The latter scenario is probably worse since, we are solving at each iteration a WLS problem with d_{i+1} features rather than $d_{i+1} - d_i$ features. We cannot claim for certain that these two scenarios are more efficient than learning afresh in step $i+1$, but the following two results along with experiments on real industrial data lead us to strongly believe that this is the case.

As the log-likelihood of the predictions increases after each iteration in step i, the variances and hence, the weight matrix approach the true weight matrix. Hence, the optimal weight matrix in step i represents the variances much more accurately than starting with the default, which is uniform weights.

Remark 2 *Using the optimal weight matrix and target of step i as a starting point for step $i+1$ has a higher log-likelihood solution than starting with the defaults, i.e., uniform weight matrix and the original target (Pregibon, 1981; Wang, 1987; McCullagh and Nelder, 1990).*

Based on Remark 2, one might ask, does starting from a higher likelihood point guarantee us faster convergence or fewer iterations? Loosely speaking, in the proposition that follows, we show that better initialization leads to no worse a solution after the same number of iterations.

Proposition 1 *When using IRLS, if two feasible points β_1 and β_2 are in the neighborhood of the same local optimum γ of a log-likelihood function where the Hessians exist and the first partial derivatives are non-zero, with $\mathcal{L}(\beta_1|X,Y) \geq \mathcal{L}(\beta_2|X,Y)$, then a tight upper bound on the distance of the solutions starting at each of these points to γ after t iterations has the following relationship, $\eta_t(\beta_1) \leq \eta_t(\beta_2)$, where $\eta_t(.)$ is a function that takes as input the starting point and outputs the required upper bound on the distance at iteration t.*

Proof Let γ be the root of the log-likelihood function $\mathcal{L}(.)$. The log-likelihood is always conditioned on inputs and outputs but for notational conciseness we do not repeatedly write this here. Let β_j^t denote the solution of IRLS at iteration t starting from the initial point β_j. Thus, by Taylor expansion starting from β_1 at iteration t we have

$$\mathcal{L}(\gamma) = \mathcal{L}(\beta_1^t) + \bigtriangledown\mathcal{L}(\beta_1^t)^T(\gamma - \beta_1^t) + \frac{1}{2}(\gamma - \beta_1^t)^T F(\beta_1^t)(\gamma - \beta_1^t),$$

where $\bigtriangledown$ denotes the gradient and $F(.)$ denotes the hessian. Since, γ is the root of the above function and after pre-multiplying by the inverse of the Jacobian at β_1^t we have

$$\begin{aligned}\gamma - \beta_1^t + J^{-1}(\beta_1^t)\mathcal{L}(\beta_1^t) &= -\frac{1}{2}J^{-1}(\beta_1^t)(\gamma - \beta_j^t)^T F(\beta_1^t)(\gamma - \beta_1^t) \\ \gamma - \beta_1^{t+1} &= -\frac{1}{2}J^{-1}(\beta_1^t)(\gamma - \beta_j^t)^T F(\beta_1^t)(\gamma - \beta_1^t)\end{aligned},$$

since, $\beta_1^{t+1} = \beta_1^t - J^{-1}(\beta_1^t)\mathcal{L}(\beta_1^t)$. Now if we take norm on both sides we get

$$|\gamma - \beta_1^{t+1}| = \frac{1}{2}|J^{-1}(\beta_1^t)F(\beta_1^t)||\gamma - \beta_1^t|^2.$$

Let $\Delta_1^t = |\gamma - \beta_1^t|$. Thus

$$\Delta_1^{t+1} = \frac{1}{2}|J^{-1}(\beta_1^t)F(\beta_1^t)|{\Delta_1^t}^2.$$

Let $\mathcal{N}_j$ denote the neighborhood around γ such that $\mathcal{N}_j = \{\beta|\mathcal{L}(\beta) \geq \mathcal{L}(\beta_j)\}$. Let $u_j = sup_{\beta\in\mathcal{N}_j}\frac{1}{2}|J^{-1}(\beta)F(\beta)|$. We thus have

$$\begin{aligned}\Delta_1^{t+1} &\leq u_1{\Delta_1^t}^2 \\ &\leq u_1^t{\Delta_1^1}^2 = \eta_{t+1}(\beta_1)\end{aligned},$$

where $\eta_{t+1}(\beta_1) = u_1^t{\Delta_1^1}^2$. Similarly, if we started at β_2 we would have

$$\Delta_2^{t+1} \leq \eta_{t+1}(\beta_2).$$

Now if $\mathcal{L}(\beta_1) \geq \mathcal{L}(\beta_2)$, then $\mathcal{N}_1 \subset \mathcal{N}_2$ and hence, $u_1 \leq u_2$. Moreover, $\Delta_1^1 \leq \Delta_2^1$. Based on these two facts we would have

$$\eta_{t+1}(\beta_1) \leq \eta_{t+1}(\beta_2).$$

■

The assumptions in Proposition 1 about local Hessians and the Jacobian matrix are standard assumptions used in proving convergence of Newton-Raphson's method (Luenberger, 1984). These are not additional assumptions that we make.

Based on Remark 2 and Proposition 1, we can say that starting from the optimal weight matrix and target of the previous step i will lead to fewer iterations in the current step $i+1$, which means faster convergence.

Note that our suggested technique is also more efficient than doing a warm start in the current step using the optimal weight matrix and target from the previous step. This is because even in the worst case where none of the conditions are satisfied, finding the solution of the corresponding WLS problem in the current step is, as discussed before, more efficient using Algorithm 1 rather than learning from scratch.

3.3 Other Regression Techniques

In the previous sections, we showed that using Algorithm 1 just by itself or using it as a core function in other algorithms can lead to optimally and efficiently solving regression problems that use a rich class of regression techniques. This naturally implies that questions I(a), I(b) and I(c) in Section 2 have been answered positively for these cases. In this section, we try to answer these questions for arbitrary linear or non-linear regression algorithms.

If we again use Algorithm 1 for an arbitrary regression algorithm, it may very well provide accurate predictions but is likely to be inefficient. The most expensive computation in Algorithm 1 would be computing the residual of fitting X_i onto each column of x_{i+1}. This could potentially lead to running the chosen regression algorithm $d_{i+1} - d_i$ times. We want to save on these computations and therefore, we relax Algorithm 1, where we simply fit x_{i+1} to $R^Y_{X_i}$ rather than fitting $R^{x_{i+1}}_{X_i}$ to $R^Y_{X_i}$. Let us refer to this new version of Algorithm 1 by Algorithm $1^{(r)}$. Let us now see if Algorithm $1^{(r)}$ also results in a positive response to questions I(a), I(b) and I(c).

Let us denote the error or residual of the optimal model built in step i (previous step) by δ_{prev}. If AFOR optimizes the residual of this model—which leads to the best AFOR model—based on x_{i+1}, then we denote its error by δ^{bst}_{AFOR}. Let the error of our relaxed method be denoted by $\delta_{1^{(r)}}$. With this, we have the following result,

Proposition 2 *Algorithm $1^{(r)}$ in step $i+1$ is no less accurate than the model in step i and AFOR in step $i+1$, i.e., $\delta_{1^{(r)}} \leq \delta^{bst}_{AFOR} \leq \delta_{prev}$.*

Proof We have $\delta_{prev} = R^Y_{X_i}$ and $\delta^{bst}_{AFOR} = R^{R^Y_{X_i}}_{x_{i+1}}$. Thus, δ^{bst}_{AFOR} can be written as, $\delta^{bst}_{AFOR} = R^{\delta_{prev}}_{x_{i+1}}$. Since in AFOR we optimize the residual of the model at the previous step, we have

$$\delta_{prev} \geq \delta^{bst}_{AFOR}.$$

In our relaxed method, we first fit x_{i+1} to $R^Y_{X_i}$. We then fit X_i to $Y^{(r)} = Y - \hat{Y}^{R^Y_{X_i}}_{x_{i+1}}$ and whose error maybe denoted by $\delta_{1^{(r)}} = R^{Y^{(r)}}_{X_i}$. Thus, $\delta_{1^{(r)}}$ can be written as, $\delta_{1^{(r)}} = R^{\delta^{bst}_{AFOR}}_{X_i}$ and since we optimize the residual of AFOR we have

$$\delta^{bst}_{AFOR} \geq \delta_{1^{(r)}}.$$

■

Proposition 2 implies a positive response to questions I(a) and I(b). It is also easy to see that Algorithm $1^{(r)}$ is more efficient than learning over the whole space and hence, we also have a positive response for I(c). If we further relax Algorithm 1 to exclude fitting X_i to $Y^{(r)}$, then our method is reduced to AFOR.

4. Addressing Question II

In the previous sections we provided techniques to update an existing model based on a new set of available features. In this section, we want to tackle the complimentary problem of updating a model in a particular step based on a new batch.

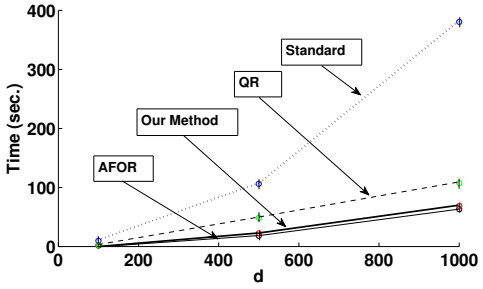


Figure 3: Total training time for OLS summed over the three steps for the different methods with different number of features d added at each step.

There are numerous works in online learning (Blum, 1996; Smale and Yao, 2005; Bottou and Cun, 2003), where existing models are updated using stochastic techniques in time proportional to the newly added information. A common update procedure for many parametric models is to perform additive updates. Thus, in step i and based on the data up until batch b, if we have a model M_i and we learn a model based on batch $b+1$, m_i^{b+1}, then the model based on data up until batch $b+1$ would be given by

$$M_i^{b+1} = M_i^b + \nu(m_i^{b+1} - M_i^b),$$

where $\nu^b \in [0,1]$ is the learning rate when batch $b+1$ becomes available. It is reasonable to make ν^b a function of N_{b+1} and $S = \sum_{j=1}^{b} N_j$, where the learning rate is higher when m_i^{b+1} is obtained by learning over a relatively larger data set. For example, $\nu^b = \frac{N_{b+1}}{S_{b+1}}$. Other possible updates are multiplicative updates (Arora et al., 2005), where the original model M_i^b can be updated based on its error in relation to the error of m_i^{b+1}.

5. Experiments

In this section, we empirically validate our claims through synthetic and real data experiments.

5.1 Synthetic Experiments

We consider the setting where we have three feature subsets or steps (i.e., $K = 3$) each of d dimensions. We generate data from a $3d+1$ dimensional Gaussian, where the first d dimensions make up the first representation X_1, the next d make up the next representation X_2, while the remaining but the last make up the third representation X_3. The final dimension corresponds to the target. We set the variances of each of the variables to 1. Since we want to model a realistic scenario, where the correlation between the target and the features is low and non-uniform, we set the correlation of the target with the features in the a) first representation to 0.1, b) second representation to 0.2 and c) third representation to 0.3. The other off-diagonal entries in the correlation matrix are set to 0.5. With this

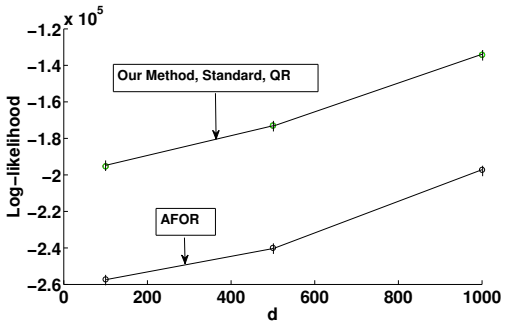


Figure 4: Average log-likelihood of the final model for the different methods with different number of features d added at each step for OLS.

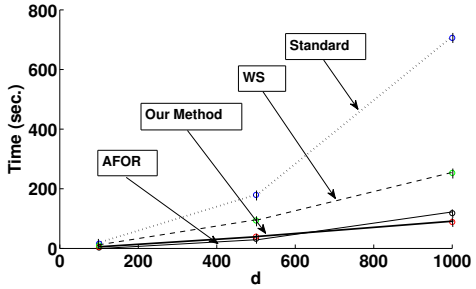


Figure 5: Total training time summed over the three steps for the different methods with different number of features d added at each step for logistic regression.

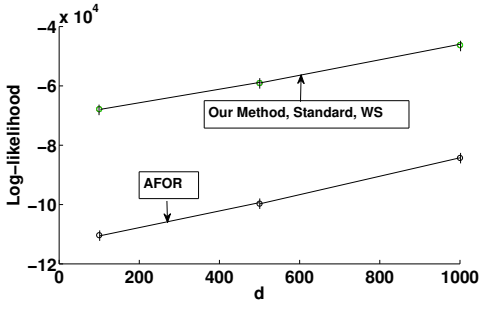


Figure 6: Average log-likelihood of the final model for the different methods with different number of features d added at each step for logistic regression.

correlation matrix and the mean set to zero we generate 100 data sets of 10000 points for each of the three values of d, namely: a) $d = 100$, b) $d = 500$ and c) $d = 1000$.

We compare the performance of our algorithms with a) learning from scratch i.e., standard, b) AFOR and c) QR decomposition based updates using Givens rotations (Stewart, 2001; Golub and Loan, 1996) or warm starting (QR and WS), for two regression methods namely, OLS regression and logistic regression ($\epsilon_\beta = \epsilon_\mathcal{D} = 0.1$). QR decomposition (Stewart, 2001) is only efficient in the OLS case since for logistic regression we have to perform the decomposition—not simply update—at every step as the optimal weight matrix and target keep changing. Thus, we use QR only for OLS, while warm starts are only applicable for logistic regression. For logistic regression, we discretize the target where we insert a value of 1 if the target has value ≥ 0, otherwise we insert a 0. We perform 10-fold cross validation and report the total training time summed over the three steps and the average test likelihood of the final model for each value of d with a 95% confidence interval.

In Figure 3, we see that the standard method takes significantly more time than the other methods. Our method is almost as fast as AFOR which is very promising and much faster than QR based updates. The performance gain of our method compared to the other optimal methods improves as d increases. In Figure 4, we see that AFOR is significantly worse in terms of accuracy than the other methods which are all optimal. In Figure 5, we see a similar trend as in the OLS case. However, the speedup of our method is greater and for the case with the highest d, our method is even faster than AFOR. A possible reason for this is that in one or more steps the updating using Algorithm 1 is much better than starting from the default even with just the added features. Here again, as seen in Figure 6, AFOR has much lower accuracy than the other methods that are optimal.

5.2 Real Data Experiments

We evaluate our methods on two real industrial data sets obtained from diverse domains. The first experiment is on a chip production data set obtained from the semiconductor industry. In this empirical study we compare the different updating strategies mentioned in this paper to learn a regression model. In the second experiment, we consider a finance data set, where we want to find which sources of information (or feature sets) are predictive of the final revenue. In this case, we run LogitGOMP (Lozano et al., 2011) to select the feature sets. When a feature set is selected, we update the current model using the different updating strategies. We then compare the runs of LogitGOMP based on these different strategies. Through this experiment we show that our method can also be effectively used to speed up sophisticated group variable selection techniques in real world settings. In both the experiments we add another straw man, which learns only on the features available at the particular step. We refer to this method, which learns only on the **a**dditional **f**eatures as AF.

5.3 Regression in Chip Manufacturing

We first provide some background of the chip manufacturing process and describe the general setup. We then discuss the major takeaways from the conducted experiments.

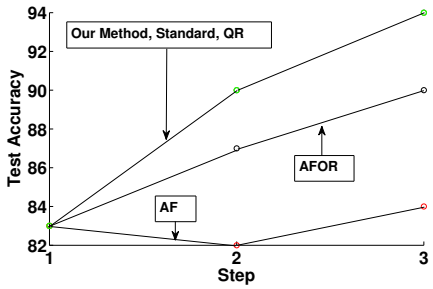


Figure 7: Test set accuracy of the methods in classifying wafers as in spec or out of spec at the respective steps for OLS.

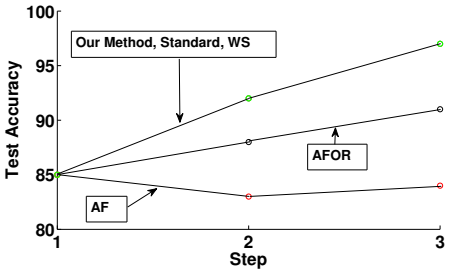


Figure 8: Test set accuracy of the methods in classifying wafers as in spec or out of spec at the respective steps for logistic regression.

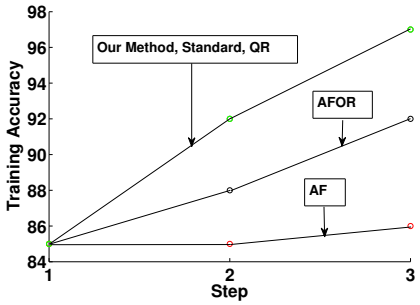


Figure 9: Training set accuracy of the different methods in classifying wafers as in spec or out of spec at the respective steps for OLS.

	Step 2			Step 3		
	Time	$\mathcal{L}_{tr}(.)$	$\mathcal{L}_{tst}(.)$	Time	$\mathcal{L}_{tr}(.)$	$\mathcal{L}_{tst}(.)$
Our Method	**20.6**	**-1991.8**	**-2156.2**	**17.8**	**-1742.1**	**-1895.5**
Standard	100.1	**-1991.8**	**-2156.2**	145.9	**-1742.1**	**-1895.5**
AFOR	**20.4**	-2693.4	-3482.5	**17.5**	-2176.7	-3285.2
AF	**20.2**	-3793.7	-4755.5	**17.4**	-3893.7	-4712.3
QR	72.3	**-1991.8**	**-2156.2**	67.5	**-1742.1**	**-1895.5**

Table 3: Average time (in sec.) the least squares methods take to train in each of the steps, the average training log-likelihood value $\mathcal{L}_{tr}(.)$ in the respective steps and the average test log-likelihood value $\mathcal{L}_{tst}(.)$ in the respective steps.

	Step 2			Step 3		
	Time	$\mathcal{L}_{tr}(.)$	$\mathcal{L}_{tst}(.)$	Time	$\mathcal{L}_{tr}(.)$	$\mathcal{L}_{tst}(.)$
Our Method	216.2	**-0.5**	**-1.7**	385.7	**-0.1**	**-1.2**
Standard	2085.8	**-0.5**	**-1.7**	3960.9	**-0.1**	**-1.2**
AFOR	**206.2**	-8.9	-12.8	**374.7**	-7.3	-11.4
AF	**206.2**	-20.9	-22.8	**374.4**	-27.3	-31.4
WS	556.7	**-0.5**	**-1.7**	791.2	**-0.1**	**-1.2**

Table 4: Average time (in sec.) the logistic regression approaches take to train in each of the steps, the average training log-likelihood value $\mathcal{L}_{tr}(.)$ in the respective steps and the average test log-likelihood value $\mathcal{L}_{tst}(.)$ in the respective steps.

5.3.1 Setup

We consider a real semiconductor process of microprocessor or chip production. In our data, a single data point is a wafer, which is a group of chips, and measurements which correspond to input features that are made on this wafer throughout its production. The target that is used to evaluate the quality of the wafer, in this case, is the speed of the wafer, which is the median of the speeds of its chips. Speed is usually used to diagnose the health of a wafer and needs to be within specifications. A slow wafer is undesirable for obvious reasons, but a fast wafer is also bad since it consumes too much power and can lead to overheating. The wafer speed prediction problem is quite challenging since the measurements are noisy and the speeds vary considerably from wafer to wafer relative to the spec. The wafers indicated as out of spec are usually discarded to save time and money in downstream processing. In some cases, they are re-routed for corrective action. In some isolated cases, wafers that are not too far out of spec may even be processed further and queued for low-end products. It takes a few months to produce a wafer as it goes through many complex high precision steps. Though the overall processing time maybe in the order of months, as each process takes a significant amount of time, the wafers move from one step to the next within a few minutes. It is thus important that we quickly estimate the

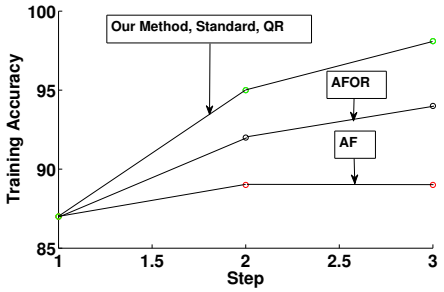


Figure 10: Training set accuracy of the methods in classifying wafers as in spec or out of spec at the respective steps for logistic regression.

speed after measurements from the finished step become available if we are to take any of the previously mentioned remedial actions.

We consider three critical steps in the manufacturing process with our data spanning over three batches. The first is the wafer polishing step referred to as chemical-mechanical planarization. Here the wafer is smoothed with removal of unnecessary material. During this step, various pressures—viz. condition head pressure, head zone pressures, etc.—and torques are measured indicating the amount of force the wafer is subjected to. The second step we consider is the etching step, where any remaining abnormalities in the photo-resistance on the wafer are removed by plasma ashing. Here the quantity and temperature of the plasma are controlled among other things and corresponding temperatures, pressures and concentrations are measured. The third and final step we consider is the rapid thermal processing step. In this step the electrical properties such as the material dielectric are altered. To alter the electrical properties, the wafer is subjected to sudden temperature ramps at tightly controlled pressures and chemical flows. Hence, here ramp up temperatures, ramp up rates, cool-down rates and various pressures and flows are measured.

By the time the wafer reaches the first step, 2287 measurements are taken. By the second step we have 3317 measurements. Finally we have 4284 measurements by the third step. Each of the three batches have 8926 wafers. In each of the steps, we train over the first batch and test over the second. We then train over the second batch and test over the third. We report the average training time and the average train and test log-likelihoods[3] for each step in Tables 3 and 4.

In this experiment, we compare the performance of our algorithms with a) learning from scratch, i.e., standard, b) AFOR, c) AF and d) QR decomposition based updates (Stewart, 2001) or warm starting (QR and WS), for two regression methods, namely, OLS regression and logistic regression ($\epsilon_\beta = \epsilon_{\mathcal{D}} = 0.1$) as in the synthetic case. The metric we use to evaluate the approaches is the time it takes for the local models to find the solution and the corresponding log-likelihood value at the solution. The higher the log-likelihood and the less time it takes to get to it, the better the method. For logistic regression, we discretize

3. We do not report the confidence intervals since, the variances are insignificant.

Approaches	Time	$\mathcal{L}_{tr}(.)$	$\mathcal{L}_{tst}(.)$
Our Method	**2.2** $\pm$ 0.1	**-561.3** $\pm$ 1.1	**-782.2** $\pm$ 1.3
Standard	10.4 $\pm$ 0.2	**-561.3** $\pm$ 1.1	**-782.2** $\pm$ 1.3
AFOR	2.9 $\pm$ 0.1	-605.2 $\pm$ 1.3	-841.6 $\pm$ 1.9
AF	2.8 $\pm$ 0.1	-1605.2 $\pm$ 1.8	-1841.6 $\pm$ 2.1
WS	7.8 $\pm$ 0.2	**-561.3** $\pm$ 1.1	**-782.2** $\pm$ 1.3

Table 5: Average running time (in seconds) of LogitGOMP, the average training log-likelihood value $\mathcal{L}_{tr}(.)$ and the average test log-likelihood value $\mathcal{L}_{tst}(.)$ with a 95% confidence interval.

the target based on specifications to denote either within spec by 1 or out of spec by 0. In all the three batches, roughly 20% of the wafers are out of spec.

5.3.2 Observations

In Tables 3 and 4, we see the results for step 2 and step 3 across the three batches. We do not show step 1 since we have to learn from scratch for all the models. The results show that our methods are optimal and significantly more efficient than the standard method. Our methods are also more efficient than QR and WS. AF and AFOR are efficient since they learn only on the additional features but are sub-optimal. Our methods are almost as efficient as AF and AFOR, which is very encouraging.

In Figures 7, 8, 9 and 10 we observe the test and training set accuracy in classifying wafers as within spec or out of spec based on the predictions obtained from OLS and logistic regression. The OLS predictions of the wafer speed are easily categorized into the two classes by considering the acceptable speed range given in the specification. In all the four figures we see that our method, which is optimal, performs significantly better than AFOR and AF as new features become available, even though it has comparable running time to both of them, as is seen in Tables 3 and 4. The poor performance of AF and AFOR indicates that the features across the various steps are correlated and are required to build an accurate predictive model.

5.4 Group Feature Selection in Finance

We consider a financial data set which is composed of three sources of information. Each of the three sources spans over two years and the data we have is at a weekly level. Thus, the number of data points is 104. The first source contains aging information of different deals, that is, how much time the different deals were in different financial stages before they were either won or lost. There are 105 features in this group. The second source is competitive information about different size bids made and how many of those were won. There are 21 features that characterize this group. The third source has information about important product launches over this two year period. There are five important product lines leading to five features. We want to find which of these groups have a significant effect in determining the revenue. Given that we run LogitGOMP, which learns a logistic

regression model, we normalize the revenue, our target, using the sum over the two year period. We divide the data into 8 quarters and perform 8-fold cross-validation as predicting an entire quarter is of more interest than predicting randomly scattered weeks.

Table 5 shows that our method ($\epsilon_\beta = \epsilon_\mathcal{D} = 0.1$) is almost 5 times faster than the standard method and about 3.5 times faster than WS. Interestingly, it is also faster than AF and AFOR. The reason for this is that, in the second step, condition two in Algorithm 2 is satisfied and we have the optimal weight matrix and target from the previous update step in LogitGOMP, which is a better starting point than starting from the default during updates.

6. Discussion

It is important to note that the problem addressed in this paper is somewhat complimentary to stagewise learning. Stagewise learning mainly addresses the problem of feature selection, where coefficients are estimated for one feature at a time. Forward selection (Weisberg, 1980) and least angle regression (Efron et al., 2004) are examples of stagewise learning techniques. On the other hand, ours is a meta-learning technique—not limited to any particular regression algorithm—that considers all new features simultaneously with the final goal of learning a predictive model accurately and efficiently.

Our methods are also more general than the incremental feature learning method based on autoencoders to update an existing model(Zhou et al., 2012). This method makes the strong assumption of bounded input and output and the number of features considered at each step is a free parameter. The method is somewhat similar to AFOR as only new features are used to optimize the residual of the model learned over old features. Moreover, our methods are also more general than QR decomposition (Stewart, 2001), which is mainly used for efficient updates in OLS problems.

In the future, it would be interesting to update an existing model simultaneously based on added features and data points rather than doing it sequentially. One may be able to merge the ideas in this paper with the vast online learning literature. However, the main challenge in developing such a method would be guaranteeing accuracy while maintaining efficiency.

Acknowledgments

We would like to thank Katherine Dhurandhar for proofreading the paper.

References

S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta algorithm and applications. *Theory of Computing*, 8:121–164, 2005.

A. Blum. On-line algorithms in machine learning. In *Workshop on On-Line Algorithms, Dagstuhl*, pages 306–325. Springer, 1996.

L. Bottou and Y. Cun. Large scale online learning. In *Neural Information Processing Systems (NIPS)*, pages 77–84, 2003.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, 3rd edition, 1996.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009.

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

A. Lozano, G. Swirszcz, and N. Abe. Grouped orthogonal matching pursuit for variable selection and prediction. In *Neural Information Processing Systems*, 2009.

A. Lozano, G. Swirszcz, and N. Abe. Group orthogonal matching pursuit for logistic regression. In *Artificial Intelligence and Statistics*, pages 452–460, 2011.

D. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley; 2nd edition, 1984.

P. McCullagh and J. Nelder. *Generalized Linear Models, Second Edition*. Chapman and Hall, 1990.

D. Pregibon. Logistic regression diagnostics. *Annals of Statistics*, 9:704–724, 1981.

Y. Saad. Analysis of augmented Krylov subspace methods. *SIAM Journal on Matrix Analysis and Applications*, 18(2):435–449, April 1997.

S. Smale and Y. Yao. Online learning algorithms. *Found. Comp. Math*, 6:145–170, 2005.

G. Stewart. *Matrix Algorithms*. SIAM, 2001.

R. J. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

R. J. Tibshirani and J. Taylor. The solution path of the generalized LASSO. *The Annals of Statistics*, 39(3):1335–1371, 2011.

P. Wang. Residual plots for detecting nonlinearity in generalized linear models. *Technometrics*, 29(4):435–438, 1987.

S. Weisberg. *Applied Linear Regression*. New York: Wiley, 1980.

G. Zhou, K. Sohn, and H. Lee. Online incremental feature learning with denoising autoencoder. In *Artificial Intelligence and Statistics*, pages 1453–1461, 2012.

Journal of Machine Learning Research 15 (2014) 2629-2652 Submitted 10/13; Revised 4/14; Published 8/14

Bayesian Estimation of Causal Direction in Acyclic Structural Equation Models with Individual-specific Confounder Variables and Non-Gaussian Distributions

Shohei Shimizu SSHIMIZU@AR.SANKEN.OSAKA-U.AC.JP
The Institute of Scientific and Industrial Research
Osaka University
Mihogaoka 8-1, Ibaraki, Osaka 567-0047, Japan

Kenneth Bollen BOLLEN@UNC.EDU
Department of Sociology, CB 3210 Hamilton Hall
University of North Carolina
Chapel Hill, NC 27599-3210
U.S.A.

Editor: David Maxwell Chickering

Abstract

Several existing methods have been shown to consistently estimate causal direction assuming linear or some form of nonlinear relationship and no latent confounders. However, the estimation results could be distorted if either assumption is violated. We develop an approach to determining the possible causal direction between two observed variables when latent confounding variables are present. We first propose a new linear non-Gaussian acyclic structural equation model with individual-specific effects that are sometimes the source of confounding. Thus, modeling individual-specific effects as latent variables allows latent confounding to be considered. We then propose an empirical Bayesian approach for estimating possible causal direction using the new model. We demonstrate the effectiveness of our method using artificial and real-world data.

Keywords: structural equation models, Bayesian networks, estimation of causal direction, latent confounding variables, non-Gaussianity

1. Introduction

Aids to uncover the causal structure of variables from observational data are welcomed additions to the field of machine learning (Pearl, 2000; Spirtes et al., 1993). One conventional approach makes use of Bayesian networks (Pearl, 2000; Spirtes et al., 1993). However, these suffer from the identifiability problem. That is, many different causal structures give the same conditional independence between variables, and in many cases one cannot uniquely estimate the underlying causal structure without prior knowledge (Pearl, 2000; Spirtes et al., 1993).

To address these issues, Shimizu et al. (2006) proposed LiNGAM (Linear Non-Gaussian Acyclic Model), a variant of Bayesian networks (Pearl, 2000; Spirtes et al., 1993) and structural equation models (Bollen, 1989). Unlike conventional Bayesian networks, LiNGAM is a fully identifiable model (Shimizu et al., 2006), and has recently attracted much attention in

machine learning (Spirtes et al., 2010; Moneta et al., 2011). If causal relations exist among variables, LiNGAM uses their non-Gaussian distributions to identify the causal structure among the variables. LiNGAM is closely related to independent component analysis (ICA) (Hyvärinen et al., 2001b); the identifiability proof and estimation algorithm are partly based on the ICA theory. The idea of LiNGAM has been extended in many directions, including to nonlinear cases (Hoyer et al., 2009; Lacerda et al., 2008; Hyvärinen et al., 2010; Zhang and Hyvärinen, 2009; Peters et al., 2011a).

Many causal discovery methods including LiNGAM make the strong assumption of no latent confounders (Spirtes and Glymour, 1991; Dodge and Rousson, 2001; Shimizu et al., 2006; Hyvärinen and Smith, 2013; Hoyer et al., 2009; Zhang and Hyvärinen, 2009). These methods have been used in various application fields (Ramsey et al., 2014; Rosenström et al., 2012; Smith et al., 2011; Statnikov et al., 2012; Moneta et al., 2013). However, in many areas of empirical science, it is often difficult to accept the estimation results because latent confounders are ignored. In theory, we could take a non-Gaussian approach (Hoyer et al., 2008b) that uses an extension of ICA with more latent variables than observed variables (overcomplete ICA) to formally consider latent confounders in the framework of LiNGAM. Unfortunately, current versions of the overcomplete ICA algorithms are not very computationally reliable since they often suffer from local optima (Entner and Hoyer, 2011).

Thus, in this paper, we propose an alternative Bayesian approach to develop a method that is computationally simple in the sense that no iterative search in the parameter space is required and it is capable of finding the possible causal direction of two observed variables in the presence of latent confounders. We first propose a variant of LiNGAM with individual-specific effects. Individual differences are sometimes the source of confounding (von Eye and Bergman, 2003). Thus, modeling certain individual-specific effects as latent variables allows a type of latent confounding to be considered. A latent confounding variable is an unobserved variable that exerts a causal influence on more than one observed variables (Hoyer et al., 2008b). The new model is still linear but allows any number of latent confounders. We then present a Bayesian approach for estimating the model by integrating out some of the large number of parameters, which is of the same order as the sample size. Such a Bayesian approach is often used in the field of mixed models (Demidenko, 2004) and multilevel models (Kreft and De Leeuw, 1998), although estimation of causal direction is not a topic studied within it.

Granger causality (Granger, 1969) is another popular method to aid detection of causal direction. His method depends on the temporal ordering of variables whereas our method does not. Therefore, our method can be applied to cases where temporal information is not available, i.e., cross-sectional data, as well as those where it is available, i.e., time-series data.

The remainder of this paper is organized as follows. We first review LiNGAM (Shimizu et al., 2006) and its extension to latent confounder cases (Hoyer et al., 2008b) in Section 2. In Section 3, we propose a new mixed-LiNGAM model, which is a variant of LiNGAM with individual-specific effects. We also propose an empirical Bayesian approach for learning the model. We empirically evaluate the performance of our method using artificial and real-world sociology data in Sections 4 and 5, respectively, and present our conclusions in Section 6.

2. Background

In this section, we first review the linear non-Gaussian structural equation model known as LiNGAM (Shimizu et al., 2006). We then discuss an extension of LiNGAM to cases where latent confounding variables exist (Hoyer et al., 2008b).

In LiNGAM (Shimizu et al., 2006), causal relations between observed variables x_l ($l = 1, \cdots, d$) are modeled as

$$x_l = \mu_l + \sum_{k(m)<k(l)} b_{lm} x_m + e_l, \tag{1}$$

where $k(l)$ is a causal ordering of the variables x_l. The causal orders $k(l)$ ($l = 1, \cdots, d$) are unknown and to be estimated. In this ordering, the variables x_l form a directed acyclic graph (DAG) so that no later variable determines, i.e., has a directed path to, any earlier variable in the DAG. The variables e_l are latent continuous variables called error variables, μ_l are intercepts or regression constants, and b_{lm} are connection strengths or regression coefficients.

In matrix form, the LiNGAM model in Equation (1) is written as

$$\boldsymbol{x} = \boldsymbol{\mu} + \mathbf{B}\boldsymbol{x} + \boldsymbol{e}, \tag{2}$$

where the vector $\boldsymbol{\mu}$ collects constants μ_l, the connection strength matrix $\mathbf{B}$ collects regression coefficients (or connection strengths) b_{lm}, and the vectors $\boldsymbol{x}$ and $\boldsymbol{e}$ collect observed variables x_l and error variables e_l, respectively. The zero/non-zero pattern of b_{lm} corresponds to the absence/existence pattern of directed edges (direct effects). It can be shown that it is always possible to perform simultaneous, equal row and column permutations on the connection strength matrix $\mathbf{B}$ to cause it to become *strictly* lower triangular, based on the acyclicity assumption (Bollen, 1989). Here, strict lower triangularity is defined as a lower triangular structure with the diagonal consisting entirely of zeros. Errors e_l follow non-Gaussian distributions with zero mean and non-zero variance, and are jointly independent. This model without assuming non-Gaussianity distribution is called a fully recursive model in conventional structural equation models (Bollen, 1989). The non-Gaussianity assumption on e_l enables the identification of a causal ordering $k(l)$ and the coefficients b_{lm} based only on $\boldsymbol{x}$ (Shimizu et al., 2006), unlike conventional Bayesian networks based on the Gaussianity assumption on e_l (Spirtes et al., 1993). To illustrate the LiNGAM model, the following example is considered, whose corresponding directed acyclic graph is provided in Figure 1:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 3 \\ -5 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix}.$$

In this example, x_3 is *equal to* error e_3 and is exogenous since it is not affected by either of the other two variables x_1 and x_2. Thus, x_3 is in the first position of such a causal ordering such that $\mathbf{B}$ is strictly lower triangular, x_1 is in the second, and x_2 is the third, i.e., $k(3) = 1$, $k(1) = 2$, and $k(2) = 3$. If we permute the variables x_1 to x_3 according to the causal ordering, we have

$$\begin{bmatrix} x_3 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 3 & 0 & 0 \\ 0 & -5 & 0 \end{bmatrix} \begin{bmatrix} x_3 \\ x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} e_3 \\ e_1 \\ e_2 \end{bmatrix}.$$

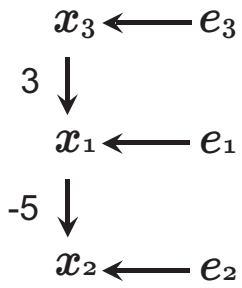


Figure 1: An example graph of LiNGAMs

It can be seen that the resulting connection strength (or regression coefficient) matrix is strictly lower triangular.

Several computationally efficient algorithms for estimating the model have been proposed (Shimizu et al., 2006, 2011; Hyvärinen and Smith, 2013). As with ICA, LiNGAM is identifiable under the assumptions of non-Gaussianity and independence among error variables (Shimizu et al., 2006; Comon, 1994; Eriksson and Koivunen, 2003).[1] However, for the estimation methods to be consistent, additional assumptions, e.g., the existence of their moments or some other statistic, must be made to ensure that the statistics computed in the estimation algorithms exist. The idea of LiNGAM can be generalized to nonlinear cases (Hoyer et al., 2009; Tillman et al., 2010; Zhang and Hyvärinen, 2009; Peters et al., 2011b).

The assumption of independence among e_l means that there is no latent confounding variable (Shimizu et al., 2006). A latent confounding variable is an unobserved variable that contributes to the values of more than one observed variable (Hoyer et al., 2008b). However, in many applications, there often exist latent confounding variables. If such latent confounders are completely ignored, the estimation results can be seriously biased (Pearl, 2000; Spirtes et al., 1993; Bollen, 1989). Therefore, in Hoyer et al. (2008b), LiNGAM with latent confounders, called latent variable LiNGAM, was proposed, and the model can be formulated as follows:

$$x_l = \mu_l + \sum_{k(m)<k(l)} b_{lm} x_m + \sum_{q=1}^{Q} \lambda_{lq} f_q + e_l,$$

where f_q are non-Gaussian individual-specific effects f_q with zero mean and unit variance and λ_{lq} denote the regression coefficients (connection strengths) from f_q to x_l. This model is written in matrix form as follows:

$$\boldsymbol{x} = \boldsymbol{\mu} + \mathbf{B}\boldsymbol{x} + \boldsymbol{\Lambda}\boldsymbol{f} + \boldsymbol{e}, \tag{3}$$

where the difference from LiNGAM in Equation (2) is the existence of a latent confounding variable vector $\boldsymbol{f}$. The vector $\boldsymbol{f}$ collects f_q. The matrix $\boldsymbol{\Lambda}$ collects λ_{lq} and is assumed to

1. Comon (1994) and Eriksson and Koivunen (2003) established the identifiability of ICA based on the characteristic functions of variables. Moments of some variables may not exist, but their characteristic functions always exist.

be of full column rank. Another way to represent latent confounder cases would be to use dependent error variables. Denoting $\boldsymbol{\Lambda f} + \boldsymbol{e}$ in Equation (3) by $\tilde{\boldsymbol{e}}$, we have

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{\mu} + \mathbf{B}\boldsymbol{x} + \boldsymbol{\Lambda f} + \boldsymbol{e} \\ &= \boldsymbol{\mu} + \mathbf{B}\boldsymbol{x} + \tilde{\boldsymbol{e}}, \end{aligned}$$

where $\tilde{e}_i$ are dependent due to the latent confounders f_q. Observed variables that are equal to dependent errors $\tilde{e}_i$ are connected by bi-directed arcs in their graphs. An example graph is given in Figure 4. This representation can be more general since it is easier to extend it to represent nonlinearly dependent errors. In this paper, however, we use the aforementioned representation using independent errors and latent confounders since linear relations of the observed variables, latent confounders, and errors are necessary for our approach.

Without loss of generality, the latent confounders f_q are assumed to be jointly independent since any dependent latent confounders can be remodeled by linear combinations of independent latent variables if the underlying model is linear acyclic and the error variables are independent (Hoyer et al., 2008b). To illustrate this, the following example model is considered:

$$\begin{aligned} \bar{f}_1 &= e_{\bar{f}_1} && (4) \\ \bar{f}_2 &= \omega_{21}\bar{f}_1 + e_{\bar{f}_2} && (5) \\ x_1 &= \lambda_{11}\bar{f}_1 + e_1 \\ x_2 &= \lambda_{21}\bar{f}_1 + e_2 \\ x_3 &= \lambda_{32}\bar{f}_2 + e_3 \\ x_4 &= b_{43}x_3 + \lambda_{42}\bar{f}_2 + e_4, \end{aligned}$$

where errors $e_{\bar{f}_1}(=\bar{f}_1)$, $e_{\bar{f}_2}$, and e_1–e_4 are non-Gaussian and independent. The associated graph is shown in Figure 2. The relations of $\bar{f}_1$, $\bar{f}_2$, and x_1–x_4 are represented by a directed acyclic graph and latent confounders $\bar{f}_1$ and $\bar{f}_2$ are dependent. In matrix form, this example model can be written as

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ b_{43} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} \lambda_{11} & 0 \\ \lambda_{21} & 0 \\ 0 & \lambda_{32} \\ 0 & \lambda_{42} \end{bmatrix} \begin{bmatrix} \bar{f}_1 \\ \bar{f}_2 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix}.$$

The relations of $\bar{f}_1$ and $\bar{f}_2$ to $e_{\bar{f}_1}$ and $e_{\bar{f}_2}$ in Equations (4)–(5):

$$\begin{bmatrix} \bar{f}_1 \\ \bar{f}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \omega_{21} & 1 \end{bmatrix} \begin{bmatrix} e_{\bar{f}_1} \\ e_{\bar{f}_2} \end{bmatrix},$$

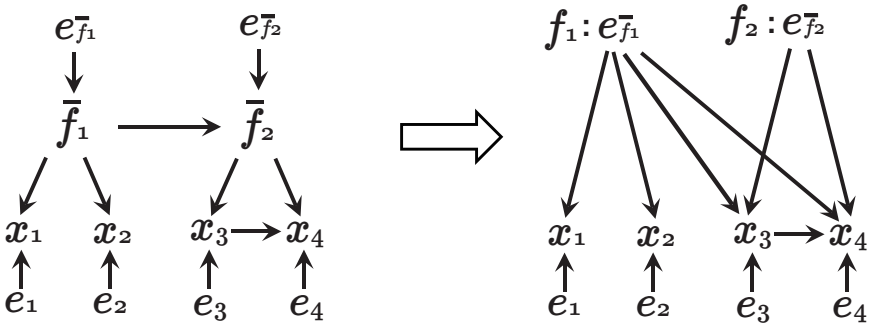


Figure 2: An example graph to illustrate the idea of independent latent confounders.

we obtain

$$\underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}}_{\boldsymbol{x}} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ b_{43} & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}}_{\boldsymbol{x}} + \underbrace{\begin{bmatrix} \lambda_{11} & 0 \\ \lambda_{21} & 0 \\ \lambda_{32}\omega_{21} & \lambda_{32} \\ \lambda_{42}\omega_{21} & \lambda_{42} \end{bmatrix}}_{\mathbf{\Lambda}} \underbrace{\begin{bmatrix} e_{\bar{f}_1} \\ e_{\bar{f}_2} \end{bmatrix}}_{\boldsymbol{f}} + \underbrace{\begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix}}_{\boldsymbol{e}}.$$

This is a latent variable LiNGAM in Equation (3) taking $f_1 = e_{\bar{f}_1}$ and $f_2 = e_{\bar{f}_2}$ since $e_{\bar{f}_1}$ and $e_{\bar{f}_2}$ are non-Gaussian and independent.

Moreover, the faithfulness of x_l and f_q to the generating graph is assumed. The faithfulness assumption (Spirtes et al., 1993) here means that when multiple causal paths exist from one variable to another, their combined effect does not equal exactly zero (Hoyer et al., 2008b). The faithfulness assumption can be considered to be not very restrictive from the Bayesian viewpoint (Spirtes et al., 1993) since the probability of having exactly the parameter values that do not satisfy faithfulness is zero (Meek, 1995).

In the framework of latent variable LiNGAM, it has been shown (Hoyer et al., 2008b) that the following three models are distinguishable based on observed data,[2] i.e., the three

2. If one or more error variables or latent confounders are Gaussian, it cannot be ensured that Models 3 to 5 will be distinguishable. Hoyer et al. (2008a) considered cases with one or more Gaussian error variables in the context of basic LiNGAM.

different causal structures induce different data distributions:

$$\text{Model 3}: \begin{cases} x_1 = & \sum_{q=1}^{Q} \lambda_{1q} f_q + e_1 \\ x_2 = & \sum_{q=1}^{Q} \lambda_{2q} f_q + e_2, \end{cases}$$
$$\text{Model 4}: \begin{cases} x_1 = & \sum_{q=1}^{Q} \lambda_{1q} f_q + e_1 \\ x_2 = b_{21} x_1 + & \sum_{q=1}^{Q} \lambda_{2q} f_q + e_2, \end{cases}$$
$$\text{Model 5}: \begin{cases} x_1 = b_{12} x_2 + & \sum_{q=1}^{Q} \lambda_{1q} f_q + e_1 \\ x_2 = & \sum_{q=1}^{Q} \lambda_{2q} f_q + e_2, \end{cases},$$

where $\lambda_{1q}\lambda_{2q} \neq 0$ due to the definition of latent confounders, that is, that they contribute to determining the values of more than two variables.

An estimation method based on overcomplete ICA (Lewicki and Sejnowski, 2000) explicitly modeling all the latent confounders f_q was proposed (Hoyer et al., 2008b). However, in current practice, overcomplete ICA estimation algorithms often get stuck in local optima and are not sufficiently reliable (Entner and Hoyer, 2011). A Bayesian approach for estimating the latent variable LiNGAM in Equation (3) has been proposed in Henao and Winther (2011). These previous approaches that explicitly model latent confounders (Hoyer et al., 2008b; Henao and Winther, 2011) need to select the number of latent confounders, and which can be quite large. This could lead to further computational difficulty and statistically unreliable estimates.

In Chen and Chan (2013), a simple approach based on fourth-order cumulants for estimating latent variable LiNGAM was proposed. Their approach does not need to explicitly model the latent confounders, however it requires the latent confounders f_q to be Gaussian. The development of nonlinear methods that incorporate latent confounders is ongoing (Zhang et al., 2010).

None of these latent confounder methods incorporate the individual-specific effects that we model in the next section to consider latent confounders f_q in the latent variable LiNGAM of Equation (3).

3. Linear Non-Gaussian Acyclic Structural Equation Model with Individual-specific Effects

In this section, we propose a new Bayesian method for learning the possible causal direction of two observed variables in the presence of latent confounding variables, assuming that the causal relations are acyclic, i.e., there is not a feedback relation.

3.1 Model

The LiNGAM (Shimizu et al., 2006) for observation i can be described as follows:

$$x_l^{(i)} = \mu_l + \sum_{k(m)<k(l)} b_{lm} x_m^{(i)} + e_l^{(i)}.$$

The random variables $e_l^{(i)}$ are non-Gaussian and independent. The distributions of $e_l^{(i)}$ $(i = 1, \cdots, n)$ are commonly assumed to be identical[3] for every l. A linear non-Gaussian acyclic structural equation model with individual-specific effects for observation i is formulated as follows:

$$x_l^{(i)} = \mu_l + \tilde{\mu}_l^{(i)} + \sum_{k(m)<k(l)} b_{lm} x_m^{(i)} + e_l^{(i)},$$

where the difference from LiNGAM is the existence of individual-specific effects $\tilde{\mu}_l^{(i)}$. The parameters $\tilde{\mu}_l^{(i)}$ are independent of $e_l^{(i)}$ and are correlated with $x_l^{(i)}$ through the structural equations in our Bayesian approach, introduced below. This means that the observations are generated from the identifiable LiNGAM, possibly with different parameter values of the means $\mu_l + \tilde{\mu}_l^{(i)}$. We call this a mixed-LiNGAM, named after mixed models (Demidenko, 2004), as it has effects μ_l and b_{lm} that are common to all the observations and individual-specific effects $\tilde{\mu}_l^{(i)}$. We note that causal orderings of variables $k(l)$ $(l = 1, \cdots, d)$ are identical for all the observations in the sample.

To use a Bayesian approach for estimating the mixed-LiNGAM, we need to model the distributions of error variables e_l and prior distributions of the parameters including individual-specific effects $\tilde{\mu}_l^{(i)}$, unlike previous LiNGAM methods (Shimizu et al., 2006; Hoyer et al., 2008b). These individual-specific effects, whose number is of the same order as the sample size, are integrated out in the Bayesian method developed in Section 3.2, assuming an informative prior for them similar to the estimation of conventional mixed models (Demidenko, 2004). More details on the distributions of error variables and prior distributions of parameters are given in Section 3.2. These distributional assumptions were implied to be robust to some extent to their violations, at least in the artificial data experiments of Section 4.

We now relate the mixed-LiNGAM model above with the latent variable LiNGAM (Hoyer et al., 2008b). The latent variable LiNGAM in Equation (3) for observation i is written as follows:

$$x_l^{(i)} = \mu_l + \sum_{k(m)<k(l)} b_{lm} x_m^{(i)} + \underbrace{\sum_{q=1}^{Q} \lambda_{lq} f_q^{(i)}}_{\tilde{\mu}_l^{(i)}} + e_l^{(i)}.$$

This is a mixed-LiNGAM taking $\tilde{\mu}_l^{(i)} = \sum_{q=1}^{Q} \lambda_{lq} f_q^{(i)}$. In contrast to the previous approaches for latent variable LiNGAM (Hoyer et al., 2008b; Henao and Winther, 2011), we do not explicitly model the latent confounders f_q and rather simply include their sums $\tilde{\mu}_l^{(i)} = \sum_{q=1}^{Q} \lambda_{lq} f_q^{(i)}$ in our model as its parameters since our main interest lies in estimation of the causal relation of observed variables x_l and not in the estimation of their relations with latent confounders f_q. Our method does not estimate λ_{lq} or the number of latent confounders Q.

3. Relaxing this identically distributed assumption would lead to more general modeling of individual differences, however, this goes beyond the scope of the paper.

3.2 Estimation of Possible Causal Direction

We apply a Bayesian approach to estimate the possible causal direction of two observed variables using the mixed-LiNGAM proposed above. We compare the following two mixed-LiNGAM models with opposite possible directions of causation. Model 1 is

$$\begin{aligned} x_1^{(i)} &= \mu_1 + \tilde{\mu}_1^{(i)} + e_1^{(i)} \\ x_2^{(i)} &= \mu_2 + \tilde{\mu}_2^{(i)} + b_{21} x_1^{(i)} + e_2^{(i)}, \end{aligned}$$

where b_{21} is non-zero. In Model 1, x_2 does not cause x_1. The second model, Model 2, is

$$\begin{aligned} x_1^{(i)} &= \mu_1 + \tilde{\mu}_1^{(i)} + b_{12} x_2^{(i)} + e_1^{(i)} \\ x_2^{(i)} &= \mu_2 + \tilde{\mu}_2^{(i)} + e_2^{(i)}, \end{aligned}$$

where b_{12} is non-zero. In Model 2, x_1 does not cause x_2. The two models have the same number of parameters, but opposite possible directions of causation.

Once the possible causal direction is estimated, one can see if the common causal coefficient (connection strength) b_{21} or b_{12} is likely to be zero by examining its posterior distribution.[4] We focus here on estimating the possible direction of causation as in many previous works (Dodge and Rousson, 2001; Hoyer et al., 2009; Zhang and Hyvärinen, 2009; Chen and Chan, 2013; Hyvärinen and Smith, 2013), and do not go to the computation of the posterior distribution[5] since estimation of the possible causal direction of two observed variables in the presence of latent confounders has been a very challenging problem in causal inference and is the main topic of this paper.

We apply standard Bayesian model selection techniques to help assess the causal direction of x_1 and x_2. We use the log-marginal likelihood for comparing the two models. The model with the larger log-marginal likelihood is regarded as the closest to the true model (Kass and Raftery, 1995).

Let $\mathcal{D}$ be the observed data set $[\boldsymbol{x}^{(1)^T}, \cdots, \boldsymbol{x}^{(n)^T}]^T$, where $\boldsymbol{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}]^T$. Denote Models 1 and 2 by M_1 and M_2. The log-marginal likelihoods of M_1 and M_2 are

$$\begin{aligned} \log\{p(M_r|\mathcal{D})\} &= \log\{p(\mathcal{D}|M_r)p(M_r)/p(\mathcal{D})\} \\ &= \log\{p(\mathcal{D}|M_r)\} + \log\{p(M_r)\} - \log p(\mathcal{D}) \\ &= \log\{\int p(\mathcal{D}|\boldsymbol{\theta}_r, M_r) p(\boldsymbol{\theta}_r|M_r, \boldsymbol{\eta}_r) d\boldsymbol{\theta}_r\} \\ &\quad + \log p(M_r) - \log p(\mathcal{D}) \ (r = 1, 2), \end{aligned}$$

where $\boldsymbol{\eta}_1, \boldsymbol{\eta}_2$ are the hyper-parameter vectors regarding the distributions of the parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, respectively. Since the last term $\log p(\mathcal{D})$ is constant with respect to M_r, we can drop it. To select suitable values for these hyper-parameters, we take an ordinary empirical Bayesian approach. First, we compute the log-marginal likelihood for every combination

4. Chickering and Pearl (1996) considered a discrete variable model with *known* possible causal direction and proposed a Bayesian approach for computing the posterior distributions of causal effects in the presence of latent confounders.
5. Point estimates of the parameters including the common causal connection strengths b_{12} and b_{21} can be obtained by taking their posterior means based on their posterior distributions, for example.

of the two models M_r and a number of candidate hyper-parameter values of $\boldsymbol{\eta}_r$. Next, we take the model and hyper-parameter values that give the largest log-marginal likelihood, and finally estimate that the model with the largest log-marginal likelihood is better than the other model.

In basic LiNGAM (Shimizu et al., 2006), we have (Hyvärinen et al., 2010; Hoyer and Hyttinen, 2009)

$$p(\boldsymbol{x}) = \prod_l p_{e_l}\left(x_l - \mu_l - \sum_{k(m)<k(l)} b_{lm} x_m\right).$$

Thus, in the same manner, the likelihoods under mixed-LiNGAM $p(\mathcal{D}|\boldsymbol{\theta}_r, M_r)$ $(r = 1, 2)$ are given by

$$\begin{aligned} p(\mathcal{D}|\boldsymbol{\theta}_r, M_r) &= \Pi_{i=1}^n \, p(\boldsymbol{x}^{(i)}|\boldsymbol{\theta}_r, M_r) \\ &= \begin{cases} \Pi_{i=1}^n \, p_{e_1^{(i)}}(x_1^{(i)} - \mu_1 - \tilde{\mu}_1^{(i)}|\boldsymbol{\theta}_1, M_1) \\ \times \, p_{e_2^{(i)}}(x_2^{(i)} - \mu_2 - \tilde{\mu}_2^{(i)} - b_{21} x_1^{(i)}|\boldsymbol{\theta}_1, M_1) \;\; \text{for } M_1 \\ \Pi_{i=1}^n \, p_{e_1^{(i)}}(x_1^{(i)} - \mu_1 - \tilde{\mu}_1^{(i)} - b_{12} x_2^{(i)}|\boldsymbol{\theta}_2, M_2) \\ \times \, p_{e_2^{(i)}}(x_2^{(i)} - \mu_2 - \tilde{\mu}_2^{(i)}|\boldsymbol{\theta}_2, M_2) \text{ for } M_2 \end{cases}. \end{aligned}$$

We model the parameters and their prior distributions as follows.[6] The prior probabilities of M_1 and M_2 are uniform:

$$p(M_1) = p(M_2).$$

The distributions of the error variables $e_1^{(i)}$ and $e_2^{(i)}$ are modeled by Laplace distributions with zero mean and variances of $\text{var}(e_1^{(i)}) = h_1^2$ and $\text{var}(e_2^{(i)}) = h_2^2$ as follows:

$$\begin{aligned} p_{e_1^{(i)}} &= \mathit{Laplace}(0, |h_1|/\sqrt{2}) \\ p_{e_2^{(i)}} &= \mathit{Laplace}(0, |h_2|/\sqrt{2}). \end{aligned}$$

Here, we simply use a symmetric super-Gaussian distribution, i.e., the Laplace distribution, to model $p_{e_1^{(i)}}$ and $p_{e_2^{(i)}}$, as suggested in Hyvärinen and Smith (2013). Such super-Gaussian distributions have been reported to often work well in non-Gaussian estimation methods including independent component analysis and LiNGAM (Hyvärinen et al., 2001b; Hyvärinen and Smith, 2013). In some cases, a wider class of non-Gaussian distributions might provide a better model for $p_{e_1^{(i)}}$ and $p_{e_2^{(i)}}$, *e.g.*, the generalized Gaussian family (Hyvärinen et al., 2001b), a finite mixture of Gaussians, or an exponential family distribution combining the Gaussian and Laplace distributions (Hoyer and Hyttinen, 2009).

The parameter vectors $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are written as follows:

$$\begin{aligned} \boldsymbol{\theta}_1 &= [\mu_l, b_{21}, h_l, \tilde{\mu}_l^{(i)}]^T \quad (l = 1, 2; i = 1, \cdots, n) \\ \boldsymbol{\theta}_2 &= [\mu_l, b_{12}, h_l, \tilde{\mu}_l^{(i)}]^T \quad (l = 1, 2; i = 1, \cdots, n). \end{aligned}$$

6. This is an example. The modeling method could depend on the domain knowledge.

The prior distributions of common effects are Gaussian as follows:

$$\begin{aligned}
\mu_1 &\sim N(0, \tau_{\mu_1}^{cmmn}) \\
\mu_2 &\sim N(0, \tau_{\mu_2}^{cmmn}) \\
b_{12} &\sim N(0, \tau_{b_{12}}^{cmmn}) \\
b_{21} &\sim N(0, \tau_{b_{21}}^{cmmn}) \\
h_1 &\sim N(0, \tau_{h_1}^{cmmn}) \\
h_2 &\sim N(0, \tau_{h_2}^{cmmn}),
\end{aligned}$$

where $\tau_{\mu_1}^{cmmn}$, $\tau_{\mu_2}^{cmmn}$, $\tau_{b_{12}}^{cmmn}$, $\tau_{b_{21}}^{cmmn}$, $\tau_{h_1}^{cmmn}$ and $\tau_{h_2}^{cmmn}$ are constants.

Generally speaking, we could use various informative prior distributions for the individual-specific effects and then compare candidate priors using the standard model selection approach based on the marginal likelihoods. Below we provide two examples.

If the data is generated from a latent variable LiNGAM, a special case of mixed-LiNGAM, as shown in Section 3.1, the individual-specific effects are the sums of many non-Gaussian independent latent confounders f_q and are dependent. The central limit theorem states that the sum of independent variables becomes increasingly close to the Gaussian (Billingsley, 1986). Therefore, in many cases, it could be practical to approximate the non-Gaussian distribution of a variable that is the sum of many non-Gaussian and independent variables by a bell-shaped curve distribution (Sogawa et al., 2011; Chen and Chan, 2013). This motivates us to model the prior distribution of individual-specific effects by the multivariate t-distribution as follows:

$$\begin{bmatrix} \tilde{\mu}_1^{(i)} \\ \tilde{\mu}_2^{(i)} \end{bmatrix} = \mathrm{diag}\left(\left[\sqrt{\tau_1^{indvdl}}, \sqrt{\tau_2^{indvdl}}\right]^T\right) \mathbf{C}^{-1/2}\boldsymbol{u}, \tag{6}$$

where τ_1^{indvdl} and τ_2^{indvdl} are constants, $\boldsymbol{u} \sim t_\nu(\mathbf{0}, \boldsymbol{\Sigma})$ and $\boldsymbol{\Sigma} = [\sigma_{ab}]$ is a symmetric scale matrix whose diagonal elements are 1s. A random variable vector $\boldsymbol{u}$ that follows the multivariate t-distribution $t_\nu(\mathbf{0}, \boldsymbol{\Sigma})$ can be created by $\frac{\boldsymbol{y}}{\sqrt{v/\nu}}$, where $\boldsymbol{y}$ follows the Gaussian distribution $N(\mathbf{0}, \boldsymbol{\Sigma})$, v follows the chi-squared distribution with ν degrees of freedom, and $\boldsymbol{y}$ and v are statistically independent (Kotz and Nadarajah, 2004). Note that u_i have energy correlations (Hyvärinen et al., 2001a), i.e., correlations of squares $\mathrm{cov}(u_i^2, u_j^2) > 0$ due to the common variable v. $\mathbf{C}$ is a diagonal matrix whose diagonal elements give the variance of elements of $\boldsymbol{u}$, i.e., $\mathbf{C} = \frac{\nu}{\nu-2}\mathrm{diag}(\boldsymbol{\Sigma})$ for $\nu > 2$. The degree of freedom ν is here taken to be six. The kurtosis of the univariate Student's t-distribution with six degrees of freedom is three, the same as that of the Laplace distribution.

The hyper-parameter vectors $\boldsymbol{\eta}_1$ and $\boldsymbol{\eta}_2$ are

$$\boldsymbol{\eta}_l = [\tau_{\mu_1}^{cmmn}, \tau_{\mu_2}^{cmmn}, \tau_{b_{12}}^{cmmn}, \tau_{b_{21}}^{cmmn}, \tau_{h_1}^{cmmn}, \tau_{h_2}^{cmmn}, \tau_1^{indvdl}, \tau_2^{indvdl}, \sigma_{21}]^T \quad (l = 1, 2).$$

We want to take the constants $\tau_{\mu_1}^{cmmn}$, $\tau_{\mu_2}^{cmmn}$, $\tau_{b_{12}}^{cmmn}$, $\tau_{b_{21}}^{cmmn}$, $\tau_{h_1}^{cmmn}$ and $\tau_{h_2}^{cmmn}$ to be sufficiently large so that the priors for the common effects are not very informative. It depends on the scales of variables when these constants are sufficiently large. In the experiments in Sections 4–5, we set $\tau_{\mu_1}^{cmmn} = \tau_{b_{12}}^{cmmn} = \tau_{h_1}^{cmmn} = 10^2 \times \widehat{\mathrm{var}}\,(x_1)$ and $\tau_{\mu_2}^{cmmn} =$

$\tau_{b_{21}}^{cmmn} = \tau_{h_2}^{cmmn} = 10^2 \times \widehat{\text{var}}\ (x_2)$ so that they reflect the scales of the corresponding variables.

Moreover, we take an empirical Bayesian approach for the individual-specific effects. We test $\tau_l^{indvdl} = 0, 0.2^2 \times \widehat{\text{var}}(x_l), ..., 0.8^2 \times \widehat{\text{var}}(x_l), 1.0^2 \times \widehat{\text{var}}(x_l)$ $(l = 1, 2)$. That is, we uniformly vary the hyper-parameter value from that with no individual-specific effects, i.e., 0, to a larger value, i.e., $1.0^2 \times \widehat{\text{var}}(x_l)$, which implies very large individual differences. Further, we test $\sigma_{12} = 0, \pm 0.3, \pm 0.5, \pm 0.7, \pm 0.9$, i.e., the value with zero correlation and larger values with stronger correlations. This means that we test uncorrelated individual-specific effects as well as correlated ones. We take the ordinary Monte Carlo sampling approach to compute the log-marginal likelihoods with 1000 samples for the parameter vectors $\boldsymbol{\theta}_r$ $(r = 1, 2)$.

The assumptions for our model are summarized in Table 1. Generally speaking, if the actual probability density function of individual-specific effects is unimodal and most often provides zero or very small absolute values and with few large values, i.e., many of the individual-specific effects are close to zero and many individuals have similar intercepts, the estimation is likely to work. If the individuals have very different intercepts, the estimation will not work very well.

An alternative way of modeling the prior distribution of individual-specific effects would be to use the multivariate Gaussian distribution as follows:

$$\begin{bmatrix} \tilde{\mu}_1^{(i)} \\ \tilde{\mu}_2^{(i)} \end{bmatrix} = \text{diag}\left(\left[\sqrt{\tau_1^{indvdl}}, \sqrt{\tau_2^{indvdl}} \right]^T \right) \boldsymbol{z},$$

where τ_1^{indvdl} and τ_2^{indvdl} are constants, $\boldsymbol{z} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$ and $\boldsymbol{\Sigma} = [\sigma_{ab}]$ is a symmetric scale matrix whose diagonal elements are 1s. This Gaussian prior would be effective if the Gaussian approximation based on the central limit theorem works well, although a non-Gaussian prior would be more consistent with the non-Gaussian latent variable LiNGAM in Equation (3). Gaussian individual-specific effects or latent confounders would not lead to losing the identifiability (Chen and Chan, 2013) since each observation still is generated by the identifiable non-Gaussian LiNGAM. However, if errors are Gaussian, there is no guarantee that our method can find correct possible causal direction. We could detect their Gaussianity by comparing our mixed-LiNGAM models with Gaussian error models based on their log-marginal likelihoods. If the errors are actually Gaussian or close to be Gaussian, Gaussian error models would provide larger log-marginal likelihoods. This would detect situations where our approach cannot find causal direction.

4. Experiments on Artificial Data

We compared our method with seven methods for estimating the possible causal direction between two variables: i) LvLiNGAM[7] (Hoyer et al., 2008b); ii) SLIM[8](Henao and Winther, 2011) iii) LiNGAM-GC-UK (Chen and Chan, 2013); iv) ICA-LiNGAM[9] (Shimizu et al., 2006); v) DirectLiNGAM[10] (Shimizu et al., 2011); vi) Pairwise LiNGAM[11] (Hyvärinen

7. The code is available at `http://www.cs.helsinki.fi/u/phoyer/code/lvlingam.tar.gz`.
8. The code is available at `http://cogsys.imm.dtu.dk/slim/`.
9. The code is available at `http://www.cs.helsinki.fi/group/neuroinf/lingam/lingam.tar.gz`.
10. The code is available at `http://www.ar.sanken.osaka-u.ac.jp/~sshimizu/code/Dlingamcode.html`.
11. The code is available at `http://www.cs.helsinki.fi/u/ahyvarin/code/pwcausal/`.

Model: $x_l^{(i)} = \mu_l + \tilde{\mu}_l^{(i)} + \sum_{k(m)<k(l)} b_{lm} x_m^{(i)} + e_l^{(i)} \quad (l, m = 1, 2;\ l \neq m)$,
where b_{lm} are non-zero.
$e_l^{(i)}$ $(l = 1, 2; i = 1, \cdots, n)$ are i.i.d..
e_l $(l = 1, 2)$ are mutually independent.
e_l $(l = 1, 2)$ follow Laplace distributions with zero mean and standard deviations $|h_l|$.

Prior distributions:
μ_l, b_{lm} and h_l $(l = 1, 2; m = 1, 2; l \neq m)$ follow Gaussian distributions with zero mean and variance $\tau_{\mu_l}^{cmmn}$, $\tau_{b_{lm}}^{cmmn}$ and $\tau_{h_l}^{cmmn}$.
$\tilde{\mu}_l^{(i)}$ $(l = 1, 2; i = 1, \cdots, n)$ are the sum of latent confounders $f_q^{(i)}$: $\sum_{q=1}^{Q} \lambda_{lq} f_q^{(i)}$ and are independent of $e_l^{(i)}$.
$\tilde{\mu}_l^{(i)}$ $(l = 1, 2; i = 1, \cdots, n)$ are i.i.d..
μ_l $(l = 1, 2)$ follow multivariate t-distributions with ν degrees of freedom, zero mean, variances τ_l^{indvdl} and correlation σ_{12} (here, $\nu = 6$).

Hyper-parameters:
$\tau_{\mu_l}^{cmmn}$, $\tau_{b_{lm}}^{cmmn}$ and $\tau_{h_l}^{cmmn}$ $(l = 1, 2; m = 1, 2; l \neq m)$ are set to be large values so that the priors are not very informative.
τ_l^{indvdl} $(l = 1, 2)$ are uniformly varied from zero to large values.
σ_{12} are uniformly varied in the interval between -0.9 and 0.9.

Table 1: Summary of the assumptions for our mixed-LiNGAM model

and Smith, 2013); vii) Post-nonlinear causal model (PNL) [12] (Zhang and Hyvärinen, 2009). Their assumptions are summarized in Table 2. The first seven methods assume linearity, and the eighth allows a very wide variety of nonlinear relations. The last four methods assume that there are no latent confounders. We tested the prior t- and Gaussian distributions for individual-specific effects in our approach. LvLiNGAM and SLIM require to specify the number of latent confounders. We tested 1 and 4 latent confounder(s) for LvLiNGAM since its current implementation cannot handle more than four latent confounders, whereas we tested 1, 4 and 10 latent confounders(s) for SLIM. LiNGAM-GC-UK (Chen and Chan, 2013) assumes that errors are simultaneously super-Gaussian or sub-Gaussian and that latent confounders are Gaussian.

	Functional form?	Latent confounders allowed?	Number of latent confounders necessary to be specified?	Iterative search in the parameter space required?	Distributional assumptions necessary?
Our approach	Linear	Yes	No	No	Yes
LvLiNGAM	Linear	Yes	Yes	Yes	No[13]
SLIM	Linear	Yes	Yes	No	Yes
LiNGAM-GC-UK	Linear	Yes	No	No	Yes
ICA-LiNGAM	Linear	No	N/A	Yes	No
DirectLiNGAM	Linear	No	N/A	No	No
Pairwise LiNGAM	Linear	No	N/A	No	No
PNL	Nonlinear	No	N/A	Yes	No

Table 2: Summary of the assumptions of eight methods

12. The code is available at `http://webdav.tuebingen.mpg.de/causality/CauseOrEffect_NICA.rar`.
13. Their current implementation of LvLiNGAM in Footnote 7 assumes a non-Gaussian distribution, which is a mixture of two Gaussian distributions.

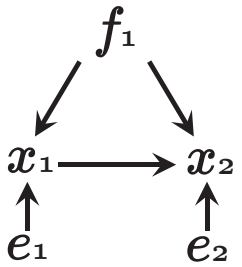


Figure 3: The associated graph of the model used to generate artificial data when the number of latent confounders $Q = 1$.

We generated data using the following latent variable LiNGAM with Q latent confounding variables, which is a mixed-LiNGAM:

$$\begin{aligned} x_1^{(i)} &= \mu_1 + \sum_{q=1}^{Q} \lambda_{1q} f_q^{(i)} + e_1^{(i)} \\ x_2^{(i)} &= \mu_2 + b_{21} x_1^{(i)} + \sum_{q=1}^{Q} \lambda_{2q} f_q^{(i)} + e_2^{(i)}, \end{aligned}$$

where μ_1 and μ_2 were randomly generated from $N(0,1)$, and $b_{21}, \lambda_{1q}, \lambda_{2q}$ were randomly generated from the interval $(-1.5,-0.5) \cup (0.5,1.5)$. We tested various numbers of latent confounders $Q = 0, 1, 6, 12$. The zero values indicate that there are no latent confounders. An example graph used to generate artificial data is given in Figure 3.

The distributions of the error variables e_1, e_2, and latent confounders f_q were identical for all observations. The distributions of the error variables e_1, e_2, and latent confounders f_q were randomly selected from the 18 non-Gaussian distributions used in Bach and Jordan (2002) to see if the Laplace distribution assumption on error variables and t- or Gaussian distribution assumption on individual-specific effects in our method were robust to different non-Gaussian distributions. These include symmetric/non-symmetric distributions, super-Gaussian/sub-Gaussian distributions, and strongly/weakly non-Gaussian distributions. The variances of e_1 and e_2 were randomly selected from the interval $(0.5^2, 1.5^2)$. The variances of f_q were 1s.

We permuted the variables according to a random ordering and inputted them to the eight estimation methods. We conducted 100 trials, with sample sizes of 50, 100, and 200. For the data with the number of latent confounders $Q = 0$, all the methods should find the correct causal direction for large enough sample sizes, as there were no latent confounders, which here means no individual-specific effects. The last four comparative methods should find the data with the number of latent confounders $Q = 1, 6, 12$ very difficult to analyze, because, unlike the other approaches, they assume no latent confounders.

To evaluate the performance of the algorithms, we counted the number of successful discoveries of possible causal direction and estimated their standard errors.

Looking at Table 3 as a whole there are several general observations that we can make. First though none of the procedures is infallible, several of them do quite well in that they choose the correct causal direction about 90% of the time. Second, overall our approach is the most successful across the conditions of the simulation. Specifically, in all but the cases of no confounding variables, one or both of our approaches have the highest percentages of success. In the situation of no confounding variables, ICA-LiNGAM, DirectLiNGAM, and Pairwise LINGAM have higher success percentages than our procedures. These generalizations need qualifications in that there are sampling errors that affect the estimates. Formal tests of significance across all conditions would be complicated. It would require taking account of multiple testing and the dependencies of the simulated samples under the same sample size and number of confounders. However, the standard errors of the estimated percentages serve to caution the reader not to judge the percentages alone without recognizing sampling variability. For instance, when there are no confounders and a sample size of 50, the ICA-LiNGAM procedure appears best with 93% success, but the success percentages of our two approaches fall within two standard errors of the 93% estimate. Alternatively, in the rows with 6 confounders and sample size 50 our approach with 88% success and a standard error of 3.25 appears sufficiently far from the success percentages of the other methods besides ours to make sampling fluctuations an unlikely explanation. In sum, taking all the evidence together, our approaches performed quite well and deserve further investigation under additional simulation conditions.

Table 4 shows the average computational times. The computational complexity of the current implementation of our methods is clearly larger than that of the other linear methods ICA-LiNGAM, DirectLiNGAM, Pairwise LiNGAM, LvLiNGAM with 1 latent confounder, SLIM and LiNGAM-GC-UK and comparable to LvLiNGAM with 4 latent confounders and the nonlinear method PNL.

The MATLAB code for performing these experiments is available on our website.[14]

5. An Experiment on Real-world Data

We analyzed the General Social Survey data set, taken from a sociological data repository (`http://www.norc.org/GSS+Website/`). The data consisted of six observed variables: x_1: prestige of father's occupation, x_2: son's income, x_3: father's education, x_4: prestige of son's occupation, x_5: son's education, and x_6: number of siblings.[15] The sample selection was conducted based on the following criteria: i) non-farm background; ii) ages 35–44; iii) white; iv) male; v) in the labor force at the time of the survey; vi) not missing data for any of the covariates; and vii) data taken from 1972–2006. The sample size was 1380.

The possible directions were determined based on the domain knowledge in Duncan et al. (1972), shown in Figure 4. Note that there is no direct causal link from x_1, x_3, and x_6 to x_2 in the figure, however it is expected that each of these variables has non-zero total causal effects on x_2 given their indirect effects on x_2. The causal relations of x_1, x_3, and x_6 usually are not modeled in the literature since there are many other determinants of these three exogenous observed variables that are not part of the model. However, the possible

14. The URL is `http://www.ar.sanken.osaka-u.ac.jp/~sshimizu/code/mixedlingamcode.html`.
15. Although x_6 is discrete, it can be considered as continuous because it is an ordinal scale with many points.

	Sample size		
	50	100	200
Number of latent confounders $Q = 0$:			
Our approach (t-distributed individual-specific effects)	88 (3.25)	91 (2.86)	86 (3.47)
Our approach (Gaussian individual-specific effects)	91 (2.86)	87 (3.36)	91 (2.86)
LvLiNGAM (1 latent confounder)	73 (4.44)	83 (3.76)	83 (3.76)
LvLiNGAM (4 latent confounders)	52 (5.00)	68 (4.66)	66 (4.74)
SLIM (1 latent confounder)	29 (4.54)	30 (4.58)	25 (4.33)
SLIM (4 latent confounders)	34 (4.74)	31 (4.62)	36 (4.80)
SLIM (10 latent confounders)	30 (4.58)	29 (4.54)	30 (4.58)
LiNGAM-GC-UK	33 (4.70)	28 (4.49)	35 (4.77)
ICA-LiNGAM	<u>93</u> (2.55)	93 (2.55)	96 (1.96)
DirectLiNGAM	87 (3.36)	<u>95</u> (2.18)	<u>97</u> (1.71)
Pairwise LiNGAM	89 (3.13)	<u>95</u> (2.18)	95 (2.18)
Post-nonlinear causal model	74 (4.39)	71 (4.54)	75 (4.33)
Number of latent confounders $Q = 1$:			
Our approach (t-distributed individual-specific effects)	<u>83</u> (3.76)	80 (4.00)	<u>80</u> (4.00)
Our approach (Gaussian individual-specific effects)	79 (4.07)	<u>87</u> (3.36)	69 (4.62)
LvLiNGAM (1 latent confounder)	66 (4.74)	71 (4.54)	73 (4.44)
LvLiNGAM (4 latent confounders)	63 (4.83)	58 (4.94)	67 (4.70)
SLIM (1 latent confounder)	40 (4.90)	47 (4.99)	25 (4.33)
SLIM (4 latent confounders)	40 (4.90)	34 (4.74)	44 (4.96)
SLIM (10 latent confounders)	47 (4.99)	39 (4.88)	41 (4.92)
LiNGAM-GC-UK	24 (4.27)	32 (4.66)	32 (4.66)
ICA-LiNGAM	74 (4.39)	71 (4.54)	67 (4.70)
DirectLiNGAM	48 (5.00)	52 (5.00)	54 (4.98)
Pairwise LiNGAM	54 (4.98)	58 (4.94)	61 (4.88)
Post-nonlinear causal model	55 (4.97)	58 (4.94)	57 (4.95)
Number of latent confounders $Q = 6$:			
Our approach (t-distributed individual-specific effects)	<u>88</u> (3.25)	81 (3.92)	<u>87</u> (3.36)
Our approach (Gaussian individual-specific effects)	84 (3.67)	<u>85</u> (3.57)	<u>87</u> (3.36)
LvLiNGAM (1 latent confounder)	58 (4.94)	70 (4.58)	70 (4.58)
LvLiNGAM (4 latent confounders)	64 (4.80)	61 (4.88)	63 (4.83)
SLIM (1 latent confounder)	50 (5.00)	63 (4.83)	47 (4.99)
SLIM (4 latent confounders)	45 (4.97)	47 (4.99)	43 (4.95)
SLIM (10 latent confounders)	58 (4.94)	48 (5.00)	58 (4.94)
LiNGAM-GC-UK	29 (4.54)	28 (4.49)	21 (4.07)
ICA-LiNGAM	74 (4.39)	72 (4.49)	47 (4.99)
DirectLiNGAM	37 (4.83)	48 (5.00)	39 (4.88)
Pairwise LiNGAM	48 (5.00)	51 (5.00)	37 (4.83)
Post-nonlinear causal model	55 (4.97)	42 (4.94)	46 (4.98)
Number of latent confounders $Q = 12$:			
Our approach (t-distributed individual-specific effects)	88 (3.25)	86 (3.47)	89 (3.13)
Our approach (Gaussian individual-specific effects)	<u>91</u> (2.86)	<u>89</u> (3.13)	<u>91</u> (2.86)
LvLiNGAM (1 latent confounder)	52 (5.00)	55 (4.97)	65 (4.77)
LvLiNGAM (4 latent confounders)	65 (4.77)	58 (4.94)	64 (4.80)
SLIM (1 latent confounder)	51 (5.00)	55 (4.97)	60 (4.90)
SLIM (4 latent confounders)	45 (4.97)	51 (5.00)	63 (4.83)
SLIM (10 latent confounders)	61 (4.88)	54 (4.98)	54 (4.98)
LiNGAM-GC-UK	21 (4.07)	25 (4.33)	29 (4.54)
ICA-LiNGAM	68 (4.66)	72 (4.49)	72 (4.49)
DirectLiNGAM	37 (4.83)	39 (4.88)	38 (4.85)
Pairwise LiNGAM	56 (4.96)	42 (4.94)	43 (4.95)
Post-nonlinear causal model	51 (5.00)	43 (4.95)	46 (4.98)

Largest numbers of successful discoveries were underlined.
Standard errors are shown in parentheses, which are computed assuming that the number of successes follow a binomial distribution.

Table 3: Number of successful discoveries (100 trials)

	Sample size		
	50	100	200
Number of latent confounders $Q = 0$			
Our approach (t-distributed individual-specific effects)	27.20	56.93	141.84
Our approach (Gaussian individual-specific effects)	35.48	69.59	117.10
LvLiNGAM (1 latent confounder)	2.41	2.55	9.91
LvLiNGAM (4 latent confounders)	22.25	30.12	87.96
SLIM (1 latent confounder)	5.89	6.25	6.81
SLIM (4 latent confounders)	7.60	8.14	9.13
SLIM (10 latent confounders)	10.88	12.02	13.96
LiNGAM-GC-UK	0.00	0.00	0.00
ICA-LiNGAM	0.04	0.03	0.02
DirectLiNGAM	0.00	0.01	0.01
Pairwise LiNGAM	0.00	0.00	0.00
Post-nonlinear causal model	19.59	27.68	57.37
Number of latent confounders $Q = 1$:			
Our approach (t-distributed individual-specific effects)	35.87	65.55	131.25
Our approach (Gaussian individual-specific effects)	37.12	75.11	114.37
LvLiNGAM (1 latent confounder)	2.40	2.53	13.93
LvLiNGAM (4 latent confounders)	21.50	29.50	92.19
SLIM (1 latent confounder)	5.88	6.01	6.69
SLIM (4 latent confounders)	7.59	8.19	8.96
SLIM (10 latent confounders)	10.96	11.79	13.68
LiNGAM-GC-UK	0.00	0.00	0.00
ICA-LiNGAM	0.05	0.03	0.03
DirectLiNGAM	0.01	0.01	0.01
Pairwise LiNGAM	0.00	0.00	0.00
Post-nonlinear causal model	18.17	28.83	51.63
Number of latent confounders $Q = 6$:			
Our approach (t-distributed individual-specific effects)	42.66	76.29	132.43
Our approach (Gaussian individual-specific effects)	33.13	69.07	104.83
LvLiNGAM (1 latent confounder)	2.40	2.56	9.38
LvLiNGAM (4 latent confounders)	22.17	30.12	83.01
SLIM (1 latent confounder)	5.89	6.22	6.77
SLIM (4 latent confounders)	7.58	8.18	9.11
SLIM (10 latent confounders)	11.03	12.02	13.91
LiNGAM-GC-UK	0.00	0.00	0.00
ICA-LiNGAM	0.06	0.05	0.05
DirectLiNGAM	0.01	0.01	0.01
Pairwise LiNGAM	0.00	0.00	0.00
Post-nonlinear causal model	18.71	29.62	52.21
Number of latent confounders $Q = 12$:			
Our approach (t-distributed individual-specific effects)	29.16	59.30	134.89
Our approach (Gaussian individual-specific effects)	32.18	68.14	104.76
LvLiNGAM (1 latent confounder)	2.35	2.50	13.58
LvLiNGAM (4 latent confounders)	21.51	30.10	94.08
SLIM (1 latent confounder)	5.90	6.03	6.62
SLIM (4 latent confounders)	7.58	7.99	8.97
SLIM (10 latent confounders)	10.92	11.68	13.74
LiNGAM-GC-UK	0.00	0.00	0.00
ICA-LiNGAM	0.07	0.08	0.07
DirectLiNGAM	0.01	0.02	0.02
Pairwise LiNGAM	0.00	0.00	0.00
Post-nonlinear causal model	18.21	29.21	51.89

Table 4: Average CPU time (s)

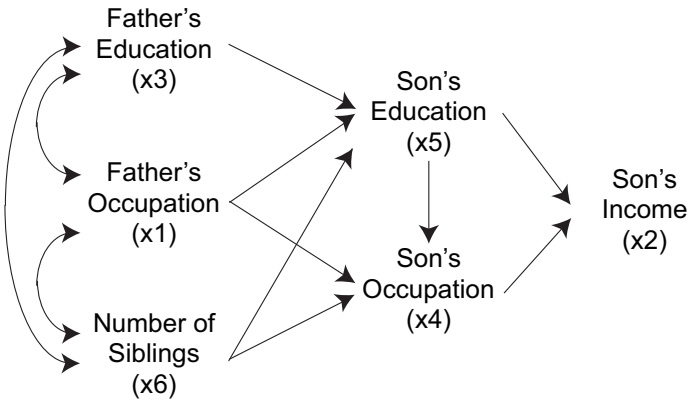


Figure 4: Status attainment model based on domain knowledge. Usually, the relations of x_1, x_3, and x_6, represented by bi-directed arcs, are not modeled.

causal directions among the three variables would be $x_1 \leftarrow x_3$, $x_6 \leftarrow x_1$, and $x_6 \leftarrow x_3$ based on their temporal orders.

Table 5 shows the numbers of successes and precisions. Our mixed-LiNGAM approach with the t-distributed individual-specific effects gave the largest number of successful discoveries 12 and achieved the highest precision, i.e., num. successes / num. pairs $= 12/15 = 0.80$. The second best method was our mixed-LiNGAM approach with the Gaussian individual-specific effects, which found one less correct possible directions than the t-distribution version. The third best method was LvLiNGAM with 1 latent confounder, which found two less correct possible directions than the t-distribution version. This would be mainly because our two methods allow individual-specific effects and the other methods do not.

Table 6 shows the estimated hyper-parameter values of our mixed-LiNGAM approach with the t-distributed individual-specific effects that performed best in the sociology data experiment. Either the estimated hyper-parameter $\hat{\tau}_1^{indvdl}$ or $\hat{\tau}_2^{indvdl}$ that represents the magnitudes of individual differences was non-zero in all pairs except (x_4, x_5). The non-ignorable influence of latent confounders was implied between the pairs (x_2, x_4), (x_2, x_6) and (x_3, x_6) since both $\hat{\tau}_1^{indvdl}$ or $\hat{\tau}_2^{indvdl}$ were non-zero for the pairs. In addition, for the pair (x_2, x_6), there might exist some nonlinear influence of latent confounders, since $\hat{\sigma}_{12}$ is zero, i.e., the individual-specific effects were linearly uncorrelated but dependent.[16] If $\hat{\sigma}_{12}$ were larger, it would have implied a larger linear influence of the latent confounders on the pair (x_2, x_6). The estimates of the hyper-parameter τ_1^{indvdl} were very large for the pairs (x_2, x_6) and (x_4, x_1), which implied very large individual differences regarding x_2 and x_4 respectively. This might imply that the estimated directions could be less reliable, although they were correct in this example.

Another point is that both our methods with t-distributed and Gaussian individual-specific effects failed to find the possible direction $x_5 \leftarrow x_1$, although the causal relation is

16. Two variables that follow the multivariate t-distribution are dependent, even when they are uncorrelated, as stated in Section 3.2.

expected to occur from the domain knowledge (Duncan et al., 1972). This failure would be attributed to the model misspecification since the sample size was very large. Since the estimate of the hyper-parameter τ_1^{indvdl} regarding x_5 was zero, the influence of latent confounders might be small for this pair, although the estimate of τ_2^{indvdl} was not small and the individual difference regarding x_5 seemed substantial. Modeling both latent confounders and nonlinear relations and/or allowing a wider class of non-Gaussian distributions might lead to better performance. This is an important line of future research.

Possible directions	Our approach		LvLiNGAM		SLIM		
	t-dist.	Gaussian	Num. lat. conf.		Num. lat. conf.		
			1	4	1	4	10
$x_1(FO) \leftarrow x_3(FE)$	✓	✓		✓			✓
$x_2(SI) \leftarrow x_1(FO)$	✓	✓				✓	✓
$x_2(SI) \leftarrow x_3(FE)$	✓	✓			✓	✓	✓
$x_2(SI) \leftarrow x_4(SO)$	✓	✓			✓	✓	
$x_2(SI) \leftarrow x_5(SE)$	✓	✓	✓	✓	✓		✓
$x_2(SI) \leftarrow x_6(NS)$	✓	✓	✓	✓			
$x_4(SO) \leftarrow x_1(FO)$	✓	✓	✓	✓	✓	✓	✓
$x_4(SO) \leftarrow x_3(FE)$	✓	✓	✓		✓	✓	✓
$x_4(SO) \leftarrow x_5(SE)$	✓	✓	✓	✓			
$x_4(SO) \leftarrow x_6(NS)$	✓	✓	✓	✓	✓		
$x_5(SE) \leftarrow x_1(FO)$					✓		
$x_5(SE) \leftarrow x_3(FE)$	✓		✓	✓	✓	✓	
$x_5(SE) \leftarrow x_6(NS)$	✓	✓	✓	✓	✓		
$x_6(NS) \leftarrow x_1(FO)$			✓				✓
$x_6(NS) \leftarrow x_3(FE)$			✓	✓		✓	✓
Num. of successes	12	11	10	9	9	7	8
Precisions	0.80	0.73	0.67	0.60	0.60	0.47	0.53

Possible directions	LiNGAM-GC-UK	ICA	Direct	Pairwise	PNL
$x_1(FO) \leftarrow x_3(FE)$		✓	✓		
$x_2(SI) \leftarrow x_1(FO)$		✓	✓		✓
$x_2(SI) \leftarrow x_3(FE)$		✓			✓
$x_2(SI) \leftarrow x_4(SO)$		✓	✓		✓
$x_2(SI) \leftarrow x_5(SE)$		✓			✓
$x_2(SI) \leftarrow x_6(NS)$		✓			✓
$x_4(SO) \leftarrow x_1(FO)$			✓	✓	
$x_4(SO) \leftarrow x_3(FE)$			✓		✓
$x_4(SO) \leftarrow x_5(SE)$		✓			✓
$x_4(SO) \leftarrow x_6(NS)$		✓			
$x_5(SE) \leftarrow x_1(FO)$	✓		✓	✓	
$x_5(SE) \leftarrow x_3(FE)$			✓		✓
$x_5(SE) \leftarrow x_6(NS)$					
$x_6(NS) \leftarrow x_1(FO)$	✓		✓		
$x_6(NS) \leftarrow x_3(FE)$	✓		✓		✓
Num. of successes	3	8	9	2	9
Precisions	0.20	0.53	0.60	0.13	0.60

FO: Father's Occupation
FE: Father's Education
SI: Son's Income
SO: Son's Occupation
SE: Son's Education
NS: Number of Siblings

ICA: ICA-LiNGAM (Shimizu et al., 2006)
Direct: DirectLiNGAM (Shimizu et al., 2011)
Pairwise: Pairwise LiNGAM (Hyvärinen and Smith, 2013)
PNL: Post-nonlinear causal model
(Zhang and Hyvärinen, 2009)

Table 5: Comparison of eight methods

Pairs analyzed	Possible directions	Estimated directions	$\hat{\tau}_1^{indvdl}$	$\hat{\tau}_2^{indvdl}$	$\hat{\sigma}_{12}$
$(x_1(FO), x_3(FE))$	$\leftarrow$	$\leftarrow$	$0.4^2\widehat{\text{var}}(x_1)$	0	-0.7
$(x_2(SI), x_1(FO))$	$\leftarrow$	$\leftarrow$	$0.8^2\widehat{\text{var}}(x_2)$	0	0.3
$(x_2(SI), x_3(FE))$	$\leftarrow$	$\leftarrow$	$0.8^2\widehat{\text{var}}(x_2)$	0	-0.5
$(x_2(SI), x_4(SO))$	$\leftarrow$	$\leftarrow$	$0.2^2\widehat{\text{var}}(x_2)$	$0.4^2\widehat{\text{var}}(x_4)$	-0.5
$(x_2(SI), x_5(SE))$	$\leftarrow$	$\leftarrow$	0	$0.4^2\widehat{\text{var}}(x_5)$	0
$(x_2(SI), x_6(NS))$	$\leftarrow$	$\leftarrow$	$1.0^2\widehat{\text{var}}(x_2)$	$0.6^2\widehat{\text{var}}(x_6)$	0
$(x_4(SO), x_1(FO))$	$\leftarrow$	$\leftarrow$	$1.0^2\widehat{\text{var}}(x_4)$	0	0.9
$(x_4(SO), x_3(FE))$	$\leftarrow$	$\leftarrow$	0	$0.2^2\widehat{\text{var}}(x_3)$	-0.3
$(x_4(SO), x_5(SE))$	$\leftarrow$	$\leftarrow$	0	0	-0.3
$(x_4(SO), x_6(NS))$	$\leftarrow$	$\leftarrow$	$0.6^2\widehat{\text{var}}(x_4)$	0	-0.7
$(x_5(SE), x_1(FO))$	$\leftarrow$	$\rightarrow$	0	$0.8^2\widehat{\text{var}}(x_1)$	0.3
$(x_5(SE), x_3(FE))$	$\leftarrow$	$\leftarrow$	$0.6^2\widehat{\text{var}}(x_5)$	0	-0.5
$(x_5(SE), x_6(NS))$	$\leftarrow$	$\leftarrow$	$0.2^2\widehat{\text{var}}(x_5)$	0	-0.3
$(x_6(NS), x_1(FO))$	$\leftarrow$	$\rightarrow$	$0.2^2\widehat{\text{var}}(x_6)$	0	-0.9
$(x_6(NS), x_3(FE))$	$\leftarrow$	$\rightarrow$	$0.2^2\widehat{\text{var}}(x_6)$	$0.6^2\widehat{\text{var}}(x_3)$	0.5

FO: Father's Occupation
FE: Father's Education
SI: Son's Income
SO: Son's Occupation
SE: Son's Education
NS: Number of Siblings

τ_1^{indvdl} and τ_2^{indvdl} represent the variances of the individual-specific effects for the variable pairs in the left-most column.
σ_{12} represents the correlation parameter value of the individual-specific effects for the variable pairs in the left-most column.

Table 6: Estimated hyper-parameter values of our method with t-distributed individual-specific effects

6. Conclusions and Future Work

We proposed a new variant of LiNGAM that incorporated individual-specific effects in order to allow latent confounders. We further proposed an empirical Bayesian approach to estimate the possible causal direction of two observed variables based on the new model. In experiments on artificial data and real-world sociology data, the performance of our method was better than or at least comparable to that of existing methods.

For more than two variables, one approach would be to apply our method on every pair of the variables. Then, we can estimate a causal ordering of all the variables by integrating the estimation results. This approach is computationally much simper than trying all the possible causal orderings. Once a causal ordering of the variables is estimated, the remaining problem is to estimate regression coefficients or their posterior distributions. Then, one can see if there are direct causal connections between these variables. Although this could still be computationally challenging for large numbers of variables, the problem reduces to a significantly simpler one by identifying their causal orders. Thus, it is sensible to develop methods that can estimate causal direction of two variables allowing latent confounders.

A reviewer suggested that we can generalize our model to more than two variables. Instead of a two-equation system in Table 1 we could have any number of equations each with an individual-specific confounder variable, although this approach would be computationally challenging.

Future work will focus on extending the model to allow cyclic and nonlinear relations and a wider class of non-Gaussian distributions as well as evaluating our method on various real-world data. Another important direction is to investigate the degree to which the model selection is sensitive to the choice of prior distributions.

Acknowledgments

S.S. was supported by KAKENHI #24700275. We thank Aapo Hyvärinen, Ricardo Silva and three reviewers for their helpful comments.

References

F. R. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

P. Billingsley. *Probability and Measure*. Wiley-Interscience, 1986.

K. Bollen. *Structural Equations with Latent Variables*. John Wiley & Sons, 1989.

Z. Chen and L. Chan. Causality in linear nonGaussian acyclic models in the presence of latent Gaussian confounders. *Neural Computation*, 25(6):1605–1641, 2013.

D. M. Chickering and J. Pearl. A clinician's tool for analyzing non-compliance. In *Proc. 13th National Conference on Artificial Intelligence (AAAI1996)*, pages 1269–1276, 1996.

P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36:62–83, 1994.

E. Demidenko. *Mixed Models: Theory and applications*. Wiley-Interscience, 2004.

Y. Dodge and V. Rousson. On asymmetric properties of the correlation coefficient in the regression setting. *The American Statistician*, 55(1):51–54, 2001.

O. D. Duncan, D. L. Featherman, and B. Duncan. *Socioeconomic Background and Achievement*. Seminar Press, New York, 1972.

D. Entner and P. O. Hoyer. Discovering unconfounded causal relationships using linear non-gaussian models. In *New Frontiers in Artificial Intelligence, Lecture Notes in Computer Science*, volume 6797, pages 181–195, 2011.

J. Eriksson and V. Koivunen. Identifiability and separability of linear ICA models revisited. In *Proc. Fourth International Conference on Independent Component Analysis and Blind Signal Separation (ICA2003)*, pages 23–27, 2003.

C. W. J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–438, 1969.

R. Henao and O. Winther. Sparse linear identifiable multivariate modeling. *Journal of Machine Learning Research*, 12:863–905, 2011.

P. O. Hoyer and A. Hyttinen. Bayesian discovery of linear acyclic causal models. In *Proc. 25th Conference on Uncertainty in Artificial Intelligence (UAI2009)*, pages 240–248, 2009.

P. O. Hoyer, A. Hyvärinen, R. Scheines, P. Spirtes, J. Ramsey, G. Lacerda, and S. Shimizu. Causal discovery of linear acyclic models with arbitrary distributions. In *Proc. 24th Conference on Uncertainty in Artificial Intelligence (UAI2008)*, pages 282–289, 2008a.

P. O. Hoyer, S. Shimizu, A. Kerminen, and M. Palviainen. Estimation of causal effects using linear non-Gaussian causal models with hidden variables. *International Journal of Approximate Reasoning*, 49(2):362–378, 2008b.

P. O. Hoyer, D. Janzing, J. Mooij, J. Peters, and B. Schölkopf. Nonlinear causal discovery with additive noise models. In *Advances in Neural Information Processing Systems 21*, pages 689–696. 2009.

A. Hyvärinen and S. M. Smith. Pairwise likelihood ratios for estimation of non-Gaussian structural equation models. *Journal of Machine Learning Research*, 14:111–152, 2013.

A. Hyvärinen, P. O. Hoyer, and M. Inki. Topographic independent component analysis. *Neural Computation*, 13(7):1527–1558, 2001a.

A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley, New York, 2001b.

A. Hyvärinen, K. Zhang, S. Shimizu, and P. O. Hoyer. Estimation of a structural vector autoregressive model using non-Gaussianity. *Journal of Machine Learning Research*, 11: 1709–1731, 2010.

R. E. Kass and A. E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795, 1995.

S. Kotz and S. Nadarajah. *Multivariate t-distributions and Their Applications*. Cambridge University Press, 2004.

I. G. G. Kreft and J. De Leeuw. *Introducing Multilevel Modeling*. Sage, 1998.

G. Lacerda, P. Spirtes, J. Ramsey, and P. O. Hoyer. Discovering cyclic causal models by independent components analysis. In *Proc. 24th Conference on Uncertainty in Artificial Intelligence (UAI2008)*, pages 366–374, 2008.

M. Lewicki and T. J. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12(2):337–365, 2000.

C. Meek. Strong completeness and faithfulness in Bayesian networks. In *Proc. 11th Conference on Uncertainty in Artificial Intelligence*, pages 411–418, 1995.

A. Moneta, N. Chlaß, D. Entner, and P. Hoyer. Causal search in structural vector autoregressive models. In *Journal of Machine Learning Research: Workshop and Conference Proceedings, Causality in Time Series (Proc. NIPS2009 Mini-Symposium on Causality in Time Series)*, volume 12, pages 95–114, 2011.

A. Moneta, D. Entner, P.O. Hoyer, and A. Coad. Causal inference by independent component analysis: Theory and applications. *Oxford Bulletin of Economics and Statistics*, 75 (5):705–730, 2013.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000. (2nd ed. 2009).

J. Peters, D. Janzing, and B. Schölkopf. Causal inference on discrete data using additive noise models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12): 2436–2450, 2011a.

J. Peters, J. Mooij, D. Janzing, and B. Schölkopf. Identifiability of causal graphs using functional models. In *Proc. 27th Conference on Uncertainty in Artificial Intelligence (UAI2011)*, pages 589–598, 2011b.

J. D. Ramsey, R. Sanchez-Romero, and C. Glymour. Non-Gaussian methods and high-pass filters in the estimation of effective connections. *NeuroImage*, 84(1):986–1006, 2014.

T. Rosenström, M. Jokela, S. Puttonen, M. Hintsanen, L. Pulkki-Råback, J. S. Viikari, O. T. Raitakari, and L. Keltikangas-Järvinen. Pairwise measures of causal direction in the epidemiology of sleep problems and depression. *PloS ONE*, 7(11):e50841, 2012.

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.

S.M. Smith, K.L. Miller, G. Salimi-Khorshidi, M. Webster, C.F. Beckmann, T.E. Nichols, J.D. Ramsey, and M.W. Woolrich. Network modelling methods for FMRI. *NeuroImage*, 54(2):875–891, 2011.

Y. Sogawa, S. Shimizu, T. Shimamura, A. Hyvärinen, T. Washio, and S. Imoto. Estimating exogenous variables in data with more variables than observations. *Neural Networks*, 24 (8):875–880, 2011.

P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9:67–72, 1991.

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

P. Spirtes, C. Glymour, R. Scheines, and R. Tillman. Automated search for causal relations: Theory and practice. In R. Dechter, H. Geffner, and J. Halpern, editors, *Heuristics, Probability, and Causality: A Tribute to Judea Pearl*, pages 467–506. College Publications, 2010.

A. Statnikov, M. Henaff, N. I. Lytkin, and C. F. Aliferis. New methods for separating causes from effects in genomics data. *BMC Genomics*, 13(Suppl 8):S22, 2012.

R. E. Tillman, A. Gretton, and P. Spirtes. Nonlinear directed acyclic structure learning with weakly additive noise models. In *Advances in Neural Information Processing Systems 22*, pages 1847–1855, 2010.

A. von Eye and L. R. Bergman. Research strategies in developmental psychopathology: Dimensional identity and the person-oriented approach. *Development and Psychopathology*, 15(3):553–580, 2003.

K. Zhang and A. Hyvärinen. On the identifiability of the post-nonlinear causal model. In *Proc. 25th Conference in Uncertainty in Artificial Intelligence (UAI2009)*, pages 647–655, 2009.

K. Zhang, B. Schölkopf, and D. Janzing. Invariant Gaussian process latent variable models and application in causal discovery. In *Proc. 26th Conference in Uncertainty in Artificial Intelligence (UAI2010)*, pages 717–724, 2010.

Journal of Machine Learning Research 15 (2014) 2653-2687 Submitted 11/12; Revised 12/13; Published 8/14

A Truncated EM Approach for Spike-and-Slab Sparse Coding

Abdul-Saboor Sheikh* SHEIKH@TU-BERLIN.DE
Jacquelyn A. Shelton* SHELTON@TU-BERLIN.DE
Faculty of Electrical Engineering and Computer Science
Technical University Berlin
Marchstr. 23, 10587 Berlin, Germany

Jörg Lücke* JOERG.LUECKE@UNI-OLDENBURG.DE
Cluster of Excellence Hearing4all and Faculty VI
University of Oldenburg
26115 Oldenburg, Germany

Editor: Aapo Hyvärinen

Abstract

We study inference and learning based on a sparse coding model with 'spike-and-slab' prior. As in standard sparse coding, the model used assumes independent latent sources that linearly combine to generate data points. However, instead of using a standard sparse prior such as a Laplace distribution, we study the application of a more flexible 'spike-and-slab' distribution which models the absence or presence of a source's contribution independently of its strength if it contributes. We investigate two approaches to optimize the parameters of spike-and-slab sparse coding: a novel truncated EM approach and, for comparison, an approach based on standard factored variational distributions. The truncated approach can be regarded as a variational approach with truncated posteriors as variational distributions. In applications to source separation we find that both approaches improve the state-of-the-art in a number of standard benchmarks, which argues for the use of 'spike-and-slab' priors for the corresponding data domains. Furthermore, we find that the truncated EM approach improves on the standard factored approach in source separation tasks—which hints to biases introduced by assuming posterior independence in the factored variational approach. Likewise, on a standard benchmark for image denoising, we find that the truncated EM approach improves on the factored variational approach. While the performance of the factored approach saturates with increasing numbers of hidden dimensions, the performance of the truncated approach improves the state-of-the-art for higher noise levels.

Keywords: sparse coding, spike-and-slab distributions, approximate EM, variational Bayes, unsupervised learning, source separation, denoising

∗. Parts of this study were done at the Frankfurt Institute for Advanced Studies, Goethe-University Frankfurt am Main, Germany, the previous affiliation of the authors.

1. Introduction

Much attention has recently been devoted to studying sparse coding models with 'spike-and-slab' distribution as a prior over the latent variables (Goodfellow et al., 2013; Mohamed et al., 2012; Lücke and Sheikh, 2012; Titsias and Lazaro-Gredilla, 2011; Carbonetto and Stephen, 2011; Knowles and Ghahramani, 2011; Yoshida and West, 2010). In general, a 'spike-and-slab' distribution is comprised of a binary (the 'spike') and a continuous (the 'slab') part. The distribution generates a random variable by multiplying together the two parts such that the resulting value is either exactly zero (due to the binary random variable being zero) or it is a value drawn from a distribution governing the continuous part. In sparse coding models, employing spike-and-slab as a prior allows for modeling the presence or absence of latents independently of their contributions in generating an observation. For example, piano keys (as latent variables) are either pressed or not (binary part), and if they are pressed, they result in sounds with different intensities (continuous part). The sounds generated by a piano are also sparse in the sense that of all keys only a relatively small number is pressed on average.

Spike-and-slab distributions can flexibly model an array of sparse distributions, making them desirable for many types of data. Algorithms based on spike-and-slab distributions have successfully been used, e.g., for deep learning and transfer learning (Goodfellow et al., 2013), regression (West, 2003; Carvalho et al., 2008; Carbonetto and Stephen, 2011; Titsias and Lazaro-Gredilla, 2011), or denoising (Zhou et al., 2009; Titsias and Lazaro-Gredilla, 2011), and often represent the state-of-the-art on given benchmarks (compare Titsias and Lazaro-Gredilla, 2011; Goodfellow et al., 2013).

The general challenge with spike-and-slab sparse coding models lies in the optimization of the model parameters. Whereas the standard Laplacian prior used for sparse coding results in uni-modal posterior distributions, the spike-and-slab prior results in multi-modal posteriors (see, e.g., Titsias and Lazaro-Gredilla, 2011; Lücke and Sheikh, 2012). Figure 1 shows typical posterior distributions for spike-and-slab sparse coding (the model will be formally defined in the next section). The figure illustrates posterior examples for the case of a two-dimensional observed and a two-dimensional hidden space. As can be observed, the posteriors have multiple modes; and the number modes increases exponentially with the dimensionality of the hidden space (Titsias and Lazaro-Gredilla, 2011; Lücke and Sheikh, 2012). The multi-modal structure of the posteriors argues against the application of the standard maximum a-posteriori (MAP) approaches (Mairal et al., 2009; Lee et al., 2007; Olshausen and Field, 1997) or Gaussian approximations of the posterior (Seeger, 2008; Ribeiro and Opper, 2011) because they rely on uni-modal posteriors. The approaches that have been proposed in the literature are, consequently, MCMC based methods (e.g., Carvalho et al., 2008; Zhou et al., 2009; Mohamed et al., 2012) and variational EM methodologies (e.g., Zhou et al., 2009; Titsias and Lazaro-Gredilla, 2011; Goodfellow et al., 2013). While MCMC approaches are more general and more accurate given sufficient computational resources, variational approaches are usually more efficient. Especially in high dimensional hidden spaces, the multi-modality of the posteriors is a particular challenge for MCMC approaches; consequently, recent applications to large hidden spaces have been based on variational EM optimization (Titsias and Lazaro-Gredilla, 2011; Goodfellow et al., 2013). The variational approaches applied to spike-and-slab models thus far (see Rattray et al.,

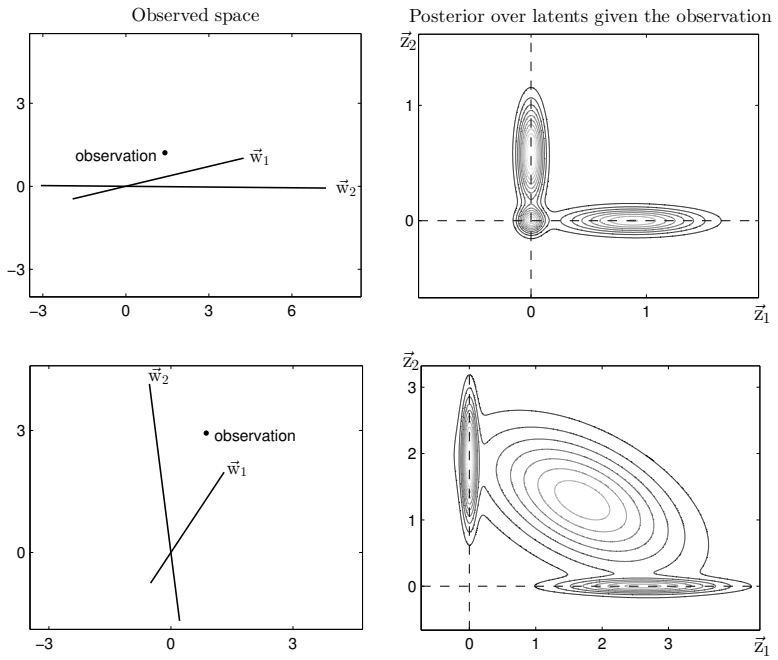


Figure 1: Left figures visualize observations generated by two different instantiations of the spike-and-slab sparse coding model (1) to (3). Solid lines are the generating bases vectors. Right figures illustrate the corresponding exact posteriors over latents computed using (13) and (15) given observations and generating model parameters. The probability mass seen just along the axes or around the origin actually lies exactly on the axis. Here we have spread the mass for visualization purposes by slightly augmenting zero diagonal entries of the posterior covariance matrix in (15).

2009; Yoshida and West, 2010; Titsias and Lazaro-Gredilla, 2011; Goodfellow et al., 2013) assume a factorization of the posteriors over the latent dimensions, that is the hidden dimensions are assumed to be independent a-posteriori. This means that any dependencies such as explaining-away effects including correlations (compare Figure 1) are ignored and not accounted for. But what consequences does such a negligence of posterior structure have? Does it result in biased parameter estimates and is it relevant for practical tasks? Biases induced by factored variational inference in latent variable models have indeed been observed before (MacKay, 2001; Ilin and Valpola, 2005; Turner and Sahani, 2011). For instance, in source separation tasks, optimization through factored inference can be biased towards finding mixing matrices that represent orthogonal sparse directions, because such solutions are most consistent with the assumed a-posteriori independence (see Ilin and Valpola, 2005, for a detailed discussion). Therefore, the posterior independence assumption in general may result in suboptimal solutions.

In this work we study an approximate EM approach for spike-and-slab sparse coding which does not assume a-posteriori independence and which can model multiple modes. The novel approach can be considered as a variational EM approach but instead of using factored distributions or Gaussians, it is based on posterior distributions truncated to regions of high probability mass (Lücke and Eggert, 2010). Such truncated EM approaches have recently been applied to different models (see e.g., Puertas et al., 2010; Shelton et al., 2011; Dai and Lücke, 2012; Bornschein et al., 2013). In contrast to the previously studied factored variational approaches (Titsias and Lazaro-Gredilla, 2011; Mohamed et al., 2012; Goodfellow et al., 2013), the truncated approach will furthermore take advantage of the fact that in the case of a Gaussian slab and Gaussian noise model, integrals over the continuous latents can be obtained in closed-form (Lücke and Sheikh, 2012). This implies that the posteriors over latent space can be computed exactly if the sums over the binary part are exhaustively evaluated over exponentially many states. This enumeration of the binary part becomes computationally intractable for high-dimensional hidden spaces. However, by applying the truncated variational approach exclusively to the binary part of the hidden space, we can still fully benefit from the analytical tractability of the continuous integrals.

In this study, we systematically compare the truncated approach to a recently suggested factored variational approach (Titsias and Lazaro-Gredilla, 2011). A direct comparison of the two variational approaches will allow for answering the questions about potential drawbacks and biases of both optimization procedures. As approaches assuming factored variational approximations have recently shown state-of-the-art performances (Titsias and Lazaro-Gredilla, 2011; Goodfellow et al., 2013), understanding their strengths and weaknesses is crucial for further advancements of sparse coding approaches and their many applications. Comparison with other approaches that are not necessarily based on the spike-and-slab model will allow for accessing the potential advantages of the spike-and-slab model itself.

In Section 2 we will introduce the used spike-and-slab sparse coding generative model, and briefly discuss the factored variational approach which has recently been applied for parameter optimization. In Section 3 we derive the closed-form EM parameter update equations for the introduced spike-and-slab model. Based on these equations, in Section 4 we derive the truncated EM algorithm for efficient learning in high dimensions. In Section 5, we numerically evaluate the algorithm and compare it to factored variational and other approaches. Finally, in Section 6 we discuss the results. The Appendix present details of the derivations and experiments.

2. Spike-and-slab Sparse Coding

The spike-and-slab sparse coding model assumes like standard sparse coding a linear superposition of basis functions, independent latents, and Gaussian observation noise. The main difference is that a spike-and-slab distribution is used as a prior. Spike-and-slab distributions have long been used for different models (e.g., Mitchell and Beauchamp, 1988, among many others) and also variants of sparse coding with spike-and-slab priors have been studied previously (compare West, 2003; Garrigues and Olshausen, 2007; Knowles and Ghahramani, 2007; Teh et al., 2007; Carvalho et al., 2008; Paisley and Carin, 2009; Zhou et al., 2009). In this work we study a generalization of the spike-and-slab sparse coding model studied by

Lücke and Sheikh (2012). The data generation process in the model assumes an independent Bernoulli prior for each component of the the binary latent vector $\vec{s} \in \{0,1\}^H$ and a multivariate Gaussian prior for the continuous latent vector $\vec{z} \in \mathbb{R}^H$:

$$p(\vec{s}\,|\Theta) = \mathcal{B}(\vec{s};\vec{\pi}) = \prod_{h=1}^{H} \pi_h^{s_h}\,(1-\pi_h)^{1-s_h}, \tag{1}$$

$$p(\vec{z}\,|\Theta) = \mathcal{N}(\vec{z};\,\vec{\mu},\Psi), \tag{2}$$

where π_h defines the probability of s_h being equal to one and where $\vec{\mu}$ and Ψ parameterize the mean and covariance of $\vec{z}$, respectively. The parameters $\vec{\mu} \in \mathbb{R}^H$ and $\Psi \in \mathbb{R}^{H\times H}$ parameterizing the Gaussian slab in (2) generalize the spike-and-slab model used in (Lücke and Sheikh, 2012). A point-wise multiplication of the two latent vectors, i.e., $(\vec{s}\odot\vec{z})_h = s_h\,z_h$ generates a 'spike-and-slab' distributed variable $(\vec{s}\odot\vec{z})$, which has either continuous values or exact zero entries. Given such a latent vector, a D-dimensional observation $\vec{y} \in \mathbb{R}^D$ is generated by linearly superimposing a set of basis functions W and by adding Gaussian noise:

$$p(\vec{y}\,|\,\vec{s},\vec{z},\Theta) = \mathcal{N}(\vec{y};\,W(\vec{s}\odot\vec{z}),\Sigma), \tag{3}$$

where each column of the matrix $W \in \mathbb{R}^{D\times H}$ is a basis function $W = (\vec{w}_1,\ldots,\vec{w}_H)$ and where the matrix $\Sigma \in \mathbb{R}^{D\times D}$ parameterizes the observation noise. Full rank covariances Σ can flexibly parametrize noise and have been found beneficial in noisy environments (Dalen and Gales, 2008; Ranzato and Hinton, 2010; Dalen and Gales, 2011). Nevertheless the model can also be constrained to have homoscedastic noise (i.e., $\Sigma = \sigma^2 I$). We use $\Theta = (W,\Sigma,\vec{\pi},\vec{\mu},\Psi)$ to denote all the model parameters. Having a spike-and-slab prior implies that for high levels of sparsity (low values of π_h) the latents assume exact zeros with a high probability. This is an important distinction compared to the Laplace or Cauchy distributions used for standard sparse coding (Olshausen and Field, 1997).

The spike-and-slab sparse coding algorithm we derive in this work is based on the model (1) to (3). The factored variational approach (Multi–Task and Multiple Kernel Learning, MTMKL; Titsias and Lazaro-Gredilla, 2011) that we use for detailed comparison is based on a similar model. The MTMKL model is both a constrained and generalized version of the model we study. On one hand, it is more constrained by assuming the same sparsity for each latent, i.e., $\pi_h = \pi_{h'}$ (for all h,h'); and by using a diagonal covariance matrix for the observation noise, $\Sigma = \mathrm{diag}(\sigma_1^2,\ldots,\sigma_D^2)$. On the other hand, it is a generalization by drawing the basis functions W from Gaussian processes. The model (1) to (3) can then be recovered as a special case of the MTMKL model if the Gaussian processes are Dirac delta functions. For parameter optimization, the MTMKL model uses a standard factored variational optimization. In the case of spike-and-slab models, this factored approach means that the exact posterior $p(\vec{s},\vec{z}\,|\,\vec{y})$ is approximated by a variational distribution $q_n(\vec{s},\vec{z};\,\Theta)$ which assumes the combined latents to be independent a-posteriori (compare Zhou et al., 2009; Titsias and Lazaro-Gredilla, 2011; Goodfellow et al., 2013):

$$q_n(\vec{s},\vec{z};\Theta) = \prod_{h=1}^{H} q_n^{(h)}(s_h,z_h;\Theta),$$

where $q_n^{(h)}$ are distributions only depending on s_h and z_h and not on any of the other latents. A detailed account of the MTMKL optimization algorithm is given by Titsias and Lazaro-Gredilla (2011) and for later numerical experiments on the model, we used the source code provided along with that publication.[1] Further comparisons will include the spike-and-slab sparse coding model by Zhou et al. (2009). The generative model is similar to the spike-and-slab model in Equations (1) to (3) but uses a Beta process prior to parameterize the Bernoulli (the "spike") distribution and assumes homoscedastic observation noise. Inference in their model is based on factored variational EM or Gibbs sampling. As this model is closely related to ours, we use it as another instance for comparison in our numerical experiments in order to assess the influence of different inference method choices. This comparison allows us to explore differences of training the model with a sampling-based approach, as they yield many of the same benefits of our inference method (e.g., flexible representation of uncertainty), but where generally more computational resources are necessary.

3. Expectation Maximization for Parameter Optimization

In order to learn the model parameters Θ given a set of N independent data points $\{\vec{y}^{(n)}\}_{n=1,\ldots,N}$ with $\vec{y}^{(n)} \in \mathbb{R}^D$, we maximize the data likelihood $\mathcal{L} = \prod_{n=1}^{N} p(\vec{y}^{(n)} \,|\, \Theta)$ by applying the Expectation Maximization (EM) algorithm. Instead of directly maximizing the likelihood, the EM algorithm (in the form studied by Neal and Hinton, 1998) maximizes the free-energy, a lower bound of the log-likelihood given by:

$$\mathcal{F}(\Theta^{\text{old}}, \Theta) = \sum_{n=1}^{N} \Big\langle \log p(\vec{y}^{(n)}, \vec{s}, \vec{z} \,|\, \Theta) \Big\rangle_n + H(\Theta^{\text{old}}), \tag{4}$$

where $\langle \cdot \rangle_n$ denotes the expectation under the posterior over the latents $\vec{s}$ and $\vec{z}$ given $\vec{y}^{(n)}$

$$\big\langle f(\vec{s}, \vec{z}) \big\rangle_n = \sum_{\vec{s}} \int_{\vec{z}} p(\vec{s}, \vec{z} \,|\, \vec{y}^{(n)}, \Theta^{\text{old}})\, f(\vec{s}, \vec{z})\, \mathrm{d}\vec{z} \tag{5}$$

and $H(\Theta^{\text{old}}) = -\sum_{\vec{s}} \int_{\vec{z}} p(\vec{s}, \vec{z} \,|\, \vec{y}^{(n)}, \Theta^{\text{old}}) \log(p(\vec{s}, \vec{z} \,|\, \vec{y}^{(n)}, \Theta^{\text{old}}))\, \mathrm{d}\vec{z}$ is the Shannon entropy, which only depends on parameter values held fixed during the optimization of $\mathcal{F}$ w.r.t. Θ in the M-step. Here $\sum_{\vec{s}}$ is a summation over all possible binary vectors $\vec{s}$.

The EM algorithm iteratively optimizes the free-energy by alternating between two steps. First, in the E-step given the current parameters Θ^{old}, the relevant expectation values under the posterior $p(\vec{s}, \vec{z} \,|\, \vec{y}^{(n)}, \Theta^{\text{old}})$ are computed. Next, the M-step uses these posterior expectations and maximizes the free-energy $\mathcal{F}(\Theta^{\text{old}}, \Theta)$ w.r.t. Θ. Iteratively applying E- and M-steps locally maximizes the data likelihood. In the following section we will first derive the M-step equations which themselves will require expectation values over the posteriors (5). The required expressions and approximations for these expectations (the E-step) will be derived afterwards.

3.1 M-step Parameter Updates

The M-step parameter updates of the model are canonically obtained by setting the derivatives of the free-energy (4) w.r.t. the second argument to zero. Details of the derivations

1. We downloaded the code from `http://www.well.ox.ac.uk/~mtitsias/code/varSparseCode.tar.gz`.

are given in Appendix A and the resulting update equations are as follows:

$$W = \frac{\sum_{n=1}^{N} \vec{y}^{(n)} \left\langle \vec{s} \odot \vec{z} \right\rangle_n^{\mathrm{T}}}{\sum_{n=1}^{N} \left\langle (\vec{s} \odot \vec{z})(\vec{s} \odot \vec{z})^{\mathrm{T}} \right\rangle_n}, \tag{6}$$

$$\vec{\pi} = \frac{1}{N} \sum_{n=1}^{N} \left\langle \vec{s} \right\rangle_n, \tag{7}$$

$$\vec{\mu} = \frac{\sum_{n=1}^{N} \left\langle \vec{s} \odot \vec{z} \right\rangle_n}{\sum_{n=1}^{N} \left\langle \vec{s} \right\rangle_n}, \tag{8}$$

$$\Psi = \sum_{n=1}^{N} \left[\left\langle (\vec{s} \odot \vec{z})(\vec{s} \odot \vec{z})^{\mathrm{T}} \right\rangle_n - \left\langle \vec{s}\,\vec{s}^{\mathrm{T}} \right\rangle_n \odot \vec{\mu}\vec{\mu}^{\mathrm{T}} \right] \odot \left(\sum_{n=1}^{N} \left[\left\langle \vec{s}\,\vec{s}^{\mathrm{T}} \right\rangle_n \right] \right)^{-1}, \tag{9}$$

$$\text{and} \quad \Sigma = \frac{1}{N} \sum_{n=1}^{N} \left[\vec{y}^{(n)} (\vec{y}^{(n)})^{\mathrm{T}} - W \left[\left\langle (\vec{s} \odot \vec{z}) \right\rangle_n \left\langle (\vec{s} \odot \vec{z}) \right\rangle_n^{\mathrm{T}} \right] W^{\mathrm{T}} \right]. \tag{10}$$

3.2 E-step Expectation Values

The M-step equations (6) to (10) require expectation values w.r.t. the posterior distribution be computed over the whole latent space, which requires either analytical solutions or approximations of integrals/sums over the latent space. For the derivation of closed-form E-step equations it is useful to know that the discrete latent variable $\vec{s}$ can be combined with the basis function matrix W so that we can rewrite (3) as

$$p(\vec{y} \,|\, \vec{s}, \vec{z}, \Theta) = \mathcal{N}(\vec{y};\, \tilde{W}_{\vec{s}}\, \vec{z}, \Sigma),$$

where we have defined $(\tilde{W}_{\vec{s}})_{dh} = W_{dh} s_h$ such that $W(\vec{s} \odot \vec{z}) = \tilde{W}_{\vec{s}}\, \vec{z}$.

Here the data likelihood $p(\vec{y} \,|\, \Theta)$ can be derived in closed-form after marginalizing the joint $p(\vec{y}, \vec{s}, \vec{z} \,|\, \Theta)$ over $\vec{z}$:

$$\begin{aligned} p(\vec{y}, \vec{s} | \, \Theta) &= \mathcal{B}(\vec{s}; \vec{\pi}) \int \mathcal{N}(\vec{y}; \tilde{W}_{\vec{s}}\, \vec{z}, \Sigma)\, \mathcal{N}(\vec{z}; \vec{\mu}, \Psi)\, \mathrm{d}\vec{z} \\ &= \mathcal{B}(\vec{s}; \vec{\pi})\ \mathcal{N}(\vec{y}; \tilde{W}_{\vec{s}}\, \vec{\mu}, C_{\vec{s}}), \end{aligned} \tag{11}$$

where $C_{\vec{s}} = \Sigma + \tilde{W}_{\vec{s}}\, \Psi \tilde{W}_{\vec{s}}^{T}$. The second step follows from standard identities for Gaussian random variables (e.g., Bishop, 2006). We can then sum the resulting expression over $\vec{s}$ to obtain

$$p(\vec{y} \,|\, \Theta) = \sum_{\vec{s}} \mathcal{B}(\vec{s}; \vec{\pi})\ \mathcal{N}(\vec{y}; \tilde{W}_{\vec{s}}\, \vec{\mu}, C_{\vec{s}}). \tag{12}$$

Thus, the marginal distribution takes the form of a Gaussian mixture model with 2^H mixture components indexed by $\vec{s}$. However, unlike in a standard Gaussian mixture model, the mixing proportions and the parameters of the mixture components are not independent but coupled together. Therefore, the following steps will lead to closed-form EM updates that are notably not a consequence of closed-form EM for classical Gaussian mixtures. In contrast, Gaussian mixture models assume independent mixing proportions and independent

component parameters. By using Equations (11) and (12) the posterior over the binary latents $p(\vec{s}\,|\,\vec{y},\Theta)$ is given by:

$$p(\vec{s}\,|\,\vec{y},\Theta) = \frac{p(\vec{s},\vec{y}\,|\,\Theta)}{p(\vec{y}\,|\,\Theta)} = \frac{\mathcal{B}(\vec{s};\vec{\pi})\;\mathcal{N}(\vec{y};\tilde{W}_{\vec{s}}\,\vec{\mu},C_{\vec{s}})}{\sum_{\vec{s}'}\mathcal{B}(\vec{s}';\vec{\pi})\;\mathcal{N}(\vec{y};\;\tilde{W}_{\vec{s}'}\vec{\mu},C_{\vec{s}'})}. \tag{13}$$

We can now consider the factorization of the posterior $p(\vec{s},\vec{z}\,|\,\vec{y},\Theta)$ into the posterior over the binary part $p(\vec{s}\,|\,\vec{y},\Theta)$ and the posterior over the continuous part given the binary state $p(\vec{z}\,|\,\vec{s},\vec{y},\Theta)$:

$$p(\vec{s},\vec{z}\,|\,\vec{y},\Theta) = p(\vec{s}\,|\,\vec{y},\Theta)\,p(\vec{z}\,|\,\vec{s},\vec{y},\Theta). \tag{14}$$

Like the first factor in (14), the second factor is also analytically tractable and given by:

$$\begin{aligned} p(\vec{z}\,|\,\vec{s},\vec{y},\Theta) &= \frac{p(\vec{s}\,|\,\Theta)\,p(\vec{z}\,|\,\Theta)\,p(\vec{y}\,|\,\vec{z},\vec{s},\Theta)}{p(\vec{s}\,|\,\Theta)\int p(\vec{y}\,|\,\vec{z},\vec{s},\Theta)\,p(\vec{z}\,|\,\Theta)\mathrm{d}\vec{z}} \\ &\propto \mathcal{N}(\vec{z};\vec{\mu},\Psi)\,\mathcal{N}(\vec{y};\tilde{W}_{\vec{s}}\,\vec{z},\Sigma) \\ &= \mathcal{N}(\vec{z};\vec{\kappa}_{\vec{s}},\Lambda_{\vec{s}}), \end{aligned}$$

where the last step again follows from standard Gaussian identities with definitions

$$\begin{aligned} \Lambda_{\vec{s}} &= (\tilde{W}_{\vec{s}}^{\mathrm{T}}\,\Sigma^{-1}\,\tilde{W}_{\vec{s}} + \Psi_{\vec{s}}^{-1})^{-1}, \\ \vec{\kappa}_{\vec{s}}^{\,(n)} &= (\vec{s}\odot\vec{\mu}) + \Lambda_{\vec{s}}\tilde{W}_{\vec{s}}^{\mathrm{T}}\,\Sigma^{-1}\,(\vec{y}^{\,(n)} - \tilde{W}_{\vec{s}}\,\vec{\mu}) \end{aligned} \tag{15}$$

and with $\Psi_{\vec{s}} = \Psi\big(\mathrm{diag}(\vec{s})\big)$. The full posterior distribution can thus be written as

$$p(\vec{s},\vec{z}\,|\,\vec{y}^{\,(n)},\Theta) \;\;=\;\; \frac{\mathcal{B}(\vec{s};\vec{\pi})\,\mathcal{N}(\vec{y}^{\,(n)};\tilde{W}_{\vec{s}}\,\vec{\mu},C_{\vec{s}})\,\mathcal{N}(\vec{z};\;\vec{\kappa}_{\vec{s}}^{\,(n)},\Lambda_{\vec{s}})}{\sum_{\vec{s}'}\mathcal{B}(\vec{s}';\vec{\pi})\,\mathcal{N}(\vec{y}^{\,(n)};\tilde{W}_{\vec{s}'}\,\vec{\mu},C_{\vec{s}'})}. \tag{16}$$

Equation (16) represents the crucial result for the computation of the E-step below because, first, it shows that the posterior does not involve analytically intractable integrals and, second, for fixed $\vec{s}$ and $\vec{y}^{\,(n)}$ the dependency on $\vec{z}$ follows a Gaussian distribution. This special form allows for the derivation of analytical expressions for the expectation values as required for the M-step updates. Because of the Gaussian form, the integrations over the continuous part are straight-forward and the expectation values required for the M-step are given as follows:

$$\big\langle\vec{s}\big\rangle_n \;=\; \sum_{\vec{s}} q_n(\vec{s};\Theta)\,\vec{s}, \tag{17}$$

$$\big\langle\vec{s}\,\vec{s}^{\mathrm{T}}\big\rangle_n \;=\; \sum_{\vec{s}} q_n(\vec{s};\Theta)\,\vec{s}\,\vec{s}^{\mathrm{T}}, \tag{18}$$

$$\big\langle\vec{s}\odot\vec{z}\big\rangle_n \;=\; \sum_{\vec{s}} q_n(\vec{s};\Theta)\,\vec{\kappa}_{\vec{s}}^{\,(n)}, \tag{19}$$

$$\text{and}\quad \big\langle(\vec{s}\odot\vec{z})(\vec{s}\odot\vec{z})^{\mathrm{T}}\big\rangle_n \;=\; \sum_{\vec{s}} q_n(\vec{s};\Theta)\,\big(\Lambda_{\vec{s}} + \vec{\kappa}_{\vec{s}}^{\,(n)}(\vec{\kappa}_{\vec{s}}^{\,(n)})^{\mathrm{T}}\big). \tag{20}$$

In all of the expressions above, the left-hand-sides are expectation values over the full latent space w.r.t. the posterior $p(\vec{s},\vec{z}\,|\,\vec{y}^{\,(n)},\Theta)$, whereas the right-hand-sides now take the form of

expectation values only over the binary part w.r.t. the posterior $p(\vec{s}\,|\,\vec{y}^{\,(n)},\Theta)$ in Equation (13). The derivations of E-step equations (17) to (20) are a generalization of the derivations by Lücke and Sheikh (2012). While Gaussian identities and marginalization have been used to obtain analytical results for mixture-of-Gaussians priors before (e.g. Moulines et al., 1997; Attias, 1999; Olshausen and Millman, 2000; Garrigues and Olshausen, 2007), the above equations are the first closed-form solutions for the spike-and-slab model (first appearing in Lücke and Sheikh, 2012). The observation that the Gaussian slab and Gaussian noise model allows for analytically tractable integrals has, in parallel work, also been pointed out by Mohamed et al. (2012).

Iteratively computing the E-step equations (17) to (20) using the current parameters Θ and the M-step equations (6) to (10), represents a closed-form and exact EM algorithm which increases the data likelihood of the model to (possibly local) maxima.

4. Truncated EM

While being exact, the execution of the above EM algorithm results in considerable computational costs for larger-scale problems. Without approximations, the computational resources required scale exponentially with the number of hidden dimensions H. This can be seen by considering the expected values w.r.t. the posterior $p(\vec{s}\,|\,\vec{y},\Theta)$ above, which each require a summation over all binary vectors $\vec{s}\in\{0,1\}^H$. For tasks involving low dimensional hidden spaces, the exact algorithm is still applicable. For higher dimensional problems approximations are required, however. Still, we can make use of the closed-form EM solutions by applying an approximation solely to the binary part. Instead of sampling-based or factored approximations to the posterior $p(\vec{s},\vec{z}\,|\,\vec{y},\Theta)$, we use a truncated approximation to the posterior $p(\vec{s}\,|\,\vec{y}^{\,(n)},\Theta)$ in Equation (13). The truncated approximation is defined to be proportional to the true posteriors on subspaces of the latent space with high probability mass (compare *Expectation Truncation*, Lücke and Eggert, 2010). More concretely, a posterior distribution $p(\vec{s}\,|\,\vec{y}^{\,(n)},\Theta)$ is approximated by a distribution $q_n(\vec{s};\Theta)$ that only has support on a subset $\mathcal{K}_n\subseteq\{0,1\}^H$ of the state space:

$$q_n(\vec{s};\Theta)=\frac{p(\vec{s},\vec{y}^{\,(n)}\,|\,\Theta)}{\sum\limits_{\vec{s}\,'\in\mathcal{K}_n}p(\vec{s}\,',\vec{y}^{\,(n)}\,|\,\Theta)}\,\delta(\vec{s}\in\mathcal{K}_n),\tag{21}$$

where $\delta(\vec{s}\in\mathcal{K}_n)$ is an indicator function, i.e., $\delta(\vec{s}\in\mathcal{K}_n)=1$ if $\vec{s}\in\mathcal{K}_n$ and zero otherwise.

The basic assumption behind the approximation in (21) is that the posterior over the entire hidden space is concentrated in small volumes, which is represented by the reduced support of subset $\mathcal{K}_n$. When using a spike-and-slab sparse coding model to gain a generative understanding of the data, sparsity in the posterior distribution usually emerges naturally. We can see an illustration of this in Figure 2 (generation details in Section 5.4). Figure 2A shows (for three typical data points) how much posterior mass is carried by each of the $H=10$ latent dimensions. Figure 2B shows (for the same data points) histograms of the posterior mass marginalized across the whole range of hyperplanes spanned by the 10–dimensional latent space. Figure 2A indicates that only a subset of the H latents is significantly relevant for encoding the posterior, while Figure 2B allows us to observe that the posterior mass is primarily contained within low-dimensional hyperplanes of the H–

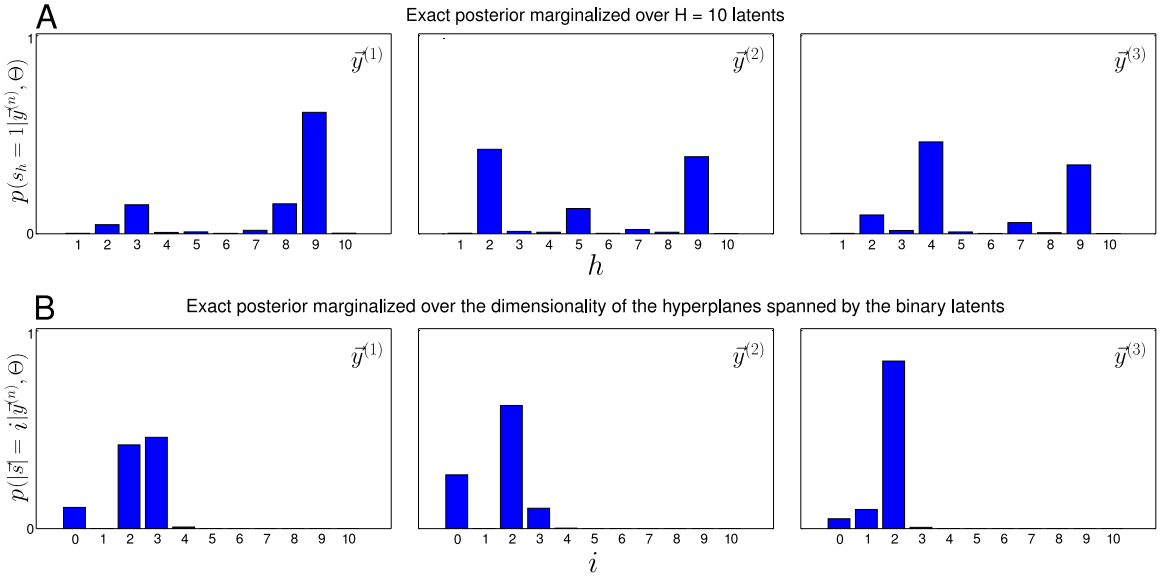


Figure 2: Visualization of the exact posterior probabilities of the spike-and-slab model with $H = 10$ latents, computed for three given data points $\vec{y}^{(n)}$. The model was trained on natural data (see Section 5.4 for more details). **A** Histograms of the posterior mass over the H latents: $p(s_h = 1 \,|\, \vec{y}^{(n)}, \Theta) = \sum_{\vec{s}_{\text{with}\, s_h=1}} p(\vec{s} \,|\, \vec{y}^{(n)}, \Theta) / \sum_{\vec{s}} p(\vec{s} \,|\, \vec{y}^{(n)}, \Theta)$. Low values for most h imply that these latents can be neglected (i.e., clamped to zero) for a posterior approximation. **B** Histograms of the posterior mass over the hyperplanes of increasing dimensionality i: $p(|\vec{s}| = i \,|\, \vec{y}^{(n)}, \Theta) = \sum_{\vec{s}_{\text{where}}|\vec{s}|=i} p(\vec{s} \,|\, \vec{y}^{(n)}, \Theta) / \sum_{i'=0}^{H} \sum_{\vec{s}_{\text{where}}|\vec{s}|=i'} p(\vec{s} \,|\, \vec{y}^{(n)}, \Theta)$. In case of all the three examples presented here, subspaces with $i > 4$ can be neglected as another approximation step for posterior estimation.

dimensional hidden space. In other words, given a data point we find that most of the posterior mass is concentrated in low-dimensional subspaces spanned by $H' \ll H$ of the latent dimensions. The sparse nature of the posterior as illustrated by Figure 2 allows us to apply approximation (21), where we define the subsets $\mathcal{K}_n$ based on index sets $I_n \subseteq \{1, \ldots, H\}$, which contain the indices of H' most relevant sparse latents (compare Figure 2A) for the corresponding data points $\vec{y}^{(n)}$:

$$\mathcal{K}_n = \{\vec{s} \,|\, \textstyle\sum_h s_h \leq \gamma \text{ and } \forall h \notin I_n : s_h = 0\} \cup \mathcal{U}, \tag{22}$$

where the indices comprising I_n have the highest value of a selection (or scoring) function $\mathcal{S}_h(\vec{y}^{(n)}, \Theta)$ (which we define later). The set $\mathcal{U}$ is defined as $\mathcal{U} = \{\vec{s} \,|\, \sum_h s_h = 1\}$ and ensures that $\mathcal{K}_n$ contains all singleton states (compare Lücke and Eggert, 2010). Otherwise, $\mathcal{K}_n$ only contains vectors with at most γ non-zero entries and with non-zero entries only permitted for $h \in I_n$. The parameter $\gamma \leqslant H'$ sets the maximal dimensionality of the considered hyper-planes (compare Figure 2B). It was empirically shown by Lücke and Eggert (2010)

that for appropriately defined subspaces $\mathcal{K}_n$, the KL-divergence between the true posteriors and their truncated approximations converges to values close to zero.

If we now use the concrete expressions of the sparse spike-and-slab model (Equations (1) to (3)) for the variational distribution in Equation (21), the truncated approximation is given by:

$$p(\vec{s}\,|\,\vec{y}^{\,(n)},\Theta) \;\approx\; q_n(\vec{s};\Theta) \;\;=\;\; \frac{\mathcal{N}(\vec{y}^{\,(n)};\tilde{W}_{\vec{s}}\,\vec{\mu},C_{\vec{s}})\,\mathcal{B}(\vec{s};\vec{\pi})}{\sum_{\vec{s}'\in\mathcal{K}_n}\mathcal{N}(\vec{y}^{\,(n)};\tilde{W}_{\vec{s}'}\,\vec{\mu},C_{\vec{s}'})\,\mathcal{B}(\vec{s}';\vec{\pi})}\,\delta(\vec{s}\in\mathcal{K}_n). \tag{23}$$

The approximation can now be used to compute the expectation values which are required for the M-step equations. If we use the variational distributions in Equation (23) for $q_n(\vec{s};\Theta)$ on the right-hand-sides of Equations (17) to (20), we obtain:

$$\sum_{\vec{s}} q_n(\vec{s};\Theta)\,f(\vec{s}) \;\;=\;\; \frac{\sum_{\vec{s}\in\mathcal{K}_n}\mathcal{N}(\vec{y}^{\,(n)};\tilde{W}_{\vec{s}}\,\vec{\mu},C_{\vec{s}})\,\mathcal{B}(\vec{s};\vec{\pi})\,f(\vec{s})}{\sum_{\vec{s}'\in\mathcal{K}_n}\mathcal{N}(\vec{y}^{\,(n)};\tilde{W}_{\vec{s}'}\,\vec{\mu},C_{\vec{s}'})\,\mathcal{B}(\vec{s}';\vec{\pi})}, \tag{24}$$

where $f(\vec{s})$ denotes any of the (possibly parameter dependent) functions of (17) to (20). Instead of having to compute sums over the entire binary state space with 2^H states, only sums over subsets $\mathcal{K}_n$ have to be computed. Since for many applications the posterior mass is finally concentrated in small volumes of the state space, the approximation quality can stay high even for relatively small sets $\mathcal{K}_n$.

Note that the definition of $q_n(\vec{s};\Theta)$ in Equation (21) neither assumes uni-modality like MAP approximations (Mairal et al., 2009; Lee et al., 2007; Olshausen and Field, 1997) or Gaussian approximations of the posterior (Ribeiro and Opper, 2011; Seeger, 2008), nor does it assume a-posteriori independence of the latents as factored approximations (Jordan et al., 1999; Goodfellow et al., 2013; Titsias and Lazaro-Gredilla, 2011). The approximation scheme we have introduced here exploits the inherent property of the sparse spike-and-slab model to have posterior probabilities concentrated in low-dimensional subspaces. The quality of our approximated posterior $q_n(\vec{s};\Theta)$ primarily depends on an appropriate selection of the relevant subspaces $\mathcal{K}_n$ (see Section 4.2 below).

The truncated approximation is similar to factored variational approximations or MAP approximations in the sense that it can be formulated as an approximate distribution $q_n(\vec{s};\Theta)$ within the free-energy formulation by Neal and Hinton (1998). Within this formulation, $q_n(\vec{s};\Theta)$ is often referred to as *variational* approximation, and we therefore refer to our approximation as *truncated variational EM*. Like factored variational approaches, we here aim to minimize the KL-divergence between the true posterior and the approximation in Equation (21). However, we do not use variational parameters and a gradient based optimization of such parameters for the minimization. Our approach is therefore not a variational approach in the sense of classical variational calculus.

4.1 Computational Complexity

The truncated E-step defined by (17) to (20) with (24) scales with the approximation parameters γ and H' which can be defined independently of the latent dimensionality H. The complexity scales as $\mathcal{O}\big(N\sum_{\gamma'=0}^{\gamma}\binom{H'}{\gamma'}(D{+}\gamma')^3\big)$, where the D^3 term can be dropped from the cubic expansion if the observed noise Σ is considered to be diagonal or homoscedastic.

Also the truncated approximation yields sparse matrices in Equations (21) and (23) which results in more efficient and tractable matrix operations.

Although the total number of data points N above defines a theoretical upper bound, in practice we can further benefit from the preselection step of the truncated approach to achieve significantly improved runtime performances. Clustering the data points using the index sets I_n saves us from redundantly performing various computationally expensive operations involved in Equations (15) and (23), that given a state $\vec{s} \in \mathcal{K}_n$ are independent of individual data points sharing the same subspace $\mathcal{K}_n$. Furthermore, such a batch processing strategy is also readily parallelizable as the truncated E-step can be performed independently for individual data clusters (see Appendix C for details). Using the batch execution mode we have observed an average runtime speedup of up to an order of magnitude.

4.2 Selection Function

To compose appropriate subspaces $\mathcal{K}_n$ a selection function $\mathcal{S}_h(\vec{y}^{(n)}, \Theta)$ is defined, which prior to each E-step allows us to select the relevant H' hidden dimensions (i.e., the elements of the index sets I_n) for a given observation $\vec{y}^{(n)}$. A selection function is essentially a heuristic-based scoring mechanism, that ranks all the latents based on their potential for being among the generating causes of a given observation. Selection functions can be based on upper bounds for probabilities $p(s_h = 1 \,|\, \vec{y}^{(n)}, \Theta)$ (compare Lücke and Eggert, 2010; Puertas et al., 2010) or deterministic functions such as the scalar product between a basis vector and a data point (derived from noiseless limits applied to observed space; compare Lücke and Eggert, 2010; Bornschein et al., 2013).

For the sparse coding model under consideration we define a selection function as follows:

$$\mathcal{S}_h(\vec{y}^{(n)}, \Theta) = \mathcal{N}(\vec{y}^{(n)}; \tilde{W}_{\vec{s}_h}\vec{\mu}, C_{\vec{s}_h}) \propto p(\vec{y}^{(n)} \,|\, \vec{s} = \vec{s}_h, \Theta), \tag{25}$$

where $\vec{s}_h$ represents a singleton state in which only the entry h is non-zero. The selection function (25) is basically the data likelihood given a singleton state $\vec{s}_h$. The function does not take into account the probability of the state itself (i.e., $p(\vec{s}_h \,|\, \Theta)$), as this may introduce a bias against less active latent dimensions. Similar to previously used selection functions (compare e.g., Lücke and Eggert, 2010; Puertas et al., 2010), in order to maintain a linear scaling behavior w.r.t. the number of latents, the selection function introduced here avoids computationally demanding higher-order combinatorics of the latents by only assessing one-to-one correspondences between individual latents and an observed data point. In the next section we empirically evaluate the efficacy of our selection function by means of numerical experiments that are based on the KL-divergence between the exact and the approximated posteriors computed from the subspaces $\mathcal{K}_n$.

Equations (21) to (23) replace the computation of the expectation values w.r.t. the exact posterior, and represent the approximate EM algorithm used in the experiments section. The algorithm will be applied without any further mechanisms such as annealing as we found it to be very robust in the form derived above. Furthermore, no data preprocessing such as mean subtraction or variance normalization will be used in any of the experiments. To distinguish the algorithm from others in comparative experiments, we will refer to it as *Gaussian Sparse Coding* (GSC) algorithm in order to emphasize the special Gaussian case of the spike-and-slab model used.

5. Numerical Experiments

We investigate the performance of the GSC algorithm on artificial data as well as various realistic source separation and denoising benchmarks. For all experiments the algorithm was implemented to run in parallel on multiple CPUs with no dependency on their arrangement as physically collocated arrays with shared memory or distributed among multiple compute nodes (see Bornschein et al., 2010, for more details). We further extended the basic technique to make our implementation more efficient and suitable for parallelization by applying the batch execution (the observation discussed in Section 4.1 on Computational Complexity and Appendix C). In all the experiments, the GSC model parameters were randomly initialized.[2] The choice of GSC truncation parameters H' and γ is in general straight-forward: the larger they are the closer the match to exact EM but the higher are also the computational costs. The truncation parameters are therefore capped by the available computational resources. However, empirically we observed that often much smaller values were sufficient than those that are maximally affordable.[3] Note that factored variational approaches do not usually offer such a trade-off between the exactness and computational demand of their inference schemes by means of a simple parameter adjustment.

5.1 Reliability of the Selection Function

To assess the reliability of the selection function we perform a number of experiments on small scale artificial data generated by the model, such that we can compute both the exact (13) and truncated (23) posteriors. To control for the quality of the truncated posterior approximation—and thus the selection function—we compute the ratio between posterior mass within the truncated space $\mathcal{K}_n$ and the overall posterior mass (compare Lücke and Eggert, 2010):

$$Q^{(n)} \;=\; \frac{\sum_{\vec{s}\in\mathcal{K}_n}\int_{\vec{z}} p(\vec{s},\vec{z}\,|\,\vec{y}^{\,(n)},\Theta)\,\mathrm{d}\vec{z}}{\sum_{\vec{s}'}\int_{\vec{z}'} p(\vec{s}',\vec{z}'\,|\,\vec{y}^{\,(n)},\Theta)\,\mathrm{d}\vec{z}'} \;=\; \frac{\sum_{\vec{s}\in\mathcal{K}_n}\mathcal{B}(\vec{s};\vec{\pi})\,\mathcal{N}(\vec{y}^{\,(n)};\tilde{W}_{\vec{s}}\,\vec{\mu},C_{\vec{s}})}{\sum_{\vec{s}'}\mathcal{B}(\vec{s}';\vec{\pi})\,\mathcal{N}(\vec{y}^{\,(n)};\tilde{W}_{\vec{s}'}\,\vec{\mu},C_{\vec{s}'})}, \qquad (26)$$

where the integrals over the latent $\vec{z}$ in (26) are again given in closed-form. The metric $Q^{(n)}$ ranges from zero to one and is directly related to the KL-divergence between the approximation q_n in Equation (23) and the true posterior:

$$D_{KL}(q_n(\vec{s},\vec{z};\,\Theta),p(\vec{s},\vec{z}\,|\,\vec{y}^{\,(n)},\Theta)) \;=\; -\log(Q^{(n)})\,.$$

If $Q^{(n)}$ is close to one, the KL-divergence is close to zero.

Data for the control experiments were generated by linearly superimposing basis functions that take the form of horizontal and vertical bars (see e.g., Földiák, 1990; Hoyer, 2002) on a $D = D_2 \times D_2$ pixel grid, where $D_2 = H/2$. This gives us D_2 possible horizontal as well as vertical locations for bars of length D_2, which together form our generating bases W^{gen}. Each bar is then randomly assigned either a positive or negative value with

2. We randomly and uniformly initialized the π_h between 0.05 and 0.95. $\vec{\mu}$ was initialized with normally distributed random values and the diagonal of Ψ was initialized with strictly positive uniformly distributed random values. We set Σ to the covariance across the data points, and the elements of W were independently drawn from a normal distribution with zero mean and unit variance.
3. Compare Appendix B for trade-off between complexity and accuracy of the truncated EM approach.

magnitude 10. We set the sparsity such that there are on average two active bars per data point, i.e., $\pi_h^{\text{gen}} = 2/H$ for all $h \in H$. We assume homoscedastic[4] observed noise $\Sigma^{\text{gen}} = \sigma^2 I_D$, where $\sigma^2 = 2.0$. The mean of the generating slab is i.i.d. drawn from a Gaussian: $\vec{\mu}^{\text{gen}} \sim \mathcal{N}(0, 5)$, and the covariance of the slab is $\Psi^{\text{gen}} = I_H$. We generate $N = 1000$ data points. We run experiments with different sets of values for the truncation parameters $(H', \gamma) \in \{(4, 4), (5, 4), (5, 3)\}$ for each $H \in \{10, 12\}$. Each run consists of 50 EM iterations and after each run we compute the Q-value over all the data points. For all the experiments we find the average Q-values to be above 0.99, which shows that the state subspaces (22) constructed from the H' latents chosen through the selection function (25) contain almost the entire posterior probability mass in this case. The small fraction of remaining posterior mass lies in other discrete subspaces and its principle form is known to not contain any heavy tails (see Equation (16)). The contribution of the truncated posterior mass to parameter updates can therefore be considered negligible.

5.2 Consistency

Prior to delving into a comparative analysis of GSC with other methods, we assess the consistency of the approach by applying both its exact and truncated variational inference schemes on the task of recovering sparse latent directions w.r.t. increasing numbers of training data. For this experiment we work with synthetic data generated by the GSC model itself. Moreover, we also apply the truncated variational inference on standard sparse coding data generated with a standard Laplace prior(Olshausen and Field, 1996; Lee et al., 2007). Taking into account the computational demands of the exact inference, we set both the hidden as well as observed dimensions (H and D respectively) to 10. For the experiment we exponentially increase N from 1000 to 512000. For each trial in the experiment we generate a new ground-truth mixing matrix $W^{\text{gen}} \in \mathbb{R}^{D \times H}$ by randomly generating a set of H orthogonal bases and perturbing them with a Gaussian noise with zero mean and a variance of 2.0. We set the sparsity parameters π_h to $1/H$, while the observed noise is assumed to be homoscedastic with $\sigma = 1.0$. When generating data with a spike-and-slab prior, the slab is considered to have its mean at zero with an identity covariance matrix, i.e., $\mu_h = 0.0$ for all $h \in H$ and $\Psi^{\text{gen}} = I_H$, respectively. In each trial after performing 100 EM iterations and inferring the whole set of GSC parameters Θ, we quantify the quality of the inference in terms of how well the inferred bases W align with the corresponding ground truth bases W^{gen}. As a measure of discrepancy between the generating and the recovered bases we use the Amari index (Amari et al., 1995):

$$A(W) \;=\; \frac{1}{2H(H-1)} \sum_{h,h'=1}^{H} \Big(\frac{|O_{hh'}|}{\max_{h''} |O_{hh''}|} + \frac{|O_{hh'}|}{\max_{h''} |O_{h''h'}|} \Big) - \frac{1}{H-1}, \qquad (27)$$

where $O_{hh'} = \big(W^{-1} W^{\text{gen}}\big)_{hh'}$. The Amari index is either positive or zero. It is zero only when the basis vectors of W and W^{gen} represent the same set of orientations, which in our case implies a precise recovery of the (ground truth) sparse directions.

4. To infer homoscedastic noise we set in the M-step the updated noise matrix Σ to $\sigma^2 I_D$ where $\sigma^2 = \text{Tr}\big(\Sigma\big)/D$. This is equivalent to parameter update for σ^2 if the model originally assumes homoscedastic noise.

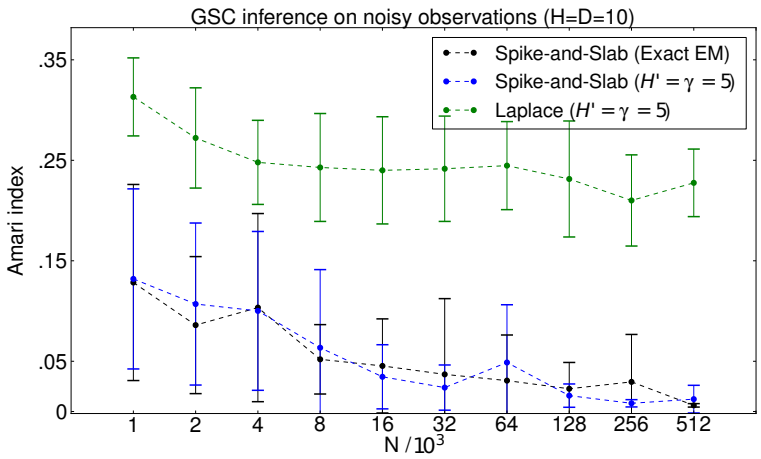


Figure 3: Numerical experiment investigating the consistency of the exact as well as the truncated variational GSC algorithm for increasing numbers of data points. The curves show results for the recovery of sparse directions for different numbers of data points. Data points were generated by both the spike-and-slab generative model (black and blue) and a standard sparse coding model with Laplace prior (green). The curves show the mean Amari index and standard deviations computed based on 15 repetitions of the learning algorithm.

Figure 3 summarizes the results of the experiment. Each error bar in the plot extends one standard deviation on both sides of its corresponding mean Amari index, which is computed from 15 repetitions. The black curve shows the results of the exact GSC inference on spike-and-slab generated data, while the blue and green curves illustrate the results of the truncated variational inference ($H' = \gamma = 5$) on data generated by spike-and-slab and Laplace priors respectively. For data generated with the spike-and-slab prior, we observe a gradually more accurate recovery of the sparse directions, as the mean Amari indices gradually converge towards the minimum value of zero for increasing numbers of training data. The minimum Amari index values that we obtain for the black and blue curves for $N \in \{128K, 256K, 512K\}$ are all below 6×10^{-3}. For the standard sparse coding data, we also see an improvement in performance with more data; however, higher mean values of the Amari index in this case can presumably be attributed to the model mismatch.

5.3 Recovery of Sparse Directions on Synthetic Data

In our first comparison with other methods, we measure the performances of GSC (using the truncated variational approximation) and MTMKL (which uses a factored variational approximation) approaches on the sparse latent direction recovery task given synthetic data generated by standard sparse coding models. In one set of experiments we generate data using sparse coding with Cauchy prior (Olshausen and Field, 1996), and in another set of experiments we use the standard Laplace distribution as a prior (Olshausen and Field, 1996; Lee et al., 2007). For each trial in the experiments a new mixing matrix

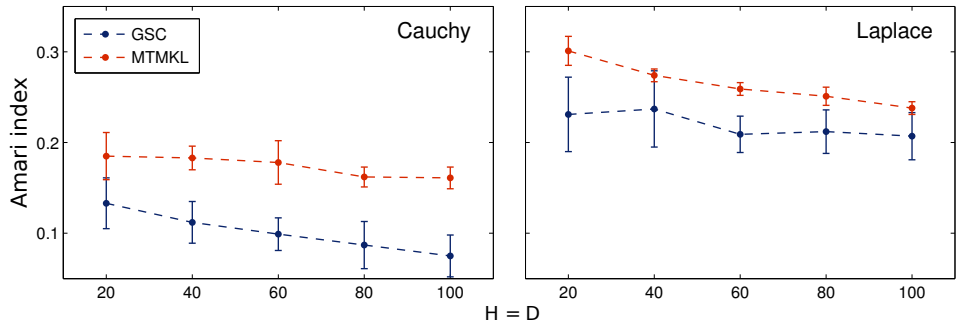


Figure 4: Performance of GSC (with $H' = \gamma = \frac{H}{10}$) vs. MTMKL on data generated by standard sparse coding models both with Cauchy and Laplace priors. Performance compared on the Amari index (27).

W^{gen} was generated without any constraints on the sparse directions (i.e., matrices were non-orthogonal in general). In both sets of experiments we simultaneously vary both the observed and latent dimensions D and H between 20 and 100, and repeat 15 trials per given dimensionality. For each trial we randomly generated a new data set of $N = 5000$ noisy observations with $\Sigma^{\text{gen}} = I_D$. Per trial, we perform 50 iterations of both algorithms. The GSC truncation parameters H' and γ were set to $\frac{H}{10}$. We assess the performances of the algorithms w.r.t. the Amari index (27).

The results for GSC and MTMKL in Figure 4 show that both approaches do relatively well in recovering the sparse directions, which shows that they are robust against the model mismatch imposed by generating from models with other priors. Furthermore, we observe that the GSC approach consistently recovers the sparse directions more accurately.

5.4 Source Separation

On synthetic data we have seen that spike-and-slab sparse coding can effectively recover sparse directions such as those generated by standard sparse coding models. As many signals such as acoustic speech data are sparse, and as different sources mix linearly, the assumptions of sparse coding match such data well. Source separation is consequently a natural application domain of sparse coding approaches, and well suited for benchmarking novel spike-and-slab as well as other sparse coding algorithms. To systematically study the a-posteriori independence assumption in factored variational approaches, we monitor the recovery of sparse directions of GSC and MTMKL for an increasing degree of the mixing matrix's non-orthogonality. Figure 5 shows the performance of both the methods based on three different source separation benchmarks obtained from (ICALAB; Cichocki et al., 2007). The error bars show two standard deviations estimated based on 15 trials per experiment. The x-axis in the figure represents the degree of orthogonality of the ground truth mixing bases W^{gen}. Starting from strictly orthogonal at the left, the bases were made increasingly non-orthogonal by randomly generating orthogonal bases and adding Gaussian distributed noise to them with $\sigma \in \{4, 10, 20\}$, respectively. For Figure 5 no observation

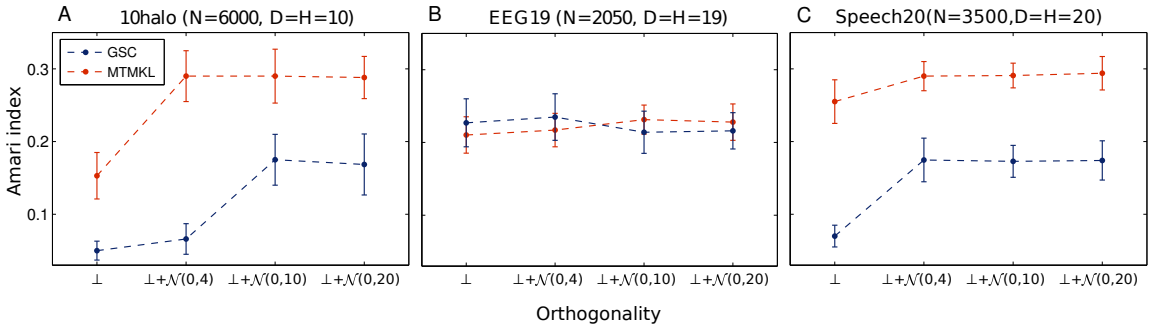


Figure 5: Performance of GSC vs. MTMKL on source separation benchmarks with varying degrees of orthogonality of the mixing bases. The orthogonality on the x-axis varies from being orthogonal $\perp$ to increasingly non-orthogonal mixing as randomly generated orthogonal bases are perturbed by adding Gaussian noise $\mathcal{N}(0, \sigma)$ to them. No observation noise was assumed for these experiments. Performances are compared on the Amari index (27).

noise was added to the mixed sources. For both the algorithms we performed 100 iterations per run.[5] The GSC truncation parameters H' and γ were set to 10 for all the following experiments, therefore for *10halo* the GSC inference was exact. As can be observed, both approaches recover the sparse directions well. While performance on the EEG19 data set is the same, GSC consistently performs better than MTMKL on *10halo* and *Speech20*. If observation noise is added, the difference can become still more pronounced for some data sets. Figure 6 shows the performance in the case of *Speech*20 (with added Gaussian noise with $\sigma = 2.0$), for instance. Along the x-axis orthogonality decreases, again. While the performance of MTMKL decreases with decreasing orthogonality, performance of GSC increases in this case. For other data sets increased observation noise may not have such effects, however (see Appendix, Figure 12 for two examples).

Next we look at MAP based sparse coding algorithms for the source separation task. Publicly available methods which we compare with are (SPAMS; Mairal et al., 2009) and the efficient sparse coding algorithms (ESCA; Lee et al., 2007). These methods are based on linear regression with lasso regularization, where sparsity is induced by introducing a parameter-regulated penalty term in the objective function,[6] which penalizes the L_1-norm of regressors (or latent variables). In a probabilistic context this is equivalent to assuming a Laplace prior on the regressors. In this experiment we test the performance on another set of ICALAB (Cichocki et al., 2007) benchmarks used previously (Suzuki and Sugiyama, 2011; Lücke and Sheikh, 2012). Following Suzuki and Sugiyama (2011) we use $N = 200$ and $N = 500$ data points from each benchmark and generate observed data by mixing the

5. For the MTMKL algorithm we observed convergence after 100 iterations while the GSC algorithm continued to improve with more iterations. However, allowing the same number of iterations to both the algorithms, the reported results are obtained with 100 iterations.
6. For both the algorithms compared here, optimal values for sparsity controlling regularization parameters were chosen through cross-validation.

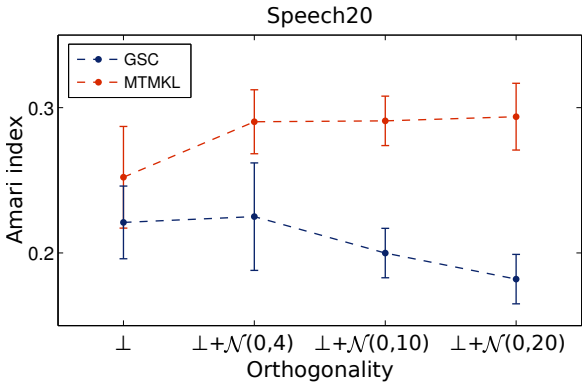


Figure 6: Performance of GSC vs. MTMKL in terms of the Amari index (27) on the Speech20 benchmark with varying degrees of orthogonality of the mixing bases and Gaussian noise (with $\sigma = 2$) added to observed data. The orthogonality on the x-axis varies from being orthogonal $\perp$ to increasingly non-orthogonal mixing as randomly generated orthogonal bases are perturbed by adding Gaussian noise $\mathcal{N}(0, \sigma)$ to them.

benchmark sources with randomly generated orthogonal bases and adding no noise to the observed data. For each experiment we performed 50 trials with a new randomly generated orthogonal data mixing matrix W^{gen} and new parameter initialization in each trial. The GSC inference was exact for these experiments with better results obtained with observed noise constrained to be homoscedastic. We performed up to 350 iterations of the GSC algorithm (with more iterations continuing to improve the performance) while for the other algorithms we observed convergence between 100 and 300 iterations.

Table 1 lists the performances of the algorithms. As can be observed, the spike-and-slab based models perform better than the standard sparse coding models for all except of one experiment (Sergio7, 200 data points) where SPAMS performs comparably well (or slightly better). Among the spike-and-slab models, GSC performs best for all settings with 500 data points, while MTMKL is better in two cases for 200 data points.[7] Further improvements on some settings in Table 1 can be obtained by algorithms constrained to assume orthogonal bases (Suzuki and Sugiyama, 2011; Lücke and Sheikh, 2012). However, for *10halo* and *speech4* GSC and MTMKL are better without such an explicit constraint.

Figure 2 was generated in a similar fashion on the *10halo* data set. There we computed the exact posterior (13) over $H = 10$ latent dimensions, thus the approximation parameters were $\gamma = H' = H$ (exact E-step). After performing 50 EM iterations and learning all the model parameters, we then visualized marginalized posteriors for a given data point along each column of the figure. The top row of the figure allows us get an idea of how

7. In Table 1 the results do not necessarily improve with an increased number of data points. However, the data points considered here are not independent samples. Following Suzuki and Sugiyama (2011) we always took consecutive 200 or 500 data points (after an offset) from each of the benchmarks. Therefore, due to time-dependencies in the signals, the underlying data point statistics change with the number of data points.

data sets			Amari index - mean (std.)			
name	H = D	N	GSC	MTMKL	SPAMS	ESCA
10halo	10	200	0.27(.04)	**0.21(.05)**	0.28(0)	0.31(.02)
		500	**0.17(.03)**	0.20(.03)	0.29(0)	0.29(.02)
Sergio7	7	200	**0.19(.05)**	**0.19(.03)**	**0.18(0)**	0.27(.04)
		500	**0.13(.04)**	0.23(.04)	0.19(0)	0.18(.04)
Speech4	4	200	**0.13(.04)**	**0.14(.03)**	0.18(0)	0.23(.02)
		500	**0.10(.04)**	0.14(.08)	0.16(0)	0.17(0)
c5signals	5	200	0.29(.08)	**0.24(.08)**	0.39(0)	0.47(.05)
		500	**0.31(.06)**	**0.32(.03)**	0.42(0)	0.48(.05)

Table 1: Performance of GSC, MTMKL and other publicly available sparse coding algorithms on benchmarks for source separation. Performances are compared based on the Amari index (27). Bold values highlight the best performing algorithm(s).

concentrated and sparse a data point is in terms of the latents contributing to its posterior mass. The bottom row of the figure on the other hand allows us to observe the sparsity in the posterior w.r.t. the dimensionality of the hyperplanes spanned by the latents, with a posterior mass accumulation in low-dimensional hyperplanes.

5.5 Computational Complexity vs. Performance

In terms of computational complexity, GSC and MTMKL algorithms are significantly different, so we also looked at the trade-off between their computational costs versus performance. Subfigures A and B in Figure 7 show performance against compute time for both algorithms. The error bars for the Speech20 plot were generated from 15 trials per experiment. For MTMKL we obtained the plot by increasing the number of iterations from 50 to 100 and 1000, while for the GSC plot we performed 100 iterations with $H' = \gamma \in [2, 3, 5, 7, 10]$. For the image denoising task (described next), the MTMKL plot was generated from a run with $H = 64$ latents and the number of iterations going up to 12K. The GSC plot was generated from $H = 400$ latents with H' and γ being 10 and 5 respectively. The last point on the GSC (blue) curve corresponds to the $120th$ EM iteration. As can be observed for both tasks, the performance of MTMKL saturates from certain runtime values onwards. GSC on the other hand continues to show improved performance with increasing computational resources.

For the denoising task we also compared the performance of both the algorithms against an increasing number of latents H. While the computational cost of the MTMKL algorithm increases linearly w.r.t. H, the runtime cost of the truncated variational GSC remains virtually unaffected by it, since it scales w.r.t. the parameters H' and γ (see Section 4.1). In this experiment we performed 65 iterations of the GSC algorithm for $H \in \{64, 256\}$ and up to 120 iterations for $H = 400$. For MTMKL we performed up to 120 iterations for each given H. Figure 7C summarizes the results of this experiment. In the figure we can see a constant performance increase for GSC, while for MTMKL we actually observe a slight

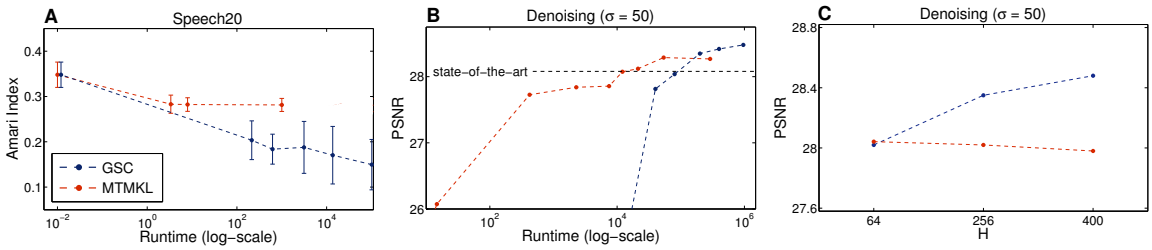


Figure 7: **A,B:** Runtime vs. performance comparison of GSC (blue) and MTMKL (red) on source separation and denoising tasks. Source separation is compared on the Amari index (the lower the better) while the denoising is compared on the peak signal-to-noise (PSNR) ratio (the higher the better). **C**: Performance of GSC (blue) and MTMKL (red) on the denoising task against an increasing number of latents.

decrease in performance. This is in conformity with what Titsias and Lazaro-Gredilla (2011) report in their work that for the denoising task they observed no performance improvements for larger number of latents.

5.6 Image Denoising

Finally, we investigate performance of the GSC algorithm on the standard "house" benchmark for denoising which has been used for the evaluation of similar approaches (e.g., Li and Liu, 2009; Zhou et al., 2009) including the MTMKL spike-and-slab approach. The MTMKL approach currently represents the state-of-the-art on this benchmark (see Titsias and Lazaro-Gredilla, 2011). We also compare with the approach by Zhou et al. (2009) as a representative sampling-based optimization scheme. For the task a noisy input image is generated by adding Gaussian noise (with zero mean and standard deviation determining the noise level) to the 256×256 image (see Figure 8). Following the previous studies, we generated $62,001$ overlapping (shifted by 1 pixel) 8×8 patches from the noisy image. We then applied 65 iterations of the GSC algorithm for $H \in \{64, 256\}$ for different noise levels $\sigma \in \{15, 25, 50\}$. The truncation parameters H' and γ for each run are listed in Table 2. We assumed homoscedastic observed noise with a priori unknown variance in all these experiments (as the MTMKL model).

A comprehensive comparison of the denoising results of the various algorithms is shown in Table 2, where performance is measured in terms of the peak signal-to-noise (PSNR) ratio. We found that for the low noise level ($\sigma = 15$) GSC is competitive with other approaches but with MTMKL performing slightly better. For the higher noise levels of $\sigma = 25$ and $\sigma = 50$, GSC outperforms all the other approaches including the MTMKL approach that represented the state-of-the-art. In Figure 8 we show our result for noise level $\sigma = 25$. The figure contains both the noisy and the GSC denoised image along with the inferred sparsity vector $\vec{\pi}$ and all bases with appearance probabilities significantly larger than zero (sorted from high such probabilities to low ones). We also applied GSC with higher numbers of

latent dimensions: Although for low noise levels of $\sigma = 15$ and $\sigma = 25$ we did not measure significant improvements, we observed a further increase for $\sigma = 50$. For instance, with $H = 400$, $H' = 10$ and $\gamma = 8$, we obtained for $\sigma = 50$ a PSNR of 28.48dB.

As for source separation described in Section 5.5, we also compared performance vs. computational demand of both algorithms for the task of image denoising. As illustrated in A and B of Figure 7, MTMKL performs better when computational resources are relatively limited. However, when increasingly more computational resources are made available, MTMKL does not improve much further on its performance while GSC performance continuously increases and eventually outperforms MTMKL on this task.

	PSNR (dB)						
Noise	Noisy img	MTMKL$^{exp.}$	K-SVD$^{mis.}$	*K-SVD^{match}	Beta pr.	GSC (H=64)	GSC (H=256)
σ=15	24.59	**34.29**	30.67	34.22	34.19	32.68 (H'=10,γ=8)	33.78 (H'=18,γ=3)
σ=25	20.22	31.88	31.52	32.08	31.89	31.10 (H'=10,γ=8)	**32.01** (H'=18,γ=3)
σ=50	14.59	28.08	19.60	27.07	27.85	28.02 (H'=10,γ=8)	**28.35** (H'=10,γ=8)

Table 2: Comparison of the GSC algorithm with other methods applied to the "house" benchmark. The compared methods are: MTMKL (Titsias and Lazaro-Gredilla, 2011), K-SVD (Li and Liu, 2009), and Beta process (Zhou et al., 2009). Bold values highlight the best performing algorithm(s). *High values for K-SVD matched are not made bold-faced as the method assumes the noise variance to be known a-priori (see Li and Liu, 2009).

6. Discussion

The last years have seen a surge in the application of sparse coding algorithms to different research domains, along with developments of new sparse coding approaches with increased capabilities. There are currently different lines of research followed for developing new algorithms: one direction is based on the standard sparse coding algorithm (Olshausen and Field, 1996) with Laplace prior and parameter optimization using maximum a-posteriori (MAP) estimates of the latent posteriors for efficient training. This original approach has since been made more efficient and precise. Many sparse coding algorithms based on the MAP estimation are continuously being developed and are successfully applied in a variety of settings (e.g., Lee et al., 2007; Mairal et al., 2009). Another line of research aims at a fully Bayesian description of sparse coding and emphasizes greater flexibility by using different (possibly non-Gaussian) noise models and estimations of the number of hidden dimensions. The great challenge of these general models is the procedure of parameter estimation. For instance, the model by Mohamed et al. (2012) uses Bayesian methodology involving conjugate priors and hyper-parameters in combination with Laplace approximation and different sampling schemes.

A line of research falling in between conventional and fully Bayesian approaches is represented by the truncated variational approach studied here and by other very recent develop-

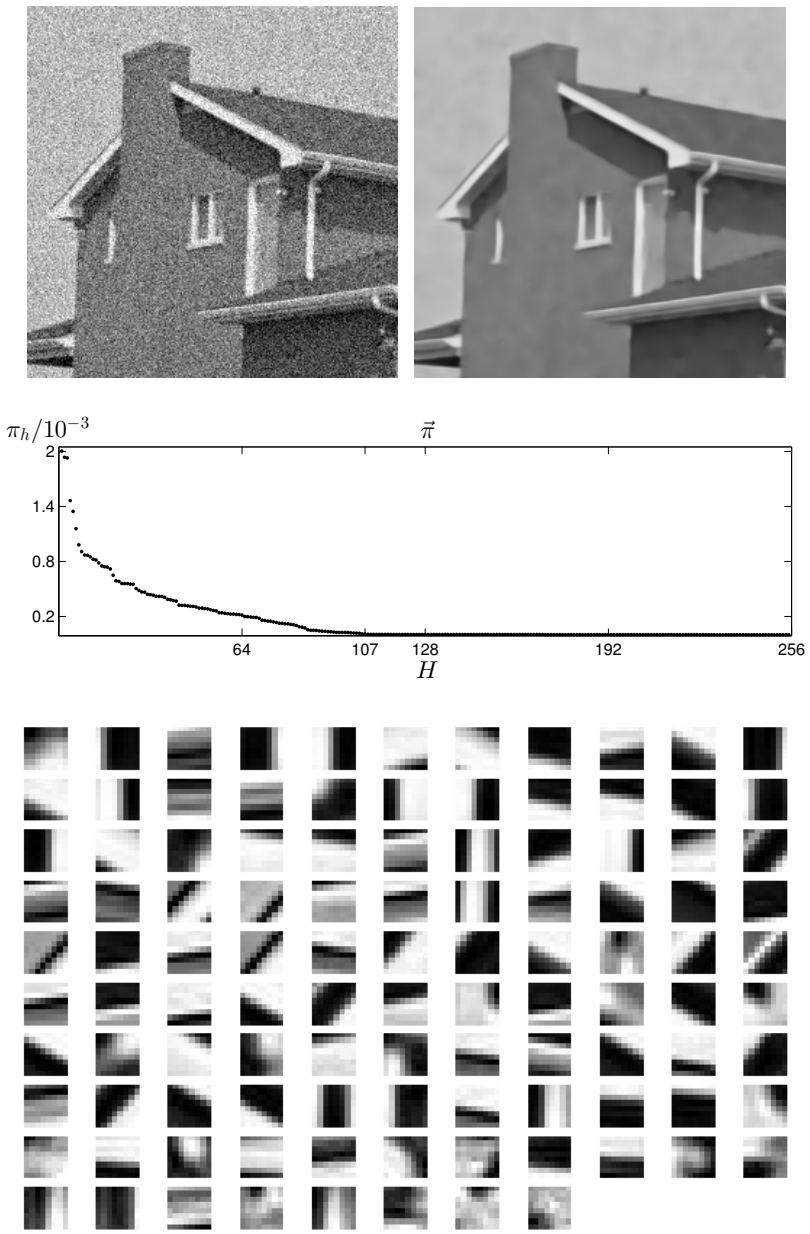


Figure 8: Top left: Noisy "house" image with $\sigma = 25$. Top right: GSC denoised image. Middle: Inferred sparsity values π_h in descending order indicate that finally around 107 of in total 256 latent dimensions significantly contribute to model the data. Bottom: Basis functions (ordered from left to right, top to bottom) corresponding to the first 107 latent dimensions sorted w.r.t. the decreasing sparsity values π_h .

ments (Titsias and Lazaro-Gredilla, 2011; Goodfellow et al., 2013). While these approaches are all based on spike-and-slab generalizations of sparse coding (like fully Bayesian approaches), they maintain deterministic approximation procedures for parameter optimization. Variational approximations allow for applications to large hidden spaces which pose a challenge for sampling approaches especially in cases of multi-modal posteriors. Using the novel and existing approaches in different experiments of this study, we have confirmed the advantages of spike-and-slab priors for sparse coding, and the scalability of variational approximations for such models. The newly developed truncated variational algorithm scales almost linearly with the number of hidden dimensions for fixed truncation parameters (see for instance the scaling behavior in supplemental Figure 10 for H going up to 1024). The MTMKL algorithm by Titsias and Lazaro-Gredilla (2011) has been applied on the same scale. Using a similar approach also based on factored distributions Goodfellow et al. (2013) report results for up to a couple of thousands latent dimensions (albeit on small input dimensions and having a more constrained generative model). Sampling based algorithms for non-parametric and fully Bayesian approaches are more general but have not been applied to such large scales.

A main focus of this work and reasoning behind the algorithm's development is due to the long-known biases introduced by factored variational approximations (MacKay, 2001; Ilin and Valpola, 2005; Turner and Sahani, 2011). Our systematic comparison of the GSC algorithm to the method by Titsias and Lazaro-Gredilla (2011) confirms the earlier observation (Ilin and Valpola, 2005) that factored variational approaches are biased towards orthogonal bases. If we compare the performance of both algorithms on the recovery of non-orthogonal sparse directions, the performance of the factored variational approach is consistently lower than the performance of the truncated variational algorithm (Figure 4). The same applies for experiments for unmixing real signals in which we increased the non-orthogonality (Figure 5A,C; suppl. Figure 12); although for some data performance is very similar (Figure 5B). Also if sources are mixed orthogonally, we usually observe better performance of the truncated variational approach (Table 1), which is presumably due to the more general underlying prior (i.e., a fully parameterized Gaussian slab). Overall, GSC is the best performing algorithm on source separation tasks involving non-orthogonal sparse directions (compare Suzuki and Sugiyama, 2011, for algorithms constrained to orthogonal bases). For some data sets with few data points, we observed an equal or better performance of the MTMKL approach, which can be explained by their Bayesian treatment of the model parameters (see Table 1, performance with 200 data points). Notably, both approaches are consistently better on source separation benchmarks than the standard sparse coding approaches SPAMS (Mairal et al., 2009) and ESCA (Lee et al., 2007) (see Table 1). This may be taken as evidence for the better suitability of a spike-and-slab prior for such types of data.

For source separation our approach (like conventional sparse coding or ICA) seeks to infer sparse directions by capturing the sparse, latent structures from the spatial domain of the input signals. However, when dealing with data that also carry a temporal structure (e.g., speech or EEG recordings), other approaches which explicitly model temporal regularities such as Hidden Markov Models (HMMs) may as well be a more natural and (depending on the task) a more suitable choice. Such methodologies can in principle be combined with the sparse coding approaches studied and compared here to form more com-

prehensive models for spatio-temporal data, which can yield improved performance on blind source separation tasks (compare e.g., Gael et al., 2008; Mysore et al., 2010).

In the last experiment of this study, we finally compared the performance of factored and truncated variational approximations on a standard image denoising task (see Table 2). The high PSNR values observed for both approaches again in general speak for the strengths of spike-and-slab sparse coding. The MTMKL model represented the state-of-the-art on this benchmark, so far. Differences of MTMKL to previous approaches are small, but this is due to the nature of such long-standing benchmarks (compare, e.g., the MNIST data set). For the same denoising task with standard noise levels of $\sigma = 25$ and $\sigma = 50$ we found the GSC model to further improve the state-of-the-art (compare Table 2 with data by Li and Liu, 2009, Zhou et al., 2009, Titsias and Lazaro-Gredilla, 2011). While we observed a continuous increase of performance with the number of hidden dimensions used for GSC, the MTMKL algorithm (Titsias and Lazaro-Gredilla, 2011) is reported to reach saturation at $H = 64$ latent dimensions. As the learned sparse directions become less and less orthogonal the more over-complete the setting gets, this saturation may again be due to the bias introduced by the factored approach. GSC with $H = 256$ improves the state-of-the-art with 32.01dB for $\sigma = 25$ and with 28.35dB for $\sigma = 50$ (with even higher PSNR for $H = 400$). As we assume an independent Bernoulli prior per latent dimension, GSC can also prune out latent dimensions by inferring very low values of π_h for the bases that make negligible contribution in the inference procedure. This can be observed in Figure 8, where for the application of GSC to the denoising task with $\sigma = 25$, we found only about 107 of the 256 basis functions to have significant probabilities to contribute to the task. This means that GSC with about 100 basis functions can be expected to achieve almost the same performance as GSC with 256 basis functions. However, in practice we observed that the average performance increases with more basis functions because local optima can more efficiently be avoided. This observation is not limited to the particular approach studied here; also for other approaches to sparse learning, efficient avoidance of local optima has been reported if the number of assumed hidden dimensions was increased (e.g. Spratling, 2006; Lücke and Sahani, 2008). In comparison to MTMKL, GSC can make use of significantly more basis functions. It uses about 100 functions while MTMKL performance saturates at about 64 as mentioned previously. On the other hand, we found MTMKL to perform better on the low noise level setting (see $\sigma = 15$ in Table 2) or when relatively limited computational resources are available (see Figure 7).

In conclusion, we have studied a novel learning algorithm for sparse coding with spike-and-slab prior and compared it with a number of sparse coding approaches including other spike-and-slab based methods. The results we obtained show that the truncated EM approach is a competitive method. It shows that posterior dependencies and multi-modality can be captured by a scalable deterministic approximation. Furthermore, the direct comparison with a factored variational approach in source separation experiments confirms earlier observations that assumptions of a-posteriori independence introduces biases, and that avoiding such biases, e.g. by a truncated approach, improves the state-of-the-art on source separation benchmarks as well as on standard denoising tasks. However, we also find that under certain constraints and settings, factored variational learning for spike-and-slab sparse coding may perform as well or better. In general, our results argue in favor of

spike-and-slab sparse coding models and recent efforts for developing improved algorithms for inference and learning in such models.

Acknowledgements

We acknowledge funding by the German Research Foundation (DFG), grant LU 1196/4-2, and by the German Ministry of Research and Education (BMBF), grant 01GQ0840 (BFNT Frankfurt). Furthermore, we acknowledge support by the Frankfurt Center for Scientific Computing (CSC Frankfurt).

Appendix A. Derivation of M-step Equations

Our goal is to optimize the free-energy w.r.t. Θ:

$$\begin{aligned}
\mathcal{F}(\Theta^{\text{old}}, \Theta) &= \sum_{n=1}^{N} \Big\langle \log p(\vec{y}^{\,(n)}, \vec{s}, \vec{z} \,|\, \Theta) \Big\rangle_n + H(\Theta^{\text{old}}) \\
&= \sum_{n=1}^{N} \sum_{\vec{s}} \int_{\vec{z}} p(\vec{s}, \vec{z} \,|\, \vec{y}^{\,(n)}, \Theta^{\text{old}}) \\
&\quad \Big[\log\big(p(\vec{y}^{\,(n)} \,|\, \vec{s}, \vec{z}, \Theta)\big) + \log\big(p(\vec{z}\,|\vec{s},\, \Theta)\big) + \log\big(p(\vec{s}\,|\,\Theta)\big) \Big] \,\mathrm{d}\vec{z} \,+\, H(\Theta^{\text{old}}) \,,
\end{aligned}$$

where

$$\begin{aligned}
\log\big(p(\vec{y}^{\,(n)} \,|\, \vec{s}, \vec{z}, \Theta)\big) &= -\frac{1}{2}\big(\log(2\pi^D) + \log|\Sigma|\big) \\
&\quad -\frac{1}{2}\Big(\vec{y}^{\,(n)} - W(\vec{s}\odot\vec{z})\Big)^{\mathrm{T}} \Sigma^{-1} \Big(\vec{y}^{\,(n)} - W(\vec{s}\odot\vec{z})\Big) \,, \\
\log\big(p(\vec{z}\,|\vec{s},\, \Theta)\big) &= -\frac{1}{2}\Big(\log(2\pi^{|\vec{s}|}) + \log|\Psi \odot \vec{s}\,\vec{s}^{\mathrm{T}}|\Big) \\
&\quad -\frac{1}{2}\big(\,(\vec{z} - \vec{\mu})\odot\vec{s}\,\big)^{\mathrm{T}} \big(\Psi \odot \vec{s}\,\vec{s}^{\mathrm{T}}\big)^{-1} \big(\,(\vec{z} - \vec{\mu})\odot\vec{s}\,\big) \\
\text{and} \qquad \log\big(p(\vec{s}\,|\Theta)\big) &= \sum_{h=1}^{H} \log\big(\pi_h^{s_h}\,(1-\pi_h)^{1-s_h}\big) \,.
\end{aligned}$$

The free-energy thus takes the form:

$$\begin{aligned}
\mathcal{F}(\Theta^{\text{old}}, \Theta) &= \sum_{n=1}^{N} \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\text{old}}) \\
&\quad \Big[-\tfrac{1}{2}\big(\log(2\pi^D) + \log|\Sigma|\big) - \tfrac{1}{2}\Big(\vec{y}^{\,(n)} - W(\vec{s}\odot\vec{z})\Big)^{\mathrm{T}} \Sigma^{-1} \Big(\vec{y}^{\,(n)} - W(\vec{s}\odot\vec{z})\Big) \\
&\quad -\tfrac{1}{2}\big(\log(2\pi^{|\vec{s}|}) + \log|\Psi \odot \vec{s}\,\vec{s}^{\mathrm{T}}|\big) \\
&\quad -\frac{1}{2}\big(\,(\vec{z} - \vec{\mu})\odot\vec{s}\,\big)^{\mathrm{T}} \big(\Psi \odot \vec{s}\,\vec{s}^{\mathrm{T}}\big)^{-1} \big(\,(\vec{z} - \vec{\mu})\odot\vec{s}\,\big) \\
&\quad + \textstyle\sum_{h=1}^{H} \log\big(\pi_h^{s_h}\,(1-\pi_h)^{1-s_h}\big) \Big] \,\mathrm{d}\vec{z} \,+\, H(\Theta^{\text{old}}) \,,
\end{aligned}$$

where $q_n(\vec{s},\vec{z};\Theta^{\text{old}})$ denotes the posterior $p(\vec{s},\vec{z}\,|\,\vec{y}^{\,(n)},\Theta^{\text{old}})$. Now we can derive the M-step equations (6) to (10) by canonically setting the derivatives of the free-energy above w.r.t. each parameter in Θ to zero.

A.1 Optimization of the Data Noise

Let us start with the derivation of the M-step equation for Σ:

$$
\begin{aligned}
&\frac{\partial}{\partial\Sigma}\mathcal{F}(\Theta^{\text{old}},\Theta)\\
&\quad=\sum_{n=1}^{N}\sum_{\vec{s}}\int_{\vec{z}} q_n(\vec{s},\vec{z};\Theta^{\text{old}})\\
&\qquad\qquad\Big[-\tfrac{1}{2}\tfrac{\partial}{\partial\Sigma}\left(\log|\Sigma|\right)-\tfrac{1}{2}\tfrac{\partial}{\partial\Sigma}\left(\vec{y}^{\,(n)}-W(\vec{s}\odot\vec{z})\right)^{\mathrm{T}}\Sigma^{-1}\left(\vec{y}^{\,(n)}-W(\vec{s}\odot\vec{z})\right)\Big]\mathrm{d}\vec{z}\\
&\quad=\sum_{n=1}^{N}\sum_{\vec{s}}\int_{\vec{z}} q_n(\vec{s},\vec{z};\Theta^{\text{old}})\\
&\qquad\qquad\Big[-\tfrac{1}{2}\Sigma^{-1}+\tfrac{1}{2}\Sigma^{-2}\left(\vec{y}^{\,(n)}-W(\vec{s}\odot\vec{z})\right)\left(\vec{y}^{\,(n)}-W(\vec{s}\odot\vec{z})\right)^{\mathrm{T}}\Big]\mathrm{d}\vec{z}\stackrel{!}{=}0
\end{aligned}
$$

$$
\begin{aligned}
\Rightarrow\Sigma&=\frac{1}{N}\sum_{n=1}^{N}\sum_{\vec{s}}\int_{\vec{z}} q_n(\vec{s},\vec{z};\Theta^{\text{old}})\Big[\left(\vec{y}^{\,(n)}-W(\vec{s}\odot\vec{z})\right)\left(\vec{y}^{\,(n)}-W(\vec{s}\odot\vec{z})\right)^{\mathrm{T}}\Big]\mathrm{d}\vec{z}\\
&=\frac{1}{N}\sum_{n=1}^{N}\Big[\Big(\vec{y}^{\,(n)}-W\big\langle(\vec{s}\odot\vec{z})\big\rangle_n\Big)\Big(\vec{y}^{\,(n)}-W\big\langle(\vec{s}\odot\vec{z})\big\rangle_n\Big)^{\mathrm{T}}\\
&\qquad\qquad+W\big[\big\langle(\vec{s}\odot\vec{z})(\vec{s}\odot\vec{z})^{\mathrm{T}}\big\rangle_n-\big\langle(\vec{s}\odot\vec{z})\big\rangle_n\big\langle(\vec{s}\odot\vec{z})\big\rangle_n^{\mathrm{T}}\big]W^{\mathrm{T}}\Big]\\
&=\frac{1}{N}\sum_{n=1}^{N}\Big[\vec{y}^{\,(n)}(\vec{y}^{\,(n)})^{\mathrm{T}}-W\big[\big\langle(\vec{s}\odot\vec{z})\big\rangle_n\big\langle(\vec{s}\odot\vec{z})\big\rangle_n^{\mathrm{T}}\big]W^{\mathrm{T}}\Big],
\end{aligned}
$$

where $\langle\cdot\rangle_n$ denotes the expectation value in Equation (5).

A.2 Optimization of the Bases

We will now derive the M-step update for the basis functions W:

$$
\begin{aligned}
&\frac{\partial}{\partial W}\mathcal{F}(\Theta^{\text{old}},\Theta)\\
&\quad=\sum_{n=1}^{N}\sum_{\vec{s}}\int_{\vec{z}} q_n(\vec{s},\vec{z};\Theta^{\text{old}})\Big[-\tfrac{1}{2}\tfrac{\partial}{\partial W}\left(\vec{y}^{\,(n)}-W(\vec{s}\odot\vec{z})\right)^{\mathrm{T}}\Sigma^{-1}\left(\vec{y}^{\,(n)}-W(\vec{s}\odot\vec{z})\right)\Big]\mathrm{d}\vec{z}\\
&\quad=\sum_{n=1}^{N}\sum_{\vec{s}}\int_{\vec{z}} q_n(\vec{s},\vec{z};\Theta^{\text{old}})\left[-\tfrac{1}{\Sigma}\left(\vec{y}^{\,(n)}(\vec{s}\odot\vec{z})^{\mathrm{T}}-W(\vec{s}\odot\vec{z})(\vec{s}\odot\vec{z})^{\mathrm{T}}\right)\right]\mathrm{d}\vec{z}\stackrel{!}{=}0
\end{aligned}
$$

$$\Rightarrow W = \frac{\sum_{n=1}^{N} \vec{y}^{(n)} \left\langle \vec{s} \odot \vec{z} \right\rangle_n^{\mathrm{T}}}{\sum_{n=1}^{N} \left\langle (\vec{s} \odot \vec{z})(\vec{s} \odot \vec{z})^{\mathrm{T}} \right\rangle_n}.$$

A.3 Optimization of the Sparsity Parameter

Here we take the derivative of the free-energy w.r.t. $\vec{\pi}$:

$$\begin{aligned}
\frac{\partial}{\partial \vec{\pi}} \mathcal{F}(\Theta^{\mathrm{old}}, \Theta) &= \sum_{n=1}^{N} \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\mathrm{old}}) \left[\frac{\partial}{\partial \vec{\pi}} \Big(\vec{s} \log \vec{\pi} + (1 - \vec{s})\, log(1 - \vec{\pi}) \Big) \right] \mathrm{d}\vec{z} \\
&= \sum_{n=1}^{N} \sum_{\vec{s}} q_n(\vec{s}; \Theta^{\mathrm{old}}) \left[\frac{\vec{s}}{\vec{\pi}} - \frac{(1-\vec{s})}{(1-\vec{\pi})} \right] \stackrel{!}{=} 0 \\
\\
\Rightarrow \vec{\pi} &= \frac{1}{N} \sum_{n=1}^{N} \left\langle \vec{s} \right\rangle_n.
\end{aligned}$$

A.4 Optimization of the Latent Mean

Now we derive the M-step update for the mean $\vec{\mu}$ of the Gaussian slab:

$$\begin{aligned}
\frac{\partial}{\partial \vec{\mu}} \mathcal{F}(\Theta^{\mathrm{old}}, \Theta) &= \sum_{n=1}^{N} \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\mathrm{old}}) \\
&\qquad \left[-\tfrac{1}{2} \tfrac{\partial}{\partial \vec{\mu}} \big((\vec{z} - \vec{\mu}) \odot \vec{s} \big)^{\mathrm{T}} \left(\Psi \odot \vec{s}\,\vec{s}^{\mathrm{T}} \right)^{-1} \big((\vec{z} - \vec{\mu}) \odot \vec{s} \big) \right] \mathrm{d}\vec{z} \\
&= \sum_{n=1}^{N} \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\mathrm{old}}) \left[\left(\Psi \odot \vec{s}\,\vec{s}^{\mathrm{T}} \right)^{-1} \Big((\vec{z} - \vec{\mu}) \odot \vec{s} \Big) \right] \mathrm{d}\vec{z} \stackrel{!}{=} 0 \\
\\
\Rightarrow \vec{\mu} &= \frac{\sum_{n=1}^{N} \left\langle \vec{s} \odot \vec{z} \right\rangle_n}{\sum_{n=1}^{N} \left\langle \vec{s} \right\rangle_n}.
\end{aligned}$$

A.5 Optimization of the Latent Covariance

Lastly we derive the M-step update for the latent covariance Ψ:

$$\begin{aligned}
&\frac{\partial}{\partial \Psi} \mathcal{F}(\Theta^{\mathrm{old}}, \Theta) \\
&\quad = \sum_{n=1}^{N} \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\mathrm{old}}) \\
&\qquad \left[-\tfrac{1}{2} \tfrac{\partial}{\partial \Psi} \left(\log \left| \Psi \odot \vec{s}\,\vec{s}^{\mathrm{T}} \right| \right) - \tfrac{1}{2} \tfrac{\partial}{\partial \Psi} \big((\vec{z} - \vec{\mu}) \odot \vec{s} \big)^{\mathrm{T}} \left(\Psi \odot \vec{s}\,\vec{s}^{\mathrm{T}} \right)^{-1} \big((\vec{z} - \vec{\mu}) \odot \vec{s} \big) \right] \mathrm{d}\vec{z}
\end{aligned}$$

$$= \sum_{n=1}^{N} \sum_{\vec{s}} \int_{\vec{z}} q_n(\vec{s}, \vec{z}; \Theta^{\text{old}}) \Big[-\tfrac{1}{2}\big(\Psi \odot \vec{s}\,\vec{s}^{\mathrm{T}}\big)^{-1} + \tfrac{1}{2}\big(\Psi \odot \vec{s}\,\vec{s}^{\mathrm{T}}\big)^{-2}\big((\vec{z} - \vec{\mu}) \odot \vec{s} \big) \big((\vec{z} - \vec{\mu}) \odot \vec{s} \big)^{\mathrm{T}} \Big] \mathrm{d}\vec{z} \stackrel{!}{=} 0$$

$$\Rightarrow \Psi = \sum_{n=1}^{N} \Big[\big\langle (\vec{z} - \vec{\mu})\,(\vec{z} - \vec{\mu})^{\mathrm{T}} \odot \vec{s}\,\vec{s}^{\mathrm{T}} \big\rangle_n \Big] \odot \Big(\sum_{n=1}^{N} \Big[\big\langle \vec{s}\,\vec{s}^{\mathrm{T}} \big\rangle_n \Big] \Big)^{-1}$$
$$= \sum_{n=1}^{N} \Big[\big\langle (\vec{s} \odot \vec{z})(\vec{s} \odot \vec{z})^{\mathrm{T}} \big\rangle_n - \big\langle \vec{s}\,\vec{s}^{\mathrm{T}} \big\rangle_n \odot \vec{\mu}\vec{\mu}^{\mathrm{T}} \Big] \odot \Big(\sum_{n=1}^{N} \Big[\big\langle \vec{s}\,\vec{s}^{\mathrm{T}} \big\rangle_n \Big] \Big)^{-1}.$$

Appendix B. Performance vs. Complexity Trade-Off

If the approximation parameters H' and γ are held constant, the computational cost of the algorithm scales with the computational cost of the selection function. If the latter cost scales linearly with H (as is the case here), then so does the overall computational complexity (compare complexity considerations by Lücke and Eggert, 2010)). This is consistent with numerical experiments in which we measured the increase in computational demand (see Figure 10). In experiments with H increasing from 16 to 1024, we observed a, finally, close to linear increase of computational costs. However, a larger H implies a larger number of parameters, and thus may require more data points to prevent over-fitting. Although a larger data set increases computational demand, our truncated approximation algorithm allows us to take advantage of parallel computing architecture in order to more efficiently deal with large data sets (see Appendix C for details). Therefore in practice, we can weaken the extent of an increase in computational cost due to a higher demand for data. Furthermore, we examined the benefit of using GSC (in terms of average speedup over EM iterations) versus the cost regarding algorithmic performance. We compared approximation parameters in the range of $H' = \gamma = [1, 10]$ and again observed the performance of the algorithm on the task of source separation (with randomly generated orthogonal ground truth mixing bases and no observed noise). Figure 9 shows that a high accuracy can still be achieved for relatively small values of H' γ which, at the same time, results in strongly reduced computational demands.

Appendix C. Dynamic Data Repartitioning for Batch/Parallel Processing

As described in Section 4, the truncated variational approach deterministically selects the most likely H' causes of a given observation $\vec{y}$ for efficiently approximating the posterior distribution over a truncated latent space. In practice one can also use the selected latent causes for applying clustering to the observed data, which allows for an efficient and parallelizable batch-mode implementation of the E-step of the truncated variational EM algorithm.

In the batch processing mode, prior to each E-step the observed data can be partitioned by clustering together the data points w.r.t. their selected latent causes. The resulting clusters can then be processed individually (e.g., on multiple compute cores) to perform

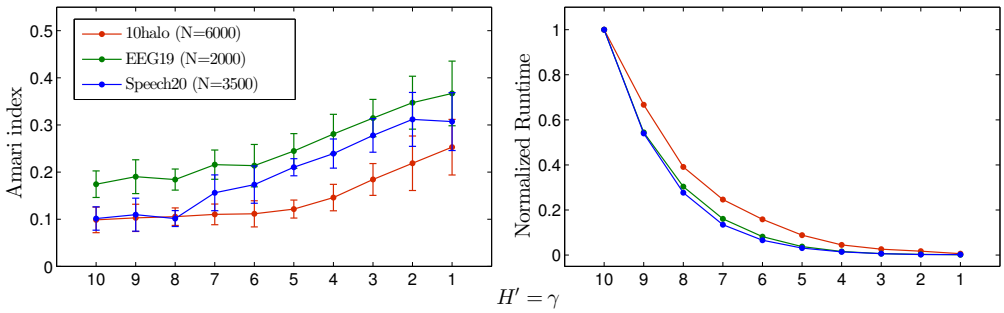


Figure 9: Performance of the GSC on 10halo, EEG19 and Speech20 benchmarks for decreasing truncation parameters H' and γ. The right plot shows how the computational demand of the truncated variational algorithm decreases with decreasing values of the truncation parameters. The runtime plots are normalized by the runtime value obtained for $H' = \gamma = 10$ for each of the benchmarks.

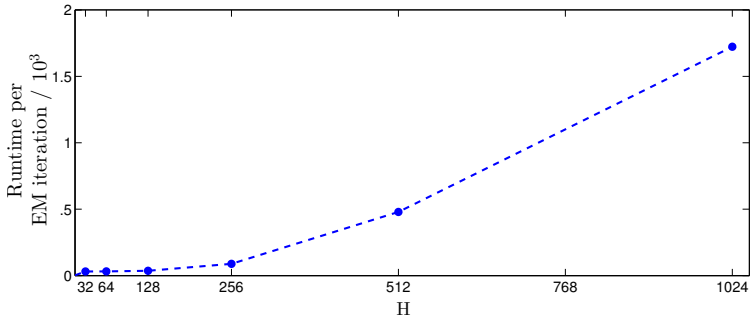


Figure 10: Time scaling behavior of GSC for increasing latent dimensions H and fixed truncation parameters H' and γ.

the E-step (Equations (21) to (23)) for all data points in a given cluster. This approach not only pursues a natural partitioning of data, but in a parallel execution environment, it can prove to be more efficient than uniformly distributing data (as in Bornschein et al., 2010) among multiple processing units. By maximizing the similarity (in latent space) of individual data points assigned to each of the processing units, we can overall minimize the number of redundant computations involved in Equations (15) and (23), that are tied to specific states of the latents. This can be observed by considering Equation (21), which is as follows:

$$p(\vec{s},\vec{z}\,|\,\vec{y}^{\,(n)},\Theta) \;\approx\; \frac{\mathcal{N}(\vec{y}^{\,(n)};\vec{\mu}_{\vec{s}},C_{\vec{s}})\,\mathcal{B}(\vec{s};\vec{\pi})\,\mathcal{N}(\vec{z};\,\vec{\kappa}^{\,(n)}_{\vec{s}},\Lambda_{\vec{s}})}{\sum_{\vec{s}'\in\mathcal{K}_n}\mathcal{N}(\vec{y}^{\,(n)};\vec{\mu}_{\vec{s}'},C_{\vec{s}'})\,\mathcal{B}(\vec{s}';\vec{\pi})}\,\delta(\vec{s}\in\mathcal{K}_n). \tag{28}$$

Here the parameters $\vec{\mu}_{\vec{s}}, C_{\vec{s}}$ and $\Lambda_{\vec{s}}$ entirely depend on a particular latent state $\vec{s}$. Also, $\vec{\kappa}^{\,(n)}_{\vec{s}}$ takes prefactors that can be precomputed given the $\vec{s}$. It turns out that to compute (28) our clustering-based, dynamic data repartitioning and redistribution strategy is more efficient than the uniform data distribution approach of Bornschein et al. (2010). This is illustrated in Figure 11, which shows empirical E-step speedup over the latter approach taken as a baseline. The error bars were generated by performing 15 trials per given data size N. For all the trials, model scale (i.e., data dimensionality) and truncation approximation parameters were kept constant.[8] Each trial was run in parallel on 24 computing nodes. The red plot in the figure also shows the speedup as a result of an intermediate approach. There we initially uniformly distributed the data samples which were then only locally clustered by each processing unit at every E-step. The blue plot on the other hand shows the speedup as a result of globally clustering and redistributing the data prior to every E-step. All the reported results here also take into account the cost of data clustering and repartitioning.

In a parallel setup, we perform the data clustering process by having each processing unit cluster its own data locally and then merging the resulting clusters globally. In order to avoid uneven data distribution, we also bound the maximum size of a cluster. Currently we pick (per iteration) top α percentile of occurring cluster sizes as the threshold.[9] Any cluster larger than α is evenly broken into smaller clusters of maximum size α. Moreover, to minimize communication overhead among computational units, we actually only cluster and redistribute the data indices. This entails that the actual data must reside in a shared memory structure which is efficiently and dynamically accessible by all the computational units. Alternatively, all the units require their own copy of the whole data set.

Here we have introduced and illustrated the gains of dynamic data repartitioning technique in the context of a specific sparse coding model, which in fact involves computationally expensive, state-dependent operations for computing posterior distributions. The technique however is inherently generic and can be straight-forwardly employed for other types of multi-causal models.

8. The observed and the latent dimensions of the GSC model were 25 and 20 respectively. The truncation approximation parameters H' and γ (maximum number of active causes in a given latent state) were 8 and 5 respectively.
9. The α for the reported experiments was 5.

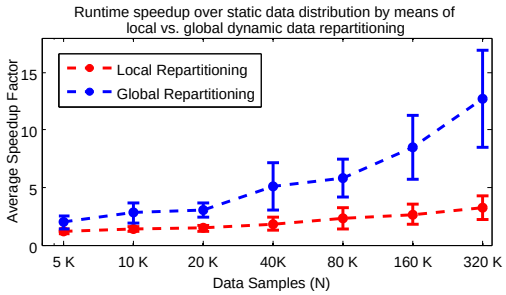


Figure 11: Runtime speedup of the truncated variational E-step (Equations (21) to (23)) with the static data distribution strategy taken as a baseline. The red plot shows the speedup when initially uniformly distributed data samples were only clustered locally by each processing unit, while the blue plot shows the speedup as a result of globally clustering and redistributing the data. The runtimes include the time taken by clustering and repartitioning modules.

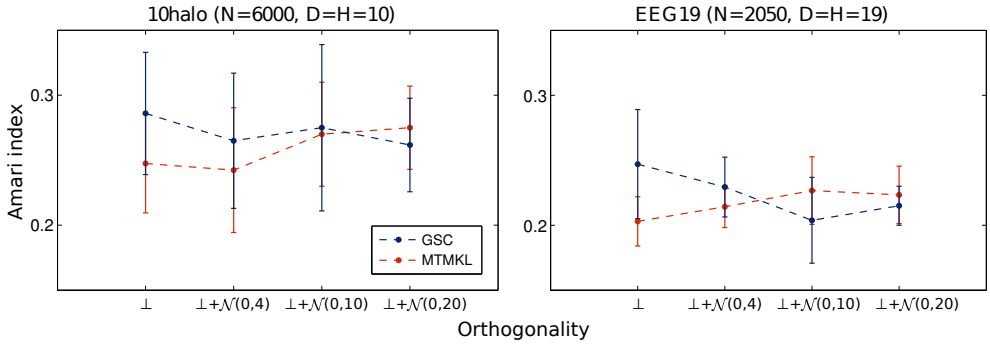


Figure 12: Source separation with observation noise. Performance of GSC vs. MTMKL on 10halo and EEG19 benchmarks with varying degrees of orthogonality of the mixing bases and Gaussian noise added to observations. Performance of GSC vs. MTMKL on the Speech20 benchmark with varying degrees of orthogonality of the mixing bases with Gaussian noise added to observed data. The orthogonality on the x-axis varies from being orthogonal $\perp$ to increasingly non-orthogonal mixing as randomly generated orthogonal bases are perturbed by adding Gaussian noise $\mathcal{N}(0, \sigma)$ to them. Performance is compared on the Amari index (27).

References

S. Amari, A. Cichocki, and H. H. Yang. A new learning algorithm for blind signal separation. In *Advances in Neural Information Processing Systems*, pages 757–763, 1995.

H. Attias. Independent factor analysis. *Neural Computation*, 11:803–851, 1999.

C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

J. Bornschein, Z. Dai, and J. Lücke. Approximate EM learning on large computer clusters. In *NIPS Workshop on Big Learning*, 2010.

J. Bornschein, M. Henniges, and J. Lücke. Are v1 simple cells optimized for visual occlusions? A comparative study. *PLoS Computational Biology*, 9(6):e1003062, 2013.

P. Carbonetto and M. Stephen. Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies. *Bayesian Analysis Journal*, 2011.

C. M. Carvalho, J. Chang, J. E. Lucas, J. R. Nevins, Q. Wang, and M. West. High-dimensional sparse factor modeling: Applications in gene expression genomics. *Journal of the American Statistical Association*, 103(484):1438–1456, 2008.

A. Cichocki, S. Amari, K. Siwek, T. Tanaka, A.H. Phan, and R. Zdunek. ICALAB-MATLAB Toolbox Version 3, 2007.

Z. Dai and J. Lücke. Autonomous cleaning of corrupted scanned documents — a generative modeling approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3338–3345, 2012.

R. C. Dalen and M. J. F. Gales. Covariance modelling for noise-robust speech recognition. In *Annual Conference of the International Speech Communication Association*, pages 2000–2003, 2008.

R. C. Dalen and M. J. F. Gales. Extended VTS for noise-robust speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 19(4), 2011.

P. Földiák. Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, 64:165–170, 1990.

J. V. Gael, Y. W. Teh, and Z. Ghahramani. The infinite factorial hidden Markov model. In *Advances in Neural Information Processing Systems*, pages 1697–1704, 2008.

P. Garrigues and B. A. Olshausen. Learning horizontal connections in a sparse coding model of natural images. In *Advances in Neural Information Processing Systems*, 2007.

I. Goodfellow, A. Courville, and Y. Bengio. Scaling up spike-and-slab models for unsupervised feature learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1902–1914, 2013.

P. O. Hoyer. Non-negative sparse coding. In *Neural Networks for Signal Processing XII: Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 557–565, 2002.

A. Ilin and H. Valpola. On the effect of the form of the posterior approximation in variational learning of ICA models. *Neural Processing Letters*, 22(2):183–204, 2005.

M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

D. Knowles and Z. Ghahramani. Infinite sparse factor analysis and infinite independent components analysis. In *Proceedings of International Conference on Independent Component Analysis and Signal Separation*, pages 381–388, 2007.

D. Knowles and Z. Ghahramani. Nonparametric Bayesian sparse factor models with application to gene expression modeling. *The Annals of Applied Statistics*, 5(2B):1534–1552, 2011.

H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, pages 801–08, 2007.

H. Li and F. Liu. Image denoising via sparse and redundant representations over learned dictionaries in wavelet domain. In *Proceedings of International Conference on Image and Graphics*, pages 754–758, 2009.

J. Lücke and J. Eggert. Expectation truncation and the benefits of preselection in training generative models. *Journal of Machine Learning Research*, 11:2855–2900, 2010.

J. Lücke and M. Sahani. Maximal causes for non-linear component extraction. *Journal of Machine Learning Research*, 9:1227–1267, 2008.

J. Lücke and A. S. Sheikh. Closed-form EM for sparse coding and its application to source separation. In *Proceedings of International Conference on Latent Variable Analysis and Signal Separation*, pages 213–221, 2012.

D. J. C. MacKay. Local minima, symmetry-breaking, and model pruning in variational free energy minimization. Online publication: www.inference.phy.cam.ac.uk/mackay/minima.ps.gz, 2001.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of International Conference on Machine Learning*, page 87, 2009.

T. J. Mitchell and J. J. Beauchamp. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1032, 1988.

S. Mohamed, K. Heller, and Z. Ghahramani. Evaluating Bayesian and L1 approaches for sparse unsupervised learning. In *Proceedings of International Conference on Machine Learning*, 2012.

E. Moulines, J.-F. Cardoso, and E. Gassiat. Maximum likelihood for blind separation and deconvolution of noisy signals using mixture models. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 5, pages 3617–3620, 1997.

G. J. Mysore, P. Smaragdis, and B. Raj. Non-negative hidden Markov modeling of audio with application to source separation. In *Proceedings of International Conference on Latent Variable Analysis and Signal Separation*, pages 140–148, 2010.

R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998.

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

B. Olshausen and D. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

B. Olshausen and K. Millman. Learning sparse codes with a mixture-of-Gaussians prior. In *Advances in Neural Information Processing Systems*, volume 12, pages 841–847, 2000.

J. Paisley and L. Carin. Nonparametric factor analysis with beta process priors. In *Proceedings of International Conference on Machine Learning*, pages 777–784, 2009.

G. Puertas, J. Bornschein, and J. Lücke. The maximal causes of natural scenes are edge filters. In *Advances in Neural Information Processing Systems*, volume 23, pages 1939–47, 2010.

M.'A. Ranzato and G. Hinton. Modeling pixel means and covariances using factorized third-order boltzmann machines. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2551–2558, 2010.

M. Rattray, O. Stegle, K. Sharp, and J. Winn. Inference algorithms and learning theory for Bayesian sparse factor analysis. *Journal of Physics: Conference Series*, 197:012002 (10pp), 2009.

F. Ribeiro and M. Opper. Expectation propagation with factorizing distributions: A gaussian approximation and performance results for simple models. *Neural Computation*, 23: 1047–1069, 2011.

M. Seeger. Bayesian inference and optimal design for the sparse linear model. *Journal of Machine Learning Research*, 9:759–813, 2008.

J. Shelton, J. Bornschein, A.-S. Sheikh, P. Berkes, and J. Lücke. Select and sample — a model of efficient neural inference and learning. In *Advances in Neural Information Processing Systems*, pages 2618–2626, 2011.

M. Spratling. Learning image components for object recognition. *Journal of Machine Learning Research*, 7:793–815, 2006.

T. Suzuki and M. Sugiyama. Least-squares independent component analysis. *Neural Computation*, 23(1):284–301, 2011.

Y. W. Teh, D. Görür, and Z. Ghahramani. Stick-breaking construction for the Indian buffet process. *Journal of Machine Learning Research*, 2:556–563, 2007.

M. Titsias and M. Lazaro-Gredilla. Spike and slab variational inference for multi-task and multiple kernel learning. In *Advances in Neural Information Processing Systems*, pages 2339–2347, 2011.

R. E. Turner and M. Sahani. Two problems with variational expectation maximisation for time-series models. In *Bayesian Time Series Models*. Cambridge University Press, 2011.

M. West. Bayesian factor regression models in the "large p, small n" paradigm. *Bayesian Statistics*, pages 723–732, 2003.

R. Yoshida and M. West. Bayesian learning in sparse graphical factor models via variational mean-field annealing. *Journal of the American Statistical Association*, 99:1771–1798, 2010.

M. Zhou, H. Chen, J. Paisley, L. Ren, G. Sapiro, and L. Carin. Non-parametric Bayesian dictionary learning for sparse image representations. In *Advances in Neural Information Processing Systems*, pages 2295–2303, 2009.

Journal of Machine Learning Research 15 (2014) 2689-2722 Submitted 7/12; Revised 9/13; Published 8/14

Efficient Occlusive Components Analysis

Marc Henniges HENNIGES@FIAS.UNI-FRANKFURT.DE
Frankfurt Institute for Advanced Studies
Goethe-University Frankfurt, 60438 Frankfurt, Germany

Richard E. Turner RET26@CAM.AC.UK
Department of Engineering
University of Cambridge, Cambridge, CB2 1PZ, UK

Maneesh Sahani MANEESH@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit
University College London, London, WC1N 3AR, UK

Julian Eggert JULIAN.EGGERT@HONDA-RI.DE
Honda Research Institute Europe GmbH
63073 Offenbach am Main, Germany

Jörg Lücke JOERG.LUECKE@UNI-OLDENBURG.DE
Cluster of Excellence Hearing4all and Faculty VI
University of Oldenburg, 26115 Oldenburg, Germany
Electrical Engineering and Computer Science
Technical University Berlin, 10587 Berlin, Germany
Frankfurt Institute for Advanced Studies (previous affiliation)
Goethe-University Frankfurt, 60438 Frankfurt, Germany

Editor: Daniel Lee

Abstract

We study unsupervised learning in a probabilistic generative model for occlusion. The model uses two types of latent variables: one indicates which objects are present in the image, and the other how they are ordered in depth. This depth order then determines how the positions and appearances of the objects present, specified in the model parameters, combine to form the image. We show that the object parameters can be learned from an unlabeled set of images in which objects occlude one another. Exact maximum-likelihood learning is intractable. Tractable approximations can be derived, however, by applying a truncated variational approach to Expectation Maximization (EM). In numerical experiments it is shown that these approximations recover the underlying set of object parameters including data noise and sparsity. Experiments on a novel version of the bars test using colored bars, and experiments on more realistic data, show that the algorithm performs well in extracting the generating components. The studied approach demonstrates that the multiple-causes generative approach can be generalized to extract occluding components, which links research on occlusion to the field of sparse coding approaches.

Keywords: generative models, occlusion, unsupervised learning, sparse coding, expectation truncation

1. Introduction

A key problem in image analysis is to learn the shape and form of objects directly from unlabeled data. Many approaches to this unsupervised learning problem have been motivated by the observation that, although the number of objects appearing across all images is vast, the number appearing in any one image is far smaller. This property, a form of sparsity, has motivated a number of algorithms including sparse coding (SC; Olshausen and Field, 1996) and non-negative matrix factorization (NMF; Lee and Seung, 1999) with its sparse variants (e.g., Hoyer, 2004). These approaches can be framed as latent-variable models, where each possible object, or part of an object, is associated with a latent variable controlling its presence or absence in a given image. Any individual "hidden cause" is rarely active, corresponding to the small number of objects present in any one image. Despite this plausible motivation, SC or NMF make severe assumptions which coarsely approximate the physical process by which images are produced. Perhaps the most crucial assumption is that in the underlying latent variable models, objects or parts thereof, combine *linearly* to form the image. In real images the combination of individual objects depends on their relative distance from the camera or eye. If two objects occupy the same region in planar space, the nearer one occludes the other, i.e., the hidden causes non-linearly compete to determine the pixel values in the region of overlap.

In this paper we extend multiple-causes models such as SC or NMF to handle occlusion. The idea of using many hidden "cause" variables to control the presence or absence of objects is retained, but these variables are augmented by another set of latent variables which determine the relative depth of the objects, much as in the z-buffer employed by computer graphics. In turn, this enables the simplistic linear combination rule to be replaced by one in which nearby objects occlude those that are more distant. One of the consequences of moving to a richer, more complex model is that inference and learning become correspondingly harder. One of the main contributions of this paper is to show how to overcome these difficulties.

The problem of occlusion has been addressed in different contexts (Jojic and Frey, 2001; Williams and Titsias, 2004; Fukushima, 2005; Eckes et al., 2006; Lücke et al., 2009; LeRoux et al., 2011; Tajima and Watanabe, 2011). Probabilistic 'sprite' models (e.g., Jojic and Frey, 2001; Williams and Titsias, 2004) assign pixels in multiple images taken from the same scene to a fixed number of image layers. The approach is most frequently applied to automatically remove foreground and background objects. Those models are in many aspects more general than the approach discussed here. However, in contrast to our approach, they model data in which objects maintain a fixed position in depth relative to the other objects. Other approaches study occlusion in the context of neural network models (Tajima and Watanabe, 2011) or generalized versions of restricted Boltzmann machines (RBMs) which incorporate occlusion (LeRoux et al., 2011). This paper takes a new and different approach which can be regarded as a generalization of sparse coding to model occlusion.

2. A Generative Model for Occlusion

The occlusion model contains three important elements. The first is a set of variables which controls the presence or absence of objects in a particular image (this part will be analogous,

e.g., to NMF or sparse coding). The second is a variable which controls the relative depths of the objects that are present. The third is the combination rule which describes how active objects which are closer occlude more distant ones. The second and third part are the distinguishing features of the model. They describe how values of observed variables are determined by the occlusion non-linearity given a set of hidden variables. While the occlusion model will be applicable to the same data as NMF or sparse coding, and while efficient inference and learning will require sparsity, explicitly modeled occlusions will define solutions different from these models with linear combination rules. Furthermore, more general features per observed variable such as color vectors can be taken into account.

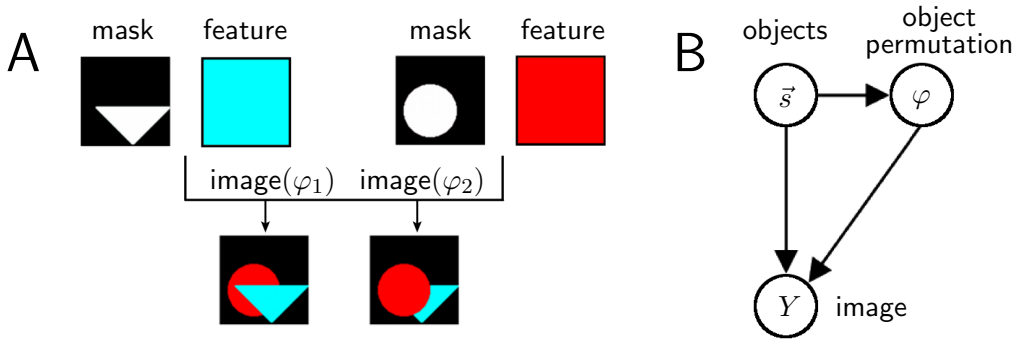


Figure 1: **A** Illustration of how object masks and features combine to generate an image. If two objects are randomly chosen ($|\vec{s}| = 2$), two different images with two different depth-orders (denoted by φ_1 and φ_2) can be generated. **B** Graphical model of the generation process with hidden variables $\vec{s}$ (object presence) and φ (depth permutation).

To model the presence or absence of objects we use H binary hidden variables $s_1, \ldots, s_H$. We assume that the presence of one object is independent of the presence of the others and, for simplicity, we also assume that each object is equally likely to be present in an image *a priori* (we refer to this probability by π). The probability for the presence and absence of objects is given by

$$p(\vec{s}\,|\,\pi) \;=\; \prod_{h=1}^{H} \text{Bernoulli}(s_h;\pi) \;=\; \prod_{h=1}^{H} \pi^{s_h}\,(1-\pi)^{1-s_h}. \tag{1}$$

Objects in a real image can be ordered by their depth and it is this ordering which determines how the objects occlude each other in regions of overlap. The depth-ordering is captured in the model by associating the active objects with a permutation. We randomly and uniformly choose a member φ of the set $\mathcal{G}(|\vec{s}|)$ which contains all permutation functions $\varphi : \{\tilde{h}_1, \ldots, \tilde{h}_{|\vec{s}|}\} \rightarrow \{1, \ldots, |\vec{s}|\}$, with $|\vec{s}| = \sum_h s_h$, where $\tilde{h}_1$ to $\tilde{h}_{|\vec{s}|}$ are the indices of the non-zero entries of $\vec{s}$. More formally, the probability of φ given $\vec{s}$ (see Figure 1B) is defined by

$$p(\varphi\,|\,\vec{s}) = \tfrac{1}{|\vec{s}|!} \quad \text{with} \quad \varphi \in \mathcal{G}(|\vec{s}|)\,. \tag{2}$$

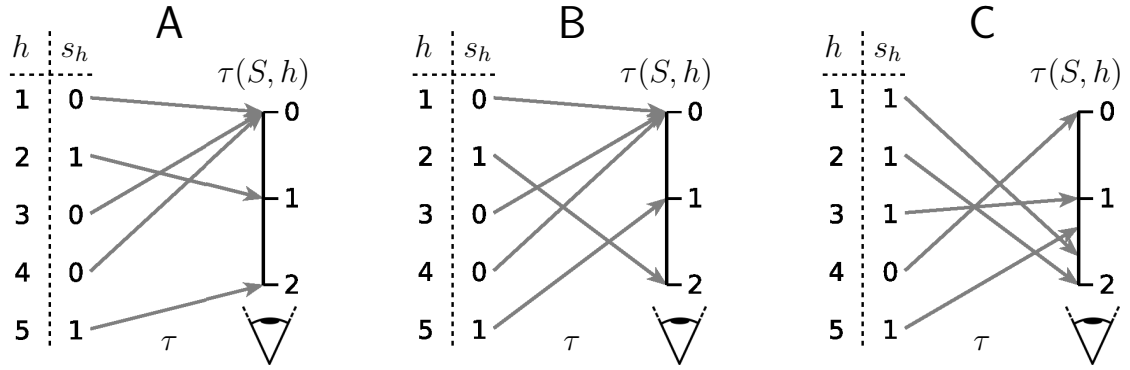


Figure 2: Visualization of the mapping $\tau(S) : \{1, \ldots, H\} \rightarrow [0, 2]$ which represents different permutations of objects in depth. The eye at the bottom illustrates the position of the observer. **A** and **B** show the two possible mappings if two causes are present. **C** shows one of the 24 mappings if four causes are present.

Note that we could have defined the prior over the order in depth (Equation 2) independently of $\vec{s}$, by choosing from $\mathcal{G}(H)$ with $p(\varphi) = \frac{1}{H!}$. But then, because the depth of absent objects ($s_h = 0$) is irrelevant, no more than $|\vec{s}|!$ distinct choices of φ would have resulted in different images.

The final stage of the generative model describes how to produce the image given a selection of active causes and an ordering in relative depth of these causes. One approach would be to choose the closest object and to set the image equal to the feature vector associated with this object. However, this would mean that every image generated from the model would comprise just one object: the closest. What is missing from this description is a notion of the extent of an object and the fact that it might only contribute to a subset of pixels in an image. For this reason, our model contains two sets of object parameters. One set of parameters, $W \in \mathbb{R}^{H \times D}$, describes whether an object contributes to a pixel and the strength of that contribution (D is the number of pixels). The vector $(W_{h1}, \ldots, W_{hD})$ is therefore described as the *mask* of object h. If an object is highly localized, this vector will contain many zero elements. The other set of parameters, $T \in \mathbb{R}^{H \times C}$, represents the features of the objects. We define one vectorial feature per object h, $\vec{T}_h \in \mathbb{R}^C$, describing, for instance, the object's RGB color ($C = 3$ in that case). Figure 1A illustrates the combination of masks and features, and Figure 1B shows the graphical model of the generation process.

Let us formalize how an image is generated given the parameters W and T and given the hidden variables $S = (\vec{s}, \varphi)$. To further abbreviate the notation, we will denote all the model's parameters by $\Theta = (W, T, \pi, \sigma)$. We define the generation of a noiseless image $\vec{\mathcal{T}}(S, \Theta)$ to be given by the following equations:

$$\vec{\mathcal{T}}_d(S,\Theta) \;=\; W_{h_o d}\,\vec{T}_{h_o} \qquad \text{where} \quad h_o \;=\; \text{argmax}_h\{\tau(S,h)\,W_{hd}\}\,, \qquad \tau(S,h) = \begin{cases} 0 & \text{if } s_h = 0 \\ \frac{3}{2} & \text{if } s_h = 1 \text{ and } |\vec{s}| = 1 \\ \frac{\varphi(h)-1}{|\vec{s}|-1} + 1 & \text{otherwise} \end{cases} \tag{3}$$

In Equation 3 the order in depth is represented by the mapping τ which intuitively can be thought of as the relative proximity of the objects. The form of this mapping has been chosen to facilitate later algebraic steps. To illustrate the combination rule of Equation 3 and the mapping τ consider Figure 1A and Figure 2. Let us assume that the mask values W_{hd} are zero or one (although we will later also allow for continuous values). As depicted in Figure 1A an object h with $s_h = 1$ occupies all image pixels with $W_{hd} = 1$ and does not occupy pixels with $W_{hd} = 0$. For all pixels with $W_{hd} = 1$ the vector $\vec{T}_h$ sets the pixels' values to a specific feature, e.g., to a specific color. The function τ maps all causes h with $s_h = 0$ to zero while all other causes are mapped to values within the interval $[1, 2]$ (see Figure 2). In this way, it assigns a proximity value $\tau(S, h) > 0$ to each present object. For a given pixel d the combination rule in Equation 3 simply states that of all objects with $W_{hd} = 1$, the most proximal is used to set the pixel property. The interval [1,2] represents a natural choice for proximity values, but any interval with boundaries greater zero would result in an equivalent generative process.

Given the latent variables and the noiseless image $\vec{\mathcal{T}}(S, \Theta)$, we take the observed variables $Y = (\vec{y}_1, \ldots, \vec{y}_D)$ to be drawn independently from a Gaussian distribution, i.e.,

$$p(Y \,|\, S, \Theta) \;=\; \prod_{d=1}^{D} p(\vec{y}_d \,|\, \vec{\mathcal{T}}_d(S, \Theta)), \qquad p(\vec{y} \,|\, \vec{t}\,) \;=\; \mathcal{N}(\vec{y}; \vec{t}, \sigma^2 \mathbb{1})\,. \tag{4}$$

Equations 1 to 4 represent a model for image generation that incorporates occlusion. We will refer to the model as the *Occlusive Components Analysis* (OCA) generative model.

3. Maximum Likelihood

One approach to learning the parameters $\Theta = (W, T, \pi, \sigma)$ of this model from data $\mathcal{Y} = \{Y^{(n)}\}_{n=1,\ldots,N}$ is to use maximum likelihood learning, that is,

$$\Theta^* \;=\; \text{argmax}_\Theta\{\mathcal{L}(\Theta)\} \quad \text{with} \quad \mathcal{L}(\Theta) \;=\; \log\left(p(Y^{(1)}, \ldots, Y^{(N)} \,|\, \Theta)\right). \tag{5}$$

However, as there is usually a large number of objects that can potentially be present in the training images, and since the likelihood involves summing over all combinations of objects and associated orderings, the computation of Equation 5 is typically intractable. More concretely, given H components the number of different sets of objects that may be present scales with 2^H. Occlusion adds additional complexity: for any subset of size γ' of the H objects that may be present there are $\gamma'!$ different depth orders. Formally, the total number of hidden states to be considered is given by

$$\text{States}_{\text{exact}}(H) \;=\; \sum_{\gamma'=0}^{H} \binom{H}{\gamma'} \gamma'!\,. \tag{6}$$

The need to consider depth-order to model occlusion means that the number of hidden states scales super-exponentially with the number of potential components H. Moreover, even if this computational tractability problem can be overcome, optimization of the likelihood is made problematic by an analytical intractability arising from the fact that the occlusion non-linearity is non-differentiable. The following section describes how to side-step both

of these intractabilities within the standard Expectation Maximization (EM) formalism for maximum likelihood learning. First, we will describe how the analytical intractability may be avoided using an approximation that softens the occlusion non-linearity, and which therefore allows parameter update equations (M-step equations) to be derived. Second, we will describe how the computational intractability can be addressed by leveraging the sparsity of visual scenes to reduce the space of solutions entertained by the posterior.

To find the maximum-likelihood parameters Θ^*, at least approximately, we use the EM formalism in the form used by Neal and Hinton (1998) and introduce the free-energy function $\mathcal{F}(\Theta, q)$ which is a function of Θ and of an unknown distribution $q(S^{(1)}, \ldots, S^{(N)})$ over the hidden variables. $\mathcal{F}(\Theta, q)$ is a lower bound of the likelihood $\mathcal{L}(\Theta)$. Approximations introduced later on can be interpreted as constraining the function q to lie within a specified class. In the model described above each image is assumed to be drawn independently and identically from an underlying distribution, $q(S^{(1)}, \ldots, S^{(N)}) = \prod_n q_n(S^{(n)}, \Theta')$, which results in the free-energy

$$\mathcal{F}(\Theta, q) \;=\; \sum_{n=1}^{N} \Big[\sum_{S} \; q_n(S\,; \Theta') \;\; \Big[\log\big(p(Y^{(n)} \,|\, S, \Theta)\big) + \log\big(p(S \,|\, \Theta)\big) \Big] \Big] + \mathbf{H}[q]\,, \quad (7)$$

where the function $\mathbf{H}[q] = -\sum_n \sum_S q_n(S\,; \Theta') \log(q_n(S\,; \Theta'))$ (the Shannon entropy) is independent of Θ. Note that $\sum_S$ in Equation 7 sums over all possible states of $S = (\vec{s}, \varphi)$, i.e., over all binary vectors and all associated permutations in depth, so that the number of terms in the sum is given by Equation 6. These large sums are the source of the computational intractability. In the EM scheme, $\mathcal{F}(\Theta, q)$ is maximized alternately with respect to the distribution q in the E-step (while the parameters Θ are kept fixed) and with respect to parameters Θ in the M-step (while q is kept fixed). Θ' refers to the model parameters of the previous iteration of the algorithm. At the end of the M-step, we thus set $\Theta' \leftarrow \Theta$. Each EM iteration increases the free-energy or leaves it unchanged. If q is unconstrained (or if any constraints imposed allow it) then the optimal setting of q in the E-step is given by the posterior distribution over the hidden states at the current parameter settings $q_n(S; \Theta') = p(S \,|\, Y^{(n)}, \Theta')$. In this case, each EM step increases the likelihood or leaves it unchanged and this process converges to a (local) maximum of the likelihood.

3.1 M-Step Equations

The M-step of EM, in which the free-energy, $\mathcal{F}$, is optimized with respect to the parameters, is usually derived by taking derivatives of $\mathcal{F}$ with respect to the parameters. Unfortunately, this standard procedure is not directly applicable because the occlusive combination rule in Equation 3 is not differentiable. However, it is possible to soften the combination rule using the differentiable approximation

$$\vec{\mathcal{T}}^{\rho}{}_d(S, \Theta) \;:=\; \frac{\sum_{h=1}^{H} (\tau(S,h)\, W_{hd})^{\rho}\, W_{hd}\, \vec{T}_h}{\sum_{h=1}^{H} (\tau(S,h)\, W_{hd})^{\rho}}, \quad (8)$$

which becomes equal to the combination rule in Equation 3 as $\rho \to \infty$. Note that for the softened combination rule with small values of ρ, the choice of the interval for the proximity values (Equation 3) can now have an effect. According to Equation 8 the hidden

states combine in the sense of a softmax operation, and different interval boundaries for the proximity values $\tau(S,h)$ change how strongly the closest cause dominates the others. Such effects can be counteracted by choosing corresponding finite values for ρ, however. For large values of ρ, differences due to different intervals become negligible again.

$\vec{\mathcal{T}}^{\rho}{}_d(S,\Theta)$ is differentiable w.r.t. the parameters W_{hd} and T_h^c (with $c \in \{1,\ldots,C\}$). For large ρ, the derivatives can be well approximated as follows:

$$\begin{array}{l} \frac{\partial}{\partial W_{id}} \vec{\mathcal{T}}^{\rho}{}_d(S,\Theta) \approx \mathcal{A}^{\rho}_{id}(S,W)\, \vec{T}_i, \\ \frac{\partial}{\partial T_i^c} \vec{\mathcal{T}}^{\rho}{}_d(S,\Theta) \approx \mathcal{A}^{\rho}_{id}(S,W)\, W_{id}\, \vec{e}_c, \end{array} \quad \text{with} \quad \begin{array}{l} \mathcal{A}^{\rho}_{id}(S,W) := \frac{(\tau(S,i)\, W_{id})^{\rho}}{\sum_{h=1}^{H} (\tau(S,h)\, W_{hd})^{\rho}}, \\ \mathcal{A}_{id}(S,W) := \lim\limits_{\rho\to\infty} \mathcal{A}^{\rho}_{id}(S,W)\,, \end{array} \tag{9}$$

where $\vec{e}_c$ is a unit vector in feature space with entry equal one at position c and zero elsewhere. The approximations on the left-hand-side above become equalities for $\rho \to \infty$. Given the approximate combination rule in Equation 9, we can compute approximations to the derivatives of $\mathcal{F}(\Theta, q)$. For large values of ρ the following holds (see Appendix B):

$$\frac{\partial}{\partial W_{id}} \mathcal{F}(\Theta,q) \approx \sum_{n=1}^{N} \left[\sum_{S} q_n(S\,;\Theta') \left(\frac{\partial}{\partial W_{id}} \vec{\mathcal{T}}^{\rho}{}_d(S,\Theta) \right)^T \vec{f}\left(\vec{y}^{\,(n)}, \vec{\mathcal{T}}^{\rho}{}_d(S,\Theta) \right) \right], \tag{10}$$

$$\frac{\partial}{\partial T_i^c} \mathcal{F}(\Theta,q) \approx \sum_{n=1}^{N} \left[\sum_{S} q_n(S\,;\Theta') \sum_{d=1}^{D} \left(\frac{\partial}{\partial T_i^c} \vec{\mathcal{T}}^{\rho}{}_d(S,\Theta) \right)^T \vec{f}\left(\vec{y}^{\,(n)}, \vec{\mathcal{T}}^{\rho}{}_d(S,\Theta) \right) \right], \tag{11}$$

$$\text{where} \quad \vec{f}(\vec{y}^{\,(n)}, \vec{t}\,) := \frac{\partial}{\partial \vec{t}} \log\left(p(\vec{y}^{\,(n)} \,|\, \vec{t}\,) \right) = -\sigma^{-2}\, (\vec{y}^{\,(n)} - \vec{t}\,).$$

Setting the derivatives in Equations 10 and 11 to zero and inserting Equations 9 yields the following necessary conditions for a maximum of the free-energy that hold in the limit $\rho \to \infty$:

$$W_{id} = \frac{\sum\limits_{n} \langle \mathcal{A}_{id}(S,W) \rangle_{q_n}\, \vec{T}_i^T\, \vec{y}_d^{\,(n)}}{\sum\limits_{n} \langle \mathcal{A}_{id}(S,W) \rangle_{q_n}\, \vec{T}_i^T\, \vec{T}_i}\,, \quad \vec{T}_i = \frac{\sum\limits_{n}\sum\limits_{d} \langle \mathcal{A}_{id}(S,W) \rangle_{q_n}\, W_{id}\, \vec{y}_d^{\,(n)}}{\sum\limits_{n}\sum\limits_{d} \langle \mathcal{A}_{id}(S,W) \rangle_{q_n}\, (W_{id})^2}\,. \tag{12}$$

Note that Equations 12 are not straightforward update rules. However, we can use them in the fixed-point sense and approximate the parameters which appear on the right-hand-side of the equations using the values from the previous iteration. For the update note that due to the multiplication of the weights and the mask, $W_{hd}\,\vec{T}_h$ in Equation 3, there is degeneracy for the object parameters: given h, the combination $\vec{\mathcal{T}}_d$ remains unchanged for the operation $\vec{T}_h \to \vec{T}_h/\varrho$ and $W_{hd} \to \varrho\, W_{hd}$ with $\varrho \neq 0$. This transformation does not leave the selection of the closest object unchanged (selection of h_o in Equation 3) because the values of W_{hd} are not binary. To remove the degeneracy and to keep the values of W_{hd} close to zero or one, we rescale after each EM iteration as follows:

$$W_{hd}^{\text{new}} = W_{hd} / \overline{W}_h, \quad \vec{T}_h^{\text{new}} = \overline{W}_h \vec{T}_h,$$

$$\text{where} \quad \overline{W}_h = \frac{1}{|\mathcal{I}|} \sum_{d\in\mathcal{I}} W_{hd} \quad \text{with} \quad \mathcal{I} = \{d \,|\, |W_{hd}| > \alpha\} \quad \text{where} \quad \alpha \in \mathbb{R}.$$

The use of $\overline{W}_h$ instead of, e.g., $W_h^{\max} = \max_d\{W_{hd}\}$ is advantageous for some data, although for many other types of data $W_h^{\max}$ works equally well. Through the influence of the scaling on the selection of closest objects, small values of W_{hd} tend to be suppressed for larger values of α and converge to zero. In general, we find the algorithm to avoid local optima more frequently if we initialize α at a small value and then slowly increase it over the EM iterations to a value near $\frac{1}{2}$. The index set $\mathcal{I}$ thus contains all entries W_{hd} at first, and only later considers exclusively entries with higher values for normalization. In this way, smaller values are suppressed only when the algorithm is already closer to an optimum than it is in the beginning of learning.

If the derivatives of the free-energy in Equation 7 w.r.t. to σ (data noise) and π (appearance frequency) are set to zero, we obtain through straightforward derivations (see Appendix B) the following two remaining update rules:

$$\sigma^{\text{new}} = \sqrt{\frac{1}{NDC}\sum_{n=1}^{N}\left\langle\sum_{d=1}^{D}\sum_{c=1}^{C}\left(y_{dc}^{(n)} - \mathcal{T}_{dc}(S,\Theta)\right)^2\right\rangle_{q_n}}\,, \tag{13}$$

$$\pi^{\text{new}} = \frac{1}{HN}\sum_{n=1}^{N}\langle|\vec{s}|\rangle_{q_n}\,. \tag{14}$$

3.2 E-Step Equations

The crucial quantities that have to be computed for update Equations 12 to 14 are expectation values w.r.t. the variational distributions $q_n(S\,;\Theta')$ in the form

$$\langle g(S,\Theta)\rangle_{q_n} \;=\; \sum_S q_n(S\,;\Theta')\,g(S,\Theta)\,, \tag{15}$$

where $g(S,\Theta)$ are functions that depend on the latent state and potentially the model parameters. The optimal choice for $q_n(S\,;\Theta')$ is the exact posterior, $q_n(S\,;\Theta') = p(S\,|\,Y^{(n)},\Theta')$, which is given by Bayes' rule, $p(S\,|\,Y^{(n)},\Theta') = \frac{p(Y^{(n)}\,|S,\Theta')\,p(S\,|\,\Theta')}{\sum_{S'}p(Y^{(n)}\,|S',\Theta')\,p(S'|\,\Theta')}$, with prior and noise distributions given by the OCA generative model in Equations 1 to 4. Unfortunately, the computation of the expectations or *sufficient statistics* in Equation 15 becomes computationally intractable in this case because of the large number of states that have to be considered (see Equation 6). To derive tractable approximations, we can, however, make use of typical properties of visual scenes: in any given scene the number of objects which are present is far smaller than the set of all objects that can potentially be present in the scene. As such, the sum over all states in Equation 15 is typically dominated by only a few terms. More specifically, components which are compatible with the observed image are the only ones to make a significant contribution to this sum, whilst components which are incompatible make only a negligible contribution. Consequently, a good approximation to the expectation values in Equation 15 can be obtained by identifying the states which carry high posterior mass and retaining only these states.

It has recently been shown (Lücke and Eggert, 2010) that this general idea (see Yuille and Kersten, 2006) can be considered as approximate variational EM. When the variational distribution q in Equation 7 is imperfectly optimized, or optimized within a constrained

space of functions, then the resulting variational EM algorithm is no longer guaranteed to converge to a local maximum of the likelihood. However, it still increases a lower bound on the likelihood, and frequently finds a good approximation to the maximum likelihood solution. The most commonly used constraint is to decompose q into a product of disjoint factors, for instance, one for each possible source object. By contrast, the approach adopted here uses a distribution truncated to a limited set $\mathcal{K}_n$ of all possible source configurations, i.e,

$$q_n(S;\Theta) = \frac{p(S \,|\, Y^{(n)},\Theta)}{\sum\limits_{S'\in\mathcal{K}_n} p(S' \,|\, Y^{(n)},\Theta)}\,\delta(S\in\mathcal{K}_n) \quad \text{with} \quad \delta(S\in\mathcal{K}_n) := \begin{cases} 1 & \text{if } S\in\mathcal{K}_n \\ 0 & \text{if } S\notin\mathcal{K}_n \end{cases}, \tag{16}$$

where $\mathcal{K}_n$ is a subset of the space of all states. If $\mathcal{K}_n$, indeed, contains most of the posterior mass given a data point $Y^{(n)}$, then $q_n(S;\Theta)$ approximates the exact posterior well. The variational approximation $q_n(S;\Theta)$ is a truncated posterior distribution that allows for the efficient estimation of the necessary expected values. The approximation is, therefore, referred to as *Expectation Truncation* (ET; Lücke and Eggert, 2010) or *truncated EM*.

In the case of the OCA generative model, we might expect good approximations if we identified a small set of candidate objects which are likely to be present in the scene, and then let $\mathcal{K}_n$ contain all of the combinations of the candidate objects. By using $q_n(S;\Theta)$ in Equation 16 as a variational distribution, the expectation values required for the M-step are of the form

$$\langle g(S,\Theta)\rangle_{q_n} = \sum_{S\in\mathcal{K}_n} \frac{p(S \,|\, Y^{(n)},\Theta)}{\sum\limits_{S'\in\mathcal{K}_n} p(S' \,|\, Y^{(n)},\Theta)}\, g(S,\Theta) = \frac{\sum\limits_{S\in\mathcal{K}_n} p(S,Y^{(n)}|\Theta')g(S,\Theta)}{\sum\limits_{S'\in\mathcal{K}_n} p(S',Y^{(n)}|\Theta')\,.} \tag{17}$$

We compute $\mathcal{K}_n$ for a given data point $Y^{(n)}$ in two stages. In the first we use a computationally inexpensive *selection* or *scoring* function (see Lücke and Eggert, 2010) to identify candidate objects. The selection function $\mathcal{S}_h\big(Y^{(n)}\big)$ seeks to assign high values to states corresponding to objects h present in the scene $Y^{(n)}$ and low values to states corresponding to objects which are not present. An ideal selection function would be monotonically related to the posterior probability of the object given the current image, but at the same time it would also be efficient to compute. The top H' states are selected as candidates and placed into an index set I_n. In the second stage we form the set $\mathcal{K}_n$ from the candidate objects. The index set I_n is used to define the set $\mathcal{K}_n$ as containing the states of all likely object combinations, i.e.,

$$\mathcal{K}_n = \{\, S \,|\, \big(\textstyle\sum_h s_h \leq \gamma \text{ and } \forall h\notin I_n : s_h = 0\big) \text{ or } \sum_j s_j \leq 1\}\,. \tag{18}$$

As an additional constraint, $\mathcal{K}_n$ does not contain combinations of more than γ objects. Furthermore, we make sure that $\mathcal{K}_n$ contains all singleton states, which proved beneficial in numerical experiments.

The selection function itself is defined as the squared distance between the observed image and the image generated by the hth component alone,

$$\mathcal{S}_h\big(Y^{(n)}\big) = -\sum_{d=1}^{D}\sum_{c=1}^{C}\big(Y^{(n)}_{cd} - \mathcal{T}_{cd}(S^{(h)};\Theta)\big)^2 \tag{19}$$

where $S^{(h)} := (\vec{s}^{(h)}, \varphi)$ with $\vec{s}^{(h)}$ being the state with only the hth object present.

Intuitively, the selection function can be thought of as a measure of the log-probability that each singleton state accounts for the current data point (also see Appendix D). Since we are only interested in the relative values of the selection function between the different components, Equation 19 contains only the exponent of the strictly monotonic exponential function without the normalization pre-factors.

3.3 Efficient EM Learning

The M-step Equations 12 to 14 together with the approximation of the expectation values in Equation 17 represent a learning algorithm for the OCA generative model. Its efficiency crucially depends on the approximation parameters H' and γ as they determine the number of latent states in $\mathcal{K}_n$ that have to be considered, that is,

$$\text{States}_{\text{ET}}(H, H', \gamma) \;=\; \sum_{\gamma'=0}^{\gamma} \binom{H'}{\gamma'} \gamma'! \;+\; (H - H')\,. \tag{20}$$

Because of the preselection of H' candidates, the combinatorics no longer scales with H. Only the number of singleton states scales linearly with H. Furthermore, the computation of the selection functions scales with H, but for the selection function specified in Equation 19 this scaling is only linear.

The potentially strongly reduced number of states in Equation 20 allows for an efficient optimization of the OCA model parameters (see Figure 8 in Appendix A for an example of such a reduction). By choosing H' and γ large enough to approximate the posteriors of the data points well, and small enough to sufficiently reduce the number of latent states that must be considered, an efficient yet accurate optimization procedure can be obtained. A crucial role for the efficiency / accuracy trade-off is played by the parameter γ which constrains the maximal number of considered components per data point (also compare Figure 8). If γ is too large, the large number of permutations that have to be considered for occlusion quickly results in computational intractabilities (scaling with $\gamma!$). If γ is too small, data points with more than γ components cannot be approximated well. Ideally we would like to learn the component parameters using as small an active set as possible (i.e., using low values of γ). However, representations of the posterior distribution which are too impoverished can result in strong biases in the parameter estimates (as is well-known, e.g., for factored variational approximations; Turner and Sahani, 2011). Since the approximation methods considered here will be at their worst for data-points that contain a large number of components, we simply discount these points and focus instead upon the data-points which are simple to learn from, thereby reducing the biases and the computational demands. This general approach was used by Lücke and Eggert (2010) who showed that such a discounting within the truncated approximate EM approach still results in approximately optimal solutions. Here we discount data-points that we believe contain more than γ components and modify the M-step equations accordingly. If we denote by $\mathcal{M}$ the subset of the N data points which are estimated to contain at most γ components, the new expressions are given

by:

$$W_{id} = \frac{\sum_{n\in\mathcal{M}} \langle \mathcal{A}_{id}(S,W)\rangle_{q_n}\, \vec{T}_i^T \vec{y}_d^{\,(n)}}{\sum_{n\in\mathcal{M}} \langle \mathcal{A}_{id}(S,W)\rangle_{q_n}\, \vec{T}_i^T \vec{T}_i} \qquad \vec{T}_i = \frac{\sum_{n\in\mathcal{M}} \sum_d \langle \mathcal{A}_{id}(S,W)\rangle_{q_n}\, W_{id}\, \vec{y}_d^{\,(n)}}{\sum_{n\in\mathcal{M}} \sum_d \langle \mathcal{A}_{id}(S,W)\rangle_{q_n}\, (W_{id})^2} \tag{21}$$

$$\sigma^{\text{new}} = \sqrt{\frac{1}{|\mathcal{M}|\, CD} \sum_{n\in\mathcal{M}} \left\langle \sum_{d=1}^{D} \sum_{c=1}^{C} \left(y_{dc}^{(n)} - \mathcal{T}_{dc}(S,\Theta) \right)^2 \right\rangle_{q_n}} \tag{22}$$

$$\pi^{\text{new}} = \frac{A(\pi)\,\pi}{B(\pi)} \frac{1}{|\mathcal{M}|} \sum_{n\in\mathcal{M}} \langle |\vec{s}| \rangle_{q_n} \;\text{ with} \tag{23}$$

$$A(\pi) = \sum_{\gamma\prime=0}^{\gamma} \binom{H}{\gamma\prime} \pi^{\gamma\prime} (1-\pi)^{H-\gamma\prime} \;\text{ and }\; B(\pi) = \sum_{\gamma\prime=0}^{\gamma} \gamma\prime \binom{H}{\gamma\prime} \pi^{\gamma\prime} (1-\pi)^{H-\gamma\prime}.$$

These modified M-step equations can be derived from a truncated free-energy (Lücke and Eggert, 2010) of the form

$$\mathcal{F}(q,\Theta) \;=\; \sum_{n\in\mathcal{M}} \sum_{S} \; q_n(S\,;\Theta') \;\log\Big(p(Y^{(n)} \,|\, S,\Theta)\, \frac{p(S\,|\,\Theta)}{\sum_{S'\in\mathcal{K}} p(S'\,|\,\Theta)} \Big), \tag{24}$$

with $q_n(S\,;\Theta')$ given in Equation 16 and with $\mathcal{K}$ being the set of all states S with at most γ non-zero components, $\mathcal{K} = \{S \,|\, |\vec{s}| \leq \gamma\}$. Details of the derivations of the update rules are given in Appendix B.

As can be observed, the update equations for W, T and σ remain essentially unchanged except for averages now running over the subset $\mathcal{M}$ instead of all data points (Equations 21 and 22). The reason is that the derivatives of the truncated free-energy w.r.t. these parameters are equal to the derivatives of the original free-energy except for reduced sums. For the derivative w.r.t. the prior parameter, the situation is different. The additional term in the denominator of the logarithm in Equation 24 results in a correction term for the update equation for π (Equation 23). Intuitively, it is clear that discounting data points with more than γ components has a direct impact on estimating the mean probability for a component to appear in a data point. The additional term in Equation 23 corrects for this.

To complete the procedure, we must determine the set $\mathcal{M}$ of all data points which are estimated to have γ active components or fewer. First note that the size of this set can be estimated given the current estimate for π. It contains an expected number of

$$N^{\text{cut}} = N \sum_{S,|\vec{s}|\leq\gamma} p(S|\pi) = N \sum_{\gamma'=0}^{\gamma} \binom{H}{\gamma'} \pi^{\gamma'} (1-\pi)^{H-\gamma'}$$

data points (analogously to the π correction factor in Appendix B.2). Following Lücke and Eggert (2010), we now compute for all N data points the sums $\sum_{S\in\mathcal{K}_n} p(S, Y^{(n)}|\Theta')$, and define the set $\mathcal{M}$ to consist of the N^{cut} largest such values. The computation of $\mathcal{M}$ does not significantly increase the complexity of the algorithm, and in numerical experiments the set $\mathcal{M}$ is, indeed, found to contain almost all data points with at most γ components.

Iterating the M-step (Equations 21 to 23) and the E-step (Equations 17) results in a learning algorithm for the OCA generative model. As will be shown numerically in the next section, the algorithm allows for a very accurate estimation of model parameters based on a strongly reduced number of latent states.

4. Experiments

In order to evaluate the OCA learning algorithm, it has been applied to artificial and real-world data. Artificial data allows for an evaluation based on ground-truth information and for a comparison with other approaches. The use of real-world data enables us to test the robustness of the method.

4.1 Initialization and Annealing

For all data points, a vector $\vec{y}_d \in [0,1]^3$ represented the RGB values of a pixel. In all trials of the experiments we initialized the parameters W_{hd} and T_h^c by independently and uniformly drawing from the interval $[0,1]$. The parameters for sparseness and standard deviation were initialized as $\pi_{\text{init}} = \frac{1}{H}$ and $\sigma_{\text{init}} = 5$, respectively.

Parameter optimization in multiple-cause models is usually a non-convex problem. For the OCA model, the strongly non-linear combination rule seems to result in even more pronounced local optima in parameter space than is the case for other models such as sparse coding. To efficiently avoid convergence to local optima, we (A) applied deterministic annealing (Ueda and Nakano, 1998; Sahani, 1999) and (B) added noise to model parameters after each EM iteration. Annealing was implemented by introducing the temperature $T = \frac{1}{\beta}$. The inverse temperature β started near 0 and was gradually increased to 1 as iterations progressed. It modified the EM updates by substituting $\pi \to \pi^\beta$, $(1-\pi) \to (1-\pi)^\beta$, and $\frac{1}{\sigma^2} \to \frac{\beta}{\sigma^2}$ in all E-step equations. We also annealed the occlusion non-linearity by setting $\rho = \frac{1}{1-\beta}$; however, once β became greater than 0.95 we set $\rho = 21$ and did not increase it further. We ran 100 iterations for each trial of learning. The inverse-temperature was set to $\beta = \frac{2}{D}$ for the first 15 iterations, then linearly increased to $\beta = 1$ over the next 15 iterations, and then kept constant until termination of the algorithm.

Additive parameter noise was drawn randomly from a normal distribution with zero mean. Its standard deviation was initially set to 0.3 for the mask parameters and at 0.05 for the prior and noise parameters. The value was kept constant for the first 10 iterations and then linearly decreased to zero over the next 30 iterations. The degeneracy parameter α was initialized at 0.2 and increased to 0.6 from iteration 25 to 35. The amount of data points used for training was linearly reduced from N to N^{cut} between iteration 15 to 30. Approximation parameters were set to $\gamma = 3$ and $H' = 5$ unless stated otherwise.

4.2 Evaluation of Optimization Results

After optimizing the parameters using the derived EM approach, we obtain different sets of parameters in different runs. In the case of available ground-truth parameters, a means to identify the best run is a comparison of the learned parameters with the ground-truth. It could, for instance, be asked if all generating components (all generating objects in our case) have been recovered successfully. However, usually the ground-truth parameters

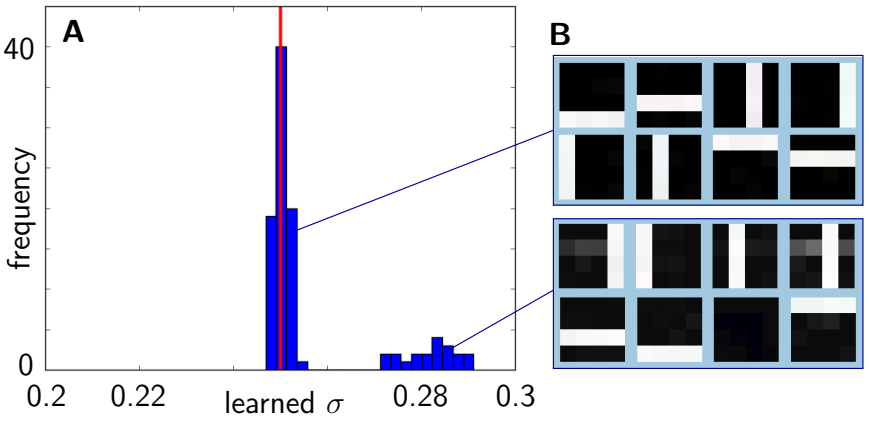


Figure 3: **A** Histogram of values for the parameter σ (Equation 13) for 100 runs of the algorithm on a standard bars test (see bars test section). The ground-truth value for σ in these runs was 0.25, indicated by the red line. As can be observed, most values lie close to this number while some values form a second mode at higher values. By thresholding the σ parameter, we can identify local optima. **B** Examples for the basis functions for the left and for the right cluster.

are not known and so another measure for the quality of a run has to be found. A good indication of the quality of the learned parameters is provided by the learned noise parameter σ (Equation 13). In fact, if we compute the derivative of the update rule for σ w.r.t. the mask and feature parameters and set these equal to zero, we obtain the same update rules for W and T as for the derivative of the free-energy. That is, maximizing the free-energy corresponds to optimizing (minimizing) the noise which the model has to assume to explain the data (see Appendix C). If this noise is small, the data are well explained by the parameters (see Figure 3 for an application to artificial data).

4.3 Colored Bars Test

The component extraction capabilities of the model were tested using the colored bars test. This test is a generalization of the classical bars test (Földiák, 1990) which has become a popular benchmark task for non-linear component extraction. In the standard bars test with $H = 8$ bars the input data are 16-dimensional vectors, representing a 4×4 grid of pixels, i.e., $D = 16$. The single bars appear at the 4 vertical and 4 horizontal positions. For the colored bars test, each of the bars has a different color. Feature values were initialized such that the color values had maximal distance to each other in one brightness plane of HSV color space. Once chosen, they remained fixed for the generation of the data set. For each image a bar appeared independently with a probability $\pi = \frac{2}{H} = 0.25$ which resulted in two bars per image on average (the standard value in the literature). For the bars chosen to be active, a ranking in depth was randomly and uniformly chosen from the permutation group to generate the image. The color of each (noiseless) image pixel was determined by the least distant bar and was black, i.e., zero, if the pixel was

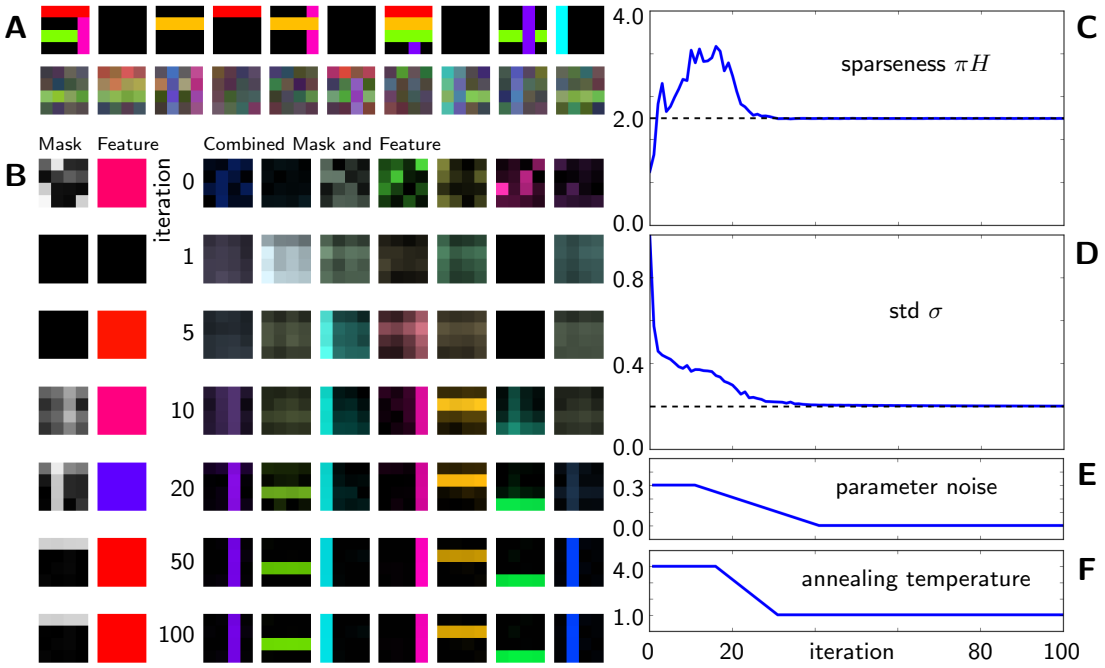


Figure 4: **A** Example of ten noiseless and noisified data points. **B** Development of the (reshaped) generative fields for the given iterations. For the first cause, mask $\vec{W}_h$ and feature $\vec{T}_h^T$ are displayed separately. The other causes are shown as product $\vec{W}_h \cdot \vec{T}_h^T$. **C** - **D** Development of data sparseness and standard deviation over the 100 iterations. **E** Magnitude of the noise which is added to the mask parameters after each iteration. **F** Annealing temperature as chosen for the algorithm.

occupied by no bar. Gaussian data noise of $\sigma = 0.25$ was added to each data point. $N = 1000$ images were generated for learning and Figure 4A shows a random selection of 10 noiseless and 10 noisy examples. The learning algorithm was applied to the colored bars test with $H = 8$ hidden units and $D = 16$ input units. The inferred approximate maximum-likelihood parameters converged to values close to the generating parameters in 97 of 100 trials. The success rate, or *reliability*, was thus 97%. Lücke et al. (2009) achieved a similar reliability of 96% with $N = 500$. Yet, here we learn, with the current version of the algorithm, more parameters, namely the data noise and the sparseness parameter. The values obtained for σ and π all lay in the interval $[0.246, 0.252]$ and $[0.241, 0.268]$, respectively. Figure 4B shows the time-course of a typical trial during learning. As can be observed, the mask values W and the feature values T converged to values close to the generating ones. More specifically, the product of each mask $\vec{W}_h$ and its color value $\vec{T}_h$ represents a true underlying bar in the right color. Reliability is affected by changes in the annealing and parameter noise schedules, i.e., by changes to those algorithmic parameters which control the mechanisms for avoiding local optima. Furthermore, we observed an effect of the approximation parameters H' and γ on the algorithm's capability to avoid local optima. Notably, smaller as well as much larger values for H' and γ lead to lower

reliabilities. In addition to increasing efficiency, Expectation Truncation, therefore, helps in avoiding local optima for this model, presumably because local optima corresponding to broad posterior distributions are avoided. A similar observation was recently reported in an application of ET to a sparse coding variant (Exarchakis et al., 2012).

4.4 Standard Bars Test

Instead of choosing the bar colors randomly as above, they can also be set to specific values. In particular, if all bar colors are white, $\vec{T} = (1, 1, 1)^T$, the classical version of the bars test is recovered. Note that the learning algorithm can be applied to this standard form without modification, even though it is impossible to recover the relative depth of the bars in this case. When the generating parameters were as above (eight bars, probability of a bar to be present $\frac{2}{8}$, $N = 1000$), all bars were successfully extracted in 80 of 100 trials (80% reliability). The estimated values of σ and π lay in the intervals $[0.247, 0.254]$ and $[0.241, 0.263]$, respectively. When learning on noiseless data, we obtained a reliability of 95%. By increasing the approximation parameters to $\gamma = 4$ and $H' = 6$, reliability changed to 91%.

For a standard setting of the parameters ($N = 500$, $H = 10$, $D = 5 \times 5$, probability of $\frac{2}{10}$ for each bar to be present) as was used in numerous previous studies (Saund, 1995; Dayan and Zemel, 1995; Hochreiter and Schmidhuber, 1999; Lücke and Sahani, 2008; Lücke and Eggert, 2010), the OCA algorithm with $\gamma = 3$ and $H' = 5$ achieved 83% for a noisy and 78% for a noiseless bars test. For $N = 1000$ data points reliability increased to 85%. For comparison, earlier generative modeling approaches such as those reported by Saund (1995) or Dayan and Zemel (1995) (both assuming a noisy-or like combination rule) achieved 27% and 69% reliability, respectively. Maximal Causes Analysis (Lücke and Sahani, 2008; Lücke and Eggert, 2010) achieved about 82% (MCA_3) reliability. And the preliminary version of the OCA algorithm (Lücke et al., 2009) achieved 86% for noiseless data. Approaches such as PCA or ICA were reported to fail in this task (Hochreiter and Schmidhuber, 1999). Furthermore, different types of objective functions and neural network approaches (Charles et al., 2002; Lücke and Malsburg, 2004; Spratling, 2006) are also successful at this task, often reporting close to 100% reliability (also see Frolov et al., 2014). The assumptions used (e.g., fixed bar appearance, noise level, parameter constraints, constraint on latent activities) are often implicit but, at the same time, can significantly facilitate learning. NMF algorithms can be successful in extracting all bars (with up to 100% reliability) but require hand-set values for sparsity constraints on hidden variables and/or parameters (see Hoyer, 2004, and for discussions Spratling, 2006, Lücke and Sahani, 2008). In general, the fewer assumptions a model makes, the more difficult it becomes to infer the parameters from a given set of data. For earlier generative models and in particular for the more general model discussed in this paper, larger data sets directly translate into higher reliabilities. A reliability of 78% for the noiseless bars test is, for the OCA algorithm discussed in this work, in this view still relatively high. Reliabilities are comparable to values for MCA and to the preliminary OCA algorithm (Lücke et al., 2009). Note, however, that the latter did use fixed values for data noise σ and bar appearance π which may explain the higher reliability.

As the recovery of optimal model parameters is the goal of the approach, we can further increase the rate of successfully recovered parameters that correspond to a representation of

all bars by considering several runs of the algorithm simultaneously. That is, given a set of N images, we can apply the algorithm M times, and use as the final result the parameters of a run with the smallest σ value. For some data sets, we even obtain two clusters of σ values (see Figure 3) where the cluster with smaller σ's represents the runs which have terminated in an optimum with parameters representing all bars. Note that clearly separable clusters are not observed for all data sets and parameter settings. In general, however, runs with small σ values tend to correspond to parameters reflecting the true underlying generative process more accurately. For the standard settings of the bars test with $D = 5 \times 5$, $N = 500$, $H = 10$, and noiseless data, the algorithm with $M = 20$ extracts all components in 50 of 50 runs. But note that each run now consists of evaluating $M = 20$ subruns. The same applies for values of M down to $M = 10$.

4.5 Inference

To briefly illustrate the algorithm's performance on an inference task, i.e., the extraction of the underlying causes and their depth order, and to show how inference can be applied to data points exceeding γ components, let us consider data points generated according to the colored bars test. Furthermore, consider the model after it has learned the parameters to represent the bars, noise level, and sparsity. Given a data point, the trained model can infer the hidden variables by applying the following procedure: We start by executing an E-step with the same values for H' and γ as used during training ($H' = 5$ and $\gamma = 3$ in this case). We then determine the maximum a-posteriori (MAP) state $\vec{s}^*$ based on the approximate posterior computed in this E-step. If this state has $|\vec{s}^*| = \gamma$ active components, we repeat the E-step with values of H' and γ increased by one each (leaving H' unchanged if $H' = H$). We terminate the procedure if the MAP state has less than γ states or if $\gamma = H' = H$. Exemplarily, Figure 5 shows three data points and the corresponding MAP states obtained with the described procedure. The data point with two components terminated after the first E-step (Figure 5A), the data points with three after γ was increased to four (Figure 5B), and the data point with four components terminated after γ was increased to five (Figure 5C,D show result for initial and final γ). For ambiguous data points, e.g., if the input contained two parallel bars, two states or more states can carry equally large probability mass due to the fact that different depth permutations do not change the image. The MAP estimate can still serve to illustrate the inference result but the approximate distribution over states represents a more accurate description in this case.

4.6 More Realistic Data

To numerically investigate the algorithm for more realistic data, it was applied to data based on pictures of objects from the COIL-100 database (Nene et al., 1996).[1] Images were scaled down to 20×20 pixels and were placed at random planar positions on a black background image of $D = 35 \times 35 = 1225$ pixels. The objects were then colored with their mean color, weighted by pixel intensity (see Figure 6A). In 100 runs, we created $N = 8000$ data points by combining $H_{\text{gen}} = 20$ of these images according to the generative model with

1. We used objects 2, 3, 4, 25, 28, 34, 47, 48, 51, 56, 58, 60, 73, 74, 77, 85, 89, 94, 97, and 112 all at 0 degree rotation.

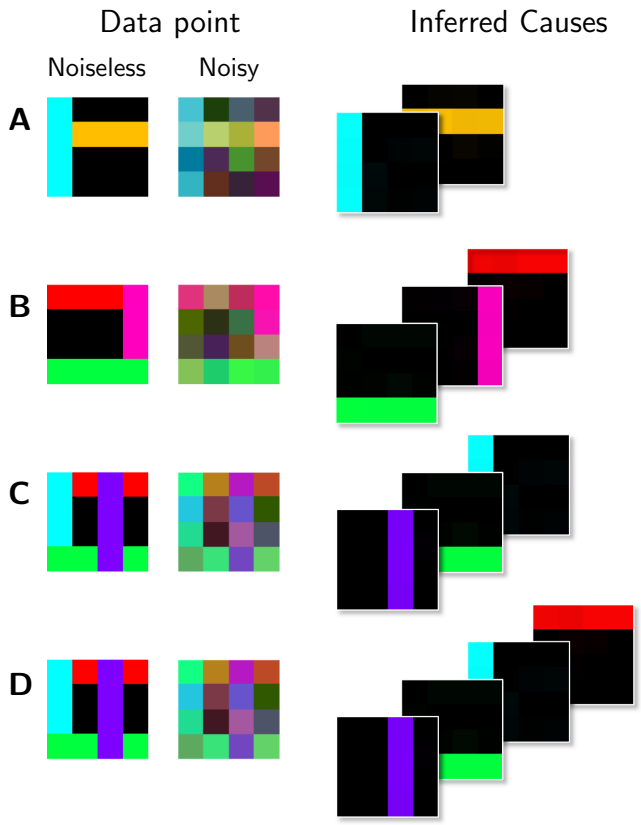


Figure 5: Examples of the inference procedure for the colored bars test. The second column shows the data points used for inference (with their noiseless versions in the first column). On the right, the causes inferred from the noisy data points are shown arranged in their inferred depth order. **A** - **D** The algorithm reliably inferred the causes for the three examples (**C** and **D** show two steps of the inference procedure). Note that we have learned the basis functions from noisy data with the same properties as those shown here.

prior parameter $\pi = \frac{2}{H_{\text{gen}}} = 0.1$, i.e., $\pi H_{\text{gen}} = 2$ and data noise $\sigma = 0.25$ (see Figure 6B). For learning, the algorithm was applied with $H = 30$ mask and feature parameters $\vec{W}_h$ and $\vec{T}_h$, i.e., 50% more than we used for data generation. Figure 6C shows the resulting mask and feature parameters for an example run (where we display each pair of feature and mask combined into one image, compare Equation 3). We obtained all 20 underlying basis functions along with 10 noisy fields in 44% of the trials. For the data noise we obtained $\sigma \in [0.251, 0.254]$ and for the sparseness parameter $\pi \in [0.062, 0.070]$, i.e., $\pi H \in [1.85, 2.11]$ for all runs. The high discrepancy in the sparseness values can be explained by the fact that we have introduced extra basis functions for learning. In the remaining 56 trials, almost all objects were extracted with usually just one and at most three objects not being

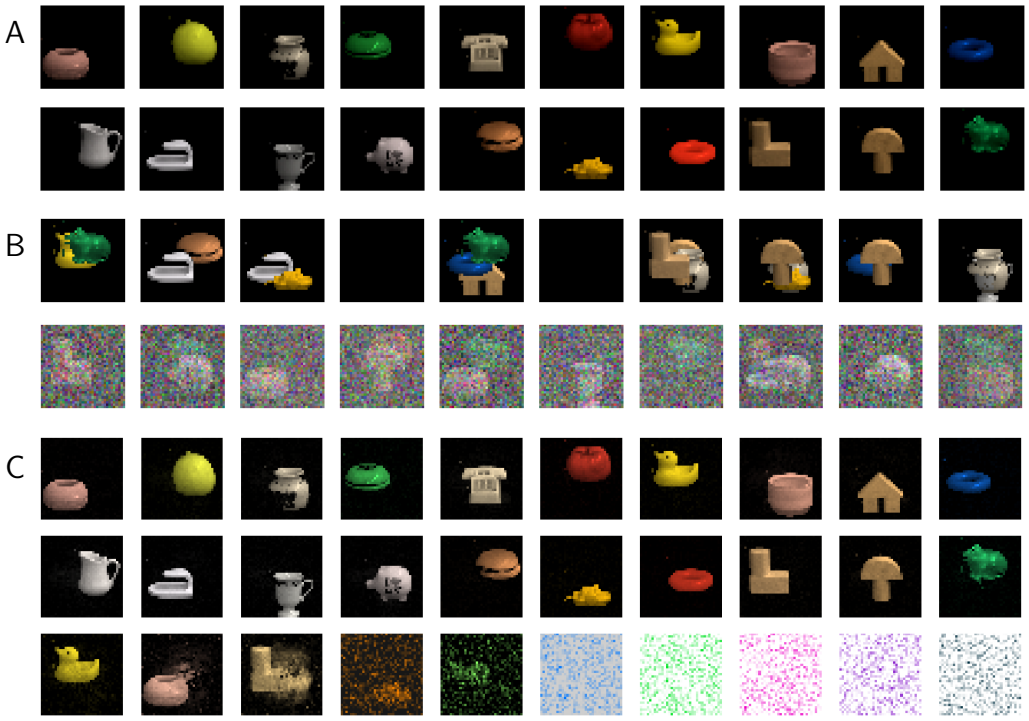


Figure 6: **A** 20 downscaled COIL images. **B** 10 noiseless and 10 noisy data points (obtained from **A** according to Equation 3). **C** 30 extracted basis functions. The first two rows display the clean extracted causes in the same order as in **A**. The third row shows the additionally learned causes which are mostly noisy fields or noisy combinations of more than one cause.

represented. Again, low values of the observation noise were found to indicate the successful extraction of all objects. By performing a series of runs and retaining the parameters of runs with the lowest learned observation noise, the reliability increased to close to 100%.

4.7 Real Data

Finally we tested the algorithm on a real world data set that includes a range of effects that are not present in synthetic data, including real-world occlusion, lighting variability due to shadows and specular variations, as well as some small translation effects. As such the data-set provides an important test of the algorithm's robustness. The data-set comprised 500 pictures of scenes consisting of up to five (toy) cars in front of a gray background. The cars could appear at different positions in depth but always in the same position in planar space (see Figure 7A). Pictures were taken from the side (as for instance a camera in a tunnel might be positioned) such that moving a car in depth had almost no effect on its vertical or horizontal position in the image (see Figure 7B). We then cut out the area of the images that contained the cars and downscaled the cut-out images to 40×165 pixels. Subsequently,

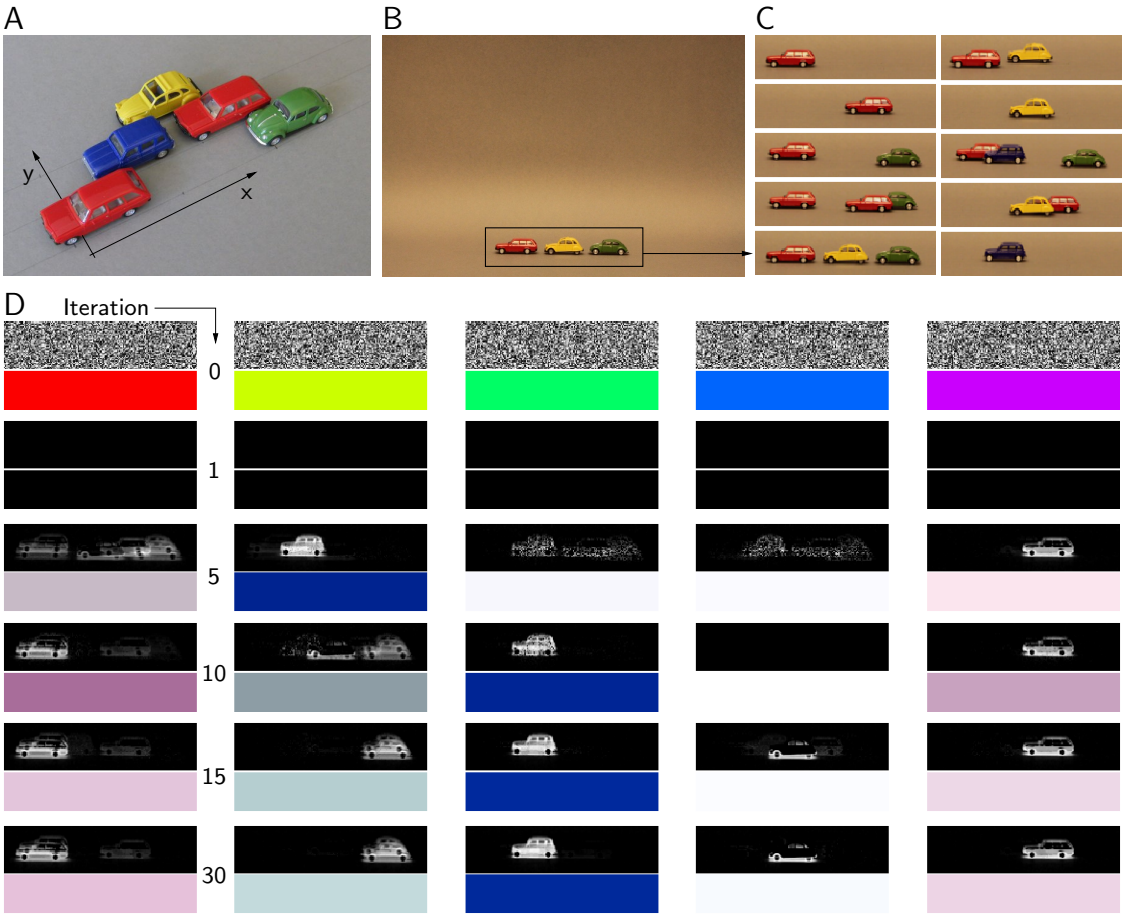


Figure 7: Numerical experiment with photographic images of cars. **A** Five cars could appear in three lanes which account for the arbitrary position in depth (y). Position in x-direction was fixed. **B** One of the 500 pictures taken of one state with three active causes. **C** 10 data points after cutting and pre-processing. **D** Generative fields (mask on top, feature below) at different iterations.

the images were normalized in luminance and the 5% lightest pixels were clamped to the same maximal value for each color channel to remove reflection effects. Figure 7C shows some example images. For learning, we then subtracted an image of an empty scene (i.e., the background) pixelwise from each input image such that pixels that did not belong to a cause became almost zero. Note that these data points can now have negative values while RGB values are usually defined to be positive. To interpret the data point as an image, one can map the values in the three color channels back to $[0, 1]$. For a homogeneous background, these images look almost the same as the original input images. We initialized the masks as random noise and the features as RGB color vectors all equally far apart from each other in color space (see Figure 7D). The inverse annealing temperature was set to $\beta = \frac{1}{15}$ and increased to 1 from iteration 5 to 25. Parameter noise was decreased between

iterations 15 and 26. Over 30 iterations we extracted the masks of all five cars along with data noise ($\sigma = 0.05$) and sparseness ($\pi = 0.17$ with $\pi_{\text{ground-truth}} = \frac{1}{H} = 0.2$). The features which had positive and negative values were mapped to $[0, 1]$ to be interpreted as color. As can be seen from the generative fields in iteration 30, not all masks were extracted cleanly. This can be explained by the fact that a different position in depth still causes a slight shift in planar space such that in some images one cause is higher or lower than in others. This smearing effect leads to a change in color because a pixel then sometimes belongs to the car and sometimes to the background which results in a color shift towards the background color (black). Another reason is that one cause does not only consist of one color but rather of a combination of the car color and background, shadow, window, and wheel color. For the yellow car which is relatively similar to the background the mask almost only represents the shadow of the car, which is the most salient part relative to the background. For the other cars, the masks correspond to representations of whole cars.

5. Discussion

We have studied learning in a multiple-cause generative model which assumes an explicit model of occlusion for component superposition. According to the OCA model assumptions, an object can appear in different depth positions for different images. This aspect reflects properties of real images and is complementary, e.g., to assumptions made by sprite models (Jojic and Frey, 2001; Williams and Titsias, 2004). A combination of sprite models and the OCA model is, therefore, a promising line of inquiry, e.g., towards systems that can learn from video data in which objects change their positions in depth. Other lines of research have also identified occlusions as an important property that has to be modeled for applications to visual data. In the context of neural network modeling, Tajima and Watanabe (2011) have recently addressed the problem (albeit with a very small number of components and in a partly supervised setting), while restricted Boltzmann machines have been augmented by LeRoux et al. (2011) to incorporate occlusions.

The directed graphical model studied here has a close connection to multiple-cause approaches such as sparse coding, NMF or ICA. All of these standard approaches use linear superpositions of elementary components to model component superposition. ICA and SC have prominently been applied to explain neural response properties, and NMF is a popular approach to learn components, e.g., for visual object recognition (e.g., Lee and Seung, 1999; Wersing and Körner, 2003; Hoyer, 2004). In the class of multiple-cause approaches our model is the first to generalize the combination rule to one that models occlusion explicitly. While non-linear combination rules have been studied before by Saund (1995); Dayan and Zemel (1995); Šingliar and Hauskrecht (2006); Frolov et al. (2014) (noisy-or), Valpola and Karhunen (2002); Honkela and Valpola (2005) (post-linear sigmoidal function) or Lücke and Sahani (2008); Puertas et al. (2010); Bornschein et al. (2013) (point-wise maximum), we go a step further and model occlusion explicitly by making the component combination dependent on an additional hidden variable for depth-ordering. As a consequence, the model requires two sets of parameters: masks and features. Masks are required because the planar space that a component occupies has to be defined. Parameterized masks are, therefore, a feature of many approaches with explicit occlusion modeling (compare Jojic and Frey, 2001; Williams and Titsias, 2004; LeRoux et al., 2011; Tajima and Watanabe, 2011). For

our model, the combination of masks and vectorial feature parameters, furthermore, allows for applications to more general sets of data than the scalar values used for SC, NMF or than in applications of sprite models (compare Jojic and Frey, 2001; Williams and Titsias, 2004). In numerical experiments we have used color images for instance. However, we can also apply our algorithm to gray-level data such as used for other algorithms. This allows for a direct quantitative comparison of the novel algorithm with state-of-the-art component extraction approaches. The reported results for the standard bars test show the competitiveness of our approach despite its larger set of parameters (compare, e.g., Spratling, 2006; Lücke and Sahani, 2008). For applications to visual data, color is the most straightforward feature to model. Possible alternatives are Gabor feature vectors which model object edges and textures, or further developments such as SIFT features (Lowe, 2004). Depending on the application, the generative model itself could also be generalized. It is, for instance, straightforward to introduce several feature vectors per cause. Although one feature (e.g., one color) per cause can represent a suitable model for many applications, it might for other applications also make sense to use multiple feature vectors per cause. In the extreme case, as many feature vectors as pixels could be used, i.e., $\vec{T}_h \rightarrow \vec{T}_{hd}$. The derivation of update rules for such features would proceed along the same lines as the derivations for single features $\vec{T}_h$. Furthermore, individual prior parameters for the frequency of object appearances could be introduced. Additional parameters could be used to model different prior probabilities for different arrangements in depth. Finally, the most interesting but also most challenging generalization direction would be the inclusion of explicit invariance models. In its current form the model uses, in common with state-of-the-art component extraction algorithms, the assumption that the component locations are fixed. Especially for images of objects, changes in planar component positions have to be addressed in general. Possible approaches that have been discussed in the literature have, for instance, been investigated by Jojic and Frey (2001) and Williams and Titsias (2004) in the context of occlusion modeling, by Eggert et al. (2004) and Wersing and Körner (2003) in the context of NMF, or by Berkes et al. (2009), Gregor and LeCun (2011) and others in the context of sparse coding. The crucial challenge of a generalization of the occlusion model studied in this work is the further increase in the dimensionality of the hidden space. By generalizing the methods as used here, such challenges could be overcome, however. On the other hand, methods such as sparse coding or NMF have proven to be useful building blocks in vision systems although they do not address translation invariance in an explicit way. As a generalization of sparse coding, the model studied here can provide a more accurate model in situations where the modeling of occlusions is important. Like for sparse coding, no prior information about the two dimensional nature of images is used in the model, i.e., learning would not suffer from a (fixed) permutation of all pixels applied to all data points. The tasks faced by the model may, therefore, appear easier for the human observer because humans make (e.g., for the COIL data) use of additional object knowledge such as of the gestalt law of proximity. This also illustrates that extensions of the model to incorporate prior knowledge about objects would further improve the approach.

To investigate robustness, we have applied the developed algorithm to real images, and observed that it is robust enough to work on non-artificial data. We do not regard this work as providing a directly applicable algorithm, however. The main goal of this study was rather to show that the challenges of a multiple-cause model with explicit occlusions can

be overcome. Replacing the standard linear superposition of sparse coding by an occlusion superposition resulted in a number of challenges that all had to be addressed:

1) The occlusion model required parameterized masks.
2) The learning equations are not closed-form.
3) Occlusion leads to a much larger combinatorial explosion of possible configurations.
4) Posterior probabilities are not unimodal.
5) Local optima in parameter space are more pronounced.

By generalizing the treatment of non-linear superpositions developed for maximal causes analysis (see Lücke and Sahani, 2008), parameter update equations were derived for all parameters of the occlusion model: for masks and features as well as data noise and sparsity (addressing points 1 and 2). The combinatoric challenge of the model's large latent space (point 3) was addressed using Expectation Truncation (ET; Lücke and Eggert, 2010) which, furthermore, does not make any assumptions about unimodal posteriors (point 4). Combined with deterministic annealing, the algorithm efficiently avoided local optima (point 5). Compared to the earlier version of the OCA learning algorithm (Lücke et al., 2009), Expectation Truncation provides a further increase of efficiency by selecting relevant latent causes using selection functions. In this way, the complexity was reduced from scaling polynomially with H (usually with H to the power of 3 or 4) to a linear scaling with H. The combinatorics of states instead only affects the much smaller space of the H' candidates selected for each data point (compare Figure 8). Given H' and γ the combinatorics is known exactly (Equation 20) and this is the main factor that determines the scalability of the algorithm. How large the values of H' and γ have to be depends on the data. A large number of objects per image will require higher values and consequently a large number of states. If the average number of objects per data point remains constant, H' and γ can be kept constant for increasing H, and the number of states that have to be evaluated will scale only linearly with H. Secondary effects may lead to the algorithm scaling super-linearly, however. Larger values of H mean a higher number of parameters which may in turn require larger data sets to prevent overfitting. Such effects can be considered as much less severe than combinatorial effects that increase the state space. Because of the generally favorable scaling with H, we could handle numerical experiments with larger numbers of latents than previously considered. For the COIL data set, the algorithm was run with $H = 30$ hidden variables and $D = 35 \times 35$ observed variables. For the colored bars test the algorithm was run with up to $H = 80$ hidden and $D = 40 \times 40$ observed variables (but extraction of all bars becomes increasingly challenging). In practice and depending on the data, learning times may differ. For some data longer learning may be required for the parameters to converge or in order to efficiently avoid local optima with slower annealing. A precise theoretical quantification of these data-dependent effects is, like for most learning algorithms, difficult. In all our empirical evaluations we found that the mechanisms in place to avoid local optima are important. We applied deterministic annealing and parameter noise for the algorithm to converge to approximately optimal global solutions, i.e., to solutions corresponding to parameters that all represented true data components (in cases when these components were known). Without annealing or parameter noise the algorithm converged to approximate global optima only in very few cases, and local optima were usually reached after a small number of steps. Both annealing and parameter noise

had an influence on the typical convergence points. With only parameter noise (i.e., without annealing), the algorithm usually converged within few EM iterations to local optima with a relatively large number of fields representing more than one component. With only annealing (i.e., without parameter noise), the algorithm often converged to local optima in which most components were represented correctly but where few generative fields represented two components. The combination of annealing and parameter noise resulted in a frequent representation of all causes (see experiments). As stated earlier, we also observed a positive effect of the approximation scheme in avoiding local optima presumably because shallow local optima corresponding to solutions with dense states are not considered by the truncated approximation.

A further improvement that followed from the application of Expectation Truncation is the availability of learning rules for data noise and sparsity. While data noise could have been inferred within the preliminary study of the occlusion model (Lücke et al., 2009), inference of sparsity requires a correction term that compensates for considering a reduced space of latent configurations, and Expectation Truncation provides a systematic way to derive such a correction (compare Appendix B.2). Data noise and sparsity parameters are, notably, not a consequence of modeling occlusion. They are potential parameters also of standard sparse coding approaches or NMF objective functions. Nonetheless, most sparse coding approaches only optimize the generative fields because of limitations induced by the usual maximum a-posteriori based learning (but see Berkes et al., 2008; Henniges et al., 2010, for exceptions). Likewise, NMF approaches focus on learning of generative fields / basis functions. Sparsity is at most indirectly inferred by standard SC or NMF through cross-validation.

To summarize, our study shows that the challenges of occlusion modeling with explicit depth orders can be overcome, and that all model parameters can be efficiently inferred. The approach complements established approaches of occlusion modeling in the literature by generalizing standard approaches such as sparse coding or NMF to incorporate one of the most salient properties of visual data.

Acknowledgments

MH, JE and JL acknowledge funding by the German Research Foundation (DFG) in the project LU 1196/4-2, by the Honda Research Institute Europe GmbH (HRI Europe), by the DFG Cluster of Excellence EXC 1077/1 (Hearing4all), and by the German Federal Ministry of Education and Research (BMBF) in the project 01GQ0840 (BFNT Frankfurt). RET acknowledges funding by EPSRC Postdoctoral Fellowship Grant EP/G050821/1, and MS acknowledges funding by The Gatsby Charitable Foundation.

Appendix A. Illustration of Hidden State Combinatorics

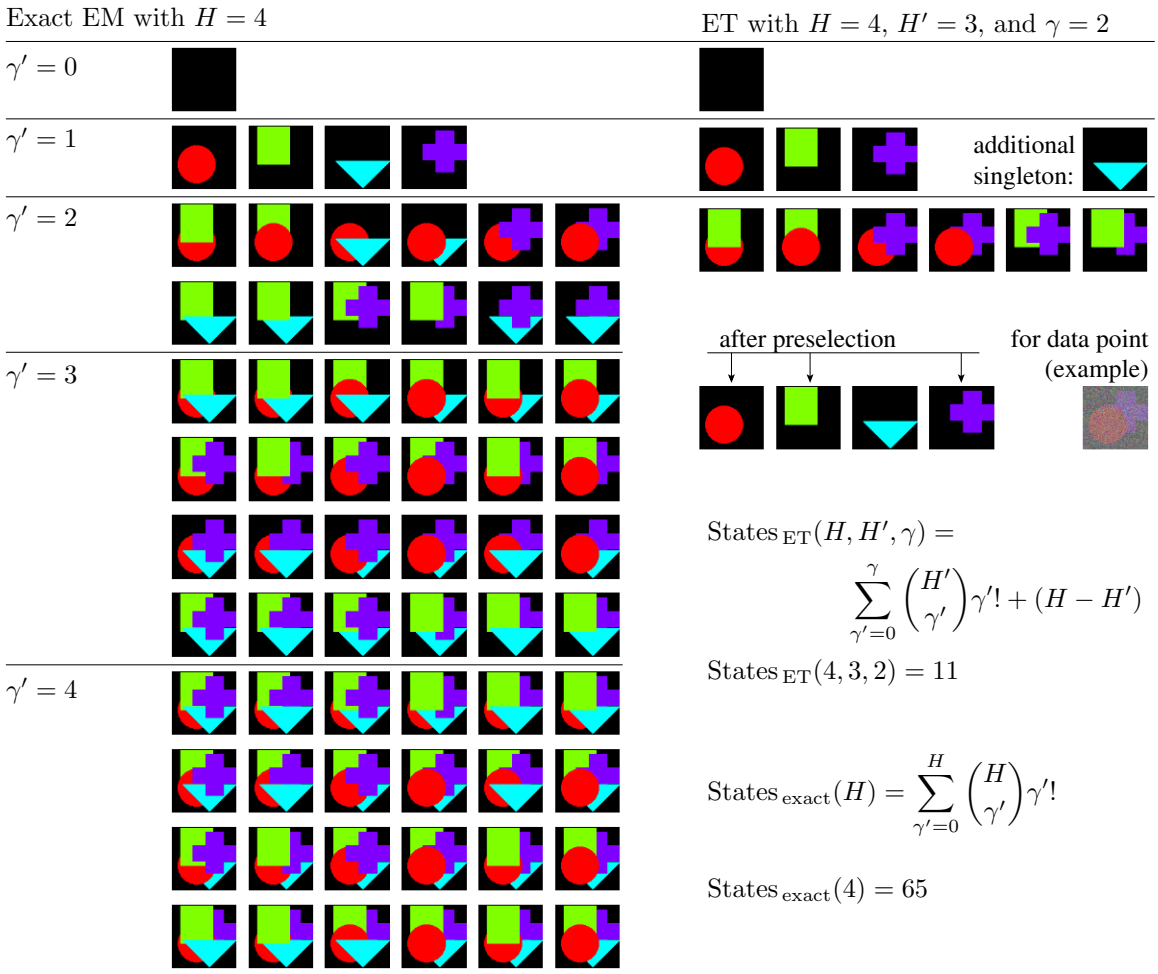


Figure 8: This figure shows all possible combinations for hidden states with given basis functions for exact EM as well as for a reduced number of combinations. The left-hand-side corresponds to exact EM and thus displays all possible combinations of $H = 4$ different causes separated for each value of $\gamma' = |\vec{s}|$. On the right-hand-side, we see all combinations of the three most relevant objects in which at most two objects appear simultaneously. The reduced number of combinations corresponds to the states evaluated by the used approximation (Equation 16) with parameters $H' = 3$ and $\gamma = 2$ in Equation 18. Given a noisy data point, the first, second, and fourth component are preselected for this example. Note that we additionally consider all singleton states. The formulas for the number of considered states are given on the bottom right.

Appendix B. Derivation of Update Rules

Our goal is to optimize the free-energy, i.e.,

$$\mathcal{F}(\Theta, q) = \sum_{n=1}^{N} \Big[\sum_{S} \; q_n(S\,; \Theta') \;\; \Big[\log\big(p(Y^{(n)} \,|\, S, \Theta)\big) + \log\big(p(S \,|\, \Theta)\big) \Big] \Big] \; + \; \mathbf{H}[q]\,,$$

where

$$p(Y^{(n)} \,|\, S, \Theta) = \prod_{d=1}^{D} p(\vec{y}_d^{\,(n)} | \vec{\mathcal{T}}_d(S, \Theta)) \text{ with } p(\vec{y}\,|\vec{t}\,) = \mathcal{N}(\vec{y}; \vec{t}, \sigma^2 \mathbb{1})\,.$$

More explicitly,

$$\begin{aligned} p(Y^{(n)} \,|\, S, \Theta) &= \prod_{d=1}^{D} \prod_{c=1}^{C} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} \left(y_{cd}^{(n)} - \mathcal{T}_{cd}(S, \Theta) \right)^2 \right) \\ &= \left(2\pi\sigma^2\right)^{-\frac{CD}{2}} \prod_{d=1}^{D} \prod_{c=1}^{C} \exp\left(-\frac{1}{2\sigma^2} \left(y_{cd}^{(n)} - \mathcal{T}_{cd}(S, \Theta) \right)^2 \right) \end{aligned}$$

and for the logarithm

$$\log\Big(p(Y^{(n)} \,|\, S, \Theta)\Big) = -\frac{CD}{2} \log\left(2\pi\sigma^2\right) - \sum_{d=1}^{D} \sum_{c=1}^{C} \frac{1}{2\sigma^2} \left(y_{cd}^{(n)} - \mathcal{T}_{cd}(S, \Theta) \right)^2 .$$

The prior term, we defined to be

$$p(S \,|\, \Theta) = \pi^{|\vec{s}\,|} (1-\pi)^{(H - |\vec{s}\,|)} \frac{1}{|\vec{s}\,|!},$$

such that

$$\log\left(p(S \,|\, \Theta)\right) = |\vec{s}\,| \log(\pi) + (H - |\vec{s}\,|) \log(1-\pi) - \sum_{\gamma=1}^{|\vec{s}\,|} \log(\gamma)\,.$$

The free-energy thus takes the form

$$\begin{aligned} \mathcal{F}(\Theta, q) = \sum_{n=1}^{N} \Big[\sum_{S} \; q_n(S\,; \Theta') \Big[&- \frac{CD}{2} \log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2} \sum_{d=1}^{D} \sum_{c=1}^{C} \left(y_{cd}^{(n)} - \mathcal{T}_{cd}(S, \Theta) \right)^2 \\ &+ |\vec{s}\,| \log(\pi) + (H - |\vec{s}\,|) \log(1-\pi) - \sum_{\gamma=1}^{|\vec{s}\,|} \log(\gamma) \Big] \Big] \\ &+ \mathbf{H}[q]\,. \end{aligned}$$

B.1 Optimization of the Data Noise

Let us start be deriving the M-step equation for σ as follows:

$$\begin{aligned}
&\frac{\partial}{\partial\sigma}\mathcal{F}(\Theta,q)\\
&=\sum_{n=1}^{N}\Bigg[\sum_{S}\ q_n(S\,;\Theta')\Bigg[-\frac{CD}{2}\frac{\partial}{\partial\sigma}\log\left(2\pi\sigma^2\right)-\left(\frac{\partial}{\partial\sigma}\frac{1}{2\sigma^2}\right)\sum_{d=1}^{D}\sum_{c=1}^{C}\left(y_{cd}^{(n)}-\mathcal{T}_{cd}(S,\Theta)\right)^2\Bigg]\Bigg]\\
&=\sum_{n=1}^{N}\Bigg[\sum_{S}\ q_n(S\,;\Theta')\Bigg[-\frac{CD}{2}\frac{4\pi\sigma}{2\pi\sigma^2}-\left(-2\,(2\sigma)^{-3}\,2\right)\sum_{d=1}^{D}\sum_{c=1}^{C}\left(y_{cd}^{(n)}-\mathcal{T}_{cd}(S,\Theta)\right)^2\Bigg]\Bigg]\\
&=\sum_{n=1}^{N}\Bigg[\sum_{S}\ q_n(S\,;\Theta')\Bigg[-\frac{CD}{\sigma}+\frac{1}{2\sigma^3}\sum_{d=1}^{D}\sum_{c=1}^{C}\left(y_{cd}^{(n)}-\mathcal{T}_{cd}(S,\Theta)\right)^2\Bigg]\Bigg]\stackrel{!}{=}0
\end{aligned}$$

$$\begin{aligned}
\Rightarrow&\frac{1}{2\sigma^3}\sum_{n=1}^{N}\Bigg[\sum_{S}\ q_n(S\,;\Theta')\sum_{d=1}^{D}\sum_{c=1}^{C}\left(y_{cd}^{(n)}-\mathcal{T}_{cd}(S,\Theta)\right)^2\Bigg]=\frac{NCD}{\sigma}\\
\Rightarrow&\sigma^2=\frac{1}{NCD}\sum_{n=1}^{N}\Bigg[\sum_{S}\ q_n(S\,;\Theta')\sum_{d=1}^{D}\sum_{c=1}^{C}\left(y_{cd}^{(n)}-\mathcal{T}_{cd}(S,\Theta)\right)^2\Bigg]
\end{aligned}$$

For ET, all we need to change is the amount of data points we consider. We thus obtain for the update of the data noise that

$$\sigma^{\text{new}}=\sqrt{\frac{1}{|\mathcal{M}|\,CD}\sum_{n\in\mathcal{M}}\left\langle\sum_{d=1}^{D}\sum_{c=1}^{C}\left(y_{dc}^{(n)}-\mathcal{T}_{dc}(S,\Theta)\right)^2\right\rangle_{q_n}}\,.$$

B.2 Optimization of the Prior Parameter

Now we will derive the M-Step equation for the update of the parameter π as follows:

$$\begin{aligned}
\frac{\partial}{\partial\pi}\mathcal{F}(\Theta,q)&=\sum_{n=1}^{N}\Bigg[\sum_{S}\ q_n(S\,;\Theta')\left[\frac{\partial}{\partial\pi}|\vec{s}|\log(\pi)+\frac{\partial}{\partial\pi}(H-|\vec{s}|)\log(1-\pi)\right]\Bigg]\\
&=\sum_{n=1}^{N}\Bigg[\sum_{S}\ q_n(S\,;\Theta')\left[\frac{|\vec{s}|}{\pi}-\frac{H-|\vec{s}|}{1-\pi}\right]\Bigg]\\
&=\sum_{n=1}^{N}\sum_{S}\ q_n(S\,;\Theta')\frac{|\vec{s}|-H\pi}{\pi(1-\pi)}\stackrel{!}{=}0
\end{aligned}$$

$$\begin{aligned}
\Rightarrow\quad&\sum_{n=1}^{N}\sum_{S}\ q_n(S\,;\Theta')|\vec{s}|=H\pi N\\
\Rightarrow\quad&\pi=\frac{1}{NH}\sum_{n=1}^{N}\sum_{S}\ q_n(S\,;\Theta')|\vec{s}|
\end{aligned}$$

With ET, we have to introduce a normalization factor, A, which changes $p(S|\pi)$ to

$$p_{\mathrm{ET}}(S|\pi) = \begin{cases} \frac{1}{A}\, p(S|\pi), & |\vec{s}| \leq \gamma \\ 0, & |\vec{s}| > \gamma \end{cases} .$$

With $\sum_S p_{\mathrm{ET}}(S|\pi) = \sum_{S \in \mathcal{K}_n} \frac{1}{A}\, p(S|\pi) \approx \sum_{S;|\vec{s}|<\gamma} \frac{1}{A}\, p(S|\pi) \stackrel{!}{=} 1$, we find that

$$\begin{aligned} A = \sum_{S,\,|\vec{s}|\leq\gamma} p(S|\,\pi) &= \sum_{S,\,|\vec{s}|\leq\gamma} \pi^{|\vec{s}|}(1-\pi)^{(H-|\vec{s}|)} \frac{1}{|\vec{s}|!} \\ &= \big[1 \times \pi^0(1-\pi)^H + H \times \pi^1(1-\pi)^{(H-1)} \\ &\quad + H(H-1) \times \pi^2(1-\pi)^{(H-2)} \frac{1}{2!} + \ldots\big] \\ &= \sum_{\gamma'=0}^{\gamma} \binom{H}{\gamma'} \pi^{\gamma'} (1-\pi)^{H-\gamma'} . \end{aligned}$$

As we are going to need it below, we define

$$B(\pi) := \sum_{\gamma'=0}^{\gamma} \gamma' \binom{H}{\gamma'} \pi^{\gamma'} (1-\pi)^{H-\gamma'}$$

and also calculate the derivative of A w.r.t. π as follows:

$$\begin{aligned} \frac{\partial}{\partial\pi} A(\pi) &= \sum_{\gamma'=0}^{\gamma} \binom{H}{\gamma'} \left[\frac{\gamma'}{\pi}\, \pi^{\gamma'}(1-\pi)^{(H-\gamma')} - \pi^{\gamma'}(1-\pi)^{(H-\gamma')}\, \frac{H-\gamma'}{1-\pi} \right] \\ &= \sum_{\gamma'=0}^{\gamma} \binom{H}{\gamma'} \left[\pi^{\gamma'}(1-\pi)^{(H-\gamma')} \left(\frac{\gamma'}{\pi} - \frac{H}{1-\pi} + \frac{\gamma'}{1-\pi} \right) \right] \\ &= \frac{1}{\pi} \sum_{\gamma'=0}^{\gamma} \gamma' \binom{H}{\gamma'} \pi^{\gamma'}(1-\pi)^{(H-\gamma')} \\ &\quad - \frac{H}{1-\pi} \sum_{\gamma'=0}^{\gamma} \binom{H}{\gamma'} \pi^{\gamma'}(1-\pi)^{(H-\gamma')} \\ &\quad + \frac{1}{1-\pi} \sum_{\gamma'=0}^{\gamma} \gamma' \binom{H}{\gamma'} \pi^{\gamma'}(1-\pi)^{(H-\gamma')} \\ &= \frac{1}{\pi} B(\pi) - \frac{H}{1-\pi} A(\pi) + \frac{1}{1-\pi} B(\pi) \\ &= \frac{B(\pi)}{\pi(1-\pi)} - \frac{HA(\pi)}{1-\pi} \end{aligned}$$

As we now take the derivative of the ET prior, we find that

$$\begin{aligned}
\frac{\partial}{\partial\pi}\log\left(p_{\mathrm{ET}}(S|\pi)\right) &= \frac{\partial}{\partial\pi}\log\left[\frac{1}{A(\pi)}\,\pi^{|\vec{s}|}(1-\pi)^{(H-|\vec{s}|)}\frac{1}{|\vec{s}|!}\right] \\
&= \frac{|\vec{s}|}{\pi} - \frac{H-|\vec{s}|}{1-\pi} - \frac{\partial}{\partial\pi}\log A(\pi) \\
&= \frac{|\vec{s}|}{\pi} - \frac{H-|\vec{s}|}{1-\pi} - \frac{1}{A(\pi)}\frac{\partial}{\partial\pi}A(\pi) \\
&= \frac{|\vec{s}|}{\pi} - \frac{H-|\vec{s}|}{1-\pi} - \frac{B(\pi)}{A(\pi)\pi(1-\pi)} + \frac{H}{1-\pi} \\
&= \frac{|\vec{s}|}{\pi} - \frac{H}{1-\pi} + \frac{|\vec{s}|}{1-\pi} - \frac{B(\pi)}{A(\pi)\pi(1-\pi)} + \frac{H}{1-\pi} \\
&= \frac{|\vec{s}|}{\pi(1-\pi)} - \frac{B(\pi)}{A(\pi)\pi(1-\pi)}\,.
\end{aligned}$$

We now have to set the free-energy with this expression equal to zero:

$$\begin{aligned}
\sum_{n\in\mathcal{M}}\sum_{S\in\mathcal{K}_n} q^{(n)}(S,\Theta')\left[\frac{|\vec{s}|}{\pi(1-\pi)} - \frac{B(\pi)}{A(\pi)\pi(1-\pi)}\right] &\stackrel{!}{=} 0 \\
\Rightarrow \sum_{n\in\mathcal{M}}\sum_{S\in\mathcal{K}_n} q^{(n)}(S,\Theta')|\vec{s}| &= \frac{B(\pi)|\mathcal{M}|}{A(\pi)} \\
\Leftrightarrow \frac{A(\pi)}{B(\pi)|\mathcal{M}|}\sum_{n\in\mathcal{M}}\sum_{S\in\mathcal{K}_n} q^{(n)}(S,\Theta')|\vec{s}| &= 1
\end{aligned}$$

In a fixed-point sense, this expression can be multiplied with π on both sides, one representing the updated π_{new} and one the old π from the iteration before:

$$\pi_{\mathrm{new}} = \frac{A(\pi)\,\pi}{B(\pi)}\,\frac{1}{|\mathcal{M}|}\sum_{n\in\mathcal{M}}\langle|\vec{s}|\rangle_{q_n}$$

B.3 Optimization of the Basis Functions

For the M-step equations of the mask and feature parameters, we observe that

$$\begin{aligned}
\frac{\partial}{\partial W_{id}}\mathcal{F}(\Theta,q) =& \frac{1}{2\sigma^2}\sum_{n=1}^{N}\sum_{S}\; q_n(S\,;\Theta')\frac{\partial}{\partial W_{id}}\sum_{d=1}^{D}\sum_{c=1}^{C}\left(y_{cd}^{(n)} - \mathcal{T}_{cd}(S,\Theta)\right)^2 \\
=& -\frac{1}{\sigma^2}\sum_{n=1}^{N}\sum_{S}\; q_n(S\,;\Theta')\sum_{c=1}^{C}\left(y_{cd}^{(n)} - \mathcal{T}_{cd}(S,\Theta)\right)\frac{\partial}{\partial W_{id}}\mathcal{T}_{cd}(S,\Theta)
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial}{\partial T_{ic}}\mathcal{F}(\Theta,q) =& \frac{1}{2\sigma^2}\sum_{n=1}^{N}\sum_{S}\; q_n(S\,;\Theta')\frac{\partial}{\partial T_{ic}}\sum_{d=1}^{D}\sum_{c=1}^{C}\left(y_{cd}^{(n)} - \mathcal{T}_{cd}(S,\Theta)\right)^2 \\
=& -\frac{1}{\sigma^2}\sum_{n=1}^{N}\sum_{S}\; q_n(S\,;\Theta')\sum_{d=1}^{D}\left(y_{cd}^{(n)} - \mathcal{T}_{cd}(S,\Theta)\right)\frac{\partial}{\partial T_{ic}}\mathcal{T}_{cd}(S,\Theta)\,.
\end{aligned}$$

We thus have to calculate the derivative of the combination rule. Since the original non-linear combination rule is not differentiable, we calculate the derivative of the approximated function and find that

$$\begin{aligned}
\frac{\partial}{\partial W_{id}}\mathcal{T}^{\rho}_{cd}(S,\Theta) =& \frac{\partial}{\partial W_{id}}\frac{\sum_{h=1}^{H}(\tau(S,h)\,W_{hd})^{\rho}\,W_{hd}\,T_{hc}}{\sum_{h=1}^{H}(\tau(S,h)\,W_{hd})^{\rho}}\left[=\frac{u'v}{v^2}-\frac{uv'}{v^2}\right]\\
=& \frac{(\tau(S,i)\,W_{id})^{\rho}T_{ic}(\rho+1)\times\sum_{h=1}^{H}(\tau(S,h)\,W_{hd})^{\rho}}{\left(\sum_{h=1}^{H}(\tau(S,h)\,W_{hd})^{\rho}\right)^2}\\
&-\frac{\sum_{h=1}^{H}(\tau(S,h)\,W_{hd})^{\rho}\,W_{hd}\,T_{hc}\times\rho\,(\tau(S,i)\,W_{id})^{\rho-1}\,\tau(S,i)}{\left(\sum_{h=1}^{H}(\tau(S,h)\,W_{hd})^{\rho}\right)^2}\\
=& \frac{(\tau(S,i)\,W_{id})^{\rho}T_{ic}(\rho+1)}{\sum_{h=1}^{H}(\tau(S,h)\,W_{hd})^{\rho}}-\mathcal{T}^{\rho}_{cd}(S,\Theta)\times\rho\,(\tau(S,i)\,W_{id})^{\rho-1}\,\tau(S,i)\\
=& \ldots .
\end{aligned}$$

As this derivation does not result in an analytically tractable solution, we introduce another approximation: The prefactor $(\tau(S,h)\,W_{hd})^{\rho}$ together with the normalizing denominator $\sum_{h=1}^{H}(\tau(S,h)\,W_{hd})^{\rho}$ simulates a differentiable step-function, i.e., its derivative will be zero almost everywhere, except for close to the point where the actual 'step' is where it is infinitely large. We will thus treat this entity as a constant prefactor. We obtain that

$$\begin{aligned}
\frac{\partial}{\partial W_{id}}\mathcal{T}^{\rho}_{cd}(S,\Theta) =& \frac{\sum_{h=1}^{H}(\tau(S,h)\,W_{hd})^{\rho}\,\frac{\partial}{\partial W_{id}}W_{hd}\,T_{hc}}{\sum_{h=1}^{H}(\tau(S,h)\,W_{hd})^{\rho}}\\
=& \frac{(\tau(S,i)\,W_{id})^{\rho}\,T_{ic}}{\sum_{h=1}^{H}(\tau(S,h)\,W_{hd})^{\rho}}\\
=& \mathcal{A}^{\rho}_{id}(S,W)\,T_{ic}\,,
\end{aligned}$$

where we defined for convenience that

$$\mathcal{A}^{\rho}_{id}(S,W) := \frac{(\tau(S,i)\,W_{id})^{\rho}}{\sum_{h=1}^{H}(\tau(S,h)\,W_{hd})^{\rho}} \text{ with } \mathcal{A}_{id}(S,W) := \lim_{\rho\to\infty}\mathcal{A}^{\rho}_{id}(S,W)\,.$$

For the feature parameter, we find that

$$\frac{\partial}{\partial T_{ic}}\mathcal{T}^{\rho}_{cd}(S,\Theta) = \mathcal{A}^{\rho}_{id}(S,W)\,W_{id}\,.$$

For large ρ, we find for a well-behaved function $f(t)$ that

$$\mathcal{A}^{\rho}_{id}(S,W)f(\mathcal{T}^{\rho}_{cd}(S,\Theta)) \approx \mathcal{A}^{\rho}_{id}(S,W)f(W_{id}T_{ic})\,.$$

When we insert this, together with the derivations above, into the free-energy, we observe that

$$\frac{\partial}{\partial W_{id}}\mathcal{F}(\Theta,q) = \frac{1}{\sigma^2}\sum_{n=1}^{N}\left[\sum_{S}\;q_n(S\,;\Theta')\sum_{c=1}^{C}\left(y^{(n)}_{cd}-W_{id}T_{ic}\right)\mathcal{A}^{\rho}_{id}(S,W)T_{ic}\right]\stackrel{!}{=}0$$

and

$$\frac{\partial}{\partial T_{ic}}\mathcal{F}(\Theta,q)=\frac{1}{\sigma^2}\sum_{n=1}^{N}\Big[\sum_{S}\;q_n(S\,;\Theta')\sum_{d=1}^{D}\Big(y_{cd}^{(n)}-W_{id}T_{ic}\Big)\,\mathcal{A}_{id}^{\rho}(S,W)W_{id}\Big]\stackrel{!}{=}0\,.$$

Then it follows that

$$\sum_{n=1}^{N}\sum_{S}\;q_n(S\,;\Theta')\mathcal{A}_{id}^{\rho}(S,W)\vec{T}_i^{T}\vec{y}_d^{\,(n)}=\sum_{n=1}^{N}\sum_{S}\;q_n(S\,;\Theta')\mathcal{A}_{id}^{\rho}(S,W)W_{id}\vec{T}_i^{T}\vec{T}_i$$

and

$$\sum_{n=1}^{N}\sum_{S}\;q_n(S\,;\Theta')\sum_{d=1}^{D}\mathcal{A}_{id}^{\rho}(S,W)y_{cd}^{(n)}W_{id}=\sum_{n=1}^{N}\sum_{S}\;q_n(S\,;\Theta')\sum_{d=1}^{D}\mathcal{A}_{id}^{\rho}(S,W)T_{ic}(W_{id})^2\,.$$

After a transformation, we find that

$$\sum_{n=1}^{N}\big\langle\mathcal{A}_{id}^{\rho}(S,W)\big\rangle_{q_n}\vec{T}_i^{T}\vec{y}_d^{\,(n)}=W_{id}\sum_{n=1}^{N}\big\langle\mathcal{A}_{id}^{\rho}(S,W)\big\rangle_{q_n}\vec{T}_i^{T}\vec{T}_i$$

and

$$\sum_{n=1}^{N}\sum_{d=1}^{D}\big\langle\mathcal{A}_{id}^{\rho}(S,W)\big\rangle_{q_n}y_{cd}^{(n)}W_{id}=T_{ic}\sum_{n=1}^{N}\sum_{d=1}^{D}\big\langle\mathcal{A}_{id}^{\rho}(S,W)\big\rangle_{q_n}(W_{id})^2\,.$$

Solving for the feature and mask parameters, we then obtain the necessary conditions for a maximum of the free-energy that need to hold in the limit $\rho\rightarrow\infty$. They are given as follows:

$$W_{id}=\frac{\displaystyle\sum_{n=1}^{N}\big\langle\mathcal{A}_{id}^{\rho}(S,W)\big\rangle_{q_n}\vec{T}_i^{T}\vec{y}_d^{\,(n)}}{\displaystyle\sum_{n=1}^{N}\big\langle\mathcal{A}_{id}^{\rho}(S,W)\big\rangle_{q_n}\vec{T}_i^{T}\vec{T}_i}$$

and

$$T_{ic}=\frac{\displaystyle\sum_{n=1}^{N}\sum_{d=1}^{D}\big\langle\mathcal{A}_{id}^{\rho}(S,W)\big\rangle_{q_n}W_{id}\,y_{cd}^{(n)}}{\displaystyle\sum_{n=1}^{N}\sum_{d=1}^{D}\big\langle\mathcal{A}_{id}^{\rho}(S,W)\big\rangle_{q_n}(W_{id})^2}\,.$$

For ET, we need to restrict the sums over the data points to only those summands corresponding to data points which can be expected to be explained by less or equal γ causes, i.e., data points which are in the set $\mathcal{M}$. The resulting update equations are given as follows:

$$W_{id}^{\text{new}}=\frac{\displaystyle\sum_{n\in\mathcal{M}}\langle\mathcal{A}_{id}(S,W)\rangle_{q_n}\vec{T}_i^{T}\vec{y}_d^{\,(n)}}{\displaystyle\sum_{n\in\mathcal{M}}\langle\mathcal{A}_{id}(S,W)\rangle_{q_n}\vec{T}_i^{T}\vec{T}_i}\,,\qquad\vec{T}_i^{\,\text{new}}=\frac{\displaystyle\sum_{n\in\mathcal{M}}\sum_{d}\langle\mathcal{A}_{id}(S,W)\rangle_{q_n}W_{id}\,\vec{y}_d^{\,(n)}}{\displaystyle\sum_{n\in\mathcal{M}}\sum_{d}\langle\mathcal{A}_{id}(S,W)\rangle_{q_n}(W_{id})^2}\,.$$

Appendix C. Influence of the Basis Functions on the Data Noise

Notably, as we alter the mask and feature values during learning, these new values have an effect on the value for σ. More specifically, we have seen that optimization runs resulting in low values for sigma closely correspond to a representation of the true underlying causes while runs with comparably high sigma values do not. For data such as provided by the bars test the final sigma values for different runs may even form corresponding clusters (Figure 3). The interplay between sigma values and object parameters will be investigated here in more detail: As we compare the derivative of the free-energy w.r.t. the masks and features

$$\frac{\partial}{\partial W_{id}}\mathcal{F}(\Theta, q) = -\frac{1}{\sigma^2}\sum_{n=1}^{N}\sum_{S} q_n(S\,;\Theta')\sum_{d=1}^{D}\sum_{c=1}^{C}\left(y_{cd}^{(n)} - \mathcal{T}_{cd}(S,\Theta)\right)\frac{\partial}{\partial W_{id}}\mathcal{T}_{cd}(S,\Theta)$$

and

$$\frac{\partial}{\partial T_{ic}}\mathcal{F}(\Theta, q) = -\frac{1}{\sigma^2}\sum_{n=1}^{N}\sum_{S} q_n(S\,;\Theta')\sum_{d=1}^{D}\sum_{c=1}^{C}\left(y_{cd}^{(n)} - \mathcal{T}_{cd}(S,\Theta)\right)\frac{\partial}{\partial T_{ic}}\mathcal{T}_{cd}(S,\Theta)$$

with the derivative of the obtained update rule for the data noise squared, again w.r.t. both basis function parameters

$$\frac{\partial}{\partial W_{id}}\sigma^2 = \frac{2}{NCD}\sum_{n=1}^{N}\sum_{S} q_n(S\,;\Theta')\sum_{d=1}^{D}\sum_{c=1}^{C}\left(y_{cd}^{(n)} - \mathcal{T}_{cd}(S,\Theta)\right)\frac{\partial}{\partial W_{id}}\mathcal{T}_{cd}(S,\Theta)$$

$$\frac{\partial}{\partial T_{ic}}\sigma^2 = \frac{2}{NCD}\sum_{n=1}^{N}\sum_{S} q_n(S\,;\Theta')\sum_{d=1}^{D}\sum_{c=1}^{C}\left(y_{cd}^{(n)} - \mathcal{T}_{cd}(S,\Theta)\right)\frac{\partial}{\partial T_{ic}}\mathcal{T}_{cd}(S,\Theta)$$

we find that these are virtually identical, except for a pre-factor which will disappear when we set the derivatives equal to zero. The optimal values for the mask and feature vectors in terms of the free-energy thus also optimize the data noise.

Appendix D. Selection Function

A straightforward selection function is given by the posterior for only one active cause which we calculate through Bayes' rule as, i.e.,

$$p(S_h|Y^{(n)},\Theta) = \frac{p(Y^{(n)}\,|S_h,\Theta)p(S_h|\Theta)}{p(Y^{(n)}\,|\Theta)}$$

where $S^{(h)} := (\vec{s}^{\,(h)},\varphi)$ with $\vec{s}^{\,(h)}$ being the state with only the hth object present.

Since we are only interested in comparing the numbers per data point, a normalization w.r.t. $p(Y^{(n)}\,|\Theta)$ is not required and does not have to be calculated for the selection function. Since the prior $p(S_h|\Theta)$ is identical for all S_h, we can omit that entity as well. We are, therefore, left with only the noise (or likelihood) term, i.e.,

$$p(Y^{(n)}\,|\,S_h,\Theta) = \left(2\pi\sigma^2\right)^{-\frac{CD}{2}}\prod_{d=1}^{D}\prod_{c=1}^{C}\exp\left(-\frac{1}{2\sigma^2}\left(y_{cd}^{(n)} - \mathcal{T}_{cd}(S_h,\Theta)\right)^2\right).$$

Since the logarithm is a strictly monotonic function, we instead can consider

$$\log\Big(p(Y^{(n)} \mid S_h, \Theta)\Big) = -\frac{CD}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{d=1}^{D}\sum_{c=1}^{C}\Big(y_{cd}^{(n)} - \mathcal{T}_{cd}(S_h, \Theta)\Big)^2 .$$

As the first term is constant for all causes, as is the prefactor $-\frac{1}{2\sigma^2}$, we omit these and are then left with

$$\mathcal{S}_h\big(Y^{(n)}\big) = -\sum_{d=1}^{D}\sum_{c=1}^{C}\big(y_{cd}^{(n)} - \mathcal{T}_{cd}(S_h; \Theta)\big)^2,$$

which is the function used to select the most likely hidden units given $Y^{(n)}$ (compare Equation 19).

References

P. Berkes, R. Turner, and M. Sahani. On sparsity and overcompleteness in image models. In *Advances in Neural Information Processing Systems*, volume 20, pages 89–96, 2008.

P. Berkes, R. E. Turner, and M. Sahani. A structured model of video reproduces primary visual cortical organisation. *PLOS Computational Biology*, 5(9):e1000495, 2009.

J. Bornschein, M. Henniges, and J. Lücke. Are V1 receptive fields shaped by low-level visual occlusions? A comparative study. *PLOS Computational Biology*, 9(6):e1003062, 2013.

D. Charles, C. Fyfe, D. MacDonald, and J. Koetsier. Unsupervised neural networks for the identification of minimum overcomplete basis in visual data. *Neurocomputing*, 47(1-4): 119–143, 2002.

P. Dayan and R. S. Zemel. Competition and multiple cause models. *Neural Computation*, 7:565 – 579, 1995.

C. Eckes, J. Triesch, and C. von der Malsburg. Analysis of cluttered scenes using an elastic matching approach for stereo images. *Neural Computation*, 18(6):1441–1471, 2006.

J. Eggert, H. Wersing, and E. Körner. Transformation-invariant representation and NMF. In *2004 IEEE International Joint Conference on Neural Networks*, pages 2535–39, 2004.

G. Exarchakis, M. Henniges, J. Eggert, and J. Lücke. Ternary sparse coding. In *Proceedings LVA/ICA*, pages 204–212, 2012.

P. Földiák. Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, 64:165–170, 1990.

A. A. Frolov, D. Husek, and P. Y. Polyakov. Two expectation-maximization algorithms for Boolean factor analysis. *Neurocomputing*, 130:83–97, 2014.

K. Fukushima. Restoring partly occluded patterns: a neural network model. *Neural Networks*, 18(1):33–43, 2005.

K. Gregor and Y. LeCun. Efficient learning of sparse invariant representations. *CoRR*, abs/1105.5307, 2011.

M. Henniges, G. Puertas, J. Bornschein, J. Eggert, and J. Lücke. Binary sparse coding. In *Proceedings LVA/ICA*, LNCS 6365, pages 450–57. Springer, 2010.

S. Hochreiter and J. Schmidhuber. Feature extraction through LOCOCODE. *Neural Computation*, 11:679–714, 1999.

A. Honkela and H. Valpola. Unsupervised variational Bayesian learning of nonlinear models. In *Advances in Neural Information Processing Systems*, volume 17, pages 593–600, 2005.

P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–69, 2004.

N. Jojic and B. Frey. Learning flexible sprites in video layers. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 199–206, 2001.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–91, 1999.

N. LeRoux, N. Heess, J. Shotton, and J. Winn. Learning a generative model of images by factoring appearance and shape. *Neural Computation*, 23:593–650, 2011.

D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

J. Lücke and J. Eggert. Expectation truncation and the benefits of preselection in training generative models. *Journal of Machine Learning Research*, 11:2855–900, 2010.

J. Lücke and C. Malsburg. Rapid processing and unsupervised learning in a model of the cortical macrocolumn. *Neural Computation*, 16:501–33, 2004.

J. Lücke and M. Sahani. Maximal causes for non-linear component extraction. *Journal of Machine Learning Research*, 9:1227–67, 2008.

J. Lücke, R. Turner, M. Sahani, and M. Henniges. Occlusive components analysis. In *Advances in Neural Information Processing Systems*, volume 22, pages 1069–77, 2009.

R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998.

S. Nene, S. Nayar, and H. Murase. Columbia object image library (COIL 100). 1996.

B. Olshausen and D. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–9, 1996.

G. Puertas, J. Bornschein, and J. Lücke. The maximal causes of natural scenes are edge filters. In *Advances in Neural Information Processing Systems*, volume 23, pages 1939–47. 2010.

M. Sahani. *Latent Variable Models for Neural Data Analysis*. PhD thesis, Caltech, 1999.

E. Saund. A multiple cause mixture model for unsupervised learning. *Neural Computation*, 7:51–71, 1995.

M. Spratling. Learning image components for object recognition. *Journal of Machine Learning Research*, 7:793–815, 2006.

S. Tajima and M. Watanabe. Acquisition of nonlinear forward optics in generative models: Two-stage "downside-up" learning for occluded vision. *Neural Networks*, 24(2):148–58, 2011.

R. Turner and M. Sahani. Two problems with variational expectation maximisation for time-series models. In D. Barber, T. Cemgil, and S. Chiappa, editors, *Bayesian time series models*, chapter 5, pages 109–130. Cambridge University Press, 2011.

N. Ueda and R. Nakano. Deterministic annealing EM algorithm. *Neural Networks*, 11(2): 271–82, 1998.

H. Valpola and J. Karhunen. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural Computation*, 14(11):2647–2692, 2002.

T. Šingliar and M. Hauskrecht. Noisy-or component analysis and its application to link analysis. *Journal of Machine Learning Research*, 7:2189–2213, 2006.

H. Wersing and E. Körner. Learning optimized features for hierarchical models of invariant object recognition. *Neural Computation*, 15(7):1559–88, 2003.

C. K. I. Williams and M. K. Titsias. Greedy learning of multiple objects in images using robust statistics and factorial learning. *Neural Computation*, 16:1039–62, 2004.

A. Yuille and D. Kersten. Vision as Bayesian inference: analysis by synthesis? *Trends in Cognitive Sciences*, 10(7):301–308, 2006.

Journal of Machine Learning Research 15 (2014) 2723-2772 Submitted 5/12; Revised 1/14; Published 8/14

Optimality of Graphlet Screening in High Dimensional Variable Selection

Jiashun Jin JIASHUN@STAT.CMU.EDU
Department of Statistics
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Cun-Hui Zhang CUNHUI@STAT.RUTGERS.EDU
Department of Statistics
Rutgers University
Piscataway, NJ 08854, USA

Qi Zhang QIZHANG@STAT.WISC.EDU
Department of Biostatistics & Medical Informatics
University of Wisconsin-Madison
Madison, WI 53705, USA

Editor: Bin Yu

Abstract

Consider a linear model $Y = X\beta + \sigma z$, where X has n rows and p columns and $z \sim N(0, I_n)$. We assume both p and n are large, including the case of $p \gg n$. The unknown signal vector β is assumed to be sparse in the sense that only a small fraction of its components is nonzero. The goal is to identify such nonzero coordinates (i.e., variable selection).

We are primarily interested in the regime where signals are both *rare and weak* so that successful variable selection is challenging but is still possible. We assume the Gram matrix $G = X'X$ is sparse in the sense that each row has relatively few large entries (diagonals of G are normalized to 1). The sparsity of G naturally induces the sparsity of the so-called *Graph of Strong Dependence* (GOSD). The key insight is that there is an interesting interplay between the signal sparsity and graph sparsity: in a broad context, the signals decompose into many small-size components of GOSD that are disconnected to each other.

We propose *Graphlet Screening* for variable selection. This is a two-step Screen and Clean procedure, where in the first step, we screen subgraphs of GOSD with sequential χ^2-tests, and in the second step, we clean with penalized MLE. The main methodological innovation is to use GOSD to guide both the screening and cleaning processes.

For any variable selection procedure $\hat{\beta}$, we measure its performance by the Hamming distance between the sign vectors of $\hat{\beta}$ and β, and assess the optimality by the minimax Hamming distance. Compared with more stringent criteria such as exact support recovery or oracle property, which demand strong signals, the Hamming distance criterion is more appropriate for weak signals since it naturally allows a small fraction of errors.

We show that in a broad class of situations, Graphlet Screening achieves the optimal rate of convergence in terms of the Hamming distance. Unlike Graphlet Screening, well-known procedures such as the L^0/L^1-penalization methods do not utilize local graphic structure for variable selection, so they generally do not achieve the optimal rate of convergence, even in very simple settings and even if the tuning parameters are ideally set.

The the presented algorithm is implemented as R-CRAN package *ScreenClean* and in *matlab* (available at `http://www.stat.cmu.edu/~jiashun/Research/software/GS-matlab/`).

Keywords: asymptotic minimaxity, graph of least favorables (GOLF), graph of strong dependence (GOSD), graphlet screening (GS), Hamming distance, phase diagram, rare and weak signal model, screen and clean, sparsity

1. Introduction

Consider a linear regression model

$$Y = X\beta + \sigma z, \qquad X = X_{n,p}, \qquad z \sim N(0, I_n). \tag{1}$$

We write

$$X = [x_1, x_2, \ldots, x_p], \qquad \text{and} \qquad X' = [X_1, X_2, \ldots, X_n], \tag{2}$$

so that x_j is the j-th design vector and X_i is the i-th sample. Motivated by the recent interest in 'Big Data', we assume both p and n are large but $p \geq n$ (though this should not be taken as a restriction). The vector β is unknown to us, but is presumably *sparse* in the sense that only a small proportion of its entries is nonzero. Calling a nonzero entry of β a signal, the main interest of this paper is to identify all signals (i.e., variable selection).

Variable selection is one of the most studied problem in statistics. However, there are important regimes where our understanding is very limited.

One of such regimes is the *rare and weak* regime, where the signals are both *rare* (or sparse) and *individually weak*. Rare and weak signals are frequently found in research areas such as Genome-wide Association Study (GWAS) or next generation sequencing. Unfortunately, despite urgent demand in applications, the literature of variable selection has been focused on the regime where the signals are rare but *individually strong*. This motivates a revisit to variable selection, focusing on the rare and weak regime.

For variable selection in this regime, we need new methods and new theoretical frameworks. In particular, we need a loss function that is appropriate for rare and weak signals to evaluate the optimality. In the literature, given a variable selection procedure $\hat{\beta}$, we usually use the probability of exact recovery $P(\mathrm{sgn}(\hat{\beta}) \neq \mathrm{sgn}(\beta))$ as the measure of loss (Fan and Li, 2001); $\mathrm{sgn}(\hat{\beta})$ and $\mathrm{sgn}(\beta)$ are the sign vectors of $\hat{\beta}$ and β respectively. In the rare and weak regime, the signals are so rare and weak that exact recovery is impossible, and the Hamming distance between $\mathrm{sgn}(\hat{\beta})$ and $\mathrm{sgn}(\beta)$ is a more appropriate measure of loss.

Our focus on the rare and weak regime and the Hamming distance loss provides new perspectives to variable selection, in methods and in theory.

Throughout this paper, we assume the diagonals of the Gram matrix

$$G = X'X \tag{3}$$

are normalized to 1 (and approximately 1 in the random design model), instead of n as often used in the literature. The difference between two normalizations is non-essential, but the signal vector β are different by a factor of $n^{1/2}$.

We also assume the Gram matrix G is 'sparse' (aka. graph sparsity) in the sense that each of its rows has relatively few large entries. Signal sparsity and graph sparsity can be simultaneously found in the following application areas.

- *Compressive sensing.* We are interested in a very high dimensional sparse vector β. The goal is to store or transmit n linear functionals of β and then reconstruct it. For $1 \leq i \leq n$, we choose a p-dimensional coefficient vector X_i and observe $Y_i = X_i'\beta + \sigma z_i$ with an error σz_i. The so-called Gaussian design is often considered (Donoho, 2006a,b; Bajwa et al., 2007), where $X_i \overset{iid}{\sim} N(0, \Omega/n)$ and Ω is sparse; the sparsity of Ω induces the sparsity of $G = X'X$.

- *Genetic Regulatory Network (GRN).* For $1 \leq i \leq n$, $W_i = (W_i(1), \ldots, W_i(p))'$ represents the expression level of p different genes of the i-th patient. Approximately, $W_i \overset{iid}{\sim} N(\alpha, \Sigma)$, where the contrast mean vector α is sparse reflecting that only few genes are differentially expressed between a normal patient and a diseased one (Peng et al., 2009). Frequently, the concentration matrix $\Omega = \Sigma^{-1}$ is believed to be sparse, and can be effectively estimated in some cases (e.g., Bickel and Levina, 2008 and Cai et al., 2010), or can be assumed as known in others, with the so-called "data about data" available (Li and Li, 2011). Let $\hat{\Omega}$ be a positive-definite estimate of Ω, the setting can be re-formulated as the linear model $(\hat{\Omega})^{1/2}Y \approx \Omega^{1/2}Y \sim N(\Omega^{1/2}\beta, I_p)$, where $\beta = \sqrt{n}\alpha$ and the Gram matrix $G \approx \Omega$, and both are sparse.

Other examples can be found in Computer Security (Ji and Jin, 2011) and Factor Analysis (Fan et al., 2011).

The sparse Gram matrix G induces a sparse graph which we call the *Graph of Strong Dependence* (GOSD), denoted by $\mathcal{G} = (V, E)$, where $V = \{1, 2, \ldots, p\}$ and there is an edge between nodes i and j if and only the design vectors x_i and x_j are *strongly correlated*. Let

$$S = S(\beta) = \{1 \leq j \leq p : \beta_j \neq 0\}$$

be the support of β and $\mathcal{G}_S$ be the subgraph of $\mathcal{G}$ formed by all nodes in S. The key insight is that, there is an interesting interaction between signal sparsity and graph sparsity, which yields the subgraph $\mathcal{G}_S$ decomposable: $\mathcal{G}_S$ splits into many "graphlet"; each "graphlet" is a small-size component and different components are not connected (in $\mathcal{G}_S$).

While we can always decompose $\mathcal{G}_S$ in this way, our emphasis in this paper is that, in many cases, the maximum size of the graphlets is small; see Lemma 1 and related discussions.

The decomposability of $\mathcal{G}_S$ motivates a new approach to variable selection, which we call *Graphlet Screening* (GS). GS is a Screen and Clean method (Wasserman and Roeder, 2009). In the screening stage, we use multivariate screening to identify candidates for all the graphlets. Let $\hat{S}$ be all the nodes that survived the screening, and let $\mathcal{G}_{\hat{S}}$ be the subgraph of GOSD formed by all nodes in $\hat{S}$. Although $\hat{S}$ is expected to be somewhat larger than S, the subgraph $\mathcal{G}_{\hat{S}}$ is still likely to resemble $\mathcal{G}_S$ in structure in the sense that it, too, splits into many small-size disconnected components. We then clean each component separately to remove false positives.

The objective of the paper is two-fold.

- To propose a "fundamentally correct" solution in the rare and weak paradigm along with a computationally fast algorithm for the solution.

- To show that GS achieves the optimal rate of convergence in terms of the Hamming distance, and achieves the optimal phase diagram for variable selection.

Phase diagram can be viewed as an optimality criterion which is especially appropriate for rare and weak signals. See Donoho and Jin (2004) and Jin (2009) for example.

In the settings we consider, most popular approaches are *not* rate optimal; we explain this in Sections 1.1-1.3. In Section 1.4, we explain the basic idea of GS and why it works.

1.1 Non-optimality of the L^0-penalization Method for Rare and Weak Signals

When $\sigma = 0$, Model (1) reduces to the "noiseless" model $Y = X\beta$. In this model, Donoho and Stark (1989) (see also Donoho and Huo, 2001) reveals a fundamental phenomenon on sparse representation. Fix (X, Y) and consider the equation $Y = X\beta$. Since $p > n$, the equation has infinitely many solutions. However, a *very sparse solution*, if exists, is *unique* under mild conditions on the design X, with all other solutions being much denser. In fact, if the sparsest solution β_0 has k elements, then all other solutions of the equation $Y = X\beta$ must have at least $(\mathrm{rank}(X) - k + 1)$ nonzero elements, and $\mathrm{rank}(X) = n$ when X is in a "general position".

From a practical viewpoint, we frequently believe that this unique sparse solution is the truth (i.e., Occam's razor). Therefore, the problem of variable selection can be solved by some *global methods* designed for finding the sparsest solution to the equation $Y = X\beta$.

Since the L^0-norm is (arguably) the most natural way to measure the sparsity of a vector, the above idea suggests that the L^0-penalization method is a "fundamentally correct" (but computationally intractable) method for variable selection, provided some mild conditions on the noise, signal and design matrix, e.g., noiseless, Signal-to-Noise Ratio (SNR) is high, or signals are sufficiently sparse (Donoho and Stark, 1989; Donoho and Huo, 2001).

Motivated by this, in the past two decades, a long list of computationally tractable algorithms have been proposed that approximate the solution of the L^0-penalization method, including the lasso, SCAD, MC+, and many more (Akaike, 1974; Candes and Tao, 2007; Efron et al., 2004; Fan and Li, 2001; Schwarz, 1978; Tibshirani, 1996; Zhang, 2010, 2011; Zhao and Yu, 2006; Zou, 2006).

With that being said, we must note that these methodologies were built upon a framework with four tightly woven core components: "signals are rare but strong", "the truth is also the sparsest solution to $Y = X\beta$", "probability of exact recovery is an appropriate loss function", and "L^0-penalization method is a fundamentally correct approach". Unfortunately, when signals are rare and weak, such a framework is no longer suitable.

- When signals are "rare and weak", the fundamental uniqueness property of the sparse solution in the noiseless case is no longer valid in the noisy case. Consider the model $Y = X\beta + \sigma z$ and suppose that a sparse β_0 is the true signal vector. There are many vectors β that are small perturbations of β_0 such that the two models $Y = X\beta + \sigma z$ and $Y = X\beta_0 + \sigma z$ are indistinguishable (i.e., all tests are asymptotically powerless). In the "rare and strong" regime, β_0 is the sparsest solution among all such "eligible" solutions of $Y = X\beta + \sigma z$. However, this claim no longer holds in the "rare and weak" regime and the principle of Occam's razor may not be as relevant as before.

- The L^0-penalization method is originally designed for "rare and strong" signals where "exact recovery" is used to measure its performance (Donoho and Stark, 1989; Donoho and Huo, 2001; Donoho, 2006a). When we must consider "rare and weak" signals and

when we use the Hamming distance as the loss function, it is unclear whether the L^0-penalization method is still "fundamentally correct".

In fact, in Section 2.8 (see also Ji and Jin, 2011), we show that the L^0-penalization method is not optimal in Hamming distance when signals are rare and weak, even with very simple designs (i.e., Gram matrix is tridiagonal or block-wise) and even when the tuning parameter is ideally set. Since the L^0-penalization method is used as the benchmark in the development of many other penalization methods, its sub-optimality is expected to imply the sub-optimality of other methods designed to match its performance (e.g., lasso, SCAD, MC+).

1.2 Limitation of Univariate Screening and UPS

Univariate Screening (also called marginal regression or Sure Screening in Fan and Lv, 2008; Genovese et al., 2012) is a well-known variable selection method. For $1 \leq j \leq p$, recall that x_j is the j-th column of X. Univariate Screening selects variables with large marginal correlations: $|(x_j, Y)|$, where $(\cdot, \cdot)$ denotes the inner product. The method is computationally fast, but it can be seriously corrupted by the so-called phenomenon of "signal cancellation" (Wasserman and Roeder, 2009). In our model (1)-(3), the SNR associated with (x_j, Y) is

$$\frac{1}{\sigma}\sum_{\ell=1}^{p}(x_j, x_\ell)\beta_\ell = \frac{\beta_j}{\sigma} + \frac{1}{\sigma}\sum_{\ell\neq j}(x_j, x_\ell)\beta_\ell.$$

"Signal cancellation" happens if SNR is significantly smaller than β_j/σ. For this reason, the success of Univariate Screening needs relatively strong conditions (e.g., Faithfulness Condition Genovese et al., 2012), under which signal cancellation does not have a major effect.

In Ji and Jin (2011), Ji and Jin proposed *Univariate Penalized Screening (UPS)* as a refinement of Univariate Screening, where it was showed to be optimal in the rare and weak paradigm, for the following two scenarios. The first scenario is where the nonzero effects of variables are all positively correlated: $(x_j\beta_j)'(x_k\beta_k) \geq 0$ for all $\{j, k\}$. This guarantees the faithfulness of the univariate association test. The second scenario is a Bernoulli model where the "signal cancellation" only has negligible effects over the Hamming distance of UPS.

With that being said, UPS attributes its success mostly to the cleaning stage; the screening stage of UPS uses nothing but Univariate Screening, so UPS does not adequately address the challenge of "signal cancellation". For this reason, we should not expect UPS to be optimal in much more general settings.

1.3 Limitations of Brute-force Multivariate Screening

One may attempt to overcome "signal cancellation" by multivariate screening, with Brute-force Multivariate Screening (BMS) being the most straightforward version. Fix an integer $1 \leq m_0 \ll p$. BMS consists of a series of screening phases, indexed by m, $1 \leq m \leq m_0$, that are increasingly more ambitious. In Phase-m BMS, we test the significance of the association between Y and any set of m different design variables $\{x_{j_1}, x_{j_2}, \ldots, x_{j_m}\}$, $j_1 < j_2 < \ldots < j_m$,

and retain all such design variables if the test is significant. The problem of BMS is, it enrolls too many candidates for screening, which is both unnecessary and unwise.

- (*Screening inefficiency*). In Phase-m of BMS, we test about $\binom{p}{m}$ hypotheses involving different subsets of m design variables. The larger the number of hypotheses we consider, the higher the threshold we need to set for the tests, in order to control the false positives. When we enroll too many candidates for hypothesis testing, we need signals that are stronger than necessary in order for them to survive the screening.

- (*Computational challenge*). Testing $\binom{p}{m}$ hypotheses is computationally infeasible when p is large, even when m is very small, e.g., $(p, m) = (10^4, 3)$.

1.4 Graphlet Screening: How It Is Different and How It Works

Graphlet Screening (GS) uses a similar screening strategy as BMS does, except for a major difference. When it comes to the test of significance between Y and design variables $\{x_{j_1}, x_{j_2}, \ldots, x_{j_m}\}$, $j_1 < j_2 < \ldots < j_m$, GS only carries out such a test if $\{j_1, j_2, \ldots, j_m\}$ is a connected subgraph of the GOSD. Otherwise, the test is safely skipped!

Fixing an appropriate threshold $\delta > 0$, we let $\Omega^{*,\delta}$ be the regularized Gram matrix:

$$\Omega^{*,\delta}(i,j) = G(i,j)1\{|G(i,j)| \geq \delta\}, \qquad 1 \leq i, j \leq p. \tag{4}$$

The GOSD $\mathcal{G} \equiv \mathcal{G}^{*,\delta} = (V, E)$ is the graph where $V = \{1, 2, \ldots, p\}$ and there is an edge between nodes i and j if and only if $\Omega^{*,\delta}(i,j) \neq 0$. See Section 2.6 for the choice of δ.

Remark. GOSD and $\mathcal{G}$ are generic terms which vary from case to case, depending on G and δ. GOSD is very different from the Bayesian conditional independence graphs (Pearl, 2000).

Fixing $m_0 \geq 1$ as in BMS, we define

$$\mathcal{A}(m_0) = \mathcal{A}(m_0; G, \delta) = \{\text{all connected subgraphs of } \mathcal{G}^{*,\delta} \text{ with size} \leq m_0\}.$$

GS is a Screen and Clean method, consisting of a graphical screening step (GS-step) and a graphical cleaning step (GC-step).

- *GS-step*. We test the significance of association between Y and $\{x_{j_1}, x_{j_2}, \ldots, x_{j_m}\}$ if and only if $\{j_1, j_2, \ldots, j_m\} \in \mathcal{A}(m_0)$ (i.e., graph guided multivariate screening). Once $\{j_1, \ldots, j_m\}$ is retained, it remains there until the end of the GS-step.

- *GC-step*. The set of surviving nodes decompose into many small-size components, which we fit separately using an efficient low-dimensional test for small graphs.

GS is similar to Wasserman and Roeder (2009) for both of them have a screening and a cleaning stage, but is more sophisticated. For clarification, note that Univariate Screening or BMS introduced earlier does not contain a cleaning stage and can be viewed as a counterpart of the GS-step.

We briefly explain why GS works. We discuss the GS-step and GC-step separately.

Consider the GS-step first. Compared with BMS, the GS-step recruits far fewer candidates for screening, so it is able to overcome the two major shortcomings of BMS aforementioned: high computational cost and low statistical efficiency. In fact, fix $K \geq 1$ and

suppose $\mathcal{G}^{*,\delta}$ is K-sparse (see Section 1.5 for the definition). By a well-known result in graph theory (Frieze and Molloy, 1999),

$$|\mathcal{A}(m_0)| \leq Cm_0p(eK)^{m_0}. \tag{5}$$

The right hand side is much smaller than the term $\binom{p}{m_0}$ as we encounter in BMS.

At the same time, recall that $S = S(\beta)$ is the support of β. Let $\mathcal{G}_S \equiv \mathcal{G}_S^{*,\delta}$ be the subgraph of $\mathcal{G}^{*,\delta}$ consisting all signal nodes. We can always split $\mathcal{G}_S^{*,\delta}$ into "graphlets" (arranged lexicographically) as follows:

$$\mathcal{G}_S^{*,\delta} = \mathcal{G}_{S,1}^{*,\delta} \cup \mathcal{G}_{S,2}^{*,\delta} \ldots \cup \mathcal{G}_{S,M}^{*,\delta}, \tag{6}$$

where each $\mathcal{G}_{S,i}^{*,\delta}$ is a component (i.e., a maximal connected subgraph) of $\mathcal{G}_S^{*,\delta}$, and different $\mathcal{G}_{S,i}^{*,\delta}$ are not connected in $\mathcal{G}_S^{*,\delta}$. Let

$$m_0^* = m_0^*(S(\beta), G, \delta) = \max_{1\leq i\leq M} |\mathcal{G}_{S,i}^{*,\delta}|$$

be the maximum size of such graphlets. Note that M also depends on $S(\beta)$, G and δ.

In many cases, m_0^* is small. One such case is when we have a Bernoulli signal model.

Lemma 1 *Fix $K \geq 1$ and $\epsilon > 0$. If $\mathcal{G}^{*,\delta}$ is K-sparse and $\mathrm{sgn}(|\beta_1|), \mathrm{sgn}(|\beta_2|), \ldots, \mathrm{sgn}(|\beta_p|)$ are iid from* Bernoulli(ϵ)*, then except for a probability $p(e\epsilon K)^{m_0+1}$, $m_0^*(S(\beta), G, \delta) \leq m_0$.*

Lemma 1 is not tied to the Bernoulli model and holds more generally. For example, it holds when $\{\mathrm{sgn}(|\beta_i|)\}_{i=1}^p$ are generated according to certain Ising models (Ising, 1925).

We recognize that in order for the GS-step to be efficient both in screening and in computation, it is sufficient that

$$m_0 \geq m_0^*. \tag{7}$$

In fact, first, if (7) holds, then for each $1 \leq \ell \leq M$, $\mathcal{G}_{S,\ell}^{*,\delta} \in \mathcal{A}(m_0)$. Therefore, at some point of the screening process of the GS-step, we must have considered a significance test between Y and the set of design variables $\{x_j : j \in \mathcal{G}_{S,\ell}^{*,\delta}\}$. Consequently, the GS-step is able to overcome the "signal cancellations" (the explanation is a little bit long, and we slightly defer it). Second, since m_0^* is small in many situations, we could choose a relatively small m_0 such that (7) holds. When m_0 is small, as long as K is small or moderately large, the GS-step is computationally feasible. In fact, the right hand side of (5) is only larger than p by a moderate factor. See Section 2.2 for more discussion on the computation complexity.

We now explain the first point above. The notations below are frequently used.

Definition 2 *For X in Models (1)-(2) and any subset $\mathcal{I} \subset \{1, 2, ..., p\}$, let $P^{\mathcal{I}} = P^{\mathcal{I}}(X)$ be the projection from $\mathbb{R}^n$ to the subspace spanned by $\{x_j : j \in \mathcal{I}\}$.*

Definition 3 *For an $n \times p$ matrix A and sets $\mathcal{I} \subset \{1, \ldots, n\}$ and $\mathcal{J} \subset \{1, \ldots, p\}$, $A^{\mathcal{I},\mathcal{J}}$ is the $|\mathcal{I}| \times |\mathcal{J}|$ sub-matrix formed by restricting the rows of A to $\mathcal{I}$ and columns to $\mathcal{J}$.*

When $p = 1$, A is a vector, and $A^{\mathcal{I}}$ is the sub-vector of A formed by restricting the rows of A to $\mathcal{I}$. When $\mathcal{I} = \{1, 2, \ldots, n\}$ (or $\mathcal{J} = \{1, 2, \ldots, p\}$), we write $A^{\mathcal{I},\mathcal{J}}$ as $A^{\otimes,\mathcal{J}}$ (or $A^{\mathcal{I},\otimes}$). Note that indices in $\mathcal{I}$ or $\mathcal{J}$ are not necessarily sorted ascendingly.

Recall that for each $1 \le \ell \le M$, at some point of the GS-step, we must have considered a significance test between Y and the set of design variables $\{x_j : j \in \mathcal{G}^{*,\delta}_{S,\ell}\}$. By (6), we rewrite Model (1) as

$$Y = \sum_{\ell=1}^{M} X^{\otimes,\mathcal{G}^{*,\delta}_{S,\ell}} \beta^{\mathcal{G}^{*,\delta}_{S,\ell}} + \sigma z, \qquad z \sim N(0, I_n).$$

The key is the set of matrices $\{X^{\otimes,\mathcal{G}^{*,\delta}_{S,\ell}} : 1 \le \ell \le M\}$ are nearly orthogonal (i.e., for any column ξ of $X^{\otimes,\mathcal{G}^{*,\delta}_{S,k}}$ and any column η of $X^{\otimes,\mathcal{G}^{*,\delta}_{S,\ell}}$, $|(\xi, \eta)|$ is small when $k \neq \ell$).

When we test the significance between Y and $\{x_j, j \in \mathcal{G}^{*,\delta}_{S,\ell}\}$, we are testing the null hypothesis $\beta^{\mathcal{G}^{*,\delta}_{S,\ell}} = 0$ against the alternative $\beta^{\mathcal{G}^{*,\delta}_{S,\ell}} \neq 0$. By the near orthogonality aforementioned, approximately, $(X^{\otimes,\mathcal{G}^{*,\delta}_{S,\ell}})'Y$ is a sufficient statistic for $\beta^{\mathcal{G}^{*,\delta}_{S,\ell}}$, and the optimal test is based on the χ^2-test statistic $\|P^{\mathcal{G}^{*,\delta}_{S,\ell}} Y\|^2$.

The near orthogonality also implies that significant "signal cancellation" only happens among signals within the same graphlet. When we screen each graphlet *as a whole* using the χ^2-statistic above, "signal cancellation" between different graphlets only has negligible effects. In this way, GS-step is able to retain all nodes in $\mathcal{G}^{*,\delta}_{S,\ell}$ in a nearly optimal way, and so overcome the challenge of "signal cancellation". This explains the first point.

Note that the GS-step consists of a sequence of sub-steps, each sub-step is associated with an element of $\mathcal{A}(m_0)$. When we screen $\mathcal{G}^{*,\delta}_{S,\ell}$ as a whole, it is possible some of the nodes have already been retained in the previous sub-steps. In this case, we implement the χ^2-test slightly differently, but the insight is similar. See Section 2.1 for details.

We now discuss the GC-step. Let $\hat{S}$ be all the surviving nodes of the GS-step, and let $\mathcal{G}^{*,\delta}_{\hat{S}}$ be the subgraph of $\mathcal{G}^{*,\delta}$ formed by confining all nodes to $\hat{S}$. Similarly, we have (a) the decomposition $\mathcal{G}^{*,\delta}_{\hat{S}} = \mathcal{G}^{*,\delta}_{\hat{S},1} \cup \mathcal{G}^{*,\delta}_{\hat{S},2} \ldots \cup \mathcal{G}^{*,\delta}_{\hat{S},\hat{M}}$, (b) the near orthogonality between the $\hat{M}$ different matrices, each is formed by $\{x_j : j \in \mathcal{G}^{*,\delta}_{\hat{S},\ell}\}$. Moreover, a carefully tuned screening stage of the GS ensures that most of the components $\mathcal{G}^{*,\delta}_{\hat{S},\ell}$ are only small perturbations of their counterparts in the decomposition of $\mathcal{G}^{*,\delta}_{S} = \mathcal{G}^{*,\delta}_{S,1} \cup \mathcal{G}^{*,\delta}_{S,2} \ldots \cup \mathcal{G}^{*,\delta}_{S,M}$ as in (6), and the maximum size of $\mathcal{G}^{*,\delta}_{\hat{S},\ell}$ is not too much larger than $m_0^* = m_0^*(S(\beta), G, \delta)$. Together, these allow us to clean $\mathcal{G}^{*,\delta}_{\hat{S},\ell}$ separately, without much loss of efficiency. Since the maximum size of $\mathcal{G}^{*,\delta}_{\hat{S},\ell}$ is small, the computational complexity in the cleaning stage is moderate.

1.5 Content

The remaining part of the paper is organized as follows. In Section 2, we show that GS achieves the minimax Hamming distance in the Asymptotic Rare and Weak (ARW) model, and use the phase diagram to visualize the optimality of GS, and to illustrate the advantage of GS over the L^0/L^1-penalization methods. In Section 3, we explain that GS attributes

its optimality to the so-called *Sure Screening* property and the *Separable After Screening* property, and use these two properties to prove our main result, Theorem 8. Section 4 contains numeric results, Section 5 discusses more connections to existing literature and possible extensions of GS, and Section 6 contains technical proofs.

Below are some notations we use in this paper. L_p denotes a generic multi-log(p) term that may vary from occurrence to occurrence; see Definition 5. For a vector $\beta \in R^p$, $\|\beta\|_q$ denotes the L^q-norm, and when $q = 2$, we drop q for simplicity. For two vectors α and β in R^p, $\alpha \circ \beta \in R^p$ denotes the vector in R^p that satisfies $(\alpha \circ \beta)_i = \alpha_i\beta_i$, $1 \le i \le p$; "$\circ$" is known as the Hadamard product.

For an $n \times p$ matrix A, $\|A\|_\infty$ denotes the matrix L^∞-norm, and $\|A\|$ denotes the spectral norm (Horn and Johnson, 1990). Recall that for two sets $\mathcal{I}$ and $\mathcal{J}$ such that $\mathcal{I} \subset \{1, 2, \ldots, n\}$ and $\mathcal{J} \subset \{1, 2, \ldots, p\}$, $A^{\mathcal{I},\mathcal{J}}$ denotes the submatrix of A formed by restricting the rows and columns of A to $\mathcal{I}$ and $\mathcal{J}$, respectively. Note that the indices in $\mathcal{I}$ and $\mathcal{J}$ are not necessarily sorted in the ascending order. In the special case where $\mathcal{I} = \{1, 2, \ldots, n\}$ (or $\mathcal{J} = \{1, 2, \ldots, p\}$), we write $A^{\mathcal{I},\mathcal{J}}$ as $A^{\otimes,\mathcal{J}}$ (or $A^{\mathcal{I},\otimes}$). In the special case where $n = p$ and A is positive definite, $\lambda_k^*(A)$ denotes the minimum eigenvalue of all the size k principal submatrices of A, $1 \le k \le p$. For X in (1), $P^{\mathcal{I}}$ denotes the projection to the column space of $X^{\otimes,\mathcal{I}}$.

Recall that in Model (1), $Y = X\beta + \sigma z$. Fixing a threshold $\delta > 0$. Let $G = X'X$ be the Gram matrix, and let $\Omega^{*,\delta}$ be the regularized Gram matrix defined by $\Omega^{*,\delta}(i,j) = G(i,j)1\{|G(i,j)| \ge \delta\}$, $1 \le i, j \le p$. Let $\mathcal{G}^{*,\delta}$ be the graph where each index in $\{1, 2, \ldots, p\}$ is a node, and there is an edge between node i and node j if and only if $\Omega^{*,\delta}(i,j) \neq 0$. We let $S(\beta)$ be the support of β, and denote $\mathcal{G}_S^{*,\delta}$ by the subgraph of $\mathcal{G}^{*,\delta}$ formed by all nodes in $S(\beta)$. We call $\mathcal{G}^{*,\delta}$ the Graph of Strong Dependence (GOSD) and sometimes write it by $\mathcal{G}$ for short. The GOSD and $\mathcal{G}$ are generic notations which depend on (G, δ) and may vary from occurrence to occurrence. We also denote $\mathcal{G}^\diamond$ by the Graph of Least Favorable (GOLF). GOLF only involves the study of the information lower bound. For an integer $K \ge 1$, a graph $\mathcal{G}$, and one of its subgraph $\mathcal{I}_0$, we write $\mathcal{I}_0 \triangleleft \mathcal{G}$ if and only if $\mathcal{I}_0$ is a component of $\mathcal{G}$ (i.e., a maximal connected subgraph of $\mathcal{G}$), and we call $\mathcal{G}$ K-sparse if its maximum degree is no greater than K.

2. Main Results

In Section 2.1, we formally introduce GS. In Section 2.2, we discuss the computational complexity of GS. In Sections 2.3-2.6, we show that GS achieves the optimal rate of convergence in the Asymptotic Rare and Weak model. In Sections 2.7-2.8, we introduce the notion of phase diagram and use it to compare GS with the L^0/L^1-penalization methods. We conclude the section with a summary in Section 2.9.

2.1 Graphlet Screening: The Procedure

GS consists of a GS-step and a GC-step. We describe two steps separately. Consider the *GS*-step first. Fix $m_0 \ge 1$ and $\delta > 0$, recall that $\mathcal{G}^{*,\delta}$ denotes the GOSD and $\mathcal{A}(m_0)$ consists of all connected subgraphs of $\mathcal{G}^{*,\delta}$ with size $\le m_0$.

- *Initial sub-step.* Let $\mathcal{U}_p^* = \emptyset$. List all elements in $\mathcal{A}(m_0)$ in the ascending order of the number of nodes it contains, with ties broken lexicographically. Since a node is thought of as connected to itself, the first p connected subgraphs on the list are simply the nodes $1, 2, \ldots, p$. We screen all connected subgraphs in the order they are listed.

- *Updating sub-step.* Let $\mathcal{I}_0$ be the connected subgraph under consideration, and let $\mathcal{U}_p^*$ be the current set of retained indices. We update $\mathcal{U}_p^*$ with a χ^2 test as follows. Let $\hat{F} = \mathcal{I}_0 \cap \mathcal{U}_p^*$ and $\hat{D} = \mathcal{I}_0 \setminus \mathcal{U}_p^*$, so that $\hat{F}$ is the set of nodes in $\mathcal{I}_0$ that have already been accepted, and $\hat{D}$ is the set of nodes in $\mathcal{I}_0$ that is currently under investigation. Note that no action is needed if $\hat{D} = \emptyset$. For a threshold $t(\hat{D}, \hat{F}) > 0$ to be determined, we update $\mathcal{U}_p^*$ by adding all nodes in $\hat{D}$ to it if

$$T(Y, \hat{D}, \hat{F}) = \|P^{\mathcal{I}_0} Y\|^2 - \|P^{\hat{F}} Y\|^2 > t(\hat{D}, \hat{F}), \tag{8}$$

  and we keep $\mathcal{U}_p^*$ the same otherwise (by default, $\|P^{\hat{F}} Y\| = 0$ if $\hat{F} = \emptyset$). We continue this process until we finish screening all connected subgraphs on the list. The final set of retained indices is denoted by $\mathcal{U}_p^*$.

See Table 1 for a recap of the procedure. In the *GS*-step, once a node is kept in any sub-stage of the screening process, it remains there until the end of the *GS*-step (however, it may be killed in the *GC*-step). This has a similar flavor to that of the Forward regression.

In principle, the procedure depends on how the connected subgraphs of the same size are initially ordered, and different ordering could give different numeric results. However, such differences are usually negligibly small. Alternatively, one could revise the procedure so that it does not depend on the ordering. For example, in the updating sub-step, we could choose to update $\mathcal{U}_p^*$ only when we finish screening all connected sub-graphs of size k, $1 \le k \le m_0$. While the theoretic results below continue to hold if we revise GS in this way, we must note that from a numeric perspective, the revision would not produce a very different result. For reasons of space, we skip discussions along this line.

The GS-step uses a set of tuning parameters:

$$\mathcal{Q} \equiv \{t(\hat{D}, \hat{F}) : (\hat{D}, \hat{F}) \text{ are as defined in (8)}\}.$$

A convenient way to set these parameters is to let $t(\hat{D}, \hat{F}) = 2\sigma^2 q \log p$ for a fixed $q > 0$ and all $(\hat{D}, \hat{F})$. More sophisticated choices are given in Section 2.6.

The *GS*-step has two important properties: *Sure Screening* and *Separable After Screening (SAS)*. With tuning parameters $\mathcal{Q}$ properly set, the Sure Screening property says that $\mathcal{U}_p^*$ retains all but a negligible fraction of the signals. The SAS property says that as a subgraph of $\mathcal{G}^{*,\delta}$, $\mathcal{U}_p^*$ decomposes into many disconnected components, each has a size $\le \ell_0$ for a fixed small integer ℓ_0. Together, these two properties enable us to reduce the original large-scale regression problem to many small-size regression problems that can be solved parallelly in the *GC*-step. See Section 3 for elaboration on these ideas.

We now discuss the *GC*-step. For any $1 \le j \le p$, we have either $j \notin \mathcal{U}_p^*$, or that there is a unique connected subgraph $\mathcal{I}_0$ such that $j \in \mathcal{I}_0 \lhd \mathcal{U}_p^*$. In the first case, we estimate β_j as 0. In the second case, for two tuning parameters $u^{gs} > 0$ and $v^{gs} > 0$, we estimate the whole set of variables $\beta^{\mathcal{I}_0}$ by minimizing the functional

$$\|P^{\mathcal{I}_0}(Y - X^{\otimes, \mathcal{I}_0}\xi)\|^2 + (u^{gs})^2 \|\xi\|_0 \tag{9}$$

GS-step:	List $\mathcal{G}^{*,\delta}$-connected submodels $\mathcal{I}_{0,k}$ with $\lvert\mathcal{I}_{0,1}\rvert \le \lvert\mathcal{I}_{0,2}\rvert \le \cdots \le m_0$
	Initialization: $\mathcal{U}_p^* = \emptyset$ and $k = 1$
	Test $H_0 : \mathcal{I}_{0,k} \cap \mathcal{U}_p^*$ against $H_1 : \mathcal{I}_{0,k}$ with χ^2 test (8)
	Update: $\mathcal{U}_p^* \leftarrow \mathcal{U}_p^* \cup \mathcal{I}_{0,k}$ if H_0 rejected, $k \leftarrow k+1$
GC-step:	As a subgraph of $\mathcal{G}^{*,\delta}$, $\mathcal{U}_p^*$ decomposes into many components $\mathcal{I}_0$
	Use the L^0-penalized test (9) to select a subset $\hat{\mathcal{I}}_0$ of each $\mathcal{I}_0$
	Return the union of $\hat{\mathcal{I}}_0$ as the selected model

Table 1: Graphlet Screening Algorithm.

over all $\lvert\mathcal{I}_0\rvert \times 1$ vectors ξ, each nonzero coordinate of which $\ge v^{gs}$ in magnitude. The resultant estimator is the final estimate of GS, and we use $\hat{\beta}^{gs} = \hat{\beta}^{gs}(Y; \delta, \mathcal{Q}, u^{gs}, v^{gs}, X, p, n)$ to denote it. See Section 1.5 for notations used in this paragraph.

Sometimes for linear models with random designs, the Gram matrix G is very noisy, and GS is more effective if we use it iteratively for a few times (≤ 5). This can be implemented in a similar way as that in Ji and Jin (2011, Section 3). Here, the main purpose of iteration is to denoise G, not for variable selection. See Ji and Jin (2011, Section 3) and Section 4 for more discussion.

2.2 Computational Complexity

If we exclude the overhead of obtaining $\mathcal{G}^{*,\delta}$, then the computation cost of GS contains two parts, that of the GS-step and that of the GC-step. In each part, the computation cost hinges on the sparsity of $\mathcal{G}^{*,\delta}$. In Section 2.3, we show that with a properly chosen δ, for a wide class of design matrices, $\mathcal{G}^{*,\delta}$ is K-sparse for some $K = K_p \le C\log^{\alpha}(p)$ as $p \to \infty$, where $\alpha > 0$ is a constant. As a result (Frieze and Molloy, 1999),

$$|\mathcal{A}(m_0)| \le pm_0(eK_p)^{m_0} \le Cm_0 p \log^{m_0\alpha}(p). \tag{10}$$

We now discuss two parts separately.

In the GS-step, the computation cost comes from that of listing all elements in $\mathcal{A}(m_0)$, and that of screening all connected-subgraphs in $\mathcal{A}(m_0)$. Fix $1 \le k \le m_0$. By (10) and the fact that every size k ($k > 1$) connected subgraph at least contains one size $k-1$ connected subgraph, greedy algorithm can be used to list all sub-graphs with size k with computational complexity $\le Cp(K_pk)^k \le Cp\log^{k\alpha}(p)$, and screening all connected subgraphs of size k has computational complexity $\le Cnp\log^{k\alpha}(p)$. Therefore, the computational complexity of the GS-step $\le Cnp(\log(p))^{(m_0+1)\alpha}$.

The computation cost of the GC-step contains the part of breaking $\mathcal{U}_p^*$ into disconnected components, and that of cleaning each component by minimizing (9). As a well-known application of the breadth-first search (Hopcroft and Tarjan, 1973), the first part $\le |\mathcal{U}_p^*|(K_p+1)$. For the second part, by the SAS property of the GS-step (i.e., Lemma 16), for a broad class of design matrices, with the tuning parameters chosen properly, there is a *fixed* integer ℓ_0 such that with overwhelming probability, $|\mathcal{I}_0| \le \ell_0$ for any $\mathcal{I}_0 \lhd \mathcal{U}_p^*$. As a result, the

total computational cost of the GC-step is no greater than $C(2^{\ell_0}\log^{\alpha}(p))|\mathcal{U}_p^*|n$, which is moderate.

The computational complexity of GS is only moderately larger than that of Univariate Screening or UPS (Ji and Jin, 2011). UPS uses univariate thresholding for screening which has a computational complexity of $O(np)$, and GS implements multivariate screening for all connected subgraphs in $\mathcal{A}(m_0)$, which has a computational complexity $\leq Cnp(\log(p))^{(m_0+1)\alpha}$. The latter is only larger by a multi-$\log(p)$ term.

2.3 Asymptotic Rare and Weak Model and Random Design Model

To analyze GS, we consider the regression model $Y = X\beta + \sigma z$ as in (1), and use an Asymptotic Rare and Weak (ARW) model for β and a random design model for X.

We introduce the ARW first. Fix parameters $\epsilon \in (0,1)$, $\tau > 0$, and $a \geq 1$. Let $b = (b_1, \ldots, b_p)'$ be the $p \times 1$ random vector where

$$b_i \overset{iid}{\sim} \text{Bernoulli}(\epsilon). \tag{11}$$

We model the signal vector β in Model (1) by

$$\beta = b \circ \mu, \tag{12}$$

where "$\circ$" denotes the Hadamard product (see Section 1.5) and $\mu \in \Theta_p^*(\tau, a)$, with

$$\Theta_p^*(\tau, a) = \{\mu \in \Theta_p(\tau), \|\mu\|_\infty \leq a\tau\}, \qquad \Theta_p(\tau) = \{\mu \in \mathbb{R}^p : |\mu_i| \geq \tau, 1 \leq i \leq p\}. \tag{13}$$

In this model, ϵ calibrates the sparsity level and τ calibrates the minimum signal strength. We are primarily interested in the case where ϵ is small and τ is smaller than the required signal strength for the exact recovery of the support of β, so the signals are both rare and weak. The constraint of $\|\mu\|_\infty \leq a\tau_p$ is mainly for technical reasons (only needed for Lemma 16); see Section 2.6 for more discussions.

We let p be the driving asymptotic parameter, and tie (ϵ, τ) to p through some fixed parameters. In detail, fixing $0 < \vartheta < 1$, we model

$$\epsilon = \epsilon_p = p^{-\vartheta}. \tag{14}$$

For any fixed ϑ, the signals become increasingly sparser as $p \to \infty$. Also, as ϑ ranges, the sparsity level ranges from very dense to very sparse, and covers all interesting cases.

It turns out that the most interesting range for τ is $\tau = \tau_p = O(\sqrt{\log(p)})$. In fact, when $\tau_p \ll \sigma\sqrt{\log(p)}$, the signals are simply too rare and weak so that successful variable selection is impossible. On the other hand, exact support recovery requires $\tau \gtrsim \sigma\sqrt{2\log p}$ for orthogonal designs and possibly even larger τ for correlated designs. In light of this, we fix $r > 0$ and calibrate τ by

$$\tau = \tau_p = \sigma\sqrt{2r\log(p)}. \tag{15}$$

Next, consider the random design model. The use of random design model is mainly for simplicity in presentation. The main results in the paper can be translated to fixed design models with a careful modification of the notations; see Corollary 7 and Section 5.

For any positive definite matrix A, let $\lambda(A)$ be the smallest eigenvalue, and let

$$\lambda_k^*(\Omega) = \min\{\lambda(A) : A \text{ is a } k \times k \text{ principle submatrix of } \Omega\}. \tag{16}$$

For m_0 as in the GS-step, let $g = g(m_0, \vartheta, r)$ be the smallest integer such that

$$g \geq \max\{m_0, (\vartheta + r)^2/(2\vartheta r)\}. \tag{17}$$

Fixing a constant $c_0 > 0$, introduce

$$\mathcal{M}_p(c_0, g) = \{\Omega : p \times p \text{ correlation matrix, } \lambda_g^*(\Omega) \geq c_0\}. \tag{18}$$

Recall X_i is the i-th row of X; see (2). In the random design model, we fix an $\Omega \in \mathcal{M}(c_0, g)$ (Ω is unknown to us), and assume

$$X_i \overset{iid}{\sim} N(0, \frac{1}{n}\Omega), \qquad 1 \leq i \leq n. \tag{19}$$

In the literature, this is called the Gaussian design, which can be found in Compressive Sensing (Bajwa et al., 2007), Computer Security (Dinur and Nissim, 2003), and other application areas.

At the same time, fixing $\kappa \in (0, 1)$, we model the sample size n by

$$n = n_p = p^\kappa. \tag{20}$$

As $p \to \infty$, n_p becomes increasingly large but is still much smaller than p. We assume

$$\kappa > (1 - \vartheta), \tag{21}$$

so that $n_p \gg p\epsilon_p$. Note $p\epsilon_p$ is approximately the total number of signals. Condition (21) is almost necessary for successful variable selection (Donoho, 2006a,b).

Definition 4 *We call model (11)-(15) for β the Asymptotic Rare Weak model* $\mathrm{ARW}(\vartheta, r, a, \mu)$*, and call model (19)-(21) for X the Random Design model* $\mathrm{RD}(\vartheta, \kappa, \Omega)$*.*

2.4 Minimax Hamming Distance

In many works on variables selection, one assesses the optimality by the 'oracle property', where the probability of non-exact recovery $P(\mathrm{sgn}(\hat{\beta}) \neq \mathrm{sgn}(\beta))$ is the loss function. When signals are rare and weak, $P(\mathrm{sgn}(\hat{\beta}) \neq \mathrm{sgn}(\beta)) \approx 1$ and 'exact recovery' is usually impossible. A more appropriate loss function is the Hamming distance between $\mathrm{sgn}(\hat{\beta})$ and $\mathrm{sgn}(\beta)$.

For any fixed β and any variable selection procedure $\hat{\beta}$, we measure the performance by the Hamming distance:

$$h_p(\hat{\beta}, \beta | X) = E\Big[\sum_{j=1}^{p} 1\big\{\mathrm{sgn}(\hat{\beta}_j) \neq \mathrm{sgn}(\beta_j)\big\}\Big|X\Big].$$

In the Asymptotic Rare Weak model, $\beta = b \circ \mu$, and (ϵ_p, τ_p) depend on p through (ϑ, r), so the overall Hamming distance for $\hat{\beta}$ is

$$H_p(\hat{\beta}; \epsilon_p, n_p, \mu, \Omega) = E_{\epsilon_p} E_\Omega\big[h_p(\hat{\beta}, \beta|X)\big] \equiv E_{\epsilon_p} E_\Omega\big[h_p(\hat{\beta}, b \circ \mu|X)\big],$$

where E_{ϵ_p} is the expectation with respect to the law of b, and E_Ω is the expectation with respect to the law of X; see (11) and (19). Finally, the minimax Hamming distance is

$$\mathrm{Hamm}_p^*(\vartheta,\kappa,r,a,\Omega) = \inf_{\hat\beta}\sup_{\mu\in\Theta_p^*(\tau_p,a)}\big\{H_p(\hat\beta;\epsilon_p,n_p,\mu,\Omega)\big\}.$$

The Hamming distance is no smaller than the sum of the expected number of signal components that are misclassified as noise and the expected number of noise components that are misclassified as signal.

2.5 Lower Bound for the Minimax Hamming Distance, and GOLF

We first construct lower bounds for "local risk" at different j, $1\le j\le p$, and then aggregate them to construct a lower bound for the global risk. One challenge we face is the least favorable configurations for different j overlap with each other. We resolve this by exploiting the sparsity of a new graph to be introduced: Graph of Least Favorable (GOLF).

To recap, the model we consider is Model (1), where

$$\beta \text{ is modeled by } \mathrm{ARW}(\vartheta,r,a,\mu), \qquad \text{and} \qquad X \text{ is modeled by } \mathrm{RD}(\vartheta,\kappa,\Omega).$$

Fix $1\le j\le p$. The "local risk" at an index j is the risk of estimating the set of variables $\{\beta_k : d(k,j)\le g\}$, where g is defined in (17) and $d(j,k)$ denotes the geodesic distance between j and k in the graph $\mathcal{G}^{*,\delta}$. The goal is to construct two subsets V_0 and V_1 and two realizations of β, $\beta^{(0)}$ and $\beta^{(1)}$ such that $j\in V_0\cup V_1$ and

$$\text{If } k\notin V_0\cup V_1,\ \beta_k^{(0)}=\beta_k^{(1)};\ \text{otherwise, } \beta_k^{(i)}\neq 0 \text{ if and only if } k\in V_i,\ i=0,1,$$

where in the special case of $V_0=V_1$, we require $\mathrm{sgn}(\beta^{(0)})\neq\mathrm{sgn}(\beta^{(1)})$. In the literature, it is known that how well we can estimate $\{\beta_k : d(k,j)\le g\}$ depends on how well we can separate two hypotheses (where $\beta^{(0)}$ and $\beta^{(1)}$ are assumed as known):

$$H_0^{(j)}: Y = X\beta^{(0)}+\sigma z \qquad \text{vs.} \qquad H_1^{(j)}: Y = X\beta^{(1)}+\sigma z, \qquad z\sim N(0,I_n). \tag{22}$$

The least favorable configuration for the local risk at index j is the quadruple $(V_0,V_1,\beta^{(0)},\beta^{(1)})$ for which two hypotheses are the most difficult to separate.

For any $V\subset\{1,2,\ldots,p\}$, let I_V be the indicator vector of V such that for any $1\le k\le p$, the k-th coordinate of I_V is 1 if $k\in V$ and is 0 otherwise. Define

$$B_V = \{I_V\circ\mu :\ \mu\in\Theta_p^*(\tau_p,a)\},$$

where we recall "$\circ$" denotes the Hadamard product (see Section 1.5). Denote for short $\theta^{(i)} = I_{V_0\cup V_1}\circ\beta^{(i)}$, and so $\beta^{(1)}-\beta^{(0)}=\theta^{(1)}-\theta^{(0)}$ and $\theta^{(i)}\in B_{V_i}$, $i=0,1$. Introduce

$$\alpha(\theta^{(0)},\theta^{(1)}) = \alpha(\theta^{(0)},\theta^{(1)};V_0,V_1,\Omega,a) = \tau_p^{-2}(\theta^{(0)}-\theta^{(1)})'\Omega(\theta^{(0)}-\theta^{(1)}).$$

For the testing problem in (22), the optimal test is to reject $H_0^{(j)}$ if and only if $(\theta^{(1)}-\theta^{(0)})'X'(Y-X\beta^{(0)})\ge t\sigma\tau_p\sqrt{\alpha(\theta^{(0)},\theta^{(1)})}$ for some threshold $t>0$ to be determined. In the ARW and RD models, $P(\beta_k\neq 0,\forall k\in V_i)\sim\epsilon_p^{|V_i|}$, $i=0,1$, and $(\theta^{(0)}-\theta^{(1)})'G(\theta^{(0)}-\theta^{(1)})\approx$

$(\theta^{(0)}-\theta^{(1)})'\Omega(\theta^{(0)}-\theta^{(1)})$, since the support of $\theta^{(0)}-\theta^{(1)}$ is contained in a small-size set $V_0\cup V_1$. Therefore the sum of Type I and Type II error of any test associated with (22) is no smaller than (up to some negligible differences)

$$\epsilon_p^{|V_0|}\bar{\Phi}(t)+\epsilon_p^{|V_1|}\Phi\big(t-(\tau_p/\sigma)[\alpha(\theta^{(0)},\theta^{(1)})]^{1/2}\big), \tag{23}$$

where $\bar{\Phi}=1-\Phi$ is the survival function of $N(0,1)$.

For a lower bound for the "local risk" at j, we first optimize the quantity in (23) over all $\theta^{(0)}\in B_{V_0}$ and $\theta^{(1)}\in B_{V_1}$, and then optimize over all (V_0,V_1) subject to $j\in V_0\cup V_1$. To this end, define $\alpha^*(V_0,V_1)=\alpha^*(V_0,V_1;a,\Omega)$, $\eta(V_0,V_1)=\eta(V_0,V_1;\vartheta,r,a,\Omega)$, and $\rho_j^*=\rho_j^*(\vartheta,r,a,\Omega)$ by

$$\begin{aligned}\alpha^*(V_0,V_1)&=\min\big\{\alpha(\theta^{(0)},\theta^{(1)};V_0,V_1,\Omega,a):\ \theta^{(i)}\in B_{V_i},i=0,1,\mathrm{sgn}(\theta^{(0)})\neq\mathrm{sgn}(\theta^{(1)})\big\},\\ \eta(V_0,V_1)&=\max\{|V_0|,|V_1|\}\vartheta+\frac{1}{4}\left[\left(\sqrt{\alpha^*(V_0,V_1)r}-\frac{\big|(|V_1|-|V_0|)\big|\vartheta}{\sqrt{\alpha^*(V_0,V_1)r}}\right)_+\right]^2,\end{aligned} \tag{24}$$

and

$$\rho_j^*(\vartheta,r,a,\Omega)=\min_{\{(V_0,V_1):j\in V_1\cup V_0\}}\eta(V_0,V_1).$$

The following shorthand notation is frequently used in this paper, which stands for a generic multi-$\log(p)$ term that may vary from one occurrence to another.

Definition 5 *$L_p>0$ denotes a multi-$\log(p)$ term such that when $p\to\infty$, for any $\delta>0$, $L_pp^{\delta}\to\infty$ and $L_pp^{-\delta}\to 0$.*

By (23) and Mills' ratio (Wasserman and Roeder, 2009), a lower bound for the "local risk" at j is

$$\begin{aligned}&\sup_{\{(V_0,V_1):\ j\in V_0\cup V_1\}}\big\{\inf_t\big[\epsilon_p^{|V_0|}\bar{\Phi}(t)+\epsilon_p^{|V_1|}\Phi\big(t-(\tau_p/\sigma)[\alpha^*(V_0,V_1)]^{1/2}\big)\big]\big\}\\ =&\sup_{\{(V_0,V_1):\ j\in V_0\cup V_1\}}\big\{L_p\exp\big(-\eta(V_0,V_1)\cdot\log(p)\big)\big\}=L_p\exp(-\rho_j^*(\vartheta,r,a,\Omega)\log(p)).\end{aligned}$$

We now aggregate such lower bounds for "local risk" for a global lower bound. Since the "least favorable" configurations of (V_0,V_1) for different j may overlap with each other, we need to consider a graph as follows. Revisit the optimization problem in (24) and let

$$(V_{0j}^*,V_{1j}^*)=\mathrm{argmin}_{\{(V_0,V_1):j\in V_1\cup V_0\}}\eta(V_0,V_1;\vartheta,r,a,\Omega). \tag{25}$$

When there is a tie, pick the pair that appears first lexicographically. Therefore, for any $1\le j\le p$, $V_{0j}^*\cup V_{1j}^*$ is uniquely defined. In Lemma 22 of the appendix, we show that $|V_{0j}^*\cup V_{1j}^*|\le(\vartheta+r)^2/(2\vartheta r)$ for all $1\le j\le p$.

We now define a new graph, Graph of Least Favorable (GOLF), $\mathcal{G}^\diamond=(V,E)$, where $V=\{1,2,\ldots,p\}$ and there is an edge between j and k if and only if $(V_{0j}^*\cup V_{1j}^*)$ and $(V_{0k}^*\cup V_{1k}^*)$ have non-empty intersections. Denote the maximum degree of GOLF by $d_p(\mathcal{G}^\diamond)$.

Theorem 6 *Fix $(\vartheta,\kappa)\in(0,1)^2$, $r>0$, and $a\geq 1$ such that $\kappa>(1-\vartheta)$, and let $\mathcal{M}_p(c_0,g)$ be as in (18). Consider Model (1) where β is modeled by $ARW(\vartheta,r,a,\mu)$ and X is modeled by $RD(\vartheta,\kappa,\Omega)$ and $\Omega\in\mathcal{M}_p(c_0,g)$ for sufficiently large p. Then as $p\to\infty$, $\mathrm{Hamm}_p^*(\vartheta,\kappa,r,a,\Omega)\geq L_p\big[d_p(\mathcal{G}^\diamond)\big]^{-1}\sum_{j=1}^p p^{-\rho_j^*(\vartheta,r,a,\Omega)}$.*

A similar claim holds for deterministic design models; the proof is similar so we omit it.

Corollary 7 *For deterministic design models, the parallel lower bound holds for the minimax Hamming distance with Ω replaced by G in the calculation of $\rho_j^*(\vartheta,r,a,\Omega)$ and $d_p(\mathcal{G}^\diamond)$.*

Remark. The lower bounds contain a factor of $\big[d_p(\mathcal{G}^\diamond)\big]^{-1}$. In many cases including that considered in our main theorem (Theorem 8), this factor is a multi-$\log(p)$ term so it does not have a major effect. In some other cases, the factor $\big[d_p(\mathcal{G}^\diamond)\big]^{-1}$ could be much smaller, say, when the GOSD has one or a few hubs, the degrees of which grow algebraically fast as p grows. In these cases, the associated GOLF may (or may not) have large-degree hubs. As a result, the lower bounds we derive could be very conservative, and can be substantially improved if we treat the hubs, neighboring nodes of the hubs, and other nodes separately. For the sake of space, we leave such discussion to future work.

Remark. A similar lower bound holds if the condition $\mu\in\Theta_p^*(\tau_p,a)$ of ARW is replaced by $\mu\in\Theta_p(\tau_p)$. In (24), suppose we replace $\Theta_p^*(\tau_p,a)$ by $\Theta_p(\tau_p)$, and the minimum is achieved at $(\theta^{(0)},\theta^{(1)})=(\theta_*^{(0)}(V_0,V_1;\Omega),\theta_*^{(1)}(V_0,V_1;\Omega))$. Let $g=g(m_0,\vartheta,r)$ be as in (17) and define

$$a_g^*(\Omega)=\max_{\{(V_0,V_1):|V_0\cup V_1|\leq g\}}\{\|\theta_*^{(0)}(V_0,V_1;\Omega)\|_\infty,\|\theta_*^{(1)}(V_0,V_1;\Omega)\|_\infty\}.$$

By elementary calculus, it is seen that for $\Omega\in\mathcal{M}_p(c_0,g)$, there is a a constant $C=C(c_0,g)$ such that $a_g^*(\Omega)\leq C$. If additionally we assume

$$a>a_g^*(\Omega), \tag{26}$$

then $\alpha^*(V_0,V_1)=\alpha^*(V_0,V_1;\Omega,a)$, $\eta(V_0,V_1;\Omega,a,\vartheta,r)$, and $\rho_j^*(\vartheta,r,a,\Omega)$ do not depend on a. Especially, we can derive an alternative formula for $\rho_j^*(\vartheta,r,a,\Omega)$; see Lemma 18 for details.

When (26) holds, $\Theta_p^*(\tau_p,a)$ is broad enough in the sense that the least favorable configurations $(V_0,V_1,\beta^{(0)},\beta^{(1)})$ for all j satisfy $\|\beta^{(i)}\|_\infty\leq a\tau_p$, $i=0,1$. Consequently, neither the minimax rate nor GS needs to adapt to a. In Section 2.6, we assume (26) holds; (26) is a mild condition for it only involves small-size sub-matrices of Ω.

2.6 Upper Bound and Optimality of Graphlet Screening

Fix constants $\gamma\in(0,1)$ and $A>0$. Let $\mathcal{M}_p(c_0,g)$ be as in (18). In this section, we further restrict Ω to the following set:

$$\mathcal{M}_p^*(\gamma,c_0,g,A)=\Big\{\Omega\in\mathcal{M}_p(c_0,g):\ \sum_{j=1}^p|\Omega(i,j)|^\gamma\leq A,\ 1\leq i\leq p\Big\}.$$

Note that any $\Omega\in\mathcal{M}_p^*(\gamma,c_0,g,A)$ is sparse in the sense that each row of Ω has relatively few large coordinates. The sparsity of Ω implies the sparsity of the Gram matrix G, since small-size sub-matrices of G approximately equal to their counterparts of Ω.

In GS, when we regularize GOSD as in (4), we set the threshold δ by

$$\delta = \delta_p = 1/\log(p). \tag{27}$$

Such a choice for threshold is mainly for convenience, and can be replaced by any term that tends to 0 logarithmically fast as $p \to \infty$.

For any subsets D and F of $\{1, 2, \ldots, p\}$, define $\omega(D, F; \Omega) = \omega(D, F; \vartheta, r, a, \Omega, p)$ by

$$\omega(D, F; \Omega) = \min_{\xi \in \mathbb{R}^{|D|}, \min_{i \in D} |\xi_i| \geq 1} \Big\{ \xi' \big(\Omega^{D,D} - \Omega^{D,F} (\Omega^{F,F})^{-1} \Omega^{F,D} \big) \xi \Big\}, \tag{28}$$

Write $\omega = \omega(\hat{D}, \hat{F}; \Omega)$ for short. We choose the tuning parameters in the GS-step in a way such that

$$t(\hat{D}, \hat{F}) = 2\sigma^2 q(\hat{D}, \hat{F}) \log p, \tag{29}$$

where $q = q(\hat{D}, \hat{F}) > 0$ satisfies

$$\begin{cases} \sqrt{q_0} \leq \sqrt{q} \leq \sqrt{\omega r} - \sqrt{\frac{(\vartheta + \omega r)^2}{4\omega r} - \frac{|\hat{D}|+1}{2}}\vartheta, & |\hat{D}| \text{ is odd } \& \ \omega r/\vartheta > |\hat{D}| + (|\hat{D}|^2 - 1)^{1/2}, \\ \sqrt{q_0} \leq \sqrt{q} \leq \sqrt{\omega r} - \sqrt{\frac{1}{4}\omega r - \frac{1}{2}|\hat{D}|\vartheta}, & |\hat{D}| \text{ is even } \& \ \omega r/\vartheta \geq 2|\hat{D}|, \\ q \text{ is a constant such that } q \geq q_0, & \text{otherwise.} \end{cases} \tag{30}$$

We set the GC-step tuning parameters by

$$u^{gs} = \sigma\sqrt{2\vartheta \log p}, \qquad v^{gs} = \tau_p = \sigma\sqrt{2r \log p}. \tag{31}$$

The main theorem of this paper is the following theorem.

Theorem 8 *Fix $m_0 \geq 1$, $(\vartheta, \gamma, \kappa) \in (0,1)^3$, $r > 0$, $c_0 > 0$, $g > 0$, $a > 1$, $A > 0$ such that $\kappa > 1 - \vartheta$ and (17) is satisfied. Consider Model (1) where β is modeled by $ARW(\vartheta, r, a, \mu)$, X is modeled by $RD(\vartheta, \kappa, \Omega)$, and where $\Omega \in \mathcal{M}_p^*(\gamma, c_0, g, A)$ and $a > a_g^*(\Omega)$ for sufficiently large p. Let $\hat{\beta}^{gs} = \hat{\beta}^{gs}(Y; \delta, \mathcal{Q}, u^{gs}, v^{gs}, X, p, n)$ be the Graphlet Screening procedure defined as in Section 2.1, where the tuning parameters $(\delta, \mathcal{Q}, u^{gs}, v^{gs})$ are set as in (27)-(31). Then as $p \to \infty$, $\sup_{\mu \in \Theta_p^*(\tau_p, a)} H_p(\hat{\beta}^{gs}; \epsilon_p, n_p, \mu, \Omega) \leq L_p \Big[p^{1-(m_0+1)\vartheta} + \sum_{j=1}^p p^{-\rho_j^*(\vartheta, r, a, \Omega)} \Big] + o(1)$.*

Note that $\rho_j^* = \rho_j^*(\vartheta, r, a, \Omega)$ does not depend on a. Also, note that in the most interesting range, $\sum_{j=1}^p p^{-\rho_j^*} \gg 1$. So if we choose m_0 properly large, e.g., $(m_0 + 1)\vartheta > 1$, then $\sup_{\mu \in \Theta_p^*(\tau_p, a)} H_p(\hat{\beta}^{gs}; \epsilon_p, n_p, \mu, \Omega) \leq L_p \sum_{j=1}^p p^{-\rho_j^*(\vartheta, r, a, \Omega)}$. Together with Theorem 6, this says that GS achieves the optimal rate of convergence, adaptively to all Ω in $\mathcal{M}_p^*(\gamma, c_0, g, A)$ and $\beta \in \Theta_p^*(\tau_p, a)$. We call this property *optimal adaptivity*. Note that since the diagonals of Ω are scaled to 1 approximately, $\kappa \equiv \log(n_p)/\log(p)$ does not have a major influence over the convergence rate, as long as (21) holds.

Remark. Theorem 8 addresses the case where (26) holds so $a > a_g^*(\Omega)$. We now briefly discuss the case where $a < a_g^*(\Omega)$. In this case, the set $\Theta_p^*(\tau_p, a)$ becomes sufficiently narrow and a starts to have some influence over the optimal rate of convergence, at least for some choices of (ϑ, r). To reflect the role of a, we modify GS as follows: (a) in the GC-step (9), limit ξ to the class where either $\xi_i = 0$ or $\tau_p \leq |\xi_i| \leq a\tau_p$, and (b) in the

GS-step, replacing the χ^2-screening by the likelihood based screening procedure; that is, when we screen $\mathcal{I}_0 = \hat{D} \cup \hat{F}$, we accept nodes in $\hat{D}$ only when $h(\hat{F}) > h(\mathcal{I}_0)$, where for any subset $D \subset \{1, 2, \ldots, p\}$, $h(D) = \min\{\frac{1}{2}\|P^D(Y - X^{\otimes,D}\xi)\|^2 + \vartheta\sigma^2\log(p)|D|\}$, where the minimum is computed over all $|D| \times 1$ vectors ξ whose nonzero elements all have magnitudes in $[\tau_p, a\tau_p]$. From a practical point of view, this modified procedure depends more on the underlying parameters and is harder to implement than is GS. However, this is the price we need to pay when a is small. Since we are primarily interested in the case of relatively larger a where $a > a_g^*(\Omega)$ holds, we skip further discussion along this line.

2.7 Phase Diagram and Examples Where $\rho_j^*(\vartheta, r, a, \Omega)$ Have Simple Forms

In general, the exponents $\rho_j^*(\vartheta, r, a, \Omega)$ may depend on Ω in a complicated way. Still, from time to time, one may want to find a simple expression for $\rho_j^*(\vartheta, r, a, \Omega)$. It turns out that in a wide class of situations, simple forms for $\rho_j^*(\vartheta, r, a, \Omega)$ are possible. The surprise is that, in many examples, $\rho_j^*(\vartheta, r, a, \Omega)$ depends more on the trade-off between the parameters ϑ and r (calibrating the signal sparsity and signal strength, respectively), rather than on the large coordinates of Ω.

We begin with the following theorem, which is proved in Ji and Jin (2011, Theorem 1.1).

Theorem 9 *Fix $(\vartheta, \kappa) \in (0, 1)$, $r > 0$, and $a > 1$ such that $\kappa > (1-\vartheta)$. Consider Model (1) where β is modeled by $ARW(\vartheta, r, a, \mu)$ and X is modeled by $RD(\vartheta, \kappa, \Omega)$. Then as $p \to \infty$,*

$$\frac{\mathrm{Hamm}_p^*(\vartheta, \kappa, r, a, \Omega)}{p^{1-\vartheta}} \gtrsim \begin{cases} 1, & 0 < r < \vartheta, \\ L_p p^{-(r-\vartheta)^2/(4r)}, & r > \vartheta. \end{cases}$$

Note that $p^{1-\vartheta}$ is approximately the number of signals. Therefore, when $r < \vartheta$, the number of selection errors can not get substantially smaller than the number of signals. This is the most difficult case where no variable selection method can be successful.

In this section, we focus on the case $r > \vartheta$, so that successful variable selection is possible. In this case, Theorem 9 says that a *universal* lower bound for the Hamming distance is $L_p p^{1-(\vartheta+r)^2/(4r)}$. An interesting question is, to what extend, this lower bound is tight.

Recall that $\lambda_k^*(\Omega)$ denotes the minimum of smallest eigenvalues across all $k \times k$ principle submatrices of Ω, as defined in (16). The following corollaries are proved in Section 6.

Corollary 10 *Suppose the conditions of Theorem 8 hold, and that additionally, $1 < r/\vartheta < 3 + 2\sqrt{2} \approx 5.828$, and $|\Omega(i,j)| \le 4\sqrt{2} - 5 \approx 0.6569$ for all $1 \le i, j \le p$, $i \ne j$. Then as $p \to \infty$, $\mathrm{Hamm}_p^*(\vartheta, \kappa, r, a, \Omega) = L_p p^{1-(\vartheta+r)^2/(4r)}$.*

Corollary 11 *Suppose the conditions of Theorem 8 hold. Also, suppose that $1 < r/\vartheta < 5 + 2\sqrt{6} \approx 9.898$, and that $\lambda_3^*(\Omega) \ge 2(5 - 2\sqrt{6}) \approx 0.2021$, $\lambda_4^*(\Omega) \ge 5 - 2\sqrt{6} \approx 0.1011$, and $|\Omega(i,j)| \le 8\sqrt{6} - 19 \approx 0.5959$ for all $1 \le i, j \le p$, $i \ne j$. Then as $p \to \infty$, $\mathrm{Hamm}_p^*(\vartheta, \kappa, r, a, \Omega) = L_p p^{1-(\vartheta+r)^2/(4r)}$.*

In these corollaries, the conditions on Ω are rather relaxed. Somewhat surprisingly, the off-diagonals of Ω do not necessarily have a major influence on the optimal rate of convergence, as one might have expected.

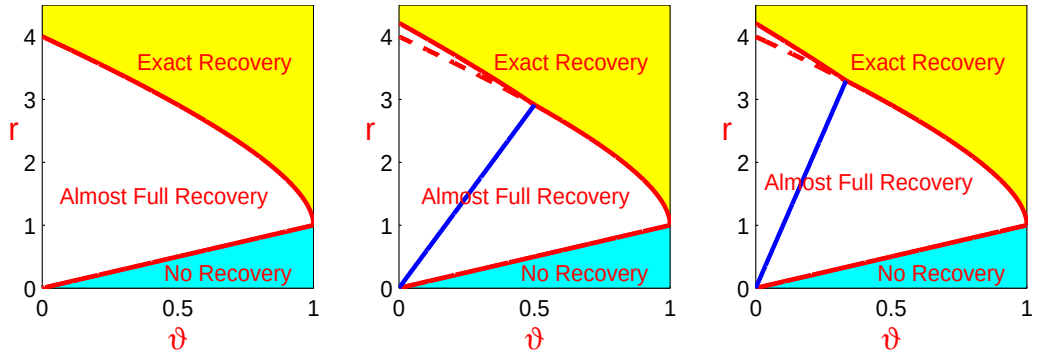


Figure 1: Phase diagram for $\Omega = I_p$ (left), for Ω satisfying conditions of Corollary 10 (middle), and for Ω satisfying conditions of Corollary 11 (right). Red line: $r = \vartheta$. Solid red curve: $r = \rho(\vartheta, \Omega)$. In each of the last two panels, the blue line intersects with the red curve at $(\vartheta, r) = (1/2, [3 + 2\sqrt{2}]/2)$ (middle) and $(\vartheta, r) = (1/3, [5 + 2\sqrt{6}]/3)$ (right), which splits the red solid curve into two parts; the part to the left is illustrative for it depends on Ω in a complicated way; the part to the right, together with the dashed red curve, represent $r = (1+\sqrt{1-\vartheta})^2$ (in the left panel, this is illustrated by the red curve).

Note also that by Theorem 8, under the condition of either Corollaries 10 or Corollary 11, GS achieves the optimal rate in that

$$\sup_{\mu \in \Theta_p^*(\tau_p, a)} H_p(\hat{\beta}^{gs}; \epsilon_p, n_p, \mu, \Omega) \leq L_p p^{1-(\vartheta+r)^2/(4r)}. \tag{32}$$

Together, Theorem 9, Corollaries 10-11, and (32) have an interesting implication on the so-called *phase diagram*. Call the two-dimensional *parameter space* $\{(\vartheta, r) : 0 < \vartheta < 1, r > 0\}$ the phase space. There are two curves $r = \vartheta$ and $r = \rho(\vartheta, \Omega)$ (the latter can be thought of as the solution of $\sum_{j=1}^p p^{-\rho_j^*(\vartheta, r, a, \Omega)} = 1$; recall that $\rho_j^*(\vartheta, r, a, \Omega)$ does not depend on a) that partition the whole phase space into three different regions:

- *Region of No Recovery.* $\{(\vartheta, r) : 0 < r < \vartheta, 0 < \vartheta < 1\}$. In this region, as $p \to \infty$, for any Ω and any procedures, the minimax Hamming error equals approximately to the total expected number of signals. This is the most difficult region, in which no procedure can be successful in the minimax sense.

- *Region of Almost Full Recovery.* $\{(\vartheta, r) : \vartheta < r < \rho(\vartheta, \Omega)\}$. In this region, as $p \to \infty$, the minimax Hamming distance satisfies $1 \ll \mathrm{Hamm}_p^*(\vartheta, \kappa, r, a, \Omega) \ll p^{1-\vartheta}$, and it is possible to recover most of the signals, but it is impossible to recover all of them.

- *Region of Exact Recovery.* In this region, as $p \to \infty$, the minimax Hamming distance $\mathrm{Hamm}_p^*(\vartheta, \kappa, r, a, \Omega) = o(1)$, and it is possible to exactly recover all signals with overwhelming probability.

In general, the function $\rho(\vartheta, \Omega)$ depends on Ω in a complicated way. However, by Theorem 9 and Corollaries 10-11, we have the following conclusions. First, for all Ω and $a > 1$, $\rho(\vartheta, \Omega) \geq (1 + \sqrt{1-\vartheta})^2$ for all $0 < \vartheta < 1$. Second, in the simplest case where $\Omega = I_p$, $\mathrm{Hamm}_p^*(\vartheta, \kappa, r, a, \Omega) = L_p p^{1-(\vartheta+r)^2/(4r)}$, and $\rho(\vartheta, \Omega) = (1 + \sqrt{1-\vartheta})^2$ for all $0 < \vartheta < 1$. Third, under the conditions of Corollary 10, $\rho(\vartheta, \Omega) = (1 + \sqrt{1-\vartheta})^2$ if $1/2 < \vartheta < 1$. Last, under the conditions of Corollary 11, $\rho(\vartheta, \Omega) = (1 + \sqrt{1-\vartheta})^2$ if $1/3 < \vartheta < 1$. The phase diagram for the last three cases are illustrated in Figure 1. The blue lines are $r/\vartheta = 3+2\sqrt{2}$ (middle) and $r/\vartheta = 5 + 2\sqrt{6}$ (right).

Corollaries 10-11 can be extended to more general situations, where r/ϑ may get arbitrary large, but consequently, we need stronger conditions on Ω. Towards this end, we note that for any (ϑ, r) such that $r > \vartheta$, we can find a unique integer $N = N(\vartheta, r)$ such that $2N - 1 \leq (\vartheta/r + r/\vartheta)/2 < 2N + 1$. Suppose that for any $2 \leq k \leq 2N - 1$,

$$\lambda_k^*(\Omega) \geq \max_{\{(k+1)/2 \leq j \leq \min\{k, N\}\}} \Big\{ \frac{(r/\vartheta + \vartheta/r)/2 - 2j + 2 + \sqrt{[(r/\vartheta + \vartheta/r)/2 - 2j + 2]^2 - 1}}{(2k - 2j + 1)(r/\vartheta)} \Big\}, \tag{33}$$

and that for any $2 \leq k \leq 2N$,

$$\lambda_k^*(\Omega) \geq \max_{\{k/2 \leq j \leq \min\{k-1, N\}\}} \Big\{ \frac{(r/\vartheta + \vartheta/r)/2 + 1 - 2j}{(k - j)(r/\vartheta)} \Big\}. \tag{34}$$

Then we have the following corollary.

Corollary 12 *Suppose the conditions in Theorem 8 and that in (33)-(34) hold. Then as* $p \to \infty$, $\mathrm{Hamm}_p^*(\vartheta, \kappa, r, a, \Omega) = L_p p^{1-(\vartheta+r)^2/(4r)}$.

The right hand sides of (33)-(34) decrease with (r/ϑ). For a constant $s_0 > 1$, (33)-(34) hold for all $1 < r/\vartheta \leq s_0$ as long as they hold for $r/\vartheta = s_0$. Hence Corollary 12 implies a similar partition of the phase diagram as do Corollaries 10-11.

Remark. Phase diagram can be viewed as a new criterion for assessing the optimality, which is especially appropriate for rare and weak signals. The phase diagram is a partition of the phase space $\{(\vartheta, r) : 0 < \vartheta < 1, r > 0\}$ into different regions where statistical inferences are distinctly different. In general, a phase diagram has the following four regions:

- An "exact recovery" region corresponding to the "rare and strong" regime in which high probability of completely correct variable selection is feasible.
- An "almost full recovery" region as a part of the "rare and weak" regime in which completely correct variable selection is not achievable with high probability but variable selection is still feasible in the sense that with high probability, the number of incorrectly selected variables is a small fraction of the total number of signals.
- A "detectable" region in which variable selection is infeasible but the detection of the existence of a signal (somewhere) is feasible (e.g., by the Higher Criticism method).
- An "undetectable" region where signals are so rare and weak that nothing can be sensibly done.

In the sparse signal detection (Donoho and Jin, 2004) and classification (Jin, 2009) problems, the main interest is to find the detectable region, so that the exact recovery and almost full recovery regions were lumped into a single "estimable" region (e.g., Donoho and Jin, 2004, Figure 1). For variable selection, the main interest is to find the boundaries of the almost full discovery region so that the detectable and non-detectable regions are lumped into a single "no recovery" region as in Ji and Jin (2011) and Figure 1 of this paper.

Variable selection in the "almost full recovery" region is a new and challenging problem. It was studied in Ji and Jin (2011) when the effect of signal cancellation is negligible, but the hardest part of the problem was unsolved in Ji and Jin (2011). This paper (the second in this area) deals with the important issue of signal cancellation, in hopes of gaining a much deeper insight on variable selection in much broader context.

2.8 Non-optimality of Subset Selection and the Lasso

Subset selection (also called the L^0-penalization method) is a well-known method for variable selection, which selects variables by minimizing the following functional:

$$\frac{1}{2}\|Y - X\beta\|^2 + \frac{1}{2}(\lambda_{ss})^2\|\beta\|_0, \tag{35}$$

where $\|\beta\|_q$ denotes the L^q-norm, $q \geq 0$, and $\lambda_{ss} > 0$ is a tuning parameter. The AIC, BIC, and RIC are methods of this type (Akaike, 1974; Schwarz, 1978; Foster and George , 1994). Subset selection is believed to have good "theoretic property", but the main drawback of this method is that it is computationally NP hard. To overcome the computational challenge, many *relaxation* methods are proposed, including but are not limited to the lasso (Chen et al., 1998; Tibshirani, 1996), SCAD (Fan and Li, 2001), MC+ (Zhang, 2010), and Dantzig selector (Candes and Tao, 2007). Take the lasso for example. The method selects variables by minimizing

$$\frac{1}{2}\|Y - X\beta\|^2 + \lambda_{lasso}\|\beta\|_1, \tag{36}$$

where the L^0-penalization is replaced by the L^1-penalization, so the functional is convex and the optimization problem is solvable in polynomial time under proper conditions.

Somewhat surprisingly, subset selection is generally *rate non-optimal* in terms of selection errors. This sub-optimality of subset selection is due to its lack of flexibility in adapting to the "local" graphic structure of the design variables. Similarly, other global relaxation methods are sub-optimal as well, as the subset selection is the "idol" these methods try to mimic. To save space, we only discuss subset selection and the lasso, but a similar conclusion can be drawn for SCAD, MC+, and Dantzig selector.

For mathematical simplicity, we illustrate the point with an idealized regression model where the Gram matrix $G = X'X$ is diagonal block-wise and has 2×2 blocks

$$G(i,j) = 1\{i = j\} + h_0 \cdot 1\{|j - i| = 1,\ max(i,j) \text{ is even}\},\ \ |h_0| < 1,\ 1 \leq i, j \leq p. \tag{37}$$

Using an idealized model is mostly for technical convenience, but the non-optimality of subset selection or the lasso holds much more broadly than what is considered here. On the other hand, using a simple model is sufficient here: if a procedure is non-optimal in an idealized case, we can not expect it to be optimal in a more general context.

At the same time, we continue to model β with the Asymptotic Rare and Weak model $\mathrm{ARW}(\vartheta, r, a, \mu)$, but where we relax the assumption of $\mu \in \Theta_p^*(\tau_p, a)$ to that of $\mu \in \Theta_p(\tau_p)$ so that the strength of each signal $\geq \tau_p$ (but there is no upper bound on the strength). Consider a variable selection procedure $\hat{\beta}^\star$, where $\star = gs, ss, lasso$, representing GS, subset selection, and the lasso and the tuning parameters for each method are ideally set. Note that for the worst-case risk considered below, the ideal tuning parameters depend on (ϑ, r, p, h_0) but do not depend on μ. Since the index groups $\{2j-1, 2j\}$ are exchangeable in (37) and the ARW models, the Hamming error of $\beta^\star$ in its worst case scenario has the form of $\sup_{\{\mu \in \Theta_p(\tau_p)\}} H_p(\hat{\beta}^\star; \epsilon_p, \mu, G) = L_p p^{1-\rho_\star(\vartheta, r, h_0)}$.

We now study $\rho_\star(\vartheta, r, h_0)$. Towards this end, we first introduce $\rho^{(3)}_{lasso}(\vartheta, r, h_0) = \big\{(2|h_0|)^{-1}[(1-h_0^2)\sqrt{r} - \sqrt{(1-h_0^2)(1-|h_0|)^2 r - 4|h_0|(1-|h_0|)\vartheta}]\big\}^2$ and $\rho^{(4)}_{lasso}(\vartheta, r, h_0) = \vartheta + \frac{(1-|h_0|)^3(1+|h_0|)}{16h_0^2}\big[(1+|h_0|)\sqrt{r} - \sqrt{(1-|h_0|)^2 r - 4|h_0|\vartheta/(1-h_0^2)}\big]^2$. We then let

$$\rho^{(1)}_{ss}(\vartheta, r, h_0) = \begin{cases} 2\vartheta, & r/\vartheta \leq 2/(1-h_0^2) \\ [2\vartheta + (1-h_0^2)r]^2/[4(1-h_0^2)r], & r/\vartheta > 2/(1-h_0^2) \end{cases},$$

$$\rho^{(2)}_{ss}(\vartheta, r, h_0) = \begin{cases} 2\vartheta, & r/\vartheta \leq 2/(1-|h_0|) \\ 2[\sqrt{2(1-|h_0|)r} - \sqrt{(1-|h_0|)r - \vartheta}]^2, & r/\vartheta > 2/(1-|h_0|) \end{cases},$$

$$\rho^{(1)}_{lasso}(\vartheta, r, h_0) = \begin{cases} 2\vartheta, & r/\vartheta \leq 2/(1-|h_0|)^2 \\ \rho^{(3)}_{lasso}(\vartheta, r, h_0), & r/\vartheta > 2/(1-|h_0|)^2 \end{cases},$$

and

$$\rho^{(2)}_{lasso}(\vartheta, r, h_0) = \begin{cases} 2\vartheta, & r/\vartheta \leq (1+|h_0|)/(1-|h_0|)^3 \\ \rho^{(4)}_{lasso}(\vartheta, r, h_0), & r/\vartheta > (1+|h_0|)/(1-|h_0|)^3 \end{cases}.$$

The following theorem is proved in Section 6.

Theorem 13 *Fix $\vartheta \in (0,1)$ and $r > 0$ such that $r > \vartheta$. Consider Model (1) where β is modeled by $ARW(\vartheta, r, a, \mu)$ and X satisfies (37). For GS, we set the tuning parameters $(\delta, m_0) = (0, 2)$, and set $(\mathcal{Q}, u^{gs}, v^{gs})$ as in (29)-(31). For subset selection as in (35) and the lasso as in (36), we set their tuning parameters ideally given that (ϑ, r) are known. Then as $p \to \infty$,*

$$\rho_{gs}(\vartheta, r, h_0) = \min\Big\{\frac{(\vartheta + r)^2}{4r},\ \vartheta + \frac{(1-|h_0|)}{2}r,\ 2\vartheta + \frac{\{[(1-h_0^2)r - \vartheta]_+\}^2}{4(1-h_0^2)r}\Big\}, \tag{38}$$

$$\rho_{ss}(\vartheta, r, h_0) = \min\Big\{\frac{(\vartheta + r)^2}{4r},\ \vartheta + \frac{(1-|h_0|)}{2}r,\ \rho^{(1)}_{ss}(\vartheta, r, h_0), \rho^{(2)}_{ss}(\vartheta, r, h_0)\Big\}, \tag{39}$$

and

$$\rho_{lasso}(\vartheta, r, h_0) = \min\Big\{\frac{(\vartheta + r)^2}{4r},\ \vartheta + \frac{(1-|h_0|)r}{2(1+\sqrt{1-h_0^2})},\ \rho^{(1)}_{lasso}(\vartheta, r, h_0),\ \rho^{(2)}_{lasso}(\vartheta, r, h_0)\Big\}. \tag{40}$$

It can be shown that $\rho_{gs}(\vartheta, r, h_0) \geq \rho_{ss}(\vartheta, r, h_0) \geq \rho_{lasso}(\vartheta, r, h_0)$, where depending on the choices of (ϑ, r, h_0), we may have equality or strict inequality (note that a larger exponent means a better error rate). This fits well with our expectation, where as far as the convergence rate is concerned, GS is optimal for all (ϑ, r, h_0), so it outperforms the subset selection, which in turn outperforms the lasso. Table 2 summarizes the exponents for some representative (ϑ, r, h_0). It is seen that differences between these exponents become increasingly prominent when h_0 increase and ϑ decrease.

$\vartheta/r/h_0$	.1/11/.8	.3/9/.8	.5/4/.8	.1/4/.4	.3/4/.4	.5/4/.4	.1/3/.2	.3/3/.2
$\star = gs$	1.1406	1.2000	0.9000	0.9907	1.1556	1.2656	0.8008	0.9075
$\star = ss$	0.8409	0.9047	0.9000	0.9093	1.1003	1.2655	0.8007	0.9075
$\star = lasso$	0.2000	0.6000	0.7500	0.4342	0.7121	1.0218	0.6021	0.8919

Table 2: The exponents $\rho_\star(\vartheta, r, h_0)$ in Theorem 13, where $\star = gs, ss, lasso$.

As in Section 2.7, each of these methods has a phase diagram plotted in Figure 2, where the phase space partitions into three regions: *Region of Exact Recovery*, *Region of Almost Full Recovery*, and *Region of No Recovery*. Interestingly, the separating boundary for the last two regions are the same for three methods, which is the line $r = \vartheta$. The boundary that separates the first two regions, however, vary significantly for different methods. For any $h_0 \in (-1, 1)$ and $\star = gs, ss, lasso$, the equation for this boundary can be obtained by setting $\rho_\star(\vartheta, r, h_0) = 1$ (the calculations are elementary so we omit them). Note that the lower the boundary is, the better the method is, and that the boundary corresponding to the lasso is discontinuous at $\vartheta = 1/2$. In the non-optimal region of either subset selection or the lasso, the Hamming errors of the procedure are much smaller than $p\epsilon_p$, so the procedure gives "almost full recovery"; however, the rate of Hamming errors is slower than that of the optimal procedure, so subset selection or the lasso is non-optimal in such regions.

Subset selection and the lasso are rate non-optimal for they are so-called *one-step* or *non-adaptive* methods (Ji and Jin, 2011), which use only one tuning parameter, and which do not adapt to the local graphic structure. The non-optimality can be best illustrated with the diagonal block-wise model presented here, where each block is a 2×2 matrix. Correspondingly, we can partition the vector β into many size 2 blocks, each of which is of the following three types (i) those have no signal, (ii) those have exactly one signal, and (iii) those have two signals. Take the subset selection for example. To best separate (i) from (ii), we need to set the tuning parameter ideally. But such a tuning parameter may not be the "best" for separating (i) from (iii). This explains the non-optimality of subset selection.

Seemingly, more complicated penalization methods that use multiple tuning parameters may have better performance than the subset selection and the lasso. However, it remains open how to design such extensions to achieve the optimal rate for general cases. To save space, we leave the study along this line to the future.

2.9 Summary

We propose GS as a new approach to variable selection. The key methodological innovation is to use the GOSD to guide the multivariate screening. While a brute-force m-variate

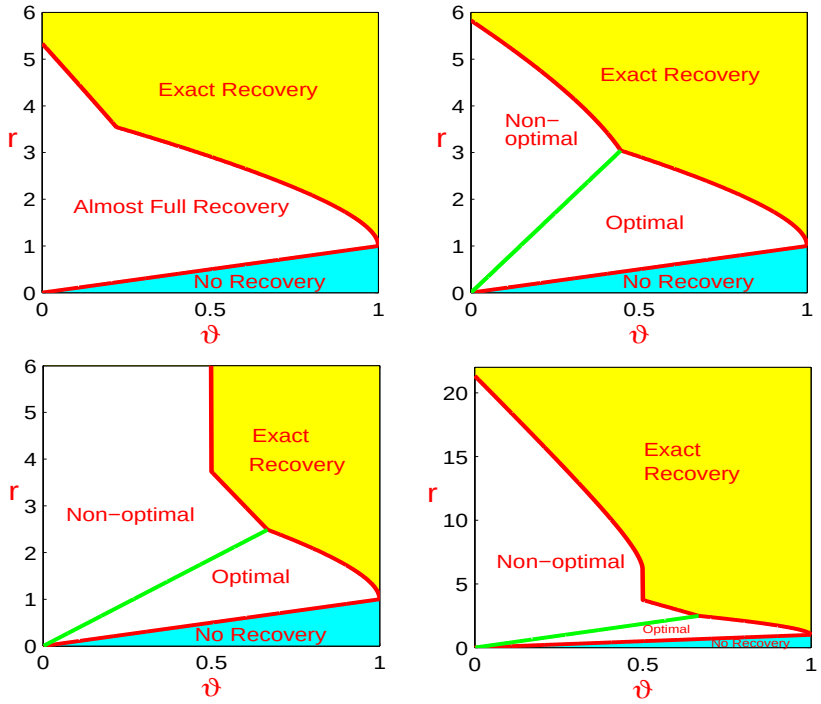


Figure 2: Phase diagrams for GS (top left), subset selection (top right), and the lasso (bottom; zoom-in on the left and zoom-out on the right), where $h_0 = 0.5$.

screening has a computation cost of $O(p^m + np)$, GS only has a computation cost of $L_p np$ (excluding the overhead of obtaining the GOSD), by utilizing graph sparsity. Note that when the design matrix G is approximately banded, say, all its large entries are confined to a diagonal band with bandwidth $\leq K$, the overhead of GS can be reduced to $O(npK)$. One such example is in Genome-Wide Association Study (GWAS), where G is the empirical Linkage Disequilibrium (LD) matrix, and K can be as small as a few tens. We remark that the lasso has a computational complexity of $O(npk)$, where k, dominated by the number steps requiring re-evaluation of the correlation between design vectors and updated residuals, could be smaller than the L_p term for GS (Wang et al., 2013).

We use *asymptotic minimaxity of the Hamming distance* as the criterion for assessing optimality. Compared with existing literature on variable selection where we use the *oracle property* or *probability of exact support recovery* to assess optimality, our approach is mathematically more demanding, yet scientifically more relevant in the rare/weak paradigm.

We have proved that GS achieves the optimal rate of convergence of Hamming errors, especially when signals are rare and weak, provided that the Gram matrix is sparse. Subset selection and the lasso are not rate optimal, even with very simple Gram matrix G and

even when the tuning parameters are ideally set. The sub-optimality of these methods is due to that they do not take advantage of the 'local' graphical structure as GS does.

GS has three key tuning parameters: q for the threshold level $t(\hat{D}, \hat{F}) = 2\sigma^2 q \log p$ in the GS-step, and $(u^{gs}, v^{gs}) = (\sigma\sqrt{2\vartheta \log p}, \sigma\sqrt{2r \log p})$ in the GC-step. While the choice of q is reasonably flexible and a sufficiently small fixed $q > 0$ is usually adequate, the choice of u^{gs} and v^{gs} are more directly tied to the signal sparsity and signal strength. Adaptive choice of these tuning parameters is a challenging direction of further research. One of our ideas to be developed in this direction is a subsampling scheme similar to the Stability Selection (Meinsausen and Buhlmann, 2010). On the other hand, as shown in our numeric results in Section 4, the performance of GS is relatively insensitive to mis-specification of (ϵ_p, τ_p); see details therein.

3. Properties of Graphlet Screening, Proof of Theorem 8

GS attributes the success to two important properties: the Sure Screening property and the *Separable After Screening* (SAS) property.

The Sure Screening property means that in the m_0-stage χ^2 screening, by picking an appropriate threshold, the set $\mathcal{U}_p^*$ (which is the set of retained indices after the GS-step) contains all but a small fraction of true signals. Asymptotically, this fraction is comparably smaller than the minimax Hamming errors, and so negligible. The SAS property means that except for a negligible probability, as a subgraph of the GOSD, $\mathcal{U}_p^*$ decomposes into many disconnected components of the GOSD, where the size of each component does not exceed a fixed integer. These two properties ensure that the original regression problem reduces to many small-size regression problems, and thus pave the way for the GC-step.

Below, we explain these ideas in detail, and conclude the section by the proof of Theorem 8. Since the only place we need the knowledge of σ is in setting the tuning parameters, so without loss of generality, we assume $\sigma = 1$ throughout this section.

First, we discuss the GS-step. For short, write $\hat{\beta} = \hat{\beta}^{gs}(Y; \delta, \mathcal{Q}, u^{gs}, v^{gs}, X, p, n)$ throughout this section. We first discuss the computation cost of the GS-step. As in Theorem 8, we take the threshold δ in $\mathcal{G}^{*,\delta}$ to be $\delta = \delta_p = 1/\log(p)$. The proof of the following lemma is similar to that of Ji and Jin (2011, Lemma 2.2), so we omit it.

Lemma 14 *Suppose the conditions of Theorem 8 hold, where we recall $\delta = 1/\log(p)$, and $\Omega^{*,\delta}$ is defined as in (4). As $p \to \infty$, with probability $1 - o(1/p^2)$, $\|\Omega - \Omega^{*,\delta}\|_\infty \leq C(\log(p))^{-(1-\gamma)}$, and $\mathcal{G}^{*,\delta}$ is K-sparse, where $K \leq C(\log(p))^{1/\gamma}$.*

Combining Lemma 14 and Frieze and Molloy (1999), it follows that with probability $1 - o(1/p^2)$, $\mathcal{G}^{*,\delta}$ has at most $p(Ce(\log(p))^{1/\gamma})^{m_0}$ connected subgraphs of size $\leq m_0$. Note that the second factor is at most logarithmically large, so the computation cost in the GS-step is at most $L_p p$ flops.

Consider the performance of the GS-step. The goal of this step is two-fold: on one hand, it tries to retain as many signals as possible during the screening; on the other hand, it tries to minimize the computation cost of the GC-step by controlling the maximum size of all components of $\mathcal{U}_p^*$. The key in the GS-step is to set the collection of thresholds $\mathcal{Q}$. The tradeoff is that, setting the thresholds too high may miss too many signals during the screening, and setting the threshold too low may increase the maximum size of the

components in $\mathcal{U}_p^*$, and so increase the computational burden of the GC-step. The following lemma characterizes the Sure Screening property of GS, and is proved in Section 6.

Lemma 15 *(Sure Screening). Suppose the settings and conditions are as in Theorem 8. In the m_0-stage χ^2 screening of the GS-step, if we set the thresholds $t(\hat{D}, \hat{F})$ as in (29), then as $p \to \infty$, for any $\Omega \in \mathcal{M}_p^*(\gamma, c_0, g, A)$, $\sum_{j=1}^p P(\beta_j \neq 0, j \notin \mathcal{U}_p^*) \leq L_p[p^{1-(m_0+1)\vartheta} + \sum_{j=1}^p p^{-\rho_j^*(\vartheta,r,a,\Omega)}] + o(1)$.*

Next, we formally state the SAS property. Viewing it as a subgraph of $\mathcal{G}^{*,\delta}$, $\mathcal{U}_p^*$ decomposes into many disconnected components $\mathcal{I}^{(k)}$, $1 \leq k \leq N$, where N is an integer that may depend on the data.

Lemma 16 *(SAS). Suppose the settings and conditions are as in Theorem 8. In the m_0-stage χ^2 screening in the GS-step, suppose we set the thresholds $t(\hat{D}, \hat{F})$ as in (29) such that $q(\hat{D}, \hat{F}) \geq q_0$ for some constant $q_0 = q_0(\vartheta, r) > 0$. As $p \to \infty$, under the conditions of Theorem 8, for any $\Omega \in \mathcal{M}_p^*(\gamma, c_0, g, A)$, there is a constant $\ell_0 = \ell_0(\vartheta, r, \kappa, \gamma, A, c_0, g) > 0$ such that with probability at least $1 - o(1/p)$, $|\mathcal{I}^{(k)}| \leq \ell_0$, $1 \leq k \leq N$.*

We remark that a more convenient way of picking q is to let

$$\begin{cases} q_0 \leq q \leq (\frac{\omega r + \vartheta}{2\omega r})^2 \omega r, & |\hat{D}| \text{ is odd } \& \ \omega r/\vartheta > |\hat{D}| + (|\hat{D}|^2 - 1)^{1/2}, \\ q_0 \leq q \leq \frac{1}{4}\omega r, & |\hat{D}| \text{ is even } \& \ \omega r/\vartheta \geq 2|\hat{D}|, \end{cases}$$

and let q be any other number otherwise, with which both lemmas continue to hold with this choice of q. Here, for short, $\omega = \omega(\hat{D}, \hat{F}; \Omega)$. Note that numerically this choice is comparably more conservative.

Together, the above two lemmas say that the GS-step makes only negligible false non-discoveries, and decomposes $\mathcal{U}_p^*$ into many disconnected components, each has a size not exceeding a fixed integer. As a result, the computation cost of the following GC-step is moderate, at least in theory.

We now discuss the GC-step. The key to understanding the GC-step is that the original regression problem reduces to many disconnected small-size regression problems. To see the point, define $\tilde{Y} = X'Y$ and recall that $G = X'X$. Let $\mathcal{I}_0 \lhd \mathcal{U}_p^*$ be a component, we limit our attention to $\mathcal{I}_0$ by considering the following regression problem:

$$\tilde{Y}^{\mathcal{I}_0} = G^{\mathcal{I}_0,\otimes}\beta + (X'z)^{\mathcal{I}_0}, \tag{41}$$

where $(X'z)^{\mathcal{I}_0} \sim N(0, G^{\mathcal{I}_0,\mathcal{I}_0}) \approx N(0, \Omega^{\mathcal{I}_0,\mathcal{I}_0})$, and $G^{\mathcal{I}_0,\otimes}$ is a $|\mathcal{I}_0| \times p$ matrix according to our notation. What is non-obvious here is that, the regression problem still involves the whole vector β, and is still high-dimensional. To see the point, letting $V = \{1, 2, \ldots, p\} \setminus \mathcal{U}_p^*$, we write $G^{\mathcal{I}_0,\otimes}\beta = G^{\mathcal{I}_0,\mathcal{I}_0}\beta^{\mathcal{I}_0} + I + II$, where $I = \sum_{\mathcal{J}_0 : \mathcal{J}_0 \lhd \mathcal{U}_p^*, \mathcal{J}_0 \neq \mathcal{I}_0} G^{\mathcal{I}_0,\mathcal{J}_0}\beta^{\mathcal{J}_0}$ and $II = G^{\mathcal{I}_0,V}\beta^V$. First, by Sure Screening property, β^V contains only a negligible number of signals, so we can think II as negligible. Second, for any $\mathcal{J}_0 \neq \mathcal{I}_0$ and $\mathcal{J}_0 \lhd \mathcal{U}_p^*$, by the SAS property, $\mathcal{I}_0$ and $\mathcal{J}_0$ are disconnected and so the matrix $G^{\mathcal{I}_0,\mathcal{J}_0}$ is a small size matrix whose coordinates are uniformly small. This heuristic is made precise in the proof of Theorem 8. It is now seen that the regression problem in (41) is indeed low-dimensional:

$$\tilde{Y}^{\mathcal{I}_0} \approx G^{\mathcal{I}_0,\mathcal{I}_0}\beta^{\mathcal{I}_0} + (X'z)^{\mathcal{I}_0} \approx N(\Omega^{\mathcal{I}_0,\mathcal{I}_0}\beta^{\mathcal{I}_0}, \Omega^{\mathcal{I}_0,\mathcal{I}_0}), \tag{42}$$

The above argument is made precise in Lemma 17, see details therein. Finally, approximately, the GC-step is to minimize $\frac{1}{2}(\tilde{Y}^{\mathcal{I}_0} - \Omega^{\mathcal{I}_0,\mathcal{I}_0}\xi)'(\Omega^{\mathcal{I}_0,\mathcal{I}_0})^{-1}(\tilde{Y}^{\mathcal{I}_0} - \Omega^{\mathcal{I}_0,\mathcal{I}_0}\xi) + \frac{1}{2}(u^{gs})^2\|\xi\|_0$, where each coordinate of ξ is either 0 or $\geq v^{gs}$ in magnitude. Comparing this with (42), the procedure is nothing but the penalized MLE of a low dimensional normal model, and the main result follows by exercising basic statistical inferences.

We remark that in the GC-step, removing the constraints on the coordinates of ξ will not give the optimal rate of convergence. This is one of the reasons why the classical subset selection procedure is rate non-optimal. Another reason why the subset selection is non-optimal is that, the procedure has only one tuning parameter, but GS has the flexibility of using different tuning parameters in the GS-step and the GC-step. See Section 2.8 for more discussion.

We are now ready for the proof of Theorem 8.

3.1 Proof of Theorem 8

For notational simplicity, we write $\rho_j^* = \rho_j^*(\vartheta, r, a, \Omega)$. By Lemma 15,

$$\sum_{j=1}^{p} P(\beta_j \neq 0, j \notin \mathcal{U}_p^*) \leq L_p[p^{1-(m_0+1)\vartheta} + \sum_{j=1}^{p} p^{-\rho_j^*}] + o(1).$$

So to show the claim, it is sufficient to show

$$\sum_{j=1}^{p} P(j \in \mathcal{U}_p^*, \mathrm{sgn}(\beta_j) \neq \mathrm{sgn}(\hat{\beta}_j)) \leq L_p[\sum_{j=1}^{p} p^{-\rho_j^*} + p^{1-(m_0+1)\vartheta}] + o(1). \tag{43}$$

Towards this end, let $S(\beta)$ be the support of β, $\Omega^{*,\delta}$ be as in (4), and $\mathcal{G}^{*,\delta}$ be the GOSD. Let $\mathcal{U}_p^*$ be the set of retained indices after the GS-step. Note that when $\mathrm{sgn}(\hat{\beta}_j) \neq 0$, there is a unique component $\mathcal{I}_0$ such that $j \in \mathcal{I}_0 \lhd \mathcal{U}_p^*$. For any connected subgraph $\mathcal{I}_0$ of $\mathcal{G}^{*,\delta}$, let $B(\mathcal{I}_0) = \{k\colon\ k \notin \mathcal{I}_0,\ \Omega^{*,\delta}(k,\ell) \neq 0 \text{ for some } \ell \in \mathcal{I}_0,\ 1 \leq k \leq p\}$. Note that when $\mathcal{I}_0$ is a component of $\mathcal{U}_p^*$, we must have $B(\mathcal{I}_0) \cap \mathcal{U}_p^* = \emptyset$ as for any node in $B(\mathcal{I}_0)$, there is at least one edge between it and some nodes in the component $\mathcal{I}_0$. As a result,

$$P(j \in \mathcal{I}_0 \lhd \mathcal{U}_p^*, B(\mathcal{I}_0) \cap S(\beta) \neq \emptyset) \leq \sum_{\mathcal{I}_0: j \in \mathcal{I}_0} \sum_{k \in B(\mathcal{I}_0)} P(k \notin \mathcal{U}_p^*, \beta_k \neq 0), \tag{44}$$

where the first summation is over all connected subgraphs that contains node j. By Lemma 16, with probability at least $1 - o(1/p)$, $\mathcal{G}^{*,\delta}$ is K-sparse with $K = C(\log(p))^{1/\gamma}$, and there is a finite integer ℓ_0 such that $|\mathcal{I}_0| \leq \ell_0$. As a result, there are at most finite $\mathcal{I}_0$ such that the event $\{j \in \mathcal{I}_0 \lhd \mathcal{U}_p^*\}$ is non-empty, and for each of such $\mathcal{I}_0$, $B(\mathcal{I}_0)$ contains at most L_p nodes. Using (44) and Lemma 15, a direct result is

$$\sum_{j=1}^{p} P(j \in \mathcal{I}_0 \lhd \mathcal{U}_p^*, B(\mathcal{I}_0) \cap S(\beta) \neq \emptyset) \leq L_p[\sum_{j=1}^{p} p^{-\rho_j^*} + p^{1-(m_0+1)\vartheta}] + o(1). \tag{45}$$

Comparing (45) with (43), to show the claim, it is sufficient to show that

$$\sum_{j=1}^{p} P(\mathrm{sgn}(\beta_j) \neq \mathrm{sgn}(\hat{\beta}_j), j \in \mathcal{I}_0 \lhd \mathcal{U}_p^*, B(\mathcal{I}_0) \cap S(\beta) = \emptyset) \leq L_p[\sum_{j=1}^{p} p^{-\rho_j^*} + p^{1-(m_0+1)\vartheta}] + o(1). \tag{46}$$

Fix $1 \leq j \leq p$ and a connected subgraph $\mathcal{I}_0$ such that $j \in \mathcal{I}_0$. For short, let S be the support of $\beta^{\mathcal{I}_0}$ and $\hat{S}$ be the support of $\hat{\beta}^{\mathcal{I}_0}$. The event $\{\mathrm{sgn}(\beta_j) \neq \mathrm{sgn}(\hat{\beta}_j), j \in \mathcal{I}_0 \lhd \mathcal{U}_p^*\}$ is identical to the event of $\{\mathrm{sgn}(\beta_j) \neq \mathrm{sgn}(\hat{\beta}_j), j \in S \cup \hat{S}\}$. Moreover, Since $\mathcal{I}_0$ has a finite size, both S and $\hat{S}$ have finite possibilities. So to show (46), it is sufficient to show that for any fixed $1 \leq j \leq p$, connected subgraph $\mathcal{I}_0$, and subsets $S_0, S_1 \subset \mathcal{I}_0$ such that $j \in S_0 \cup S_1$,

$$P(\mathrm{sgn}(\beta_j) \neq \mathrm{sgn}(\hat{\beta}_j), S = S_0, \hat{S} = S_1, j \in \mathcal{I}_0 \lhd \mathcal{U}_p^*, B(\mathcal{I}_0) \cap S(\beta) = \emptyset) \leq L_p[p^{-\rho_j^*} + p^{-(m_0+1)\vartheta}]. \tag{47}$$

We now show (47). The following lemma is proved in Ji and Jin (2011, A.4).

Lemma 17 *Suppose the conditions of Theorem 8 hold. Over the event* $\{j \in \mathcal{I}_0 \lhd \mathcal{U}_p^*\} \cap \{B(\mathcal{I}_0) \cap S(\beta) = \emptyset\}$, $\|(\Omega\beta)^{\mathcal{I}_0} - \Omega^{\mathcal{I}_0,\mathcal{I}_0}\beta^{\mathcal{I}_0}\|_\infty \leq C\tau_p(\log(p))^{-(1-\gamma)}$.

Write for short $\hat{M} = G^{\mathcal{I}_0,\mathcal{I}_0}$ and $M = \Omega^{\mathcal{I}_0,\mathcal{I}_0}$. By definitions, $\hat{\beta}^{\mathcal{I}_0}$ is the minimizer of the following functional $Q(\xi) \equiv \frac{1}{2}(\tilde{Y}^{\mathcal{I}_0} - \hat{M}\xi)'\hat{M}^{-1}(\tilde{Y}^{\mathcal{I}_0} - \hat{M}\xi) + \frac{1}{2}(u^{gs})^2\|\xi\|_0$, where ξ is an $|\mathcal{I}_0| \times 1$ vector whose coordinates are either 0 or $\geq v^{gs}$ in magnitude, $u^{gs} = \sqrt{2\vartheta\log(p)}$, and $v^{gs} = \sqrt{2r\log(p)}$. In particular, $Q(\beta^{\mathcal{I}_0}) \geq Q(\hat{\beta}^{\mathcal{I}_0})$, or equivalently

$$(\hat{\beta}^{\mathcal{I}_0} - \beta^{\mathcal{I}_0})'(\tilde{Y}^{\mathcal{I}_0} - \hat{M}\beta^{\mathcal{I}_0}) \geq \frac{1}{2}(\hat{\beta}^{\mathcal{I}_0} - \beta^{\mathcal{I}_0})'\hat{M}(\hat{\beta}^{\mathcal{I}_0} - \beta^{\mathcal{I}_0}) + (|S_1| - |S_0|)\vartheta\log(p). \tag{48}$$

Now, write for short $\delta = \tau_p^{-2}(\hat{\beta}^{\mathcal{I}_0} - \beta^{\mathcal{I}_0})'M(\hat{\beta}^{\mathcal{I}_0} - \beta^{\mathcal{I}_0})$. First, by Schwartz inequality, $[(\hat{\beta}^{\mathcal{I}_0} - \beta^{\mathcal{I}_0})'(\tilde{Y}^{\mathcal{I}_0} - \hat{M}\beta^{\mathcal{I}_0})]^2 \leq \delta\tau_p^2(\tilde{Y}^{\mathcal{I}_0} - \hat{M}\beta^{\mathcal{I}_0})'M^{-1}(\tilde{Y}^{\mathcal{I}_0} - \hat{M}\beta^{\mathcal{I}_0})$. Second, by Lemma 17, $\tilde{Y}^{\mathcal{I}_0} = w + M\beta^{\mathcal{I}_0} + rem$, where $w \sim N(0, M)$ and with probability $1 - o(1/p)$, $|rem| \leq C(\log(p))^{-(1-\gamma)}\tau_p$. Last, with probability at least $(1 - o(1/p))$, $\| \hat{M} - M \|_\infty \leq C\sqrt{\log(p)}p^{-[\kappa-(1-\vartheta)]/2}$. Inserting these into (48) gives that with probability at least $(1 - o(1/p))$, $w'M^{-1}w \geq \frac{1}{4}\left[\left(\sqrt{\delta r} + \frac{(|S_1|-|S_0|)\vartheta}{\sqrt{\delta r}}\right)_+\right]^2(2\log(p)) + O((\log(p))^\gamma)$. Since $\gamma < 1$, $O((\log(p))^\gamma)$ is negligible. We note that $w'M^{-1}w \sim \chi^2_{|\mathcal{I}_0|}(0)$. Inserting this back to (47), the left hand side $\leq \epsilon_p^{|S_0|}P(\chi^2_{|\mathcal{I}_0|}(0) \geq [(\sqrt{\delta r} + (|S_1| - |S_0|)\vartheta/\sqrt{\delta r})_+]^2(\log(p)/2)) + o(1/p)$. Assume $\mathrm{sgn}(\beta_j) \neq \mathrm{sgn}(\hat{\beta}_j)$, and fix all parameters except δ, S_0 and S_1. By arguments similar to the proof of Lemma 18, the above quantity cannot achieve its maximum in the cases where $S_0 = S_1$. Hence we only need to consider the cases where $S_0 \neq S_1$. We also only need to consider the cases where $\max(|S_0|, |S_1|) \leq m_0$, since the sum of the probabilities of other cases is controlled by $p^{1-(m_0+1)\vartheta}$. The claim follows by the definitions of ρ_j^*. ■

4. Simulations

We conduct a small-scale simulation study to investigate the numerical performance of Graphlet Screening and compare it with the lasso and the UPS. The subset selection is not included for comparison since it is computationally NP hard. We consider the experiments for both random design and fixed design, where as before, the parameters (ϵ_p, τ_p) are tied to (ϑ, r) by $\epsilon_p = p^{-\vartheta}$ and $\tau_p = \sqrt{2r\log(p)}$ (we assume $\sigma = 1$ for simplicity in this section).

In random design settings where p is not very large, we follow the spirit of the refined UPS in Ji and Jin (2011) and propose the iterative Graphlet Screening algorithm where we

iterate Graphlet Screening for a few times (≤ 5). The main purpose for the iteration is to denoise the Gram matrix; see Ji and Jin (2011, Section 3) for more discussion.

Even with the refinement as in Ji and Jin (2011, Section 3), UPS behaves poorly for most examples presented below. Over close investigations, we find out that this is due to the threshold choice in the initial U-step is too low, and increasing the threshold largely increases the performance. Note that the purpose of this step is to denoise the Gram matrix (Ji and Jin, 2011, Section 3), not for signal retainment, and so a larger threshold helps.

In this section, we use this improved version of refined UPS, but for simplicity, we still call it the refined UPS. With that being said, recall that UPS is unable to resolve the problem of signal cancellation, so it usually performs poorer than GS, especially when the effect of signal cancellation is strong. For this reason, part of the comparison is between GS and the lasso only.

The experiments with random design contain the following steps.

1. Fix $(p, \vartheta, r, \mu, \Omega)$ such that $\mu \in \Theta_p(\tau_p)$. Generate a vector $b = (b_1, b_2, \ldots, b_p)'$ such that $b_i \overset{iid}{\sim} \text{Bernoulli}(\epsilon_p)$, and set $\beta = b \circ \mu$.

2. Fix κ and let $n = n_p = p^\kappa$. Generate an $n \times p$ matrix with iid rows from $N(0, (1/n)\Omega)$.

3. Generate $Y \sim N(X\beta, I_n)$, and apply the iterative Graphlet Screening, the refined UPS and the lasso.

4. Repeat 1-3 independently, and record the average Hamming distances or the Hamming ratio, the ratio of the Hamming distance and the number of the signals.

The steps for fixed design experiments are similar, except for that $n_p = p$, $X = \Omega^{1/2}$ and we apply GS and UPS directly.

GS uses tuning parameters $(m_0, \mathcal{Q}, u^{gs}, v^{gs})$. We set $m_0 = 3$ for our experiments, which is usually large enough due to signal sparsity. The choice of $\mathcal{Q}$ is not critical, as long as the corresponding parameter q satisfies (30), and we use the maximal $\mathcal{Q}$ satisfying (30) in most experiments. Numerical studies below (e.g., Experiment 5a) support this point. In principle, the optimal choices of (u^{gs}, v^{gs}) depend on the unknown parameters (ϵ_p, τ_p), and how to estimate them in general settings is a lasting open problem (even for linear models with orthogonal designs). Fortunately, our studies (e.g., Experiment 5b-5d) show that mis-specifying parameters (ϵ_p, τ_p) by a reasonable amount does not significantly affect the performance of the procedure. For this reason, in most experiments below, assuming (ϵ_p, τ_p) are known, we set (u^{gs}, v^{gs}) as $(\sqrt{2\log(1/\epsilon_p)}, \tau_p)$. For the iterative Graphlet Screening, we use the same tuning parameters in each iteration.

For the UPS and the refined UPS, we use the tuning parameters $(u^{ups}, v^{ups}) = (u^{gs}, v^{gs})$. For both the iterative Graphlet Screening and the refined UPS, we use the following as the initial estimate: $\hat{\beta}_i = \text{sgn}(\tilde{Y}_i) \cdot 1\{|\tilde{Y}_i| \geq \tau_p\}$, $1 \leq i \leq p$, where $\tilde{Y} = X'Y$. The main purpose of initial estimate is to denoise the Gram matrix, not for screening. We use *glmnet* package (Friedman et al., 2010) to perform lasso. To be fair in comparison, we apply the lasso with all tuning parameters, and we report the Hamming error associated with the "best" tuning parameter.

The simulations contain 6 different experiments which we now describe separately.

Experiment 1. The goal of this experiment is two-fold. First, we compare GS with UPS and the lasso in the fixed design setting. Second, we investigate the minimum signal strength levels τ_p required by these three methods to yield exact recovery, respectively.

Fixing $p = 0.5 \times 10^4$, we let $\epsilon_p = p^{-\vartheta}$ for $\vartheta \in \{0.25, 0.4, 0.55\}$, and $\tau_p \in \{6, 7, 8, 9, 10\}$. We use a fixed design model where Ω is a symmetric diagonal block-wise matrix, where each block is a 2×2 matrix, with 1 on the diagonals, and ± 0.7 on the off-diagonals (the signs alternate across different blocks). Recall the $\beta = b \circ \mu$. For each pair of (ϵ_p, τ_p), we generate b as p iid samples from $Bernoulli(\epsilon_p)$, and we let μ be the vector where the signs of $\mu_i = \pm 1$ with equal probabilities, and $|\mu_i| \stackrel{iid}{\sim} 0.8\nu_{\tau_p} + 0.2h$, where ν_{τ_p} is the point mass at τ_p and $h(x)$ is the density of $\tau_p(1 + V/6)$ with $V \sim \chi^2_1$. The average Hamming errors across 40 repetitions are tabulated in Table 3. For all (ϑ, τ_p) in this experiment, GS behaves more satisfactorily than the UPS, which in turn behaves more satisfactorily than the lasso.

Suppose we say a method yields 'exact recovery' if the average Hamming error ≤ 3. Then, when $\vartheta = 0.25$, the minimum τ_p for GS to yield exact recovery is $\tau_p \approx 8$, but that for UPS and the lasso are much larger (≥ 10). For larger ϑ, the differences are less prominent, but the pattern is similar.

The comparison between GS and UPS is particularly interesting. Due to the block structure of Ω, as ϑ decreases, the signals become increasingly less sparse, and the effects of signal cancellation become increasingly stronger. As a result, the advantage of GS over the UPS becomes increasingly more prominent.

	τ_p	6	7	8	9	10
$\vartheta = 0.25$	Graphic Screening	24.7750	8.6750	2.8250	0.5250	0.1250
	UPS	48.5500	34.6250	36.3500	30.8750	33.4000
	lasso	66.4750	47.7000	43.5250	35.2500	35.0500
$\vartheta = 0.40$	Graphic Screening	6.9500	2.1500	0.4000	0.0750	0.0500
	UPS	7.7500	4.0000	2.2000	2.7750	2.4250
	lasso	12.8750	6.8000	4.3250	3.7500	2.6750
$\vartheta = 0.55$	Graphic Screening	1.8750	0.8000	0.3250	0.2250	0.1250
	UPS	1.8750	0.8000	0.3250	0.2250	0.1250
	lasso	2.5000	1.1000	0.7750	0.2750	0.1250

Table 3: Comparison of average Hamming errors (Experiment 1).

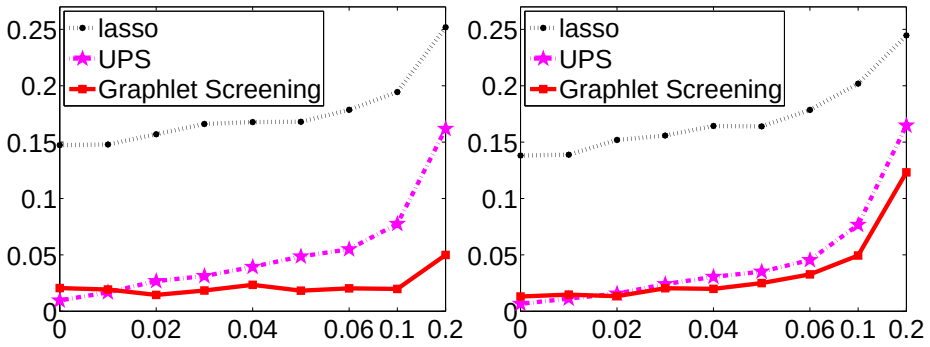
Experiment 2. In this experiment, we compare GS, UPS and the lasso in the random design setting, and investigate the effect of signal cancellation on their performances. We fix $(p, \kappa, \vartheta, r) = (0.5 \times 10^4, 0.975, 0.35, 3)$, and assume Ω is blockwise diagonal. We generate μ as in Experiment 1, but to better illustrate the difference between UPS and GS in the presence of signal cancellation, we generate the vector b differently and allow it to depend on Ω. The experiment contains 2 parts, $2a$ and $2b$.

In Experiment 2a, Ω is the block-wise matrix where each block is 2 by 2 matrix with 1 on the diagonals and $\pm.5$ on the off diagonals (the signs alternate on adjacent blocks). According to the blocks in Ω, the set of indices $\{1, 2, \ldots, p\}$ are also partitioned into blocks accordingly. For any fixed ϑ and $\eta \in \{0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.1, 0.2\}$, we ran-

domly choose $(1-2p^{-\vartheta})$ fraction of the blocks (of indices) where b is 0 at both indices, $2(1-\eta)p^{-\vartheta}$ fraction of the blocks where b is 0 at one index and 1 at the other (two indices are equally likely to be 0), $2\eta p^{-\vartheta}$ faction of the blocks where b is 1 on both indices.

Experiment 2b has similar settings, where the difference is that (a) we choose Ω to be a diagonal block matrix where each block is a 4 by 4 matrix (say, denoted by A) satisfying $A(i,j) = 1\{i=j\} + 0.4\cdot 1\{|i-j|=1\}\cdot \mathrm{sgn}(6-i-j) + 0.05\{|i-j|\geq 2\}\cdot \mathrm{sgn}(5.5-i-j)$, $1\leq i,j\leq 4$, and (b) $(1-4p^{-\vartheta})$ is the fraction of blocks where b is nonzero in $k=0$ indices, $4(1-\eta)p^{-\vartheta}$ is that for $k=1$, and $4\eta p^{-\vartheta}$ is that for $k\in\{2,3,4\}$ in total. In a block where β is nonzero at k indices, all configurations with $k=1$ are equally likely, and all those with $k\in\{2,3,4\}$ are equally likely.

The average Hamming ratio results across 40 runs for two Experiment 2a and 2b are reported in Figure 3, where UPS and GS consistently outperform the lasso. Additionally, when η is small, the effect of signal cancellation is negligible, so UPS and GS have similar performances. However, when η increases, the effects of signal cancellation grows, and the advantage of GS over UPS becomes increasingly more prominent.



(a) Blockwise diagonal Ω in 2×2 blocks(b) Blockwise diagonal Ω in 4×4 blocks

Figure 3: Hamming ratio results in Experiment 2

Through Experiment 1-2, the comparison of UPS and GS is more or less understood. For this reason, we do not include UPS for study in Experiment 3-5, but we include UPS for study in Experiment 6 where we investigate robustness of all three methods.

Experiment 3. In this experiment, we investigate how different choices of signal vector β affect the comparisons of GS and the lasso. We use a random design model, and Ω is a symmetric tri-diagonal correlation matrix where the vector on each sub-diagonal consists of blocks of $(.4,.4,-.4)'$. Fix $(p,\kappa) = (0.5\times 10^4, 0.975)$ (note $n = p^\kappa \approx 4,000$). We let $\epsilon_p = p^{-\vartheta}$ with $\vartheta\in\{0.35, 0.5\}$ and let $\tau_p\in\{6,8,10\}$. For each combination of (ϵ_p,τ_p), we consider two choices of μ. For the first choice, we let μ be the vector where all coordinates equal to τ_p (note β is still sparse). For the second one, we let μ be as in Experiment 1. The average Hamming ratios for both procedures across 40 repetitions are tabulated in Table 4.

Experiment 4. In this experiment, we generate β the same way as in Experiment 1, and investigate how different choices of design matrices affect the performance of the two methods. Setting $(p,\vartheta,\kappa) = (0.5\times 10^4, 0.35, 0.975)$ and $\tau_p\in\{6,7,8,9,10,11,12\}$, we use

τ_p		6		8		10	
Signal Strength		Equal	Unequal	Equal	Unequal	Equal	Unequal
$\vartheta = 0.35$	Graphic Screening	0.0810	0.0825	0.0018	0.0034	0	0.0003
	lasso	0.2424	0.2535	0.1445	0.1556	0.0941	0.1109
$\vartheta = 0.5$	Graphic Screening	0.0315	0.0297	0.0007	0.0007	0	0
	lasso	0.1107	0.1130	0.0320	0.0254	0.0064	0.0115

Table 4: Hamming ratio results of Experiment 3, where "Equal" and "Unequal" stand for the first and the second choices of μ, respectively.

Gaussian random design model for the study. The experiment contains 3 sub-experiments 4*a*-4*c*.

In Experiment 4*a*, we set Ω as the symmetric diagonal block-wise matrix, where each block is a 2×2 matrix, with 1 on the diagonals, and ± 0.5 on the off-diagonals (the signs alternate across different blocks). The average Hamming ratios of 40 repetitions are reported in Figure 4.

In Experiment 4*b*, we set Ω as a symmetric penta-diagonal correlation matrix, where the main diagonal are ones, the first sub-diagonal consists of blocks of $(.4, .4, -.4)'$, and the second sub-diagonal consists of blocks of $(.05, -.05)'$. The average Hamming ratios across 40 repetitions are reported in Figure 4.

In Experiment 4*c*, we generate Ω as follows. First, we generate Ω using the function *sprandsym(p,K/p)* in *matlab*. We then set the diagonals of Ω to be zero, and remove some of entries so that Ω is K-sparse for a pre-specified K. We then normalize each non-zero entry by the sum of the absolute values in that row or that column, whichever is larger, and multiply each entry by a pre-specified positive constant A. Last, we set the diagonal elements to be 1. We choose $K = 3$ and $A = 0.7$, draw 5 different Ω with this method, and for each of them, we draw (X, β, z) 10 times independently. The average Hamming ratios are reported in Figure 4. The results suggest that GS is consistently better than the lasso.

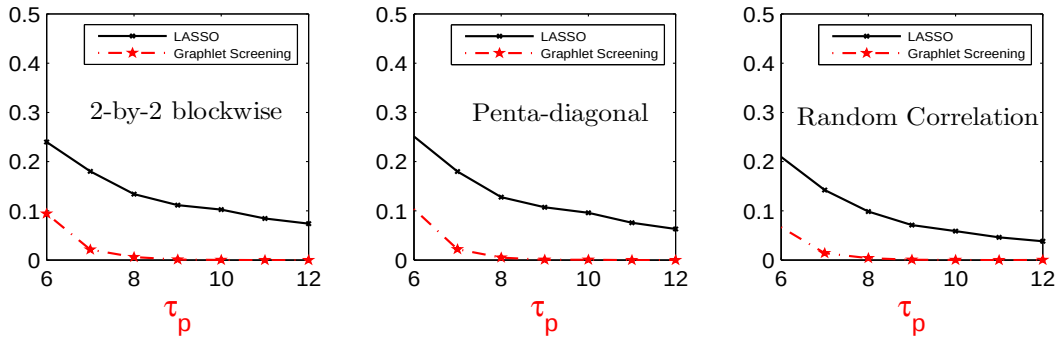


Figure 4: x-axis: τ_p. y-axis: Hamming ratios. Left to right: Experiment 4*a*, 4*b*, and 4*c*.

Experiment 5. In this experiment, we investigate how sensitive GS is with respect to the tuning parameters. The experiment contains 4 sub-experiments, 5a-5d. In Experiment 5a, we investigate how sensitive the procedure is with respect to the tuning parameter q in $\mathcal{Q}$ where we assume (ϵ_p, τ_p) are known; recall that the main results hold as long as q fall into the range given in (30). In Experiment 5b-5d, we mis-specify (ϵ_p, τ_p) by a reasonably small amount, and investigate how the mis-specification affect the performance of the procedure. For the whole experiment, we choose β the same as in Experiment 1, and Ω the same as in Experiment 4b. We use a fixed design model in Experiment 5a-5c, and a random design model in Experiment 5d. For each sub-experiment, the results are based on 40 independent repetitions. We now describe the sub-experiments with details.

In Experiment 5a, we choose $\vartheta \in \{0.35, 0.6\}$ and $r \in \{1.5, 3\}$. In GS, let $q_{max} = q_{max}(\hat{D}, \hat{F})$ be the maximum value of q satisfying (30). For each combination of (ϑ, r) and $(\hat{D}, \hat{F})$, we choose $q(\hat{D}, \hat{F}) = q_{max}(\hat{D}, \hat{F}) \times \{0.7, 0.8, 0.9, 1, 1.1, 1.2\}$ for our experiment. The results are tabulated in Table 5, which suggest that different choices of q have little influence over the variable selection errors. We must note that the larger we set $q(\hat{D}, \hat{F})$, the faster the algorithm runs.

$q(\hat{F}, \hat{D})/q_{max}(\hat{F}, \hat{D})$	0.7	0.8	0.9	1	1.1	1.2
$(\vartheta, r) = (0.35, 1.5)$	0.0782	0.0707	0.0661	0.0675	0.0684	0.0702
$(\vartheta, r) = (0.35, 3)$	0.0066	0.0049	0.0036	0.0034	0.0033	0.0032
$(\vartheta, r) = (0.6, 1.5)$	0.1417	0.1417	0.1417	0.1417	0.1417	0.1417
$(\vartheta, r) = (0.6, 3)$	0.0089	0.0089	0.0089	0.0089	0.0089	0.0089

Table 5: Hamming ratio results in Experiment 5a.

In Experiment 5b, we use the same settings as in Experiment 5a, but we assume ϑ (and so ϵ_p) is unknown (the parameter r is assumed as known, however), and let ϑ^* is the misspecified value of ϑ. We take $\vartheta^* \in \vartheta \times \{0.85, 0.925, 1, 1.075, 1.15, 1.225\}$ for the experiment.

In Experiment 5c, we use the same settings as in Experiment 5b, but we assume r (and so τ_p) is unknown (the parameter ϑ is assumed as known, however), and let r^* is the misspecified value of r. We take $r^* = r \times \{0.8, 0.9, 1, 1.1, 1.2, 1.3\}$ for the experiment.

In Experiment 5b-5c, we run GS with tuning parameters set as in Experiment 1, except ϑ or r are replaced by the misspecified counterparts ϑ^* and r^*, respectively. The results are reported in Table 6, which suggest that the mis-specifications have little effect as long as r^*/r and ϑ^*/ϑ are reasonably close to 1.

In Experiment 5d, we re-examine the mis-specification issue in the random design setting. We use the same settings as in Experiment 5b and Experiment 5c, except for (a) while we use the same Ω as in Experiment 5b, the design matrix X are generated according to the random design model as in Experiment 4b, and (b) we only investigate for the case of $r = 2$ and $\vartheta \in \{0.35, 0.6\}$. The results are summarized in Table 7, which is consistent with the results in 5b-5c.

Experiment 6. In this experiment, we investigate the robustness of all three methods for the mis-specification of the linear model (1). We use the random design setting as in Experiment 4b, except that we fix $(\vartheta, r) = (0.35, 3)$. The experiment contains 3 sub-

ϑ^*/ϑ	0.85	0.925	1	1.075	1.15	1.225
$(\vartheta, r) = (0.35, 1.5)$	0.0799	0.0753	0.0711	0.0710	0.0715	0.0746
$(\vartheta, r) = (0.35, 3)$	0.0026	0.0023	0.0029	0.0030	0.0031	0.0028
$(\vartheta, r) = (0.6, 1.5)$	0.1468	0.1313	0.1272	0.1280	0.1247	0.1296
$(\vartheta, r) = (0.6, 3)$	0.0122	0.0122	0.0139	0.0139	0.0130	0.0147
r^*/r	0.8	0.9	1	1.1	1.2	1.3
$(\vartheta, r) = (0.35, 1.5)$	0.0843	0.0731	0.0683	0.0645	0.0656	0.0687
$(\vartheta, r) = (0.35, 3)$	0.0062	0.0039	0.0029	0.0030	0.0041	0.0054
$(\vartheta, r) = (0.6, 1.5)$	0.1542	0.1365	0.1277	0.1237	0.1229	0.1261
$(\vartheta, r) = (0.6, 3)$	0.0102	0.0076	0.0085	0.0059	0.0051	0.0076

Table 6: Hamming ratio results in Experiment 5*b* (top) and in Experiment 5*c* (bottom).

ϑ^*/ϑ	0.85	0.925	1	1.075	1.15	1.225
$(\vartheta, r) = (0.35, 2)$	0.1730	0.1367	0.1145	0.1118	0.0880	0.0983
$(\vartheta, r) = (0.6, 2)$	0.0583	0.0591	0.0477	0.0487	0.0446	0.0431
r^*/r	0.8	0.9	1	1.1	1.2	1.3
$(\vartheta, r) = (0.35, 2)$	0.1881	0.1192	0.1275	0.1211	0.1474	0.1920
$(\vartheta, r) = (0.6, 2)$	0.0813	0.0515	0.0536	0.0397	0.0442	0.0510

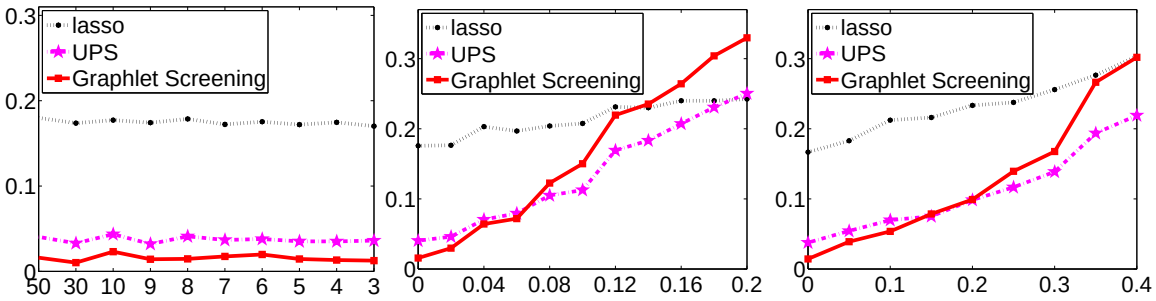
Table 7: Hamming ratio results in Experiment 5*d*.

experiments, 6*a*-6*c*, where we consider three scenarios where the linear model (1) is in question: the presence of non-Gaussianity, the presence of missing predictors, and the presence of non-linearity, correspondingly.

In Experiment 6*a*, we assume the noise vector z in Model (1) is non-Gaussian, where the coordinates are iid samples from a t-distribution with the same degree of freedom (df) (we assume that z is normalized so each coordinate has unit variance), where the df range in $\{3, 4, 5, 6, 7, 8, 9, 10, 30, 50\}$. Figure 5a shows how the Hamming ratios (based on 40 independent repetitions) change when the *df* decreases. The results suggest that all three methods are reasonably robust against non-Gaussianity, but GS continues to have the best performance.

In Experiment 6*b*, we assume that the true model is $Y = X\beta + z$ where (X, β, z) are generated as in 4*b*, but the model that is accessible to us is a misspecified model where the some of the true predictors are missing. Fix $\eta \in (0, 1)$, and let $S(\beta)$ be the support of β. For each $i \in S(\beta)$, we flip a coin that lands on head with probability η, and we retain i if and only if the coin lands on tail. Let $S^* \subset S(\beta)$ be the set of retained indices, and let $R = S^* \cup S^c$. The misspecified model we consider is then $Y = X^{\otimes,R}\beta^R + z$.

For the experiment, we let η range in $0.02 \times \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. The average Hamming ratios (based on 40 independent repetitions) are reported in Figure 5b. The results suggest that all three results are reasonably robust to missing predictors, with the lasso being the most robust. However, as long as the proportion of true predictors that are missing is reasonably small (say, $\eta \leq .1$), GS continues to outperform UPS and the lasso.



(a) non-Gaussian noise (b) Missing predictors (c) Non-linear predictors

Figure 5: Hamming ratio results in Experiment 6

In Experiment 6c, for $i = 1, \ldots, n$, the true model is an additive model in the form of $Y_i = \sum_{j=1}^{p} f_j(X_{ij})\beta_j + z_i$, but what is accessible to us is the linear model $Y_i = \sum_{j=1}^{p} X_{ij}\beta_j + z_i$ (and thus misspecified; the true model is non-linear). For experiment, we let (X, β, z) be generated as in 4b, and $S(\beta)$ be the support of β. Fixing $\eta \in (0, 1)$, for each $i \in S(\beta)$, we flip a coin that lands on head with probability η, and let $S_{nl} \subset S(\beta)$ be all indices of the heads. We then randomly split S_{nl} into two sets S_1 and S_2 evenly. For $j = 1, \ldots, p$, we define $f_j(x) = [\mathrm{sgn}(x)x^2 \cdot 1\{j \in S_1\} + (e^{\sqrt{n}x} - a_j) \cdot 1\{j \in S_2\} + x \cdot 1\{j \in S_{nl}^c\}]/c_j$, where a_j and c_j are constants such that $\{f_j(X(i, j))\}_{i=1}^{n}$ has mean 0 and variance $1/n$.

For the experiment, we let η range in $.05 \times \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$. The average Hamming ratios (based on 40 independent repetitions) are reported in Figure 5c. The results suggest that all three methods are reasonably robust to the presence of nonlinearity, and GS continues to outperform UPS and the lasso when the degree of nonlinearly is moderate (say, $\eta < .2$).

5. Connection to Existing Literature and Possible Extensions

Our idea of utilizing graph sparsity is related to the graphical lasso (Meinshausen and Buhlmann, 2006; Friedman et al., 2008), which also attempts to exploit graph structure. However, the setting we consider here is different from that in Meinshausen and Buhlmann (2006); Friedman et al. (2008), and our emphasis on precise optimality and calibration is also very different. Our method allows nearly optimal detection of very rare and weak effects, because they are based on careful analysis that has revealed a number of subtle high-dimensional effects (e.g., phase transitions) that we properly exploit. Existing methodologies are not able to exploit or capture these phenomena, and can be shown to fail at the levels of rare and weak effects where we are successful.

The paper is closely related to the recent work by Fan and Lv (2008), Ji and Jin (2011) and Genovese et al. (2012). Both Ji and Jin (2011) and this paper use a similar rare and weak signal framework and a similar random design model. However, they are different in important ways, since the technical devise developed in Ji and Jin (2011) can not be extended to the current study. For example, the lower bound derived in this paper is different and sharper than that in Ji and Jin (2011). Also, the procedure in Ji and Jin (2011) relies on marginal regression for screening. The limitation of marginal regression

is that it neglects the graph structure of GOSD for the regularized Gram matrix (1.5), so that it is incapable of picking variables that have *weak marginal* correlation but *significant joint* correlation to Y. Correct selection of such hidden significant variables, termed as the challenge of *signal cancellation* (Wasserman and Roeder, 2009), is the difficulty at the heart of the variable selection problem. One of the main innovation of GS is that it uses the graph structure to guide the screening, so that it is able to successfully overcome the challenge of signal cancellation.

Additionally, two papers have very different objectives, and consequently the underlying analysis are very different. The main results of each of these two papers can not be deduced from the other. For example, to assess optimality, Ji and Jin (2011) uses the criterion of the partition of the phase diagram, while the current paper uses the minimax Hamming distance. Given the complexity of the high dimensional variable selection, one type of optimality does not imply the other, and vice versa. Also, the main result in Ji and Jin (2011) focuses on conditions under which the optimal rate of convergence is $L_p p^{1-(\vartheta+r)^2/(4r)}$ for the *whole* phase space. While this overlaps with our Corollaries 10 and 11, we must note that Ji and Jin (2011) deals with the much more difficult cases where r/ϑ can get arbitrary large; and to ensure the success in that case, they assume very strong conditions on the design matrix and the range of the signal strength. On the other hand, the main focus of the current paper is on optimal variable selection under conditions (of the Gram matrix G as well as the signal vector β) that are as general as possible.

While the study in this paper has been focused on the Random Design model $\mathrm{RD}(\vartheta, \kappa, \Omega)$, extensions to deterministic design models are straightforward (in fact, in Corollary 7, we have already stated some results on deterministic design models), and the omission of discussion on the latter is largely for technical simplicity and the sake of space. In fact, for models with deterministic designs, since the likelihood ratio test in the derivation of the lower bound matches the penalized MLE in the cleaning step of GS, the optimality of GS follows from the *Sure Screening and Separable After Screening* properties of GS. The proof of these properties, and therefore the optimality of GS, follows the same line as those for random design as long as $\max_j |\sum_i \beta_i G(i,j) I\{\Omega^{*,\delta}(i,j) = 0\}|/\tau_p$ is small. This last condition on G holds when $p^{1-\vartheta}\|G - \Omega\|_\infty = o(1)$ with a certain $\Omega \in \mathcal{M}_p^*(\gamma, c_0, g, A)$. Alternatively, this condition holds when $p^{1-\vartheta}\|G - \Omega\|_\infty^2 \log p = o(1)$ with $\Omega \in \mathcal{M}_p^*(\gamma, c_0, g, A)$, provided that $\mathrm{sgn}(\beta_j)$ are iid symmetric random variables as in Candes and Plan (2009).

In this paper, we assume the signal vector β is independent of the design matrix X, and that β is modeled by a Bernoulli model through $\beta = b \circ \mu$. Both assumptions can be relaxed. In fact, in order for GS to work, what we really need is some *decomposability* condition similar to that in Lemma 1, where except for negligible probabilities, the maximum size of the graphlets $m_0^* = m_0^*(S(\beta), G, \delta)$ is small. In many situations, we can show that m_0^* does not exceed a fixed integer. One of such examples is as follows. Suppose for any fixed integer $m \geq 1$ and size-m subset S of $\{1, 2, \ldots, p\}$, there are constants $C > 0$ and $d > 0$ such that the conditional probability $P(\beta_j \neq 0, \forall j \in S|X) \leq Cp^{-dm}$. In fact, when such a condition holds, the claim follows since $\mathcal{G}^{*,\delta}$ has no more than $C(eK)^m$ size-m connected subgraphs if it is K-sparse. See the proof of Lemma 1 for details. Note that when $\epsilon_p = p^{-\vartheta}$ as in the ARW, then the condition holds for the Bernoulli model in Lemma 1, with $d = \vartheta$. Note also that the Bernoulli model can be replaced by some Ising models.

Another interesting direction of future research is the extension of GS to more general models such as logistic regression. The extension of the lower bound in Theorem 6 is relatively simple since the degree of GOLF can be bounded using the true β. This indicates the optimality of GS in logistic and other generalized linear models as long as proper generalized likelihood ratio or Bayes tests are used in both the *GS*- and *GC*-steps.

6. Proofs

In this section, we provide all technical proofs. We assume $\sigma = 1$ for simplicity.

6.1 Proof of Lemma 1

When $\mathcal{G}_S^{*,\delta}$ contains a connected subgraph of size $\geq m_0 + 1$, it must contain a connected subgraph with size $m_0 + 1$. By Frieze and Molloy (1999), there are $\leq p(eK)^{m_0+1}$ connected subgraph of size $m_0 + 1$. Therefore, the probability that $\mathcal{G}_S^{*,\delta}$ has a connected subgraph of size $(m_0 + 1) \leq p(eK)^{m_0+1}\epsilon_p^{m_0+1}$. Combining these gives the claim. ■

6.2 Proof of Theorem 6

Write for short $\rho_j^* = \rho_j^*(\vartheta, r, a, \Omega)$. Without loss of generality, assume $\rho_1^* \leq \rho_2^* \leq \ldots \leq \rho_p^*$. We construct indices $i_1 < i_2 < \ldots < i_m$ as follows. (a) start with $B = \{1, 2, \ldots, p\}$ and let $i_1 = 1$, (b) updating B by removing i_1 and all nodes j that are neighbors of i_1 in GOLF, let i_2 be the smallest index, (c) defining $i_3, i_4, \ldots, i_m$ by repeating (b), and terminates the process when no indices is left in B. Since each time we remove at most $d_p(\mathcal{G}^\diamond)$ nodes, it follows that

$$\sum_{j=1}^{p} p^{-\rho_j^*} \leq d_p(\mathcal{G}^\diamond) \sum_{k=1}^{m} p^{-\rho_{i_k}^*}. \tag{49}$$

For each $1 \leq j \leq p$, as before, let (V_{0j}^*, V_{1j}^*) be the least favorable configuration, and let $(\theta_{*j}^{(0)}, \theta_{*j}^{(1)}) = \mathrm{argmin}_{\{\theta^{(0)} \in B_{V_{0j}^*}, \theta^{(1)} \in B_{V_{1j}^*}, \mathrm{sgn}(\theta^{(0)}) \neq \mathrm{sgn}(\theta^{(1)})\}} \alpha(\theta^{(0)}, \theta^{(1)}; \Omega)$. By our notations, it is seen that

$$\rho_j^* = \eta(V_{0j}^*, V_{1j}^*; \Omega), \qquad \alpha^*(V_{0j}^*, V_{1j}^*; \Omega) = \alpha(\theta_{*j}^{(0)}, \theta_{*j}^{(1)}; \Omega). \tag{50}$$

In case $(\theta_{*j}^{(0)}, \theta_{*j}^{(1)})$ is not unique, pick one arbitrarily. We construct a $p \times 1$ vector μ^* as follows. Fix $j \in \{i_1, \cdots, i_m\}$. For all indices in V_{0j}^*, set the constraint of μ^* on these indices to be $\theta_{*j}^{(0)}$. For any index $i \notin \cup_{k=1}^m V_{0i_k}^*$, set $\mu_i^* = \tau_p$. Since

$$\mathrm{Hamm}_p^*(\vartheta, \kappa, r, a, \Omega) \geq \inf_{\hat{\beta}} H_p(\hat{\beta}; \epsilon_p, n_p, \mu^*, \Omega) = \inf_{\hat{\beta}} \sum_{i=1}^{p} P(\mathrm{sgn}(\hat{\beta}_j) \neq \mathrm{sgn}(\beta_j)), \tag{51}$$

it follows that

$$\mathrm{Hamm}_p^*(\vartheta, \kappa, r, a, \Omega) \geq \sum_{k=1}^{m} \sum_{j \in V_{0i_k} \cup V_{1i_k}} P(\mathrm{sgn}(\hat{\beta}_j) \neq \mathrm{sgn}(\beta_j)), \tag{52}$$

where $\beta = b \circ \mu^*$ in (51)-(52). Combining (49) and (52), to show the claim, we only need to show that for any $1 \le k \le m$ and any procedure $\hat{\beta}$,

$$\sum_{j \in V_{0i_k} \cup V_{1i_k}} P(\mathrm{sgn}(\hat{\beta}_j) \neq \mathrm{sgn}(\beta_j)) \ge L_p p^{-\rho^*_{i_k}}. \tag{53}$$

Towards this end, we write for short $V_0 = V_{0i_k}$, $V_1 = V_{1i_k}$, $V = V_0 \cup V_1$, $\theta^{(0)} = \theta^{(0)}_{*i_k}$, and $\theta^{(1)} = \theta^{(1)}_{*i_k}$. Note that by Lemma 22,

$$|V| \le (\vartheta + r)^2/(2\vartheta r).$$

Consider a test setting where under the null H_0, $\beta = \beta^{(0)} = b \circ \mu^*$ and $I_V \circ \beta^{(0)} = I_V \circ \theta^{(0)}$, and under the alternative H_1, $\beta = \beta^{(1)}$ which is constructed by keeping all coordinates of $\beta^{(0)}$ unchanged, except those coordinates in V are perturbed in a way so that $I_V \circ \beta^{(1)} = I_V \circ \theta^{(1)}$. In this construction, both $\beta^{(0)}$ and $\beta^{(1)}$ are assumed as known, but we don't know which of H_0 and H_1 is true. In the literature, it is known that $\inf_{\hat{\beta}} \sum_{j \in V} P(\mathrm{sgn}(\hat{\beta}_j) \neq \mathrm{sgn}(\beta_j))$ is not smaller than the minimum sum of Type I and Type II errors associated with this testing problem.

Note that by our construction and (50), the right hand side is $\alpha^*(V_0, V_1; \Omega)$. At the same time, it is seen the optimal test statistic is $Z \equiv (\theta^{(1)} - \theta^{(0)})' X'(Y - X\beta^{(0)})$. It is seen that up to some negligible terms, $Z \sim N(0, \alpha^*(V_0, V_1; \Omega)\tau_p^2)$ under H_0, and $Z \sim N(\alpha^*(V_0, V_1; \Omega)\tau_p^2, \alpha^*(V_0, V_1; \Omega)\tau_p^2)$ under H_1. The optimal test is to reject H_0 when $Z \ge t[\alpha^*(V_0, V_1; \Omega)]^{1/2}\tau_p$ for some threshold t, and the minimum sum of Type I and Type II error is

$$\inf_t \big\{ \epsilon_p^{|V_0|} \bar{\Phi}(t) + \epsilon_p^{|V_1|} \Phi(t - [\alpha^*(V_0, V_1; \Omega)]^{1/2}\tau_p) \big\}.$$

Here, we have used $P(H_0) \sim \epsilon_p^{|V_0|}$ and $P(H_1) \sim \epsilon_p^{|V_1|}$, as a result of the Binomial structure in β. It follows that $\sum_{j \in V} P(\mathrm{sgn}(\hat{\beta}_j) \neq \mathrm{sgn}(\beta_j)) \gtrsim \inf_t \big\{ \epsilon_p^{|V_0|} \bar{\Phi}(t) + \epsilon_p^{|V_1|} \Phi(t - [\alpha^*(V_0, V_1; \Omega)]^{1/2}\tau_p) \big\}$. Using Mills' ratio and definitions, the right hand side $\ge L_p p^{-\eta(V_0, V_1; \Omega)}$, and (53) follows by recalling (50). ■

6.3 Proof of Corollaries 10, 11, and 12

When $a > a_g^*(\Omega)$, $\rho_j^*(\vartheta, r, a, \Omega)$ does not depend on a, and have an alternative expression as follows. For any subsets D and F of $\{1, 2, \ldots, p\}$, let $\omega(D, F; \Omega)$ be as in (28). Introduce $\rho(D, F; \Omega) = \rho(D, F; \vartheta, r, a, \Omega, p)$ by

$$\rho(D, F; \Omega) = \frac{(|D| + 2|F|)\vartheta}{2} + \begin{cases} \frac{1}{4}\omega(D, F; \Omega)r, & |D| \text{ is even}, \\ \frac{\vartheta}{2} + \frac{1}{4}\big[(\sqrt{\omega(D, F; \Omega)r} - \frac{\vartheta}{\sqrt{\omega(D, F; \Omega)r}})_+\big]^2, & |D| \text{ is odd}. \end{cases} \tag{54}$$

The following lemma is proved in Section 6.3.4.

Lemma 18 *Fix $m_0 \ge 1$, $(\vartheta, \kappa) \in (0, 1)^2$, $r > 0$, $c_0 > 0$, and $g > 0$ such that $\kappa > (1 - \vartheta)$. Suppose the conditions of Theorem 6 hold, and that for sufficiently large p, (26) is satisfied. Then as $p \to \infty$, $\rho_j^*(\vartheta, r, a, \Omega)$ does not depend on a, and satisfies $\rho_j^*(\vartheta, r, a, \Omega) = \min_{\{(D,F): j \in D \cup F, D \cap F = \emptyset, D \neq \emptyset, |D \cup F| \le g\}} \rho(D, F; \Omega)$.*

We now show Corollaries 10-12. Write for short $\omega = \rho(D, F; \Omega)$, $T = r/\vartheta$, and $\lambda_k^* = \lambda_k^*(\Omega)$. The following inequality is frequently used below, the proof of which is elementary so we omit it:

$$\omega \geq \lambda_k^* |D|, \qquad \text{where } k = |D| + |F|. \tag{55}$$

To show these corollaries, it is sufficient to show for all subsets D and F of $\{1, 2, \ldots, p\}$,

$$\rho(D, F; \Omega) \geq (\vartheta + r)^2/(4r), \qquad |D| \geq 1, \tag{56}$$

where $\rho(D, F; \Omega)$ is as in (54). By basic algebra, (56) is equivalent to

$$\begin{cases} (\omega T + 1/(\omega T) - 2)1\{\omega T \geq 1\} \geq (T + 1/T - 2(|D| + 2|F|)), & |D| \text{ is odd}, \\ \omega \geq \frac{2}{T}[(T + 1/T)/2 + 1 - (|D| + 2|F|)], & |D| \text{ is even}. \end{cases} \tag{57}$$

Note that when $(|D|, |F|) = (1, 0)$, this claim holds trivially, so it is sufficient to consider the case where

$$|D| + |F| \geq 2. \tag{58}$$

We now show that (57) holds under the conditions of each of corollaries.

6.3.1 Proof of Corollary 10

In this corollary, $1 < (T + 1/T)/2 \leq 3$, and if either (a) $|D| + 2|F| \geq 3$ and $|D|$ is odd or (b) $|D| + 2|F| \geq 4$ and $|D|$ is even, the right hand side of (57) ≤ 0, so the claim holds trivially. Therefore, all we need to show is the case where $(|D|, |F|) = (2, 0)$. In this case, since each off-diagonal coordinate $\leq 4\sqrt{2} - 5 \equiv \rho_0$, it follows from definitions and basic algebra that $\omega \geq 2(1 - \rho_0) = 4(3 - 2\sqrt{2})$, and (57) follows by noting that $\frac{2}{T}[(T + 1/T)/2 + 1 - (|D| + 2|F|)] = (1 - 1/T)^2 \leq 4(3 - 2\sqrt{2})$. ■

6.3.2 Proof of Corollary 11

In this corollary, $1 < (T + 1/T)/2 \leq 5$. First, we consider the case where $|D|$ is odd. By similar argument, (57) holds trivially when $|D| + 2|F| \geq 5$, so all we need to consider is the case $(|D|, |F|) = (1, 1)$ and the case $(|D|, |F|) = (3, 0)$. In both cases, $|D| + 2|F| = 3$. By (55), when $\omega T < 1$, there must be $T < 1/\min(\lambda_2^*, 3\lambda_3^*)$. By the conditions of this corollary, it follows $T < (5 + 2\sqrt{6})/4 < 3 + 2\sqrt{2}$. When $1 < T < 3 + 2\sqrt{2}$, there is $T + 1/T - 6 < 0$, and thus (57) holds for $\omega T < 1$. When $\omega T \geq 1$, (57) holds if and only if $\omega T + \frac{1}{\omega T} - 2 \geq T + 1/T - 6$. By basic algebra, this holds if

$$\omega \geq \frac{1}{4}\big[(1 - 1/T) + \sqrt{(1 - 1/T)^2 - 4/T}\big]^2. \tag{59}$$

Note that the right hand side of (59) is a monotone in T and has a maximum of $(3 + 2\sqrt{2})(5 - 2\sqrt{6})$ at $T = (5 + 2\sqrt{6})$. Now, on one other hand, when $(|D|, |F|) = (1, 0)$, by (55) and conditions of the corollary, $\omega \geq 3\lambda_3^* > (3 + 2\sqrt{2})(5 - 2\sqrt{6})$. On the other hand, when $(|D|, |F|) = (1, 1)$, by basic algebra and that each off-diagonal coordinate of $\Omega \leq \sqrt{1 + (\sqrt{6} - \sqrt{2})}/(1 + \sqrt{3/2}) \equiv \rho_1$ in magnitude, $\omega \geq 1 - \rho_1^2 = (3 + 2\sqrt{2})(5 - 2\sqrt{6})$. Combining these gives (57).

We now consider the case where $|D|$ is even. By similar argument, (57) holds when $|D| + 2|F| \geq 6$, so all we need is to show is that (57) holds for the following three cases: $(|D|, |F|) = (4, 0), (2, 1), (2, 0)$. Equivalently, this is to show that $\omega \geq \frac{2}{T}[(T + 1/T)/2 - 3]$ in the first two cases and that $\omega \geq \frac{2}{T}[(T + 1/T)/2 - 1]$ in the last case. Similarly, by the monotonicity of the right hand side of these inequalities, all we need to show is $\omega \geq 4(5 - 2\sqrt{6})$ in the first two cases, and $\omega \geq 8(5 - 2\sqrt{6})$ in the last case. Now, on one hand, using (55), $\omega \geq 4\lambda_4^*$ in the first case, and $\omega \geq 2\lambda_3^*$ in the second case, so by the conditions of the corollary, $\omega \geq 4(5 - 2\sqrt{6})$ in the first two cases. On the other hand, in the last case, since all off-diagonal coordinates of $\Omega \leq 8\sqrt{6} - 19 \equiv \rho_0$ in magnitude, and $\omega \geq 2(1 - \rho_0) = 8(5 - 2\sqrt{6})$. Combining these gives (57). ■

6.3.3 Proof of Corollary 12

Let N be the unique integer such that $2N - 1 \leq (T + 1/T)/2 < 2N + 1$. First, we consider the case where $|D|$ is odd. Note that when $|D| + 2|F| \geq 2N + 1$, the right hand side of (57) ≤ 0, so all we need to consider is the case $|D| + 2|F| \leq 2N - 1$. Write for short $k = k(D, F) = |D| + |F|$ and $j = j(D, F) = (|D| + 2|F| + 1)/2$. By (58), definitions, and that $|D| + 2|F| \leq 2N - 1$, it is seen that $2 \leq k \leq 2N - 1$ and $(k + 1)/2 \leq j \leq \min\{k, N\}$. By the condition of the corollary, $\lambda_k^* \geq \frac{(T+1/T)/2-2j+2+\sqrt{[(T+1/T)/2-2j+2]^2-1}}{T(2k-2j+1)}$. Note that $|D| = 2k-2j+1$. Combining these with (55) gives $\omega T \geq (2k-2j+1)\lambda_k^* T \geq (T+1/T)/2 - 2j + 2 + \sqrt{[(T + 1/T)/2 - 2j + 2]^2 - 1} \geq 1$. and (57) follows by basic algebra.

We now consider the case where $|D|$ is even. Similarly, the right hand side of (57) is negative when $|D| + 2|F| \geq 2(N + 1)$, so we only need to consider the case where $|D|+2|F| \leq 2N$. Similarly, write for short $k = k(D, F) = |D|+|F|$ and $j = (|D|+2|F|)/2$. It is seen that $2 \leq k \leq 2N$ and $k/2 \leq j \leq \min\{k-1, N\}$. By the conditions of the corollary, $\lambda_k^* \geq \frac{(T+1/T)/2+1-2j}{T(k-j)}$. Note that $|D| = k - j$. It follows from (55) that $\omega \geq 2(k - j)\lambda_k^* \geq \frac{2}{T}[(T + 1/T)/2 + 1 - 2j]$, and (57) follows. ■

6.3.4 Proof of Lemma 18

Let sets V_0 and V_1 and vectors $\theta^{(0)}$ and $\theta^{(1)}$ be as in Section 2.5, and let $V = V_0 \cup V_1$. By definition, $\rho_j^*(\vartheta, r, a, \Omega)$ can be written as the minimum of I and II, where $I = \min_{\{(V_0,V_1):j\in V_1\cup V_0, V_0\neq V_1\}} \eta(V_0, V_1; \Omega)$ and $II = \min_{\{V_0:j\in V_0\cup V_1, V_0=V_1\}} \eta(V_0, V_1; \Omega)$. So to show the claim, it is sufficient to show

$$I = \min_{\{(D,F):j\in D\cup F, D\cap F=\emptyset, D\neq\emptyset, |D\cup F|\leq g\}} \rho(D, F; \Omega), \qquad II \geq I. \tag{60}$$

Consider the first claim in (60). Write for short $F = F(V_0, V_1) = V_0 \cap V_1$ and $D = D(V_0, V_1) = V \setminus F$. By the definitions, $D \neq \emptyset$. The key is to show that when $|V_0 \cup V_1| \leq g$,

$$\alpha^*(V_0, V_1; \Omega) = \omega(D, F; \Omega). \tag{61}$$

Towards this end, note that by definitions, $\alpha^*(V_0, V_1; \Omega) = \alpha(\theta_*^{(0)}, \theta_*^{(1)})$, where $(\theta_*^{(0)}, \theta_*^{(1)}) = \mathrm{argmin}_{\{\theta^{(0)}\in B_{V_0}, \theta^{(1)}\in B_{V_1}\}} \alpha(\theta^{(0)}, \theta^{(1)})$. By $a > a_g^*(\Omega)$ and the way $a_g^*(\Omega)$ is defined, $(\theta_*^{(0)}, \theta_*^{(1)})$ remains as the solution of the optimization problem if we relax the conditions $\theta^{(i)} \in B_{V_i}$ to

that of $\theta^{(i)} = I_{V_i} \circ \mu^{(i)}$, where $\mu^{(i)} \in \Theta_p(\tau_p)$ (so that upper bounds on the signal strengths are removed), $i = 0, 1$. As a result,

$$\alpha^*(V_0, V_1; \Omega) = \min_{\{\theta^{(i)} \in I_{V_i} \circ \mu^{(i)}, \mu^{(i)} \in \Theta_p(\tau_p), i=0,1,\}} \alpha(\theta^{(0)}, \theta^{(1)}). \tag{62}$$

We now study (62). For short, write $\xi = \tau_p^{-1}(\theta^{(1)} - \theta^{(0)})^V$, $\Omega_{VV} = \Omega^{V,V}$, $\xi_D = \tau_p^{-1}(\theta^{(1)} - \theta^{(0)})^D$, and similarly for Ω_{DD}, Ω_{DF}, Ω_{FD}, Ω_{FF}, and ξ_F. Without loss of generality, assume the indices in D come first in V. It follows

$$\Omega_{VV} = \begin{pmatrix} \Omega_{DD} & \Omega_{DF} \\ \Omega_{FD} & \Omega_{FF} \end{pmatrix},$$

and

$$\alpha(\theta^{(0)}, \theta^{(1)}) = \xi' \Omega_{VV} \xi = \xi_D' \Omega_{DD} \xi_D + 2\xi_D' \Omega_{DF} \xi_F + \xi_F' \Omega_{FF} \xi_F. \tag{63}$$

By definitions, it is seen that there is no constraint on the coordinates of ξ_F, so to optimize the quadratic form in (61), we need to choose ξ is a way such that $\xi_F = -\Omega_{FF}^{-1}\Omega_{FD}\xi_D$, and that ξ_D minimizes $\xi_D'(\Omega_{DD} - \Omega_{DF}\Omega_{FF}^{-1}\Omega_{FD})\xi_D$, where every coordinate of $\xi_D \geq 1$ in magnitude. Combining these with (62) gives (61).

At the same time, we rewrite

$$I = \min_{\{(D,F): j \in D \cup F, D \neq \emptyset, D \cap F = \emptyset\}} \Big\{ \min_{\{(V_0,V_1): V_0 \cup V_1 = D \cup F, V_0 \cap V_1 = F\}} \eta(V_0, V_1; \Omega) \Big\}. \tag{64}$$

By similar arguments as in the proof of Lemma 22, the subsets (V_0, V_1) that achieve the minimum of $\eta(V_0, V_1; \Omega)$ must satisfy $|V_0 \cup V_1| \leq g$. Using (61), for any fixed D and F such that $|D \cup F| \leq g$, $D \neq \emptyset$ and $D \cap F = \emptyset$, the term in the big bracket on the right hand side is $\min_{\{(V_0,V_1): V_0 \cup V_1 = D \cup F, V_0 \cap V_1 = F\}} \{ \frac{(2|F| + |D|)\vartheta}{2} + \frac{\big||V_1| - |V_0|\big|\vartheta}{2} + \frac{1}{4}[(\sqrt{\omega(D,F;\Omega)r} - \frac{\big||V_1| - |V_0|\big|\vartheta}{\sqrt{\omega(D,F;\Omega)r}})_+]^2 \}$. It is worth noting that for fixed D and F, the above quantity is monotone increasing with $\big||V_1| - |V_0|\big|$. When $|D|$ is even, the minimum is achieved at (V_0, V_1) with $|V_0| = |V_1|$, and when $|D|$ is odd, the minimum is achieved at (V_0, V_1) with $\big||V_1| - |V_0|\big| = 1$, and in both cases, the minimum is $\rho(D, F; \Omega)$. Inserting this to (64), it is seen that

$$I = \min_{\{(D,F): j \in D \cup F, D \cap F = \emptyset, D \neq \emptyset, |D \cup F| \leq g\}} \rho(D, F; \Omega),$$

which is the first claim in (60).

Consider the second claim of (60). In this case, by definitions, $V_0 = V_1$ but $\mathrm{sgn}(\theta^{(0)}) \neq \mathrm{sgn}(\theta^{(1)})$. Redefine D as the subset of V_0 where the signs of the coordinates of $\theta^{(0)}$ do not equal to those of $\theta^{(1)}$, and let $F = V \setminus D$. By definitions, it is seen that $\alpha^*(V_0, V_0; \Omega) = 4\alpha^*(F, V_0; \Omega)$, where we note $D \neq \emptyset$ and $F \neq V_0$. By the definition of $\eta(V_0, V_1; \Omega)$, it follows that $\eta(V_0, V_0; \Omega) \geq \eta(F, V_0; \Omega)$, and the claim follows. ■

6.4 Proof of Lemma 15

Write for short $\rho_j^* = \rho_j^*(\vartheta, a, r, \Omega)$. To show the claim, it is sufficient to show that for any fixed $1 \leq j \leq p$,

$$P(j \notin \mathcal{U}_p^*, \beta_j \neq 0) \leq L_p[p^{-\rho_j^*} + p^{-(m_0+1)\vartheta} + o(1/p)]. \tag{65}$$

Using Lemma 14 and Ji and Jin (2011, Lemma 3.1), there is an event A_p that depends on (X,β) such that $P(A_p^c) \le o(1/p)$ and that over the event, $\Omega^{*,\delta}$ is K-sparse with $K = C(\log(p))^{1/\gamma}$, $\|\Omega^{*,\delta}-\Omega\|_\infty \le (\log(p))^{-(1-\gamma)}$, $\|(X'X-\Omega)\beta\|_\infty \le C\|\Omega\|\sqrt{2\log(p)}p^{-[(\kappa-(1-\vartheta)]/2}$, and for all subset B with size $\le m_0$, $\|G^{B,B}-\Omega^{B,B}\|_\infty \le L_p p^{-\kappa/2}$. Recall that $\mathcal{G}^{*,\delta}$ is the GOSD and $\mathcal{G}_S^{*,\delta}$ is the subgraph of the GOSD formed by the nodes in the support of β, $S(\beta) = \{1 \le j \le p : \beta_j \ne 0\}$. When $\beta_j \ne 0$, there is a unique component $\mathcal{I}_0$ such that $j \in \mathcal{I}_0 \lhd \mathcal{G}_S^{*,\delta}$ ($A \lhd B$ means that A is component or maximal connected subgraph of B). Let B_p be the event $|\mathcal{I}_0| \le m_0$. By Frieze Frieze and Molloy (1999), it is seen that $P(B_p^c \cap A_p) \le L_p p^{-(m_0+1)\vartheta}$. So to show (65), it is sufficient to show that

$$P(j \notin \mathcal{U}_p^*, j \in \mathcal{I}_0 \lhd \mathcal{G}_S^{*,\delta}, A_p \cap B_p) \le L_p p^{-\rho_j^*}. \tag{66}$$

Now, in the screening procedure, when we screen $\mathcal{I}_0$, we have $\mathcal{I}_0 = \hat{D} \cup \hat{F}$ as in (8). Since the event $\{j \notin \mathcal{U}_p^*, j \in \mathcal{I}_0 \lhd \mathcal{G}_S^{*,\delta}\}$ is contained in the event $\{T(Y,\hat{D},\hat{F}) < t(\hat{D},\hat{F})\}$, $P(j \notin \mathcal{U}_p^*, j \in \mathcal{I}_0 \lhd \mathcal{G}_S^{*,\delta}, A_p \cap B_p) \le P(T(Y,\hat{D},\hat{F}) \le t(\hat{D},\hat{F}), j \in \mathcal{I}_0 \lhd \mathcal{G}_S^{*,\delta}, A_p \cap B_p)$, where the right hand side does not exceed

$$\sum_{(\mathcal{I}_0,D,F): j \in \mathcal{I}_0 \ \&\ \mathcal{I}_0 = D \cup F \text{ is a partition}} P(T(Y,D,F) \le t(D,F), j \in \mathcal{I}_0 \lhd \mathcal{G}_S^{*,\delta}, A_p \cap B_p);$$

note that $(\mathcal{I}_0, D, F)$ do not depend on z (but may still depend on (X,β)). First, note that over the event A_p, there are at most $(eK)^{m_0+1}$ $\mathcal{I}_0$ such that $j \in \mathcal{I}_0$ and $|\mathcal{I}_0| \le m_0$. Second, note that for each $\mathcal{I}_0$, there are only finite ways to partition it to D and F. Last, note that for any fixed j and $\mathcal{I}_0$, $P(j \in \mathcal{I}_0 \lhd \mathcal{G}_S^{*,\delta}) \le \epsilon_p^{|\mathcal{I}_0|}$. Combining these observations, to show (66), it is sufficient to show that for any such triplet $(\mathcal{I}_0, D, F)$,

$$\epsilon_p^{|\mathcal{I}_0|} P\big(T(Y,D,F) \le t(D,F) \big| \{j \in \mathcal{I}_0 \lhd \mathcal{G}_S^{*,\delta}\} \cap A_p \cap B_p\big) \le L_p p^{-\rho_j^*}. \tag{67}$$

We now show (67). Since $\lambda_{m_0}^*(\Omega) \ge C > 0$, it follows from the definition of A_p and basic algebra that for any realization of (X,β) in $A_p \cap B_p$,

$$\|(G^{\mathcal{I}_0,\mathcal{I}_0})^{-1}\|_\infty \le C. \tag{68}$$

Recall that $\tilde{Y} = X'Y$ and denote for short $y = (G^{\mathcal{I}_0,\mathcal{I}_0})^{-1}\tilde{Y}^{\mathcal{I}_0}$. It is seen that

$$y = \beta^{\mathcal{I}_0} + w + rem, \qquad w \sim N(0, (G^{\mathcal{I}_0,\mathcal{I}_0})^{-1}), \qquad rem \equiv (G^{\mathcal{I}_0,\mathcal{I}_0})^{-1} G^{\mathcal{I}_0,\mathcal{I}_0^c}\beta^{\mathcal{I}_0^c}. \tag{69}$$

Since $\mathcal{I}_0$ is a component of $\mathcal{G}_S^{*,\delta}$, $(\Omega^{*,\delta})^{\mathcal{I}_0,\mathcal{I}_0^c}\beta^{\mathcal{I}_0^c} = 0$. Therefore, we can write $rem = (G^{\mathcal{I}_0,\mathcal{I}_0})^{-1}(I+II)$, where $I = (G^{\mathcal{I}_0,\mathcal{I}_0^c} - \Omega^{\mathcal{I}_0,\mathcal{I}_0^c})\beta^{\mathcal{I}_0^c}$ and $II = [\Omega^{\mathcal{I}_0,\mathcal{I}_0^c} - (\Omega^{*,\delta})^{\mathcal{I}_0,\mathcal{I}_0^c}]\beta^{\mathcal{I}_0^c}$. By the definition of A_p, $\|I\|_\infty \le C\sqrt{2\log(p)}p^{-[\kappa-(1-\vartheta)]/2}$, and $\|II\|_\infty \le \|\Omega-\Omega^{*,\delta}\|_\infty \|\beta^{\mathcal{I}_0^c}\|_\infty \le C\tau_p(\log(p))^{-(1-\gamma)}$. Combining these with (68) gives $\|rem\|_\infty \le C\tau_p(\log(p))^{-(1-\gamma)}$.

At the same time, let y_1, w_1, and rem^1 be the restriction of y, w, and rem to indices in D, correspondingly, and let $H = [G^{D,D} - G^{D,F}(G^{F,F})^{-1}G^{F,D}]$. By (69) and direct calculations, $T(Y,D,F) = y_1'Hy_1$, $y_1 \sim N(\beta^D + rem^1, H^{-1})$, and so $T(Y,D,F)$ is distributed as $\chi^2_{|D|}(\delta)$, where the non-central parameter is $(\beta^D + rem^1)'H(\beta^D + rem^1) = \delta + O((\log(p))^\gamma)$ and $\delta \equiv (\beta^D)'H\beta^D$. Since $\lambda_{m_0}^*(\Omega) \ge C$, $\delta \ge C\tau_p^2$ and is the dominating term. It follows that

$$P(T(Y,D,F) \le t(D,F) \big| \{j \in \mathcal{I}_0 \lhd \mathcal{G}_S^{*,\delta}\} \cap A_p \cap B_p\big) \lesssim P\big(\chi^2_{|D|}(\delta) \le t(D,F)\big). \tag{70}$$

Now, first, by definitions, $\delta \geq 2\omega(D,F;\Omega)r\log(p)$, so by basic knowledge on non-central χ^2,

$$P(\chi^2_{|D|}(\delta) \leq t(D,F)) \leq P(\chi^2_{|D|}(2\omega(D,F;\Omega)r\log(p)) \leq t(D,F)). \tag{71}$$

Second, recalling $t(D,F) = 2q\log(p)$, we have

$$P(\chi^2_{|D|}(2\omega(D,F;\Omega)r\log(p)) \leq t(D,F)) \leq L_p p^{-[(\sqrt{\omega(D,F;\Omega)r}-\sqrt{q})_+]^2}. \tag{72}$$

Inserting (71)-(72) into (70) and recalling $\epsilon_p = p^{-\vartheta}$,

$$\epsilon_p^{|\mathcal{I}_0|} P(T(Y,D,F) \leq t(D,F) \big| \{j \in \mathcal{I}_0 \lhd \mathcal{G}_S^{*,\delta}\} \cap A_p \cap B_p\big) \leq L_p p^{-(|\mathcal{I}_0|\vartheta + [(\sqrt{\omega(D,F;\Omega)r}-\sqrt{q})_+]^2)}. \tag{73}$$

By the choice of q and direct calculations,

$$|\mathcal{I}_0|\vartheta + [(\sqrt{\omega(D,F;\Omega)r} - \sqrt{q})_+]^2 \geq \rho(D,F;\Omega) \geq \rho_j^*, \tag{74}$$

where $\rho(D,F;\Omega)$ as in (54). Combining (73)-(74) gives (67). ■

6.5 Proof of Lemma 16

In the screening stage, suppose we pick the threshold $t(\hat{D},\hat{F}) = 2q\log(p)$ in a way such that there is a constant $q_0(\vartheta,r,\kappa) > 0$ such that $q = q(\hat{D},\hat{F}) \geq q_0(\vartheta,r,\kappa) > 0$. Recall that $\mathcal{G}^{*,\delta}$ denotes the GOSD. Let $\mathcal{U}_p^*$ be the set of retained indices. Viewing it as a subgraph of $\mathcal{G}^{*,\delta}$, $\mathcal{U}_p^*$ decomposes into many components $\mathcal{U}_p^* = \mathcal{I}^{(1)} \cup \mathcal{I}^{(2)} \ldots \cup \mathcal{I}^{(N)}$. Recall that $\tilde{Y} = X'Y$. The following lemma is proved below.

Lemma 19 *Suppose the settings and conditions are as in Lemma 16. There exists a constant $c_1 = c_1(\vartheta,r,\kappa,\gamma,A) > 0$ such that with probability at least $1 - o(1/p)$, for any component $\mathcal{I}_0 \lhd \mathcal{U}_p^*$, $\|\tilde{Y}^{\mathcal{I}_0}\|^2 \geq 2c_1|\mathcal{I}_0|\log(p)$.*

The remaining part of the proof is similar to that of Ji and Jin (2011, Lemma 2.3) so we omit it. We note that however Lemma 19 is new and needs a much harder proof. ■

6.5.1 Proof of Lemma 19

First, we need some notations. Let $\mathcal{I}_0$ be a component of $\mathcal{U}_p^*$, and let $\mathcal{I}_0^{(i)}$, $1 \leq i \leq N_0$, be all connected subgraphs with size $\leq m_0$, listed in the order as in the GS-step, where N_0 is an integer that may depend on (X,Y). For each $1 \leq i \leq N_0$, let $\mathcal{I}_0^{(i)} = \hat{D}^{(i)} \cup \hat{F}^{(i)}$ be the *exactly the same* partition when we screen $\mathcal{I}_0^{(i)}$ in the m_0-stage χ^2-screening of the GS-step. In out list, we only keep $\mathcal{I}_0^{(i)}$ such that $\hat{D}^{(i)} \cap \mathcal{I}_0 \neq \emptyset$. Since $\mathcal{I}_0$ is a component of $\mathcal{U}_p^*$ and $\mathcal{I}_0^{(i)}$ is a connected subgraph, it follows from the way that the χ^2-screening is designed and the definition of $\hat{D}^{(i)}$ that

$$\mathcal{I}_0^{(i)} \subset \mathcal{I}_0, \qquad \text{and} \qquad \hat{D}^{(i)} = \mathcal{I}_0^{(i)} \setminus (\cup_{j=1}^{i-1}\mathcal{I}_0^{(j)}), \qquad 1 \leq i \leq N_0,$$

and

$$\mathcal{I}_0 = \hat{D}^{(1)} \cup \hat{D}^{(2)} \ldots \cup \hat{D}^{(N_0)} \text{ is a partition}, \tag{75}$$

where $\hat{F}^{(1)}$ is empty.

Now, for each $1 \leq i \leq N_0$, recall that as long as $G^{\mathcal{I}_0^{(i)},\mathcal{I}_0^{(i)}}$ is non-singular, the χ^2-test score in GS is $T(Y,\hat{D}^{(i)},\hat{F}^{(i)}) = T(Y,\hat{D}^{(i)},\hat{F}^{(i)};\mathcal{I}_0^{(i)},X,p,n) = (\tilde{Y}^{\mathcal{I}_0^{(i)}})'(G^{\mathcal{I}_0^{(i)},\mathcal{I}_0^{(i)}})^{-1}\tilde{Y}^{\mathcal{I}_0^{(i)}} - (\tilde{Y}^{\hat{F}^{(i)}})'(G^{\hat{F}^{(i)},\hat{F}^{(i)}})^{-1}\tilde{Y}^{\hat{F}^{(i)}}$. By basic algebra and direct calculations, it can be verified that $T(Y,\hat{D}^{(i)},\hat{F}^{(i)}) = \|W_i\|^2$, where $W_i = W(\tilde{Y},\hat{D}^{(i)},\hat{F}^{(i)};\mathcal{I}_0^{(i)},X,p,n)$ is defined as $W_i = V_i^{-1/2}y_i$, and for short, $V_i = G^{\hat{D}^{(i)},\hat{D}^{(i)}} - G^{\hat{D}^{(i)},\hat{F}^{(i)}}(G^{\hat{F}^{(i)},\hat{F}^{(i)}})^{-1}G^{\hat{F}^{(i)},\hat{D}^{(i)}}$, $y_i = \tilde{Y}^{\hat{D}^{(i)}} - G^{\hat{D}^{(i)},\hat{F}^{(i)}}(G^{\hat{F}^{(i)},\hat{F}^{(i)}})^{-1}\tilde{Y}^{\hat{F}^{(i)}}$. At the same time, for a constant $\delta > 0$ to be determined, define $\tilde{\Omega}$ by $\tilde{\Omega}(i,j) = G(i,j)\cdot 1\{|G(i,j)| \geq \delta\}$. The definition of $\tilde{\Omega}$ is the same as that of $\Omega^{*,\delta}$, except for that the threshold δ would be selected differently. We introduce a counterpart of W_i which we call W_i^*,

$$W_i^* = V_i^{-1/2}y_i^*. \tag{76}$$

where $y_i^* = \tilde{Y}^{\hat{D}^{(i)}} - \tilde{\Omega}^{\hat{D}^{(i)},\hat{F}^{(i)}}(\tilde{\Omega}^{\hat{F}^{(i)},\hat{F}^{(i)}})^{-1}\tilde{Y}^{\hat{F}^{(i)}}$. Let $W^* = ((W_1^*)',(W_2^*)',\ldots,(W_{N_0}^*)')'$, and define $|\mathcal{I}_0|\times|\mathcal{I}_0|$ matrices H_1 and H_2 as follows: H_1 is a diagonal block-wise matrix where the i-th block is $V_i^{-1/2}$, and $H_2 = \tilde{H}_2^{\mathcal{I}_0,\mathcal{I}_0}$, where $\tilde{H}_2$ is a $p\times p$ matrix such that for every component $\mathcal{I}_0$ of $\mathcal{U}_p^*$, and $\hat{D}^{(i)}$ and $\hat{F}^{(i)}$ defined on each component, $\tilde{H}_2^{\hat{D}^{(i)},\hat{F}^{(i)}} = -(\tilde{\Omega})^{\hat{D}^{(i)},\hat{F}^{(i)}}[(\tilde{\Omega})^{\hat{F}^{(i)},\hat{F}^{(i)}}]^{-1}$, $\tilde{H}_2^{\hat{D}^{(i)},\hat{D}^{(i)}} = I_{|\hat{D}^{(i)}|}$, and that the coordinates of $\tilde{H}_2$ are zero elsewhere. Here I_k stands for $k\times k$ identity matrix. From the definitions, it is seen that

$$W^* = H_1H_2\tilde{Y}^{\mathcal{I}_0}. \tag{77}$$

Compared with W_i, W_i^* is relatively easier to study, for it induces column-sparsity of H_2. In fact, using (Ji and Jin, 2011, Lemma 2.2, 3.1), there is an event A_p that depends on (X,β) such that $P(A_p^c) \leq o(1/p^2)$ and that over the event, for all subset B with size $\leq m_0$,

$$\|G^{B,B} - \Omega^{B,B}\|_\infty \leq L_p p^{-\kappa/2}.$$

The following lemma is proved below.

Lemma 20 *Fix $\delta > 0$ and suppose the conditions in Lemma 19 hold. Over the event A_p, there is a constant $C > 0$ such that each row and column of $\tilde{H}_2$ has no more than C nonzero coordinates.*

We are now ready to show Lemma 19. To begin with, note that since we accept $\hat{D}^{(i)}$ when we graphlet-screen $\mathcal{I}_0^{(i)}$ and $|\hat{D}^{(i)}| \leq m_0$,

$$\|W_i\|^2 \geq 2(q_0/m_0)|\hat{D}^{(i)}|\log(p). \tag{78}$$

At the same time, by basic algebra, $\|W_i - W_i^*\| \leq \|V_i^{-1/2}\|\|y_i - y_i^*\|$, and $\|y_i - y_i^*\| \leq \|G^{\hat{D}^{(i)},\hat{F}^{(i)}}(G^{\hat{F}^{(i)},\hat{F}^{(i)}})^{-1} - (\tilde{\Omega})^{\hat{D}^{(i)},\hat{F}^{(i)}}((\tilde{\Omega})^{\hat{F}^{(i)},\hat{F}^{(i)}})^{-1}\|_\infty \cdot \|\tilde{Y}^{\hat{F}^{(i)}}\|$. First, since $\lambda_{m_0}^*(\Omega) \geq C$, it is seen that over the event A_p, $\|V_i^{-1/2}\| \leq C$. Second, by similar reasons, it is not hard to see that except for probability $o(p^{-2})$, $\|G^{\hat{D}^{(i)},\hat{F}^{(i)}}(G^{\hat{F}^{(i)},\hat{F}^{(i)}})^{-1} - (\tilde{\Omega})^{\hat{D}^{(i)},\hat{F}^{(i)}}((\tilde{\Omega})^{\hat{F}^{(i)},\hat{F}^{(i)}})^{-1}\|_\infty \leq C\delta^{1-\gamma}$, and $\|\tilde{Y}^{\hat{F}^{(i)}}\| \leq C\sqrt{\log(p)} \leq C\tau_p$. Combining these gives

$$\|W_i - W_i^*\| \leq C\delta^{1-\gamma}\tau_p.$$

Inserting this to (78), if we choose δ to be a sufficiently small constant, $\|W_i^*\|^2 \geq \frac{1}{2}\|W_i\|^2 \geq (q_0/m_0)|\hat{D}^{(i)}|\log(p)$.

At the same time, by definitions, it follows from $\|V_i^{-1/2}\| \leq C$ that $\|H_1\| \leq C$. Also, since over the event A_p, each coordinate of H_2 is bounded from above by a constant in magnitude, it follows from Lemma 20 that $\|H_2\| \leq C$. Combining this with (75)-(77), it follows from basic algebra that except for probability $o(p^{-2})$, $(q_0/m_0)|\mathcal{I}_0|\log(p) \leq \|W^*\|^2 \leq \|H_1H_2\tilde{Y}^{\mathcal{I}_0}\|^2 \leq C\|\tilde{Y}^{\mathcal{I}_0}\|^2$, and the claim follows since m_0 is a fixed integer. ■

6.5.2 Proof of Lemma 20

By definitions, it is equivalent to show that over the event A_p, each row and column of $\tilde{H}_2$ has finite nonzero coordinates. It is seen that each row of $\tilde{H}_2$ has $\leq m_0$ nonzeros, so all we need to show is that each column of $\tilde{H}_2$ has finite nonzeros.

Towards this end, we introduce a new graph $\tilde{\mathcal{G}} = (V, E)$, where $V = \{1, 2, \ldots, p\}$ and nodes i and j are connected if and only if $|\tilde{\Omega}(i,j)| \neq 0$. This definition is the same as GOSD, except that $\Omega^{*,\delta}$ is substituted by $\tilde{\Omega}$. It is seen that over the event A_p, for any $\Omega \in \mathcal{M}_p^*(\gamma, c_0, g, A)$, $\tilde{\mathcal{G}}$ is K-sparse with $K \leq C\delta^{-1/\gamma}$. The key for the proof is to show that for any $k \neq \ell$ such that $\tilde{H}_2(k,\ell) \neq 0$, there is a path with length $\leq (m_0 - 1)$ in $\tilde{\mathcal{G}}$ that connects k and ℓ.

To see the point, we note that when $\tilde{H}_2(k,\ell) \neq 0$, there must be an i such that $k \in \hat{D}^{(i)}$ and $\ell \in \hat{F}^{(i)}$. We claim that there is a path in $\mathcal{I}_0^{(i)}$ (which is regarded as a subgraph of $\tilde{\mathcal{G}}$) that connects k and ℓ. In fact, if k and ℓ are not connected in $\mathcal{I}_0^{(i)}$, we can partition $\mathcal{I}_0^{(i)}$ into two separate sets of nodes such that one contains k and the other contains ℓ, and two sets are disconnected. In effect, both the matrix $\tilde{\Omega}^{\hat{D}^{(i)},\hat{D}^{(i)}}$ and $\tilde{\Omega}^{\hat{D}^{(i)},\hat{F}^{(i)}}$ can be visualized as two by two blockwise matrix, with off-diagonal blocks being 0. As a result, it is seen that $\tilde{H}_2(k,\ell) = 0$. This contradiction shows that whenever $\tilde{H}_2(k,\ell) \neq 0$, k and ℓ are connected by a path in $\mathcal{I}_0^{(i)}$. Since $|\mathcal{I}_0^{(i)}| \leq m_0$, there is a path $\leq m_0 - 1$ in $\tilde{\mathcal{G}}$ that connects k and ℓ where $k \neq \ell$.

Finally, since $\tilde{\mathcal{G}}$ is K-sparse with $K = C\delta^{-1/\gamma}$, for any fixed ℓ, there are at most finite k connecting to ℓ by a path with length $\leq (m_0 - 1)$. The claim follows. ■

6.6 Proof of Theorem 13

Since σ is known, for simplicity, we assume $\sigma = 1$. First, consider (38). By Theorem 8 and (54), $\rho_{gs} = \min_{\{(D,F):D\cap F=\emptyset, D\neq\emptyset, D\cup F\subset\{1,2\}\}} \rho(D,F;\Omega)$, where we have used that G is a diagonal block-wise matrix, each block is the same 2×2 matrix. To calculate $\rho(D,F;\Omega)$, we consider three cases (a) $(|D|,|F|) = (2,0)$, (b) $(|D|,|F|) = (1,0)$, (c) $(|D|,|F|) = (1,1)$. By definitions and direct calculations, it is seen that $\rho(D,F;\Omega) = \vartheta + [(1-|h_0|)r]/2$ in case (a), $\rho(D,F;\Omega) = (\vartheta + r)^2/(4r)$ in case (b), and $\rho(D,F;\Omega) = 2\vartheta + [(\sqrt{(1-h_0^2)r} - \vartheta/\sqrt{(1-h_0^2)r})_+]^2/4$ in case (c). Combining these gives the claim.

Next, consider (39). Similarly, by the block-wise structure of G, we can restrict our attention to the first two coordinates of β, and apply the subset selection to the size 2 subproblem where the Gram matrix is the 2×2 matrix with 1 on the diagonals and h_0 on the off-diagonals. Fix $q > 0$, and let the tuning parameter $\lambda_{ss} = \sqrt{2q_{ss}\log(p)}$. Define $f_{ss}^{(1)}(q) = \vartheta + [(\sqrt{r} - \sqrt{q})_+]^2$, $f_{ss}^{(2)}(q) = 2\vartheta + [(\sqrt{r(1-h_0^2)} - \sqrt{q})_+]^2$, and $f_{ss}^{(3)}(q) = 2\vartheta +$

$2[(\sqrt{r(1-|h_0|)}-\sqrt{q})_+]^2$, where $x_+ = \max\{x, 0\}$. The following lemma is proved below, where the key is to use Ji and Jin (2011, Lemma 4.3).

Lemma 21 *Fix $q > 0$ and suppose the conditions in Theorem 13 hold. Apply the subset selection to the aforementioned size 2 subproblem with $\lambda_{ss} = \sqrt{2q\log(p)}$. As $p \to \infty$, the worst-case Hamming error rate is $L_p p^{-f_{ss}(q)}$, where $f_{ss}(q) = f_{ss}(q, \vartheta, r, h_0) = \min\big\{\vartheta + (1 - |h_0|)r/2, q, f_{ss}^{(1)}(q), f_{ss}^{(2)}(q), f_{ss}^{(3)}(q)\big\}$.*

By direct calculations, $\rho_{ss}(\vartheta, r, h_0) = \max_{\{q>0\}} f_{ss}(\vartheta, r, h_0)$ and the claim follows.

Last, consider (40). The proof is very similar to that of the subset selection, except for that we need to use Ji and Jin (2011, Lemma 4.1), instead of Ji and Jin (2011, Lemma 4.3). For this reason, we omit the proof. ■

6.6.1 Proof of Lemma 21

By the symmetry in (35)-(36) when G is given by (37), we only need to consider the case where $h_0 \in [0, 1)$ and $\beta_1 \geq 0$. Introduce events, $A_0 = \{\beta_1 = \beta_2 = 0\}$, $A_1 = \{\beta_1 \geq \tau_p, \beta_2 = 0\}$, $A_{21} = \{\beta_1 \geq \tau_p, \beta_2 \geq \tau_p\}$, $A_{22} = \{\beta_1 \geq \tau_p, \beta_2 \leq -\tau_p\}$, $B_0 = \{\hat\beta_1 = \hat\beta_2 = 0\}$, $B_1 = \{\hat\beta_1 > 0, \hat\beta_2 = 0\}$, $B_{21} = \{\hat\beta_1 > 0, \hat\beta_2 > 0\}$ and $B_{22} = \{\hat\beta_1 > 0, \hat\beta_2 < 0\}$. It is seen that the Hamming error

$$= L_p(I + II + III), \tag{79}$$

where $I = P(A_0 \cap B_0^c)$, $II = P(A_1 \cap B_1^c)$ and $III = P(A_{21} \cap B_{21}^c) + P(A_{22} \cap B_{22}^c)$.

Let H be the 2×2 matrix with ones on the diagonals and h_0 on the off-diagonals, $\alpha = (\beta_1, \beta_2)'$, and $w = (\tilde Y_1, \tilde Y_2)$, where we recall $\tilde Y = X'Y$. It is seen that $w \sim N(H\alpha, H)$. Write for short $\lambda = \sqrt{2q\log(p)}$. Define regions on the plane of $(\tilde Y_1, \tilde Y_2)$, $D_0 = \{\max(|\tilde Y_1|, |\tilde Y_2|) > \lambda$ *or* $\tilde Y_1^2 + \tilde Y_2^2 - 2h_0\tilde Y_1\tilde Y_2 > 2\lambda^2(1-h_0^2)\}$, $D_1 = \{|\tilde Y_1| < \lambda\,,\ \tilde Y_1 < \tilde Y_2$ *or* $|\tilde Y_2 - h_0\tilde Y_1| > \lambda\sqrt{1-h_0^2}\}$, $D_{21} = \{\tilde Y_2 - h_0\tilde Y_1 < \lambda\sqrt{1-h_0^2}$ *or* $\tilde Y_1 - h_0\tilde Y_2 < \lambda\sqrt{1-h_0^2}\}$ and $D_{22} = \{\tilde Y_2 - h_0\tilde Y_1 > -\lambda\sqrt{1-h_0^2}$ *or* $\tilde Y_1 - h_0\tilde Y_2 > \lambda\sqrt{1-h_0^2}$ *or* $\tilde Y_1^2 + \tilde Y_2^2 - 2h_0\tilde Y_1\tilde Y_2 < 2\lambda^2(1-h_0^2)\}$. Using (Ji and Jin, 2011, Lemma 4.3), we have $B_0^c = \{(\tilde Y_1, \tilde Y_2)' \in D_0\}$, $B_1^c = \{(\tilde Y_1, \tilde Y_2)' \in D_1\}$, $B_{21}^c = \{(\tilde Y_1, \tilde Y_2)' \in D_{21}\}$, and $B_{22}^c = \{(\tilde Y_1, \tilde Y_2)' \in D_{22}\}$. By direct calculation and Mills' ratio, it follows that for all $\mu \in \Theta_p(\tau_p)$,

$$I = L_p \cdot (P(N(0,1) > \lambda) + P(\chi_2^2 > 2\lambda^2)) = L_p \cdot p^{-q}, \tag{80}$$

$$II \leq L_p \cdot P(N((\tau_p, h_0\tau_p)', H) \in D_1) = L_p \cdot p^{-\vartheta - \min[(\sqrt{r}-\sqrt{q})^2, (1-h_0)r/2, q]}, \tag{81}$$

and when $\beta_1 = \tau_p$ and $\beta_2 = 0$, the equality holds in (81). At the same time, note that over the event A_{21}, the worst case scenario, is where $\beta_1 = \beta_2 = \tau_p$. In such a case, $(\tilde Y_1, \tilde Y_2)' \sim N(((1+h_0)\tau_p, (1+h_0)\tau_p)', H)$. Combining this with Mills' ratio, it follows that for all $\mu \in \Theta_p(\tau_p)$,

$$P(A_{21} \cap B_{21}^c) = P((\tilde Y_1, \tilde Y_2)' \in D_{21}) \leq L_p \cdot p^{-2\vartheta - (\sqrt{r(1-h_0^2)}-\sqrt{q})_+^2}, \tag{82}$$

and the equality holds when $\beta_1 = \beta_2 = \tau_p$. Similarly, note that over the event A_{22}, in the worst case scenario, $\beta_1 = -\beta_2 = \tau_p$. In such a case, $(\tilde Y_1, \tilde Y_2)' \sim N(((1-h_0)\tau_p, -(1-h_0)\tau_p)', H)$. Combining this with Mills' ratio, it follows that for all $\mu \in \Theta_p(\tau_p)$,

$$P(A_{22} \cap B_{22}^c) = P((\tilde Y_1, \tilde Y_2)' \in D_{22}) \leq L_p \cdot p^{-2\vartheta - \min([(\sqrt{r(1-h_0^2)}-\sqrt{q})_+]^2, 2\{[\sqrt{r(1-h_0)}-\sqrt{q}]_+\}^2)}, \tag{83}$$

and the equality holds when $\beta_1 = -\beta_2 = \tau_p$. Inserting (80)-(83) into (79) gives the claim. ■

6.7 Lemma 22 and the Proof

Lemma 22 *Let (V_{0j}^*, V_{1j}^*) be defined as in (25). If the conditions of Theorem 6 hold, then $\max\{|V_{0j}^* \cup V_{1j}^*|\} \leq (\vartheta + r)^2/(2\vartheta r)$.*

Proof. Let $V_0 = \emptyset$ and $V_1 = \{j\}$. It is seen that $\alpha^*(V_0, V_1; \Omega) = 1$, and $\eta(V_0, V_1; \Omega) \leq (\vartheta+r)^2/(4r)$. Using this and the definitions of V_{0j}^* and V_{1j}^*, $\max\{|V_{0j}^*|, |V_{1j}^*|\}\vartheta \leq (\vartheta+r)^2/(4r)$ and the claim follows. ■

Acknowledgments

We thank Pengsheng Ji and Tracy Ke for help and pointers. JJ and QZ were partially supported by NSF CAREER award DMS-0908613. QZ was also partially supported by the NIH Grant P50-MH080215-02. CHZ was partially supported by the NSF Grants DMS-0906420, DMS-1106753 and NSA Grant H98230-11-1-0205. The research was supported in part by the computational resources on PittGrid.

References

H. Akaike. A new look at the statistical model identification. *IEEE Trans. Automatic Control*, 19(6):716–723, 1974.

W. Bajwa, J. Haupt, G. Raz, S. Wright and R. Nowak. Toeplitz-structured compressed sensing matrices. *Proc. IEEE Stat. Sig. Proc. Workshop*, pages 294–298, 2007.

P. Bickel and E. Levina. Covariance regularization by thresholding. *Ann. Statist.*, 36(6):2577–2604, 2008.

T. Cai, W. Liu and X. Luo. A constrained ℓ^1 minimization approach to sparse precision matrix estimation. *J. Amer. Statist. Assoc.*, 106(494):594-607, 2010.

E. Candes and Y. Plan. Near-ideal model selection by ℓ^1-minimization. *Ann. Statist.*, 37(5):2145–2177, 2009.

E. Candes and T. Tao. The Dantzig selector: statistical estimation when p is much larger than n (with discussion). *Ann. Statist.*, 35(6):2313–2404, 2007.

S. Chen, D. Donoho and M. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Computing*, 20(1):33–61, 1998.

I. Dinur and K. Nissim. Revealing information while preserving privacy. *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 202–210, ACM Press, 2003.

D. Donoho. For most large underdetermined systems of linear equations the minimal ℓ^1-norm solution is also the sparsest solution. *Comm. Pure Appl. Math.*, 59(7):907–934, 2006.

D. Donoho. Compressed sensing. *IEEE Trans. Inform. Theory*, 52(4):1289–1306, 2006.

D. Donoho and X. Huo. Uncertainty principle and ideal atomic decomposition. *IEEE Trans. Inform. Theory*, 47(7):2845-2862, 2001.

D. Donoho and J. Jin. Higher criticism for detecting sparse heterogeneous mixtures. *Ann. Statist.*, 32(3):962–994, 2004.

D. Donoho and J. Jin. Higher criticism thresholding: optimal feature selection when useful features are rare and weak. *Proc. Natl. Acad. Sci.*, 105(39):14790–14795, 2008.

D. Donoho and P. Stark. Uncertainty principles and signal recovery. *SIAM J. Appl. Math.*, 49(3):906-931, 1989.

B. Efron, H. Hastie, I. Johnstone and R. Tibshirani. Least angle regression (with discussion). *Ann. Statist.*, 32(2):407-499, 2004.

J. Fan, X. Han and W. Gu. Estimating false discovery proportion under arbitrary covariance dependence. *J. Amer. Statist. Assoc.*, 107(499):1019-1035, 2011.

J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *J. Amer. Statist. Assoc.*, 96(456):1349–1360, 2001.

J. Fan and J. Lv. Sure independence screening for ultrahigh dimensional feature space. *J. Roy. Soc. Statist. B*, 70(5):849–911, 2008.

D. Foster and E. George. The risk inflation criterion for multiple regression. *Ann. Statist.*, 22(4):1947–1975, 1994.

J. Friedman, T. Hastie and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

J. Friedman, T. Hastie and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *J. Statist. Soft.*, 33(1):1-22, 2010.
`http://cran.r-project.org/web/packages/glmnet/index.html`

A. Frieze and M. Molloy. Splitting an expander graph. *J. Algorithms*, 33(1):166–172, 1999.

C. Genovese, J. Jin, L. Wasserman and Z. Yao. A comparison of the lasso and marginal regression. *J. Mach. Learn. Res.* 13:2107-2143, 2012.

J. Hopcroft and R. Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Commun. ACM*, 16(6):372-378, 1973.

R. Horn and C. Johnson. Matrix analysis. *Cambridge University Press*, 1990.

Y. Ingster, C. Pouet and A. Tsybakov. Classification of sparse high-dimensional vectors. *Phil. Trans. R. Soc. A*, 367(1906):4427-4448, 2009.

E. Ising. A contribution to the theory of ferromagnetism. *Z.Phys.*, 31(1):253-258, 1925.

P. Ji and J. Jin. UPS delivers optimal phase diagram in high dimensional variable selection. *Ann. Statist.*, 40(1):73-103, 2011.

J. Jin. Impossibility of successful classification when useful features are rare and weak. *Proc. Natl Acad. Sci.*, 106(22):8859-8864, 2009.

C. Li and H. Li. Network-constrained regularization and variable selection for analysis of genomic data. *Bioinformatics*, 24(9):1175-1182, 2011.

N. Meinsausen and P. Buhlmann. High dimensional graphs and variable selection with the lasso. *Ann. Statist.*, 34(3):1436–1462, 2006.

N. Meinsausen and P. Buhlmann. Stability Selection (with discussion). *J. Roy. Soc. Statist. B*, 72(4):417-473, 2010.

N. Meinsausen and J. Rice. Estimating the proportion of false null hypotheses among a large number of independently tested hypotheses. *Ann. Statist.*, 34(1):323-393, 2006.

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

J. Peng, P. Wang, N. Zhou and J. Zhu. Partial correlation estimation by joint sparse regression model. *J. Amer. Statist. Assoc.*, 104(486):735-746, 2009.

G. Schwarz. Estimating the dimension of a model. *Ann. Statist.*, 6(2):461–464, 1978.

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. B*, 58(1):267–288, 1996.

M. Wainwright. Sharp threshold for high-dimensional and noisy recovery of sparsity using ℓ_1 constrained quadratic programming (lasso). *IEEE Trans. Inform. Theory*, 52(5):2183–2202, 2009.

Z. Wang, H. Liu and T. Zhang. Optimal computational and statistical rates of convergence for sparse nonconvex learning problems. *ArXiv E-print*, arXiv:1306.4960, 2013.

L. Wasserman and K. Roeder. High-dimensional variable selection. *Ann. Statist.*, 37(5):2178–2201, 2009.

F. Ye and C.-H. Zhang. Rate minimaxity of the lasso and Dantzig selection for the ℓ_q loss in ℓ_r balls. *J. Mach. Learn. Res.*, 11:3519-3540, 2010.

C.-H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *Ann. Statist.*, 38(2):894-942, 2010.

C.-H. Zhang and T. Zhang. A general theory of concave regularization for high dimensional sparse estimation problems. *Statist. Sci.*, 27(4):576-593, 2010.

T. Zhang. Adaptive forward-backward greedy algorithm for learning sparse representations. *IEEE Trans. Inform. Theory.*, 57(7):4689-4708, 2011.

P. Zhao and B. Yu. On model selection consistency of lasso. *J. Mach. Learn. Res.*, 7:2541–2567, 2006.

H. Zou. The adaptive lasso and its oracle properties. *J. Amer. Statist. Assoc.*, 101(476):1418–1429, 2006.