The Dissertation Committee for Shanshan Wu
certifies that this is the approved version of the following dissertation:

Unsupervised Learning for Large-Scale Data

Committee:

Sujay Sanghavi, Supervisor

Georgios-Alex Dimakis, Co-Supervisor

Constantine Caramanis

Adam R. Klivans

Rachel A. Ward

Unsupervised Learning for Large-Scale Data

by

Shanshan Wu

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2019

Dedicated to my family.

Acknowledgments

This thesis would not be possible without the support and encouragement from my supervisors Sujay Sanghavi and Alex Dimakis. I learned many things from them, among which the most important thing is their attitudes towards research. I would like to thank Sujay for encouraging me to attend different machine learning conferences, workshops, and seminars, especially during the early years of my PhD. I learned a lot from the talks given by senior researchers. Attending conferences and workshops also provide me opportunities to make new friends with people having similar research interests. I would like to thank Alex for all the meetings we had together, including both individual meetings and group meetings. Alex is always approachable and enthusiastic about teaching and research. I have benefited a lot from his writing and presentation skills. Finally, I would like to thank both Sujay and Alex for letting me choose research topics that are aligned with my own interests, and at the same time encouraging me to be open-minded. My PhD would be dramatically more difficult without their guidance and support.

I would like to thank my committee members Constantine Caramanis, Adam Klivans, and Rachel Ward. I have benefited a lot from their views and insights in the broad areas of machine learning and optimization. My special thanks go to Adam for his paper on learning graphical models, which inspired

me to work on this important problem.

I have learned a lot from the graduate courses taken during my PhD years. I thank Sanjay Shakkottai for his well-prepared courses on probability and stochastic processes. I am grateful for all I learned from the various algorithms and theory courses taught by Eric Price, Vijaya Ramachandran, Evdokia Nikolova, Haris Vikalo, Sujay Sanghavi, Constantine Caramanis, Alex Dimakis, and Adam Klivans. I enjoyed the data mining foundation course offered by Joydeep Ghosh.

Special thanks to all the WNCG staff for making my life easier. I thank Melanie Gulick and Melody Singleton for always being patient when listening to my questions and giving suggestions.

I have three wonderful internship experiences during my PhD studies. I would like to thank Hyokun Yun, my internship mentor at Amazon, for teaching me the basics of natural language processing. I am grateful for all the discussions I had with my collaborators at Google, including but not limited to Felix X. Yu, Dan Holtmann-Rice, Sanjiv Kumar, Dmitry Storcheus, Afshin Rostamizadeh, Petros Mol, and Natalia Ponomareva.

I would like to thank all my labmates and friends for being an important part of my PhD journey. I cherish all the memories they made along the way.

Last but not least, I would like to express my deepest gratitude to my parents and husband for their unconditional love and support.

Unsupervised Learning for Large-Scale Data

Publication No. _____

Shanshan Wu, Ph.D.
The University of Texas at Austin, 2019

Supervisor: Sujay Sanghavi
Co-Supervisor: Georgios-Alex Dimakis

Unsupervised learning involves inferring the inherent structures or patterns from unlabeled data. Since there is no label information, the fundamental challenge of unsupervised learning is that the objective function is not explicitly defined. The ubiquity of large-scale datasets adds another layer of complexity to the overall learning problem. When the data size or dimension is large, even algorithms with quadratic runtime may be prohibitive.

This thesis presents four large-scale unsupervised learning problems. We start with two density estimation problems: given samples from a one-layer ReLU generative model or a discrete pairwise graphical model, the goal is to recover the parameters of the generative model. We then move to representation learning of high-dimensional sparse data coming from one-hot encoded categorical features. We assume that there are additional but a-priori unknown structures in their support. The goal is to learn a lossless low-dimensional embedding for the given data. Our last problem is to compute

low-rank approximations of a matrix product given the individual matrices. We are interested in the setting where the matrices are too large and can only be stored in the disk. For every problem presented in this thesis, we (i) design novel and efficient algorithms to capture the inherent structure from data in an unsupervised manner; (ii) establish theoretical guarantees and compare the empirical performance with the state-of-the-art methods; and (iii) provide source code to support our experimental findings.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine learning problems can be classified as: supervised, unsupervised, and semi-supervised, based on whether the training data are labeled, unlabeled, or partially labeled. This thesis focuses on unsupervised learning, and in particular we will consider two important types of unsupervised learning problems: (i) density estimation, and (ii) representation learning. The goal of density estimation is to construct a probability distribution given the observed samples. In representation learning, the goal is to extract low-dimensional features to capture the inherent structure in the high-dimensional data. Both density estimation and representation learning have broad applications in machine learning and statistics.

The rest of this thesis is organized as follows. In each chapter, we consider a different unsupervised learning problem, and for each problem, we present new algorithms, theorectical analysis, and experimental results. We also provide source code to support the empirical findings.

In Chapter 2, we consider the problem of estimating the parameters of a d-dimensional rectified Gaussian distribution from i.i.d. samples. A rectified Gaussian distribution is defined by passing a standard Gaussian distribution

through a one-layer ReLU neural network (Definition 2.1.1). We give a simple algorithm to estimate the parameters (i.e., the weight matrix W and bias vector b of the ReLU neural network) up to an error $\epsilon\|W\|_F$ using $\widetilde{O}(1/\epsilon^2)$ samples and $\widetilde{O}(d^2/\epsilon^2)$ time (Theorem 2.4.5). This implies that we can estimate the distribution up to ϵ in total variation distance using $\widetilde{O}(\kappa^2 d^2/\epsilon^2)$ samples, where κ is the condition number of the covariance matrix (Corollary 2.4.6). Our only assumption is that the bias vector is non-negative. Without this non-negativity assumption, we show that estimating the bias vector within any error requires the number of samples at least exponential in the infinity norm of the bias vector (Claim 2).

Our algorithm is based on the key observation that vector norms and pairwise angles can be estimated separately (Algorithm 1). We use a recent result of learning from truncated samples (Algorithm 3). We also prove sample complexity lower bounds and the lower bound implies that our algorithm is optimal for parameter estimation (Theorem 2.5.1 and 2.5.2). Finally, we show an interesting connection between learning a two-layer generative model and non-negative matrix factorization (Claim 3). Experimental results are provided to support our analysis (Section 2.6).

In Chapter 3, we characterize the effectiveness of a classical algorithm for recovering the Markov graph of a general discrete pairwise graphical model from i.i.d. samples. The algorithm is (appropriately regularized) maximum conditional log-likelihood, which involves solving a convex program for each node; for Ising models this is ℓ_1-constrained logistic regression, while for more

general alphabets an $\ell_{2,1}$ group-norm constraint needs to be used. We show that this algorithm can recover any arbitrary discrete pairwise graphical model, and also characterize its sample complexity as a function of model width, alphabet size, edge parameter accuracy, and the number of variables (Theorem 3.2.1 and 3.2.3). We show that along every one of these axes, it matches or improves on all existing results and algorithms for this problem (Table 3.1).

Our analysis applies a sharp generalization error bound for logistic regression when the weight vector has an ℓ_1 constraint (or $\ell_{2,1}$ constraint) and the sample vector has an ℓ_∞ constraint (or $\ell_{2,\infty}$ constraint). We also show that the proposed convex programs can be efficiently solved in $\tilde{O}(n^2)$ running time (where n is the number of variables) under the same statistical guarantees (Section 3.2.3). We provide experimental results to support our analysis (Section 3.4).

In Chapter 4, we consider the problem of learning lossless low-dimensional embeddings for high-dimensional sparse data. Linear encoding of sparse vectors is widely popular, but is commonly data-independent – missing any possible extra (but a-priori unknown) structure beyond sparsity. We present a new method to learn linear encoders that adapt to data, while still performing well with the widely used ℓ_1 decoder. The convex ℓ_1 decoder prevents gradient propagation as needed in standard gradient-based training. Our method is based on the insight that unrolling the convex decoder into T projected sub-gradient steps can address this issue (Section 4.3). Our method can be seen as a data-driven way to learn a compressed sensing measurement matrix.

We compare the empirical performance of ten algorithms over six sparse datasets (three synthetic and three real). Our experiments show that there is indeed additional structure beyond sparsity in the real datasets. Our method is able to discover it and exploit it to create excellent reconstructions with fewer measurements (by a factor of 1.1-3x) compared to the previous state-of-the-art methods (Figure 4.2 and Figure 4.3). We illustrate an application of our method in learning label embeddings for extreme multi-label classification (Section 4.5). Our experiments show that our method is able to match or outperform the precision scores of SLEEC (Table 4.5), which is one of the state-of-the-art embedding-based approaches for extreme multi-label learning.

In Chapter 5, we present a new algorithm for computing a low rank approximation of the product $A^T B$ by taking only a single pass of the two matrices A and B (Algorithm 7). The straightforward way to do this is to (a) first sketch A and B individually, and then (b) find the top components using PCA on the sketch. Our algorithm in contrast retains additional summary information about A, B (e.g. row and column norms etc.) and uses this additional information to obtain an improved approximation from the sketches. Our key idea *Rescaled JL Embedding* is based on the observation that the vector length can be preserved via a simple rescaling operation (Definition 5.5). We prove that rescaled JL embedding outperforms the standard JL embedding in terms of preserving the inner product of two vectors (Section 5.3). Finally, we provide experimental results from an Apache Spark implementation that shows better computational and statistical performance on real-world and synthetic

evaluation datasets (Section 5.4).

Chapter 2

Learning One-Layer ReLU Generative Model

2.1 Introduction

Estimating a high-dimensional distribution from observed samples is a fundamental problem in machine learning and statistics. A popular recent generative approach is to model complex distributions by passing a simple distribution (typically a standard Gaussian) through a neural network. Parameters of the neural network are then learned from data. Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) and Variational Auto-Encoders (VAEs) (Kingma and Welling, 2013) are built on this method of modeling high-dimensional distributions.

Current methods for learning such deep generative models do not have provable guarantees or sample complexity bounds. In this project we obtain the first such results for a single-layer ReLU generative model. Specifically, we study the following problem: Assume that the latent variable z is selected from a standard Gaussian which then drives the generation of samples from

Chapter 2 is based on material from (Wu et al., 2019a). The author of this dissertation is the leading author of (Wu et al., 2019a), and contributed to the idea, the analysis, the implementation and experiments, and the writing of the paper. Source code can be found at `https://github.com/wushanshan/densityEstimation`.

a one-layer ReLU activated neural network with weights W and bias b. We observe the output samples (but *not* the latent variable realizations z) and we would like to provably learn the parameters W and b. More formally:

Definition 2.1.1. Let $W \in \mathbb{R}^{d\times k}$ be the weight matrix, and $b \in \mathbb{R}^d$ be the bias vector. We define $\mathcal{D}(W, b)$ as the distribution[1] of the random variable $x \in \mathbb{R}^d$ generated as follows:

$$x = \mathrm{ReLU}(Wz + b), \text{ where } z \sim \mathcal{N}(0, I_k). \tag{2.1}$$

Here z is a Gaussian random variable in $\mathbb{R}^k$, and I_k is a k-by-k identity matrix.

Given n samples $x_1, x_2, ..., x_n$ from some $\mathcal{D}(W, b)$ with unknown parameters W and b, the goal is to estimate W and b from the given samples. Since the ReLU operation is not invertible[2], estimating W and b via maximum likelihood is often intractable. In fact, as we will show in Section 2.4.1, even in the one-dimensional setting, the negative log-likelihood at a given sample is a non-convex function of the parameters.

2.1.1 Our Contributions

- We provide a simple and novel algorithm to estimate the parameters of $\mathcal{D}(W, b)$ from i.i.d. samples, under the assumption that b is non-negative.

[1] It is also called as a rectified Gaussian distribution, and can be used in non-negative factor analysis (Harva and Kabán, 2007).

[2] If the activation function σ (e.g., sigmoid, leaky ReLU, etc.) is invertible, then $\sigma^{-1}(X) \sim \mathcal{N}(b, WW^T)$. In that case the problem becomes learning a Gaussian from samples.

Our algorithm (Algorithm 1) takes two steps. In Step 1, we estimate b and the row norms of W using a recent result on estimation from truncated samples (Algorithm 2). In Step 2, we estimate the angles between any two row vectors of W using a simple geometric result (Fact 1).

- We prove that the proposed algorithm needs $\widetilde{O}(1/\epsilon^2)$ samples and $\widetilde{O}(d^2/\epsilon^2)$ time, in order to estimate the parameter WW^T (reps. b) within an error $\epsilon\|W\|_F^2$ (resp. $\epsilon\|W\|_F$) (see Theorem 2.4.5 for the precise bound). This implies that (for the non-degenerate case) the total variation distance between the learned distribution and the ground truth is within an error ϵ given $\widetilde{O}(\kappa^2 d^2/\epsilon^2)$ samples, where κ is the condition number of WW^T (Corollary 2.4.6).

- Without the non-negativity assumption on b, we show that estimating the parameters of $\mathcal{D}(W, b)$ within an error ϵ requires $\Omega(\exp(\|b\|_\infty^2))$ samples (Claim 2). Even when the bias vector b has negative components, our algorithm can still be used to recover part of the parameters with small amount of samples (Section 2.7.1).

- We prove two lower bounds on the sample complexity. The first lower bound (Theorem 2.5.1) says that $\Omega(1/\epsilon^2)$ samples are required in order to estimate b up to error $\epsilon\|W\|_F$, which implies that our algorithm is optimal in estimating the parameters. The second lower bound (Theorem 2.5.2) says that $\Omega(d/\epsilon^2)$ samples are required to estimate the distribution up to TV distance ϵ.

- We empirically evaluate our algorithm in terms of its dependence over the number of samples, dimension, and condition number (Figure 2.1). The empirical results are consistent with our analysis.

- We provide a new algorithm to estimate the parameters of a two-layer generative model (Algorithm 4). Our algorithm uses ideas from non-negative matrix factorization and the separability assumption (Claim 3).

2.1.2 Notation

We use capital letters to denote matrices and lower-case letters to denote vectors. We use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. For a vector $x \in \mathbb{R}^d$, we use $x(i)$ to denote its i-th coordinate. The ℓ_p norm of a vector is defined as $\|x\|_p = (\sum_i |x(i)|^p)^{1/p}$. For a matrix $W \in \mathbb{R}^{d \times k}$, we use $W(i, j)$ to denote its (i, j)-th entry. We use $W(i, :) \in \mathbb{R}^k$ and $W(:, j) \in \mathbb{R}^d$ to the denote the i-th row and the j-th column. The dot product between two vectors is $\langle x, y \rangle = \sum_i x(i)y(i)$. For any $a \in \mathbb{R}$, we use $\mathbb{R}_{>a}$ to denote the set $\mathbb{R}_{>a} := \{x \in \mathbb{R} : x > a\}$. We use $I_k \in \mathbb{R}^{k \times k}$ to denote an identity matrix.

2.2 Related Work

We briefly review the relevant work, and highlight the differences compared to our work.

Estimation from truncated samples. Given a d-dimensional distribution $\mathcal{D}$ and a subset $S \subseteq \mathbb{R}^d$, truncation means that we can only observe

samples from $\mathcal{D}$ if it falls in S. Samples falling outside S (and their counts in proportion) are not revealed. Estimating the parameters of a multivariate normal distribution from truncated samples is a fundamental problem in statistics and a breakthrough was achieved recently (Daskalakis et al., 2018) on this problem. This is different from our problem because our samples are formed by *projecting* the samples of a multivariate normal distribution onto the positive orthant instead of *truncating* to the positive orthant. Nevertheless, a single coordinate of $\mathcal{D}(W, b)$ can be viewed as a truncated univariate normal distribution (Definition 2.4.1). We use this observation and leverage on the recent results of (Daskalakis et al., 2018) to estimate b and the row norms of W (Section 2.4.3).

Learning ReLU neural networks. A recent series of work, e.g., (Ge et al., 2019; Goel et al., 2018; Li and Yuan, 2017; Zhong et al., 2017; Soltanolkotabi, 2017), considers the problem of estimating the parameters of a ReLU neural network given samples of the form $\{(x_i, y_i)\}_{i=1}^n$. Here (x_i, y_i) represents the input features and the output target, e.g., $y_i = \mathrm{ReLU}(Wx_i + b)$. This is a *supervised* learning problem, and hence, is different from our *unsupervised* density estimation problem.

Learning neural network-based generative models. Many approaches have been proposed to train a neural network to model complex distributions. Examples include GAN (Goodfellow et al., 2014) and its variants (e.g., WGAN (Arjovsky et al., 2017), DCGAN (Radford et al., 2015), etc.), VAE (Kingma and Welling, 2013), autoregressive models (Van Oord et al.,

2016), and reversible generative models (Grathwohl et al., 2019). All of those methods lack theoretical guarantees and explicit sample complexity bounds. A recent work (Nguyen et al., 2018) proves that training an autoencoder via gradient descent can possibly recover a *linear* generative model. This is different from our setting, where we focus on *non-linear* generative models. Mazumdar and Rawat (2019) also consider the problem of learning from one-layer ReLU generative models. Their modeling assumption is different from ours. They assume that the bias vector b is a random variable whose distribution satisfies certain conditions. Besides, there is no distributional assumption on the hidden variable z. By contrast, in our model, both W and b are deterministic and unknown parameters. The randomness only comes from z which is assumed to follow a standard Gaussian distribution.

2.3 Identifiability

Our first question is whether W is identifiable from the distribution $\mathcal{D}(W, b)$. Claim 1 below implies that only WW^T can be possibly identified from $\mathcal{D}(W, b)$.

Claim 1. *For any matrices satisfying $W_1 W_1^T = W_2 W_2^T$, and any vector b, $\mathcal{D}(W_1, b) = \mathcal{D}(W_2, b)$.*

Proof. Since $W_1 W_1^T = W_2 W_2^T$, there exists a unitary matrix $Q \in \mathbb{R}^{k \times k}$ that satisfies $W_2 = W_1 Q$. Since $z \sim \mathcal{N}(0, I_k)$, we have $Qz \sim \mathcal{N}(0, I_k)$. The claim then follows.

□

Identifying the bias vector b from $\mathcal{D}(W, b)$ can be impossible in some cases. For example, if W is a zero matrix, then any negative coordinate of b cannot be identified since it will be reset to zero after the ReLU operation. For the cases when b is identifiable, e.g., by assuming that every row of the W matrix has at least one non-zero element, our next claim shows an example that estimating the bias vector to be within an error of ϵ requires $\Omega(\exp(1/\epsilon^2))$ samples.

Claim 2. *For any value $\delta > 0$, there exists one-dimensional distributions $\mathcal{D}(1, b_1)$ and $\mathcal{D}(1, b_2)$ such that: (a) $|b_1 - b_2| = \delta$; (b) at least $\Omega(\exp(b_1^2/2))$ samples are required to distinguish them.*

Proof. Let $b_1 < 0$ and $b_2 = b_1 - \delta$. It is easy to check that (a) holds. To show (b), note that the probability of observing a positive (i.e., nonzero) sample from $\mathcal{D}(1, b_1)$ is upper bounded by $\mathbb{P}[\mathrm{ReLU}(z - |b_1|) > 0] = \mathbb{P}[z > |b_1|] \leq \exp(-b_1^2/2)$, where the last step follows from the standard Gaussian tail bound Wainwright (2019). The same bound holds for $\mathcal{D}(1, b_2)$. To distinguish $\mathcal{D}(1, b_1)$ and $\mathcal{D}(1, b_2)$, we need to observe at least one nonzero sample, which requires $\Omega(\exp(b_1^2/2))$ samples. □

Claim 2 indicates that in order to estimate the parameters within any error, the sample complexity should scale at least exponentially in $\|b\|_\infty^2$. This is true if b is allowed to take negative values. Intuitively, if b has large negative

values, then most of the samples would be zeros. To avoid this exponential dependence, we now assume that the bias vector is *non-negative*. In Section 2.4, we give an algorithm to provably learn the parameters of $\mathcal{D}(W, b)$ with a sample complexity that is polynomial in $1/\epsilon$ and does not depend on the values of b. In Section 2.7.1, we show that even when the bias vector has negative coordinates, our algorithm can still be able to recover part of the parameters with a small number of samples.

2.4 Algorithm

In this section, we describe a novel algorithm to estimate $WW^T \in \mathbb{R}^{d\times d}$ and $b \in \mathbb{R}^d$ from i.i.d. samples of $\mathcal{D}(W, b)$. Our goal is to estimate WW^T instead of W since W is not identifiable (Claim 1). Our only assumption is that the true b is non-negative. As discussed in Claim 2, this assumption can potentially avoid the exponential dependence in the values of b. Note that our algorithm does not require to know the dimension k of the latent variable z. All proofs can be found in the appendix.

2.4.1 Problem with Maximum Likelihood Estimation

One standard approach to parameter estimation is the maximum likelihood method: finding the parameters to maximize the likelihood of observing the given samples. There are two problems with this maximum likelihood method in our setting: 1) we cannot compute a closed-form function of the likelihood, because it involves integration over the domain $(-\infty, 0]$; 2) the

negative log-likelihood at a given sample can be a non-convex function of the parameters, as shown in the following example.

Consider the one-dimensional setting, given a sample $x = 0$, the negative log-likelihood that x comes from $\mathcal{D}(\sigma, b)$ is

$$\begin{aligned} f(b, \sigma^{-2}) &= -\ln\left(\int_{-\infty}^{0} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(z-b)^2}{2\sigma^2})\,\mathrm{d}z\right) \\ &= \frac{\ln(2\pi)}{2} - \frac{\ln(\sigma^{-2})}{2} - \ln\left(\int_{-\infty}^{0} \exp(-\frac{(z-b)^2}{2}\sigma^{-2})\,\mathrm{d}z\right). \end{aligned}$$

Here we follow the parameterization convention of Gaussian distributions[3] and write the negative log-likelihood as a function of (b, σ^{-2}). It is easy to see that $f(0, \sigma^{-2}) = \ln(2)$ for all σ because any zero-mean Gaussian distribution has half of the probability mass in $(-\infty, 0]$. This implies that $f(b, \sigma^{-2})$ cannot be a strongly-convex function. By numerically computing the integration, we have $f(0.1, 0.5) + f(0.1, 1.5) < 2 \cdot f(0.1, 1)$, which indicates that $f(b, \sigma^{-2})$ is not a convex function with respect to σ^{-2}.

2.4.2 Intuition Behind Our Algorithm

Let $W(i,:) \in \mathbb{R}^k$ be the i-th row ($i \in [d]$) of W. For any $i < j \in [d]$, the (i,j)-th entry of WW^T is

$$\langle W(i,:), W(j,:)\rangle = \|W(i,:)\|_2 \|W(j,:)\|_2 \cos(\theta_{ij}), \tag{2.2}$$

where θ_{ij} is the angle between vectors $W(i,:)$ and $W(j,:)$. Our key idea is to estimate the norms $\|W(i,:)\|_2$, $\|W(j,:)\|_2$, and the angles θ_{ij} separately, as

[3]For Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, its negative log-likelihood is a convex function with respect to (μ, σ^{-2}).

shown in Algorithm 1.

Estimating the row norms[4] $\|W(i,:)\|_2$ as well as the i-th coordinate of the bias vector $b(i) \in \mathbb{R}$ can be done by only looking at the i-th coordinate of the given samples. The idea is to view the problem as estimating the parameters of a univariate normal distribution from truncated samples[5]. This part of the algorithm is described in Section 2.4.3. To estimate $\theta_{ij} \in [0, \pi)$ for every $i < j \in [d]$, we use a simple fact that the angle between any two vectors can be estimated from their inner products with a random Gaussian vector. Details of this part can be found in Section 2.4.4.

2.4.3 Estimate $\|W(i,:)\|_2$ and $b(i)$

Without loss of generality, we fix $i = 1$ and describe how to estimate $\|W(1,:)\|_2 \in \mathbb{R}$ and $b(1) \in \mathbb{R}$ by looking at the first coordinate of the given samples.

The starting point of our algorithm is the following observation. Suppose $x \sim \mathcal{D}(W, b)$, its first coordinate can be written as

$$x(1) = \mathrm{ReLU}(W(1,:)^T z + b(1)) = \mathrm{ReLU}(y), \text{ where } y \sim \mathcal{N}(b(1), \|W(1,:)\|_2^2). \tag{2.3}$$

Because of the ReLU operation, we can only observe the samples of y when it is

[4]Without loss of generality, we can assume that $\|W(i,:)\|_2 \neq 0$ for all $i \in [d]$. If $W(i,:)$ is a zero vector, one can easily detect that and figure out the corresponding non-negative bias term.

[5]Another idea is to use the median to estimate the i-th coordinate of the bias vector. This approach will give the same sample complexity bound as that of our proposed algorithm.

Algorithm 1: Learning a single-layer ReLU generative model

Input: n i.i.d. samples $x_1, \cdots, x_n \in \mathbb{R}^d$ from $\mathcal{D}(W^*, b^*)$, b^* is non-negative.

Output: $\widehat{\Sigma} \in \mathbb{R}^{d \times d}$, $\widehat{b} \in \mathbb{R}^d$.

for $i \leftarrow 1$ **to** d **do**
  $S \leftarrow \{x_m(i), m \in [n] : x_m(i) > 0\}$;
  $\widehat{b}(i), \widehat{\Sigma}(i,i) \leftarrow \text{NormBiasEst}(S)$;
  $\widehat{b}(i) \leftarrow \max\left(0, \widehat{b}(i)\right)$;
end
for $i < j \in [d]$ **do**
  $\widehat{\theta}_{ij} \leftarrow \pi - \frac{2\pi}{n}\left(\sum_{m=1}^{n} \mathbb{1}(x_m(i) > \widehat{b}(i))\, \mathbb{1}(x_m(j) > \widehat{b}(j))\right)$;
  $\widehat{\Sigma}(i,j) \leftarrow \sqrt{\widehat{\Sigma}(i,i)\widehat{\Sigma}(j,j)}\cos(\widehat{\theta}_{ij})$;
  $\widehat{\Sigma}(j,i) \leftarrow \widehat{\Sigma}(i,j)$;
end

positive. Given samples of $x(1) \in \mathbb{R}$, let us keep the samples that have positive values (i.e., ignore the zero samples). Now the problem of estimating $b(1)$ and $\|W(1,:)\|_2$ is equivalent to estimating the parameters of a one-dimensional normal distribution using samples falling in the set $\mathbb{R}_{>0} := \{x \in \mathbb{R} : x > 0\}$.

Recently, Daskalakis et al. (2018) gave an efficient algorithm for estimating the mean and covariance matrix of a multivariate Gaussian distribution from truncated samples. We adapt their algorithm for the specific problem described above. Before describing the details, we start with a formal definition of the truncated (univariate) normal distribution.

Definition 2.4.1. The univariate normal distribution $\mathcal{N}(\mu, \sigma^2)$ has probability

density function

$$\mathcal{N}(\mu, \sigma^2; x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right), \quad \text{for } x \in \mathbb{R}. \tag{2.4}$$

Given a measurable set $S \subseteq \mathbb{R}$, the S-truncated normal distribution $\mathcal{N}(\mu, \sigma^2, S)$ is defined as

$$\mathcal{N}(\mu, \sigma^2, S; x) = \begin{cases} \frac{\mathcal{N}(\mu,\sigma^2;x)}{\int_S \mathcal{N}(\mu,\sigma^2;y)dy} & \text{if } x \in S \\ 0 & \text{if } x \notin S \end{cases}. \tag{2.5}$$

We are now ready to describe the algorithm in (Daskalakis et al., 2018) applied to our problem. The pseudocode is given in Algorithm 2. The algorithm is essentially maximum likelihood by projected stochastic gradient descent (SGD). Given a sample $x \sim \mathcal{N}(\mu^*, \sigma^{*2}, S)$, let $\ell(\mu, \sigma; x)$ be the negative log-likelihood that x is from $\mathcal{N}(\mu, \sigma^2, S)$, then $\ell(\mu, \sigma; x)$ is a convex function with respect to a reparameterization $v = [1/\sigma^2, \mu/\sigma^2] \in \mathbb{R}^2$. We use $\ell(v; x)$ to denote the negative log-likelihood after this reparameterization. Let $\bar{\ell}(v) = \mathbb{E}_x[\ell(v; x)]$ be the expected negative log-likelihood. Although it is intractable to compute $\bar{\ell}(v)$, its gradient $\nabla\bar{\ell}(v)$ with respect to v has a simple unbiased estimator. Specifically, define a random vector $g \in \mathbb{R}^2$ as

$$g = -\begin{bmatrix} -x^2/2 \\ x \end{bmatrix} + \begin{bmatrix} -z^2/2 \\ z \end{bmatrix}, \text{ where } x \sim \mathcal{N}(\mu^*, \sigma^{*2}, S), z \sim \mathcal{N}(\mu, \sigma^2, S). \tag{2.6}$$

We have that $\nabla\bar{\ell}(v) = \mathbb{E}_{x,z}[g]$, i.e., g is an unbiased estimator of $\nabla\bar{\ell}(v)$.

Eq. (2.6) indicates that one can maximize the log-likelihood via SGD, however, in order to efficiently perform this optimization, we need three extra steps.

First, the convergence rate of SGD depends on the expected gradient norm $\mathbb{E}[\|g\|_2^2]$ (Shalev-Shwartz and Ben-David, 2014, Theorem 14.11). In order to maintain a small gradient norm, we transform the given samples to a new space (so that the empirical mean and variance is well-controlled) and perform optimization in that space. After the optimization is done, the solution is transformed back to the original space. Specifically, given samples $x_1, \cdots, x_n \sim \mathcal{N}(\mu^*, \sigma^{*2}, \mathbb{R}_{>0})$, we transform them as

$$x_i \to \frac{x_i - \widehat{\mu}_0}{\widehat{\sigma}_0}, \text{ where } \widehat{\mu}_0 = \frac{1}{n}\sum_{i=1}^{n} x_i,\ \widehat{\sigma}_0^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \widehat{\mu}_0)^2. \tag{2.7}$$

In the transformed space, the problem becomes estimating parameters of a normal distribution with samples truncated to the set $\mathbb{R}_{>-\widehat{\mu}_0/\widehat{\sigma}_0} = \{x \in \mathbb{R} : x > -\widehat{\mu}_0/\widehat{\sigma}_0\}$.

Second, we need to control the strong-convexity of the objective function. This is done by projecting the parameters onto a domain where the strong-convexity is bounded. The domain D_r is parameterized by $r > 0$ and is defined as

$$D_r = \{v \in \mathbb{R}^2 : 1/r \le v(1) \le r, |v(2)| \le r\}. \tag{2.8}$$

According to (Daskalakis et al., 2018, Section 3.4), $r = O(\ln(1/\alpha)/\alpha^2)$ is a hyper-parameter that only depends on $\alpha = \int_S \mathcal{N}(\mu^*, \sigma^{*2}; y)dy$ (i.e., the probability mass of original truncation set S). In our setting, we have $\alpha \ge 1/2$. This is because the original truncation set is $\mathbb{R}_{>0}$ and $\mu^* = b(1) \ge 0$. A large value of r would lead to a small strong-convexity parameter. In our experiments, we set $r = 3$.

Third, a single run of the projected SGD algorithm only guarantees a constant probability of success. To amplify the probability of success to $1-\delta/d$, a standard procedure is to repeat the algorithm $O(\ln(d/\delta))$ times. This procedure is illustrated in Step 2-5 in Algorithm 2.

Algorithm 2: NormBiasEst

Input: Samples from $\mathcal{N}(\mu, \sigma^2, \mathbb{R}_{>0})$.
Output: $\widehat{\mu} \in \mathbb{R}$, $\widehat{\sigma^2} \in \mathbb{R}$.

1 Shift and rescale the samples using (2.7);
2 Split the samples into $B = O(\ln(d/\delta))$ batches;
3 For batch $i \in [B]$, run ProjSGD (Algorithm 3) to get $v_i \in \mathbb{R}^2$;
4 $S \leftarrow \{v_1, \cdots, v_B\}$;
5 $\widehat{v} \leftarrow \arg\min_{v_i \in S} \sum_{j \in [B]} \|v_i - v_j\|_2$;
6 Transform $\widehat{v}$ back to the original space;
7 $\widehat{\mu} \leftarrow \widehat{v}(2)/\widehat{v}(1)$, $\widehat{\sigma^2} \leftarrow 1/\widehat{v}(1)$;

Algorithm 3: ProjSGD

Input: $T = \widetilde{O}(\ln(d/\delta)/\epsilon^2)$, $\lambda > 0$.
Output: $v \in \mathbb{R}^2$.

1 Initialize $v^{(0)} = [1, 0] \in \mathbb{R}^2$;
2 **for** $t \leftarrow 1$ **to** T **do**
3 $\quad g^{(t)} \leftarrow$ Estimate the gradient using (2.6);
4 $\quad v^{(t)} \leftarrow v^{(t-1)} - g^{(t)}/(\lambda \cdot t)$;
5 $\quad v^{(t)} \leftarrow$ Project $v^{(t)}$ to the domain in (2.8);
6 **end**
7 $v \leftarrow \sum_{t=1}^{T} v^{(t)}/T$;

Lemma 2.4.1. *For any $\epsilon \in (0,1)$ and $\delta \in (0,1)$, Algorithm 1 takes $n = \widetilde{O}\left(\frac{1}{\epsilon^2}\ln(\frac{d}{\delta})\right)$ samples from $\mathcal{D}(W^*, b^*)$ (for some non-negative b^*) and outputs $\widehat{b}(i)$ and $\widehat{\Sigma}(i,i)$ for all $i \in [d]$ that satisfy*

$$(1-\epsilon)\|W^*(i,:)\|_2^2 \le \widehat{\Sigma}(i,i) \le (1+\epsilon)\|W^*(i,:)\|_2^2, \quad |\widehat{b}(i) - b^*(i)| \le \epsilon\|W^*(i,:)\|_2 \tag{2.9}$$

with probability at least $1 - \delta$.

2.4.4 Estimate θ_{ij}

To estimate the angle between any two vectors $W^*(i,:)$ and $W^*(j,:)$ (where $i \neq j \in [d]$), we will use the following result.

Fact 1. (Lemma 6.7 in (Williamson and Shmoys, 2011)). *Let* $z \sim \mathcal{N}(0, I_k)$ *be a standard Gaussian random variable in* $\mathbb{R}^k$. *For any two non-zero vectors* $u, v \in \mathbb{R}^k$, *the following holds:*

$$\mathop{\mathbb{P}}_{z \sim \mathcal{N}(0,I_k)}[u^T z > 0 \text{ and } v^T z > 0] = \frac{\pi - \theta}{2\pi}, \tag{2.10}$$

where $\theta = \arccos\left(\frac{\langle u,v \rangle}{\|u\|_2 \|v\|_2}\right)$.

Fact 1 says that the angle between any two vectors can be estimated from the sign of their inner products with a Gaussian random vector. Let $x \sim \mathcal{D}(W^*, b^*)$, since b^* is assumed to be non-negative, Fact 1 gives an unbiased estimator of the pairwise angles.

Lemma 2.4.2. *Suppose that* $x \sim \mathcal{D}(W^*, b^*)$ *and that* $b^* \in \mathbb{R}^d$ *is non-negative, for all* $i \neq j \in [d]$,

$$\mathop{\mathbb{P}}_{x \sim \mathcal{D}(W^*,b^*)}[x(i) > b^*(i) \text{ and } x(j) > b^*(j)] = \frac{\pi - \theta^*_{ij}}{2\pi}, \tag{2.11}$$

where θ^*_{ij} *is the angle between vectors* $W^*(i,:)$ *and* $W^*(j,:)$.

Proof. Since $x(i) = \text{ReLU}\left(W^*(i,:)^T z + b^*(i)\right)$ and b^* is non-negative, we have

$$\text{LHS} = \mathop{\mathbb{P}}_{z \sim \mathcal{N}(0,I_k)}[W^*(i,:)^T z > 0 \text{ and } W^*(j,:)^T z > 0] = \frac{\pi - \theta^*_{ij}}{2\pi} = \text{RHS},$$

where the second equality follows from Fact 1. □

Lemma 2.4.2 gives an unbiased estimator of θ^*_{ij}, however, it requires knowing the true bias vector b^*. In the previous section, we give an algorithm that can estimate $b^*(i)$ within an additive error of $\epsilon\|W^*(i,:)\|_2$ for all $i \in [d]$. Fortunately, this is good enough for estimating θ^*_{ij} within an additive error of ϵ, as indicated by the following lemma.

Lemma 2.4.3. *Let $x \sim \mathcal{D}(W^*, b^*)$, where b^* is non-negative. Suppose that $\widehat{b} \in \mathbb{R}^d$ is non-negative and satisfies $|\widehat{b}(i) - b^*(i)| \le \epsilon\|W^*(i,:)\|_2$ for all $i \in [d]$ and some $\epsilon > 0$. Then for all $i \neq j \in [d]$,*

$$\left|\mathop{\mathbb{P}}_{x}[x(i) > \widehat{b}(i) \text{ and } x(j) > \widehat{b}(j)] - \mathop{\mathbb{P}}_{x}[x(i) > b^*(i) \text{ and } x(j) > b^*(j)]\right| \le \epsilon. \tag{2.12}$$

Let $\mathbb{1}(\cdot)$ be the indicator function, e.g., $\mathbb{1}(x > 0) = 1$ if $x > 0$ and is 0 otherwise. Given samples $\{x_m\}_{m=1}^n$ of $\mathcal{D}(W^*, b^*)$ and an estimated bias vector $\widehat{b}$, Lemma 2.4.2 and 2.4.3 implies that θ^*_{ij} can be estimated as

$$\widehat{\theta}_{ij} = \pi - \frac{2\pi}{n}\sum_{m=1}^{n} \mathbb{1}(x_m(i) > \widehat{b}(i) \text{ and } x_m(j) > \widehat{b}(j)). \tag{2.13}$$

The following lemma shows that the estimated $\widehat{\theta}_{ij}$ is close to the true θ^*_{ij}.

Lemma 2.4.4. *For a fixed pair of $i \neq j \in [d]$, for any $\epsilon, \delta \in (0,1)$, suppose $\widehat{b}$ satisfies the condition in Lemma 2.4.3, given $80\ln(2/\delta)/\epsilon^2$ samples, with probability at least $1-\delta$, $|\cos(\widehat{\theta}_{ij}) - \cos(\theta^*_{ij})| \le \epsilon$.*

2.4.5 Estimate WW^T and b

Our overall algorithm is given in Algorithm 1. In the first for-loop, we estimate the row norms of W^* and b^*. In the second for-loop, we estimate the angles between any two row vectors of W^*.

Theorem 2.4.5. *For any $\epsilon \in (0,1)$ and $\delta \in (0,1)$, Algorithm 1 takes $n = \widetilde{O}\left(\frac{1}{\epsilon^2}\ln(\frac{d}{\delta})\right)$ samples from $\mathcal{D}(W^*, b^*)$ (for some non-negative b^*) and outputs $\widehat{\Sigma} \in \mathbb{R}^{d\times d}$ and $\widehat{b} \in \mathbb{R}^d$ that satisfy*

$$\|\widehat{\Sigma} - W^*W^{*T}\|_F \le \epsilon\|W^*\|_F^2, \quad \|\widehat{b} - b^*\|_2 \le \epsilon\|W^*\|_F \tag{2.14}$$

with probability at least $1-\delta$. Algorithm 1 runs in time $\widetilde{O}\left(\frac{d^2}{\epsilon^2}\ln(\frac{d}{\delta})\right)$ and space $\widetilde{O}\left(\frac{d}{\epsilon^2}\ln(\frac{d}{\delta}) + d^2\right)$.

Theorem 2.4.5 characterizes the sample complexity to achieve a small parameter estimation error. We are also interested in the distance between the estimated distribution and the true distribution. Let $\mathrm{TV}(A, B)$ be the total variation (TV) distance between two distributions A and B. Note that in order for the TV distance to be meaningful[6], we restrict ourselves to the non-degenerate case, i.e., when W is a full-rank square matrix. The following corollary characterizes the number of samples used by our algorithm in order to achieve a small TV distance.

[6]The TV distance between two different degenerate distributions can be a constant. As an example, let $\mathcal{N}(0, \Sigma_1)$ and $\mathcal{N}(0, \Sigma_2)$ be two Gaussian distributions in $\mathbb{R}^d$. If both Σ_1, Σ_2 have rank smaller than d, then $\mathrm{TV}(\mathcal{N}(0, \Sigma_1), \mathcal{N}(0, \Sigma_2)) = 1$ as long as $\Sigma_1 \neq \Sigma_2$.

Corollary 2.4.6. *Suppose that $W^* \in \mathbb{R}^{d\times d}$ is full-rank. Let κ be the condition number of W^*W^{*T}. For any $\epsilon \in (0, 1/2]$ and $\delta \in (0, 1)$, Algorithm 1 takes $n = \widetilde{O}\left(\frac{\kappa^2 d^2}{\epsilon^2}\ln(\frac{d}{\delta})\right)$ samples from $\mathcal{D}(W^*, b^*)$ (for some non-negative b^*) and outputs a distribution $\mathcal{D}(\widehat{\Sigma}^{1/2}, \widehat{b})$ that satisfies*

$$\mathrm{TV}\left(\mathcal{D}(\widehat{\Sigma}^{1/2}, \widehat{b}),\ \mathcal{D}(W^*, b^*)\right) \le \epsilon, \tag{2.15}$$

with probability at least $1 - \delta$. Algorithm 1 runs in time $\widetilde{O}\left(\frac{\kappa^2 d^4}{\epsilon^2}\ln(\frac{d}{\delta})\right)$ and space $\widetilde{O}\left(\frac{\kappa^2 d^3}{\epsilon^2}\ln(\frac{d}{\delta})\right)$.

2.5 Lower Bounds

In the previous section, we gave an algorithm to estimate W^*W^{*T} and b^* using i.i.d. samples from $\mathcal{D}(W^*, b^*)$, and analyzed its sample complexity. In this section, we provide lower bounds for this density estimation problem. More precisely, we try to answer the following question: what is the minimum number of samples necessary to learn $\mathcal{D}(W^*, b^*)$ up to some error measure ϵ?

Before stating our lower bounds, we first formally describe the distribution learning framework. The framework that we consider here is quite standard. It can be viewed as the minimax framework in statistics or the PAC-learning framework in learning theory. Specifically, let S be a class of distributions. Let d be some distance function between the two distributions (or between the parameters of the two distributions). We say that a distribution learning algorithm learns S with sample complexity $m(\epsilon)$ if for any distribution $p \in S$, given $m(\epsilon)$ i.i.d. samples from p, it constructs a distribution q such that

$d(p, q) \leq \epsilon$ with success probability at least $2/3$[7].

We have analyzed two types of distance functions for Algorithm 1 so far: the distance in the parameter space (Theorem 2.4.5), and the TV distance between two distributions (Corollary 2.4.6). Correspondingly, we will provide two sample complexity lower bounds in terms of these two distance functions.

Theorem 2.5.1. *(Lower bound for parameter estimation). Let $\sigma > 0$ be a fixed and known scalar. Let I_d be the identity matrix in $\mathbb{R}^d$. Let $S := \{\mathcal{D}(W, b) : W = \sigma I_d, b \in \mathbb{R}^d$ non-negative$\}$ be a class of distributions in $\mathbb{R}^d$. Any algorithm that learns S to satisfy $\|\widehat{b} - b^*\|_2 \leq \epsilon \|W^*\|_F$ with success probability at least 2/3 requires $\Omega(1/\epsilon^2)$ samples.*

Theorem 2.5.2. *(Lower bound for distribution estimation). Let $S := \{\mathcal{D}(W, 0) : W \in \mathbb{R}^{d \times d}$ full rank$\}$ be a set of distributions in $\mathbb{R}^d$. Any algorithm that learns S within total variation distance ϵ and success probability at least 2/3 requires $\Omega(d/\epsilon^2)$ samples.*

Comparing the sample complexity achieved by our algorithm (Theorem 2.4.5 and Corollary 2.4.6) and the above lower bounds, we can see that 1) our algorithm matches the sample complexity lower bound (up to log factors) for parameter estimation; 2) there is a gap between our sample complexity and the lower bound for learning distributions in TV distance. There are two possible reasons why this gap shows up.

[7]We focus on constant success probability here as standard techniques can be used to boost the success probability to $1 - \delta$ with an extra multiplicative factor $\ln(1/\delta)$ in the sample complexity.

- The lower bound given in Theorem 2.5.2 may be loose. In fact, because learning a d-dimensional Gaussian distribution up to TV distance ϵ requires $\widetilde{\Theta}(d^2/\epsilon^2)$ samples (this is both sufficient and necessary (Ashtiani et al., 2018)), it is reasonable to guess that learning rectified Gaussian distributions also requires at least $\Omega(d^2/\epsilon^2)$ samples. It is thus interesting to see if one can show a better lower bound than $\Omega(d/\epsilon^2)$.

- Our sample complexity of learning $\mathcal{D}(W, b)$ up to TV distance ϵ also depends on the condition number κ of WW^T. Intuitively, this κ dependence shows up because our algorithm estimates WW^T entry-by-entry instead of estimating the matrix as a whole. Besides, our algorithm is a *proper* learning algorithm, meaning that the output distribution belongs to the family $\mathcal{D}(W, b)$. By contrast, the lower bound proved in Theorem 2.5.2 allows any *non-proper* learning algorithm, i.e., there is no constraint on the output distribution. One interesting direction for future research is to see if one can remove this κ dependence.

2.6 Experiments

In this section, we provide empirical results to verify the correctness of our algorithm as well as the analysis. The algorithm is implemented in MATLAB. Code to reproduce our result can be found at `https://github.com/wushanshan/densityEstimation`. All experiments are done in a personal desktop. The hyper-parameters are set as $B = 1$ (in Algorithm 2), $r = 3$ and $\lambda = 0.1$ (in Algorithm 3).

We evaluate three performance metrics, as shown in Figure 2.1. The first two metrics measure the error between the estimated parameters and the ground truth. Specifically, we compute the estimation errors analyzed in Theorem 2.4.5: $\|\widehat{\Sigma} - W^* W^{*T}\|_F / \|W\|_F^2$ and $\|\widehat{b} - b\|_2 / \|W\|_F$. Besides the parameter estimation error, we are also interested in the TV distance analyzed in Corollary 2.4.6: $\mathrm{TV}\left(\mathcal{D}(\widehat{\Sigma}^{1/2}, \widehat{b}),\ \mathcal{D}(W^*, b^*)\right)$. It is difficult to compute the TV distance exactly, so we instead compute an upper bound of it. Let $KL(A||B)$ denote the KL divergence between two distributions. Let $\Sigma^* = W^* W^{*T}$. Assuming that both Σ^* and $\widehat{\Sigma}$ are full-rank, we have

$$\begin{aligned}\mathrm{TV}\left(\mathcal{D}(\widehat{\Sigma}^{1/2}, \widehat{b}),\ \mathcal{D}(W^*, b^*)\right) &\leq \mathrm{TV}\left(\mathcal{N}(\widehat{b}, \widehat{\Sigma}), \mathcal{N}(b^*, \Sigma^*)\right) \\ &\leq \sqrt{\mathrm{KL}\left(\mathcal{N}(\widehat{b}, \widehat{\Sigma}) || \mathcal{N}(b^*, \Sigma^*)\right)/2}.\end{aligned}$$

The first inequality follows from the data-processing inequality given in Lemma A.6.3 of Appendix A.6 (see also (Ashtiani et al., 2018, Fact A.5)): for any function f and random variables X, Y over the same space, $\mathrm{TV}(f(X), f(Y)) \leq \mathrm{TV}(X, Y)$. The second inequality follows from the Pinsker's inequality (Tsybakov, 2009, Lemma 2.5).

Sample Efficiency. The left plot of Figure 2.1 shows that both the parameter estimation errors and the KL divergence decrease when we have more samples. Our experimental setting is simple: we set the dimension as $d = k = 5$ and the condition number as 1; we generate W^* as a random orthonormal matrix; we generate b^* as a random normal vector, followed by a ReLU operation (to ensure non-negativity). This plot indicates that our

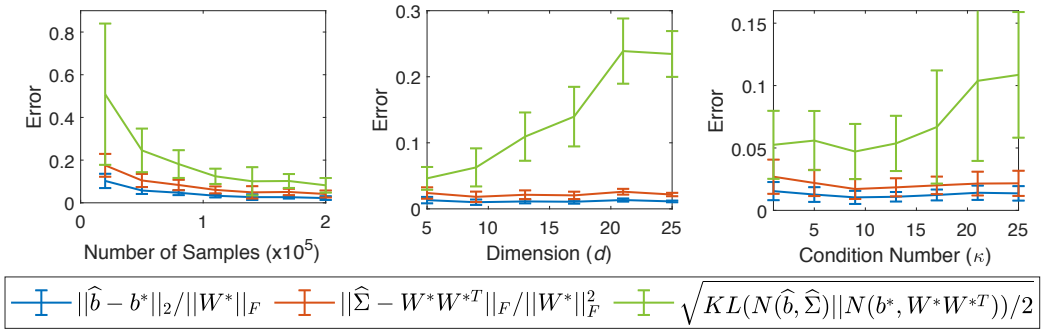


Figure 2.1: Best viewed in color. Empirical performance of our algorithm with respect to three parameters: number of samples n, dimension d, and the condition number κ. **Left:** Fix $d = 5$ and $\kappa = 1$. **Middle:** Fix $n = 5 \times 10^5$ and $\kappa = 1$. **Right:** Fix $n = 5 \times 10^5$ and $d = 5$. Every point shows the mean and standard deviation across 10 runs. Each run corresponds to a different W^* and b^*.

algorithm is able to accurately estimate the true parameters and obtain a distribution that is close to the true distribution in TV distance.

Dependence on Dimension. In the middle plot of Figure 2.1, we use 5×10^5 samples and keep the condition number to be 1. We then increase the dimension ($d = k$) from 5 to 25. Both W^* and b^* are generated in the same manner as the previous plot. As shown in the middle plot, the parameter estimation errors maintain the same value while the KL divergence increases as the dimension increases. This is consistent with our analysis, because the sample complexity in Theorem 2.4.5 is dimension-free (ignoring the log factor) while the sample complexity in Corollary 2.4.6 depends on d^2.

Dependence on Condition Number. In the right plot of Figure 2.1, we keep the dimension $d = k = 5$ and the number of samples 5×10^5 fixed. We

then increase the condition number κ of W^*W^{*T}. This plot shows the same trend as the middle plot, i.e., the parameter estimation errors remain the same while the KL divergence increases as κ increases, which is again consistent with our analysis. The number of samples required to achieve an additive estimation error (Theorem 2.4.5) does not depend on κ, while the sample complexity to guarantee a small TV distance (Corollary 2.4.6) depends on κ^2.

2.7 Open Problems

2.7.1 Negative Bias

Our algorithm relies on the assumption that the bias vector is non-negative. This assumption is required to ensure that Lemma 2.4.2 holds, which subsequently ensures that the pairwise angles between the row vectors of W^* can be correctly recovered. A weaker assumption would be allowing the bias vector b^* to be negative but constraining the largest negative values. Designing algorithms under this weaker assumption is an interesting direction for future research.

When b^* has negative components, running our algorithm can still recover part of the parameters with a small number of samples. Specifically, let $\Omega := \{i \in [d] : b^*(i) \geq 0\}$ be the set of coordinates that b^* is non-negative; let $b^*_\Omega \in \mathbb{R}^{|\Omega|}$ and $W^*_\Omega \in \mathbb{R}^{|\Omega| \times k}$ be the sub-vector and sub-matrix associated with the coordinates in Ω. Then given $O(\frac{1}{\epsilon^2}\ln(\frac{d}{\delta})$ samples, the output of our algorithm $\widehat{b} \in \mathbb{R}^d$ and $\widehat{\Sigma} \in \mathbb{R}^{d\times d}$ satisfies

$$\|\widehat{\Sigma}_{\Omega\times\Omega} - W^*_\Omega W^{*T}_\Omega\|_F \leq \epsilon\|W^*_\Omega\|_F^2, \quad \|\widehat{b}_\Omega - b^*_\Omega\|_2 \leq \epsilon\|W^*_\Omega\|_F,$$

with probability at least $1-\delta$. This is the same guarantee given in Theorem 2.4.5. The reason is that our algorithm only uses the i-th and j-th coordinates of the samples to estimate $\langle W^*(i,:), W^*(j,:)\rangle$ and $b^*(i)$, $b^*(j)$. As a result, Theorem 2.4.5 still holds for this part of the parameters.

For the rest part of the parameters, if the negative components of b^* are small (in absolute value), then the error of our algorithm will be also small. Let Ω^c be the complement of Ω. We assume that there is a value $\eta \geq 0$ such that the negative coordinates of b^* satisfy

$$b^*(i) \geq -\eta \|W^*(i,:)\|_2, \qquad \text{for all } i \in \Omega^c.$$

Given $\widetilde{O}(\ln(d)/\epsilon^2)$ samples, the output of our algorithm satisfies

$$|\widehat{b}(i) - b^*(i)| \leq \max(\eta, \epsilon)\|W^*(i,:)\|_2, \qquad \text{for all } i \in \Omega^c.$$

One can show a similar result for $\langle W^*(i,:), W^*(j,:)\rangle$, where $i \in \Omega^c$ and $j \in [d]$:

$$|\widehat{\Sigma}(i,j) - \langle W^*(i,:), W^*(j,:)\rangle\,| \leq 7\max(\eta, \epsilon)\|W^*(i,:)\|_2\|W^*(j,:)\|_2.$$

Comparing the above two equations with (A.5) and (A.8), we see that the error from the negative bias is small if $\eta = O(\epsilon)$. If η is large, i.e., if b^* have large negative components, then estimating those parameters becomes difficult (as indicated by Claim 2). In that case, maybe one should directly estimate the distribution without estimating the parameters. This is an interesting direction for future research.

2.7.2 Two-Layer Generative Model

One natural generalization of our problem is to consider distributions defined by a two-layer generative model:

Definition 2.7.1. Given $A \in \mathbb{R}^{d\times p}$, $W \in \mathbb{R}^{p\times k}$, and $b \in \mathbb{R}^p$, we define $\mathcal{D}(A, W, b)$ as the distribution of a random variable $x \in \mathbb{R}^d$ generated as follows:

$$x = A\,\mathrm{ReLU}(Wz + b), \text{ where } z \sim \mathcal{N}(0, I_k). \tag{2.16}$$

Given i.i.d. samples $x \sim \mathcal{D}(A, W, b)$, can we recover the parameters A, W, b (up to permutation and scaling of the columns of A)? While this problem seems hard in general, we find an interesting connection between this problem and non-negative matrix factorization. A non-negative matrix has all its entries being non-negative. Note that in our problem, the A matrix does *not* need to be a non-negative matrix.

Connection to Non-negative Matrix Factorization (NMF). In MNF, we are given a non-negative matrix $X \in \mathbb{R}^{d\times n}$ and an integer $p > 0$, the goal is to find two non-negative matrices $A \in \mathbb{R}^{d\times p}, M \in \mathbb{R}^{p\times n}$ such that $X = AM$. This problem is NP-hard in general (Vavasis, 2009). Arora et al. (2012) give the first polynomial-time algorithm under the "separability" condition (Donoho and Stodden, 2004):

Definition 2.7.2. The factorization $X = AM$ is called separable[8] if for each

[8]Here we define separability with respect to the M matrix while (Arora et al., 2012, Definition 5.1) defines it with respect to the A matrix, but they are equivalent definitions.

$i \in [p]$, there is a column $f(i) \in [n]$ of M such that $M(:, f(i)) \in \mathbb{R}^p$ has only one non-zero positive entry at the i-th location, i.e., $M(i, f(i)) > 0$ and $M(j, f(i)) = 0$ for $j \neq i$.

If the separability condition holds, then the algorithm proposed by (Arora et al., 2012) is guaranteed to find a separable non-negative factorization in time polynomial in n, p, d.

In our problem, we are given n samples $\{x_i\}_{i=1}^n$ from $\mathcal{D}(A, W, b)$. Stacking these samples to form a matrix $X \in \mathbb{R}^{d \times n}$ as

$$X = AM, \text{ where } M(:, i) = \text{ReLU}(Wz_i + b), i \in [n]. \tag{2.17}$$

Note that $M \in \mathbb{R}^{p \times n}$ is a non-negative matrix while A can be an arbitrary matrix. Nevertheless, if M satisfies the separability condition (Definition 2.7.2), and A has full column rank (i.e., the columns of A are linearly independent), then we can still use the same idea of (Arora et al., 2012) to *exactly* recover A and M (up to permutation and scaling of the column vectors in A). Once $M \in \mathbb{R}^{p \times n}$ is recovered, estimating W and b is the same problem as learning one-layer ReLU generative model, and hence can be done by our algorithm (Algorithm 1) assuming that b is non-negative.

The pseudocode is given in Algorithm 4. We first create a set S by normalizing each sample and removing zero and duplicated vectors. The next step is to check for each vector $v \in S$, whether v can be represented as a conical sum (i.e., non-negative linear combination) of the rest vectors in S. This can

be done by checking the feasibility of a linear program. For example, checking whether vector v can be expressed as a conical sum of two vectors w_1, w_2 is equivalent to checking whether the following linear program is feasible:

$$\min_{c_1 \geq 0,\, c_2 \geq 0} c_1 + c_2 \quad \text{s.t. } c_1 w_1 + c_2 w_2 = v.$$

We only keep a vector if it *cannot* be written as the conical sum of the other vectors. Those vectors are then stacked to form $\widehat{A}$. Let $\widehat{A}^\dagger = (\widehat{A}^T \widehat{A})^{-1} \widehat{A}^T$ be the pseudo-inverse of $\widehat{A}$. The last step is to compute $\{\widehat{A}^\dagger x_i\}_{i=1}^n$ and treat them as samples from one-layer ReLU generative model so that we can run Algorithm 1 to estimate $W^* W^{*T}$ and b^*.

Algorithm 4: Learning a two-layer ReLU generative model

Input: n i.i.d. samples $x_1, \cdots, x_n \in \mathbb{R}^d$ from $\mathcal{D}(A^*, W^*, b^*)$, b^* is non-negative, A^* has linearly independent column vectors.
Output: $\widehat{A} \in \mathbb{R}^{d \times p}$, $\widehat{\Sigma} \in \mathbb{R}^{p \times p}$, $\widehat{b} \in \mathbb{R}^p$.

$S \leftarrow \emptyset$;
for $i \leftarrow 1$ **to** n **do**
  if $x_i \neq 0$ **then**
    $S \leftarrow S \cup \{x_i / \|x_i\|_2\}$;
  end
end
Remove duplicated vectors from S;
for $v \in S$ **do**
  if v is a conical sum of the rest vectors in S **then**
    Remove v from S;
  end
end
$\widehat{A} \leftarrow$ stack vectors from S;
$\widehat{\Sigma}, \widehat{b} \leftarrow$ run Algorithm 1 with samples $\{\widehat{A}^\dagger x_i\}_{i=1}^n$.

Claim 3. *Define $X \in \mathbb{R}^{d\times n}$ and $M \in \mathbb{R}^{p\times n}$ as in (2.17). Without loss of generality, we assume that the column vectors of A^* have unit ℓ_2-norm. Let $\widehat{A}$ be the output of Algorithm 4. If A^* has full column rank, and M satisfies the separability condition in Definition 2.7.2, then there is a way to permute the column vectors of $\widehat{A}$ so that $\widehat{A} = A^*$.*

Proof. After Step 1-7, Algorithm 4 produces a set S which contains all non-zero and normalized samples. Besides, the vectors in S are unique because the duplicated ones are removed in Step 7. To prove $\widehat{A} = A^*$ (up to permutation of the columns), we only need to prove that

- (a) All the (normalized) column vectors of A^* are in S.
- (b) Except the column vectors in A^*, every vector in S can be represented as a conical sum of the rest vectors in S.
- (c) Any column vector in A^* cannot be represented as a conical sum of the rest vectors in S.

(a) is true because the M matrix satisfies the separability condition. According to Definition 2.7.2, for every column vector of A^*, there is at least one sample $x \in \mathbb{R}^d$ which is a scaled version of that column vector.

To prove (b), first note that all the vectors in S can be represented as a conical combination of the column vectors of A^*. This is because M is a non-negative matrix and the samples are $X = A^*M$. From (a), we know that

all the column vectors of A^* are also in S. Therefore, all the samples, except those that are scaled versions of A^*'s columns, can be written as a conical combination of the rest vectors in S.

We will prove (c) by contradiction. If a column vector of A^* can be written as a conical combination of the rest vectors in S, then it means that this column vector can be represented as a conical combination of the column vectors in A^*. This will violate the fact that A^* has full column rank. Hence, any column vector in A^* cannot be represented as a conical sum of the rest vectors in S. □

According to Claim 3, if M satisfies the separability condition, and A^* has full column rank, then Algorithm 4 can *exactly* recover A^* (up to permutation and scaling of the column vectors in A^*). Once A^* is recovered, estimating W and b is the same problem as learning one-layer ReLU generative model, which can be done by Algorithm 1. One problem with the above approach is that it requires the $M \in \mathbb{R}^{p \times n}$ matrix to satisfy the separability condition. This is true when, e.g., W has full row rank, and the number of samples is $\Omega(2^k)$. Developing sample-efficient algorithms for more general generative models is definitely an interesting direction for future research.

We simulate Algorithm 4 on a two-layer generative model with $k = p = 5$ and $d = 10$. We generate $A^* \in \mathbb{R}^{10 \times 5}$ as a random Gaussian matrix, $W^* \in \mathbb{R}^{5 \times 5}$ as a random orthogonal matrix, and let b^* be zero. Given n, we run 100 times of Algorithm 4, and each time we use a different set of random samples with

size n. Table 2.1 lists the fraction of runs that Algorithm 4 successfully recovers A^*. We see that the probability of success increases as we are given more samples.

Number of samples n	50	100	150
Probability of success in 100 runs	0.30	0.78	0.99

Table 2.1: We simulate a two-layer generative model: $A^* \in \mathbb{R}^{10\times 5}$ is a random Gaussian matrix, $W^* \in \mathbb{R}^{5\times 5}$ is a random orthogonal matrix, and $b^* = 0$. For a fixed number of samples, we run 100 times of Algorithm 4 with different input samples. This table shows the fraction of runs that Algorithm 4 successfully recovers A^*.

2.7.3 Learning from Noisy Samples

It is an interesting direction to design algorithms that can learn from noisy samples, e.g., samples of the form $x = \mathrm{ReLU}(W^*z + b^*) + \xi$, where $\xi \sim \mathcal{N}(0, \sigma^2 I_d)$ represents the noise. In that case, Algorithm 1 would not work because both parts of our algorithm (i.e., learn from truncated samples, and estimate the pairwise angles) require clean samples. Nevertheless, the above problem is easy when $b^* = 0$. This is because we can estimate $\|W^*(i,:)\|_2$ using the fact that $\mathbb{E}_{z,\xi}[x(i)^2] = \|W^*(i,:)\|_2^2/2$, and estimate θ_{ij}^* using the following fact (Cho and Saul, 2009):

$$\mathop{\mathbb{E}}_{z,\xi}[x(i)x(j)] = \frac{1}{2\pi}\|W^*(i,:)\|_2\|W^*(j,:)\|_2(\sin(\theta_{ij}) - (\pi - \theta_{ij})\cos(\theta_{ij}))\,.$$

2.8 Conclusion

A popular generative model nowadays is defined by passing a standard Gaussian random variable through a neural network. In this project we are interested in the following fundamental question: Given samples from this distribution, is it possible to recover the parameters of the neural network? We designed a new algorithm to provably recover the parameters of a single-layer ReLU generative model from i.i.d. samples, under the assumption that the bias vector is non-negative. We analyzed the sample complexity of the proposed algorithm in terms of two error metrics: parameter estimation error and total variation distance. We also showed an interesting connection between learning a two-layer generative model and non-negative matrix factorization.

While our focus here is parameter recovery, one interesting direction for future work is to see whether one can directly estimate the distribution in some distance without first estimating the parameters. Another interesting direction is to develop provable learning algorithms for the agnostic setting instead of the realizable setting. Besides designing new algorithms, analyzing the existing algorithms, e.g., GANs, VAEs, and reversible generative models, is also an important research direction.

Chapter 3

Structural Learning of Discrete Graphical Models

3.1 Introduction

Undirected graphical models provide a framework for modeling high dimensional distributions with dependent variables and have many applications including in computer vision (Choi et al., 2010), bio-informatics (Marbach et al., 2012), and sociology (Eagle et al., 2009). In this project we characterize the effectiveness of a natural, and already popular, algorithm for the *structure learning* problem. Structure learning is the task of finding the dependency graph of a Markov random field (MRF) given i.i.d. samples; typically one is also interested in finding estimates for the edge weights as well. We consider the structure learning problem in general (non-binary) discrete pairwise graphical models. These are MRFs where the variables take values in a discrete alphabet, but all interactions are pairwise. This includes the Ising model as a special case (which corresponds to a binary alphabet).

Chapter 3 is based on material from (Wu et al., 2019c). The author of this dissertation is the leading author of (Wu et al., 2019c), and contributed to the idea, the analysis, the implementation and experiments, and the writing of the paper. Source code can be found at `https://github.com/wushanshan/GraphLearn`.

The natural and popular algorithm we consider is (appropriately regularized) maximum conditional log-likelihood for finding the neighborhood set of any given node. For the Ising model, this becomes ℓ_1-constrained logistic regression; more generally for non-binary graphical models the regularizer becomes an $\ell_{2,1}$ norm. We show that this algorithm can recover all discrete pairwise graphical models, and characterize its sample complexity as a function of the parameters of interest: model width, alphabet size, edge parameter accuracy, and the number of variables. We match or improve dependence on each of these parameters, over all existing results for the general alphabet case when no additional assumptions are made on the model (see Table 3.1). For the specific case of Ising models, some recent work has better dependence on some parameters (see Table B.1 in Appendix B.1).

We now describe the related work, and then outline our contributions.

3.1.1 Related Work

In a classic paper, Ravikumar et al. (2010) considered the structure learning problem for Ising models. They showed that ℓ_1-regularized logistic regression provably recovers the correct dependency graph with a very small number of samples by solving a convex program for each variable. This algorithm was later generalized to multi-class logistic regression with group-sparse regularization, which can learn MRFs with higher-order interactions and non-binary variables (Jalali et al., 2011). A well-known limitation of (Ravikumar et al., 2010; Jalali et al., 2011) is that their theoretical guarantees only work

for a restricted class of models. Specifically, they require that the underlying learned model satisfies technical *incoherence* assumptions, that are difficult to validate or check.

Paper	Assumptions	Sample complexity (N)
Greedy algorithm (Hamilton et al., 2017)	1. Alphabet size $k \geq 2$ 2. Model width $\leq \lambda$ 3. Degree $\leq d$ 4. Minimum edge weight $\geq \eta$ 5. Probability of success $\geq 1-\rho$	$O(\exp(\frac{k^{O(d)}\exp(O(d^2\lambda))}{\eta^{O(1)}}) \cdot \ln(\frac{nk}{\rho}))$
Sparsitron (Klivans and Meka, 2017)	1. Alphabet size $k \geq 2$ 2. Model width $\leq \lambda$ 3. Minimum edge weight $\geq \eta$ 4. Probability of success $\geq 1-\rho$	$O(\frac{\lambda^2 k^5 \exp(14\lambda)}{\eta^4} \ln(\frac{nk}{\rho\eta}))$
$\ell_{2,1}$-constrained logistic regression (Our result)	1. Alphabet size $k \geq 2$ 2. Model width $\leq \lambda$ 3. Minimum edge weight $\geq \eta$ 4. Probability of success $\geq 1-\rho$	$O(\frac{\lambda^2 k^4 \exp(14\lambda)}{\eta^4} \ln(\frac{nk}{\rho}))$

Table 3.1: Sample complexity comparison for different graph recovery algorithms. The pairwise graphical model has alphabet size k. For $k = 2$ (i.e., Ising models), our algorithm reduces to the ℓ_1-constrained logistic regression (see Table B.1 in Appendix B.1 for related work on learning Ising models). Our sample complexity has a better dependency on the alphabet size ($\tilde{O}(k^4)$ versus $\tilde{O}(k^5)$) than that in (Klivans and Meka, 2017).

A large amount of recent work has since proposed various algorithms to obtain provable learning results for general graphical models without requiring the incoherence assumptions. We now describe the (most related part of the extensive) related work, followed by our results and comparisons (see Table 3.1). For a discrete pairwise graphical model, let n be the number of variables and k

be the alphabet size; define the model width λ as the maximum neighborhood weight (see Definition 3.2.1 and 3.2.2 for the precise definition). For structure learning algorithms, a popular approach is to focus on the sub-problem of finding the neighborhood of a single node. Once this is correctly learned, the overall graph structure is a simple union bound. Indeed all the papers we now discuss are of this type. As shown in Table 3.1, Hamilton et al. (2017) proposed a greedy algorithm to learn pairwise (and higher-order) MRFs with general alphabet. Their algorithm generalizes the approach of Bresler (2015) for learning Ising models. The sample complexity in (Hamilton et al., 2017) grows logarithmically in n, but *doubly* exponentially in the width λ. Note that an information-theoretic lower bound for learning Ising models (Santhanam and Wainwright, 2012) only has a *single-exponential* dependence on λ. Klivans and Meka (2017) provided a different algorithmic and theoretical approach by setting this up as an online learning problem and leveraging results from the Hedge algorithm therein. Their algorithm Sparsitron achieves single-exponential dependence on the width λ.

3.1.2 Our Contributions

- Our main result: We show that the $\ell_{2,1}$-constrained[1] logistic regression can be used to estimate the edge weights of a discrete pairwise graphical

[1]It may be possible to prove a similar result for the *regularized* version of the optimization problem using techniques from (Negahban et al., 2012). One needs to prove that the objective function satisfies restricted strong convexity (RSC) when the samples are from a graphical model distribution (Vuffray et al., 2016; Lokhov et al., 2018). It is interesting to see if the proof presented here is related to the RSC condition.

model from i.i.d. samples (see Theorem 3.2.3). For the special case of Ising models (see Theorem 3.2.1), this reduces to an ℓ_1-constrained logistic regression. We make no incoherence assumption on the graphical models. As shown in Table 3.1, our sample complexity scales as $\tilde{O}(k^4)$, which improves[2] the previous best result with $\tilde{O}(k^5)$ dependency[3]. The analysis applies a sharp generalization error bound for logistic regression when the weight vector has an $\ell_{2,1}$ constraint (or ℓ_1 constraint) and the sample vector has an $\ell_{2,\infty}$ constraint (or ℓ_∞ constraint) (see Lemma B.2.1 and Lemma B.2.4 in Appendix B.2). Our key insight is that a generalization bound can be used to control the squared distance between the predicted and true logistic functions (see Lemma 3.3.1 and Lemma 3.3.2 in Section 3.3.2), which then implies an ℓ_∞ norm bound between the weight vectors (see Lemma 3.3.5 and Lemma 3.3.6).

- We show that the proposed algorithms can run in $\tilde{O}(n^2)$ time without affecting the statistical guarantees (see Section 3.2.3). Note that $\tilde{O}(n^2)$ is an efficient runtime for graph recovery over n nodes. Previous algorithms

[2]This improvement essentially comes from the fact that we are using an $\ell_{2,1}$ norm constraint instead of an ℓ_1 norm constraint for learning general (i.e., non-binary) pairwise graphical models (see our remark after Theorem 3.2.3). The Sparsitron algorithm proposed by Klivans and Meka (2017) learns a ℓ_1-constrained generalized linear model. This ℓ_1-constraint gives rise to a k^5 dependency for learning non-binary pairwise graphical models.

[3]In an independent and concurrent work, Vuffray et al. (2019) generalize the Interaction Screening algorithm (Vuffray et al., 2016) to the non-binary alphabet setting as well as the high-order MRFs. Their sample complexity is $O(k^4\widehat{\gamma}^4 \exp(12\lambda)\ln(nk)/\eta^4)$ for learning pairwise non-binary graphical models (see Corollary 4 in their paper), where $\widehat{\gamma}$ is an upper bound on the ℓ_1 norm of the node-wise weight vectors. Since $\widehat{\gamma}$ can scale as $k^2\lambda$, their dependence on k can be much worse than ours.

in (Hamilton et al., 2017; Klivans and Meka, 2017) also require $\tilde{O}(n^2)$ runtime for structure learning of pairwise graphical models.

- We construct examples that violate the incoherence condition proposed in (Ravikumar et al., 2010) (see Figure 3.1). We then run ℓ_1-constrained logistic regression and show that it can recover the graph structure as long as given enough samples. This verifies our analysis and shows that our conditions for graph recovery are weaker than those in (Ravikumar et al., 2010).

- We empirically compare the proposed algorithm with the Sparsitron algorithm in (Klivans and Meka, 2017) over different alphabet sizes, and show that our algorithm needs fewer samples for graph recovery (see Figure 3.2).

3.1.3 Notation

We use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. For a vector $x \in \mathbb{R}^n$, we use x_i or $x(i)$ to denote its i-th coordinate. The ℓ_p norm of a vector is defined as $\|x\|_p = (\sum_i |x_i|^p)^{1/p}$. We use $x_{-i} \in \mathbb{R}^{n-1}$ to denote the vector after deleting the i-th coordinate. For a matrix $A \in \mathbb{R}^{n \times k}$, we use A_{ij} or $A(i, j)$ to denote its (i, j)-th entry. We use $A(i, :) \in \mathbb{R}^k$ and $A(:, j) \in \mathbb{R}^n$ to the denote the i-th row vector and the j-th column vector. The $\ell_{p,q}$ norm of a matrix $A \in \mathbb{R}^{n \times k}$ is defined as $\|A\|_{p,q} = \|[\|A(1, :)\|_p, ..., \|A(n, :)\|_p]\|_q$. We define $\|A\|_\infty = \max_{ij} |A(i, j)|$ throughout this chapter (note that this definition is different from the induced matrix norm). We use $\sigma(z) = 1/(1 + e^{-z})$ to represent the sigmoid function.

We use $\langle \cdot, \cdot \rangle$ to represent the dot product between two vectors $\langle x, y \rangle = \sum_i x_i y_i$ or two matrices $\langle A, B \rangle = \sum_{ij} A(i,j)B(i,j)$.

3.2 Main Results

We start with the special case of binary variables (i.e., Ising models), and then move to the general case with non-binary variables.

3.2.1 Learning Ising Models

We first give a definition of an Ising model distribution.

Definition 3.2.1. Let $A \in \mathbb{R}^{n \times n}$ be a symmetric weight matrix with $A_{ii} = 0$ for $i \in [n]$. Let $\theta \in \mathbb{R}^n$ be a mean-field vector. The n-variable Ising model is a distribution $\mathcal{D}(A, \theta)$ on $\{-1, 1\}^n$ that satisfies

$$\mathop{\mathbb{P}}_{Z \sim \mathcal{D}(A,\theta)}[Z = z] \propto \exp\Big(\sum_{1 \le i < j \le n} A_{ij} z_i z_j + \sum_{i \in [n]} \theta_i z_i\Big). \tag{3.1}$$

The dependency graph of $\mathcal{D}(A, \theta)$ is an undirected graph $G = (V, E)$, with vertices $V = [n]$ and edges $E = \{(i, j) : A_{ij} \neq 0\}$. The width of $\mathcal{D}(A, \theta)$ is defined as

$$\lambda(A, \theta) = \max_{i \in [n]} \Big(\sum_{j \in [n]} |A_{ij}| + |\theta_i|\Big). \tag{3.2}$$

Let $\eta(A, \theta)$ be the minimum edge weight in absolute value, i.e., $\eta(A, \theta) = \min_{i,j \in [n]: A_{ij} \neq 0} |A_{ij}|$.

One property of an Ising model distribution is that the conditional distribution of any variable given the rest variables follows a logistic function.

Let $\sigma(z) = 1/(1+e^{-z})$ be the sigmoid function.

Fact 2. *Let $Z \sim \mathcal{D}(A,\theta)$ and $Z \in \{-1,1\}^n$. For any $i \in [n]$, the conditional probability of the i-th variable $Z_i \in \{-1,1\}$ given the states of all other variables $Z_{-i} \in \{-1,1\}^{n-1}$ is*

$$\begin{aligned}\mathbb{P}[Z_i = 1 | Z_{-i} = x] &= \frac{\exp(\sum_{j\neq i} A_{ij}x_j + \theta_i)}{\exp(\sum_{j\neq i} A_{ij}x_j + \theta_i) + \exp(-\sum_{j\neq i} A_{ij}x_j - \theta_i)} \\ &= \sigma(\langle w, x' \rangle), \end{aligned} \tag{3.3}$$

where $x' = [x, 1] \in \{-1,1\}^n$, and $w = 2[A_{i1}, \cdots, A_{i(i-1)}, A_{i(i+1)}, \cdots, A_{in}, \theta_i] \in \mathbb{R}^n$. Moreover, w satisfies $\|w\|_1 \leq 2\lambda(A,\theta)$, where $\lambda(A,\theta)$ is the model width defined in Definition 3.2.1.

Following Fact 2, the natural approach to estimating the edge weights A_{ij} is to solve a logistic regression problem for each variable. For ease of notation, let us focus on the n-th variable (the algorithm directly applies to the rest variables). Given N i.i.d. samples $\{z^1, \cdots, z^N\}$, where $z^i \in \{-1,1\}^n$ from an Ising model $\mathcal{D}(A,\theta)$, we first transform the samples into $\{(x^i, y^i)\}_{i=1}^N$, where $x^i = [z_1^i, \cdots, z_{n-1}^i, 1] \in \{-1,1\}^n$ and $y^i = z_n^i \in \{-1,1\}$. By Fact 2, we know that $\mathbb{P}[y^i = 1 | x^i = x] = \sigma(\langle w^*, x \rangle)$ where $w^* = 2[A_{n1}, \cdots, A_{n(n-1)}, \theta_n] \in \mathbb{R}^n$ satisfies $\|w^*\|_1 \leq 2\lambda(A,\theta)$. Suppose that $\lambda(A,\theta) \leq \lambda$, we are then interested in recovering w^* by solving the following ℓ_1-constrained logistic regression problem

$$\hat{w} \in \arg\min_{w \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^N \ell(y^i \langle w, x^i \rangle) \qquad \text{s.t. } \|w\|_1 \leq 2\lambda, \tag{3.4}$$

where $\ell : \mathbb{R} \to \mathbb{R}$ is the loss function

$$\ell(y^i \langle w, x^i \rangle) = \ln(1 + e^{-y^i \langle w, x^i \rangle}) = \begin{cases} -\ln \sigma(\langle w, x^i \rangle), & \text{if } y^i = 1 \\ -\ln(1 - \sigma(\langle w, x^i \rangle)), & \text{if } y^i = -1 \end{cases} \tag{3.5}$$

Eq. (3.5) is essentially the negative log-likelihood of observing y^i given x^i at the current w.

Let $\hat{w}$ be a minimizer of (3.4). It is worth noting that in the high-dimensional regime ($N < n$), $\hat{w}$ may not be unique. In this case, we will show that *any* one of them would work. After solving the convex problem in (3.4), the edge weight is estimated as $\hat{A}_{nj} = \hat{w}_j/2$.

The pseudocode of the above algorithm is given in Algorithm 5. Solving the ℓ_1-constrained logistic regression problem will give us an estimator of the true edge weight. We then form the graph by keeping the edge that has estimated weight larger than $\eta/2$ (in absolute value).

Algorithm 5: Learning an Ising model via ℓ_1-constrained logistic regression

Input: N i.i.d. samples $\{z^1, \cdots, z^N\}$, where $z^m \in \{-1, 1\}^n$ for $m \in [N]$; an upper bound on $\lambda(A, \theta) \leq \lambda$; a lower bound on $\eta(A, \theta) \geq \eta > 0$.

Output: $\hat{A} \in \mathbb{R}^{n \times n}$, and an undirected graph $\hat{G}$ on n nodes.

for $i \leftarrow 1$ **to** n **do**
  $\forall m \in [N]$, $x^m \leftarrow [z^m_{-i}, 1]$, $y^m \leftarrow z^m_i$
  $\hat{w} \leftarrow \arg\min_{w \in \mathbb{R}^n} \frac{1}{N} \sum_{m=1}^N \ln(1 + e^{-y^m \langle w, x^m \rangle})$ s.t. $\|w\|_1 \leq 2\lambda$
  $\forall j \in [n]$, $\hat{A}_{ij} \leftarrow \hat{w}_{\tilde{j}}/2$, where $\tilde{j} = j$ if $j < i$ and $\tilde{j} = j - 1$ if $j > i$
end
Form an undirected graph $\hat{G}$ on n nodes with edges $\{(i, j) : |\hat{A}_{ij}| \geq \eta/2, i < j\}$.

Theorem 3.2.1. *Let $\mathcal{D}(A,\theta)$ be an unknown n-variable Ising model distribution with dependency graph G. Suppose that the $\mathcal{D}(A,\theta)$ has width $\lambda(A,\theta) \le \lambda$. Given $\rho \in (0,1)$ and $\epsilon > 0$, if the number of i.i.d. samples satisfies $N = O(\lambda^2 \exp(12\lambda)\ln(n/\rho)/\epsilon^4)$, then with probability at least $1-\rho$, Algorithm 5 produces $\hat{A}$ that satisfies*

$$\max_{i,j\in[n]} |A_{ij} - \hat{A}_{ij}| \le \epsilon. \tag{3.6}$$

Corollary 3.2.2. *In the setup of Theorem 3.2.1, suppose that the Ising model distribution $\mathcal{D}(A,\theta)$ has minimum edge weight $\eta(A,\theta) \ge \eta > 0$. If we set $\epsilon < \eta/2$ in (3.6), i.e., the sample complexity $N = O(\lambda^2 \exp(12\lambda)\ln(n/\rho)/\eta^4)$, then with probability at least $1-\rho$, Algorithm 5 recovers the dependency graph, i.e., $\hat{G} = G$.*

3.2.2 Learning Pairwise Models Over General Alphabet

Definition 3.2.2. Let k be the alphabet size. Let $\mathcal{W} = \{W_{ij} \in \mathbb{R}^{k\times k} : i \ne j \in [n]\}$ be a set of weight matrices satisfying $W_{ij} = W_{ji}^T$. Without loss of generality, we assume that every row (and column) vector of W_{ij} has zero mean. Let $\Theta = \{\theta_i \in \mathbb{R}^k : i \in [n]\}$ be a set of external field vectors. Then the n-variable pairwise graphical model $\mathcal{D}(\mathcal{W},\Theta)$ is a distribution over $[k]^n$ where

$$\mathop{\mathbb{P}}_{Z\sim\mathcal{D}(\mathcal{W},\Theta)}[Z = z] \propto \exp\Big(\sum_{1\le i<j\le n} W_{ij}(z_i,z_j) + \sum_{i\in[n]} \theta_i(z_i)\Big). \tag{3.7}$$

The dependency graph of $\mathcal{D}(\mathcal{W},\Theta)$ is an undirected graph $G = (V,E)$, with vertices $V = [n]$ and edges $E = \{(i,j) : W_{ij} \ne 0\}$. The width of $\mathcal{D}(\mathcal{W},\Theta)$ is

defined as

$$\lambda(\mathcal{W}, \Theta) = \max_{i,a}(\sum_{j \neq i} \max_{b \in [k]} |W_{ij}(a,b)| + |\theta_i(a)|). \tag{3.8}$$

We define $\eta(\mathcal{W}, \Theta) = \min_{(i,j) \in E} \max_{a,b} |W_{ij}(a,b)|$.

Remark. The assumption that W_{ij} has centered rows and columns (i.e., $\sum_b W_{ij}(a,b) = 0$ and $\sum_a W_{ij}(a,b) = 0$ for any $a, b \in [k]$) is without loss of generality (see Fact 8.2 in (Klivans and Meka, 2017)). If the a-th row of W_{ij} is not centered, i.e., $\sum_b W_{ij}(a,b) \neq 0$, we can define $W'_{ij}(a,b) = W_{ij}(a,b) - \sum_b W_{ij}(a,b)/k$ and $\theta'_i(a) = \theta_i(a) + \sum_b W_{ij}(a,b)/k$, and notice that $\mathcal{D}(\mathcal{W}, \Theta) = \mathcal{D}(\mathcal{W}', \Theta')$. Because the sets of matrices with centered rows and columns (i.e., $\{M \in \mathbb{R}^{k \times k} : \sum_b M(a,b) = 0, \forall a \in [k]\}$ and $\{M \in \mathbb{R}^{k \times k} : \sum_a M(a,b) = 0, \forall b \in [k]\}$) are two linear subspaces, alternatively projecting W_{ij} onto the two sets will converge to the intersection of the two subspaces (Von Neumann, 1949). As a result, the condition of centered rows and columns is necessary for recovering the underlying weight matrices, since otherwise different parameters can give the same distribution. Note that in the case of $k = 2$, Definition 3.2.2 is the same as Definition 3.2.1 for Ising models. To see their connection, simply define $W_{ij} \in \mathbb{R}^{2 \times 2}$ as follows: $W_{ij}(1,1) = W_{ij}(2,2) = A_{ij}$, $W_{ij}(1,2) = W_{ij}(2,1) = -A_{ij}$.

For a pairwise graphical model distribution $\mathcal{D}(\mathcal{W}, \Theta)$, the conditional distribution of any variable (when restricted to a pair of values) given all the other variables follows a logistic function, as shown in Fact 3. This is analogous to Fact 2 for the Ising model distribution.

Fact 3. *Let $Z \sim \mathcal{D}(\mathcal{W}, \Theta)$ and $Z \in [k]^n$. For any $i \in [n]$, any $\alpha \neq \beta \in [k]$, and any $x \in [k]^{n-1}$,*

$$\mathbb{P}[Z_i = \alpha | Z_i \in \{\alpha, \beta\}, Z_{-i} = x] = \sigma(\sum_{j \neq i} (W_{ij}(\alpha, x_j) - W_{ij}(\beta, x_j)) + \theta_i(\alpha) - \theta_i(\beta)). \tag{3.9}$$

Given N i.i.d. samples $\{z^1, \cdots, z^N\}$, where $z^m \in [k]^n \sim \mathcal{D}(\mathcal{W}, \Theta)$ for $m \in [N]$, the goal is to estimate matrices W_{ij} for all $i \neq j \in [n]$. For ease of notation and without loss of generality, let us consider the n-th variable. Now the goal is to estimate matrices W_{nj} for all $j \in [n-1]$.

To use Fact 3, fix a pair of values $\alpha \neq \beta \in [k]$, let S be the set of samples satisfying $z_n \in \{\alpha, \beta\}$. We next transform the samples in S to $\{(x^t, y^t)\}_{t=1}^{|S|}$ as follows: $x^t = \text{OneHotEncode}([z^t_{-n}, 1]) \in \{0,1\}^{n \times k}$, $y^t = 1$ if $z^t_n = \alpha$, and $y^t = -1$ if $z^t_n = \beta$. Here $\text{OneHotEncode}(\cdot) : [k]^n \to \{0,1\}^{n \times k}$ is a function that maps a value $t \in [k]$ to the standard basis vector $e_t \in \{0,1\}^k$, where e_t has a single 1 at the t-th entry. For each sample (x, y) in the set S, Fact 3 implies that $\mathbb{P}[y = 1|x] = \sigma(\langle w^*, x \rangle)$, where $w^* \in \mathbb{R}^{n \times k}$ satisfies

$$w^*(j,:) = W_{nj}(\alpha,:) - W_{nj}(\beta,:), \forall j \in [n-1]; \quad w^*(n,:) = [\theta_n(\alpha) - \theta_n(\beta), 0, ..., 0]. \tag{3.10}$$

Suppose that the width of $\mathcal{D}(\mathcal{W}, \Theta)$ satisfies $\lambda(\mathcal{W}, \Theta) \leq \lambda$, then w^* defined in (3.10) satisfies $\|w^*\|_{2,1} \leq 2\lambda\sqrt{k}$, where $\|w^*\|_{2,1} := \sum_j \|w^*(j,:)\|_2$. We can now form an $\ell_{2,1}$-constrained logistic regression over the samples in S:

$$w^{\alpha,\beta} \in \arg\min_{w \in \mathbb{R}^{n \times k}} \frac{1}{|S|} \sum_{t=1}^{|S|} \ln(1 + e^{-y^t \langle w, x^t \rangle}) \qquad \text{s.t. } \|w\|_{2,1} \leq 2\lambda\sqrt{k}, \tag{3.11}$$

Let $w^{\alpha,\beta}$ be a minimizer of (3.11). Without loss of generality, we can assume that the first $n-1$ rows of $w^{\alpha,\beta}$ are centered, i.e., $\sum_a w^{\alpha,\beta}(j,a) = 0$ for $j \in [n-1]$. Otherwise, we can always define a new matrix $U^{\alpha,\beta} \in \mathbb{R}^{n\times k}$ by centering the first $n-1$ rows of $w^{\alpha,\beta}$:

$$U^{\alpha,\beta}(j,b) = w^{\alpha,\beta}(j,b) - \frac{1}{k}\sum_{a\in[k]} w^{\alpha,\beta}(j,a),\ \forall j \in [n-1],\ \forall b \in [k]; \tag{3.12}$$
$$U^{\alpha,\beta}(n,b) = w^{\alpha,\beta}(n,b) + \frac{1}{k}\sum_{j\in[n-1],a\in[k]} w^{\alpha,\beta}(j,a),\ \forall b \in [k].$$

Since each row of the x matrix in (3.11) is a standard basis vector (i.e., all zeros except a single one), $\left\langle U^{\alpha,\beta}, x\right\rangle = \left\langle w^{\alpha,\beta}, x\right\rangle$, which implies that $U^{\alpha,\beta}$ is also a minimizer of (3.11).

The key step in our proof is to show that given enough samples, the obtained $U^{\alpha,\beta} \in \mathbb{R}^{n\times k}$ matrix is close to w^* defined in (3.10). Specifically, we will prove that

$$|W_{nj}(\alpha,b) - W_{nj}(\beta,b) - U^{\alpha,\beta}(j,b)| \le \epsilon, \quad \forall j \in [n-1],\ \forall \alpha,\beta,b \in [k]. \tag{3.13}$$

Recall that our goal is to estimate the original matrices W_{nj} for all $j \in [n-1]$. Summing (3.13) over $\beta \in [k]$ (suppose $U^{\alpha,\alpha} = 0$) and using the fact that $\sum_\beta W_{nj}(\beta,b) = 0$ gives

$$|W_{nj}(\alpha,b) - \frac{1}{k}\sum_{\beta\in[k]} U^{\alpha,\beta}(j,b)| \le \epsilon, \quad \forall j \in [n-1],\ \forall \alpha,b \in [k]. \tag{3.14}$$

In other words, $\hat{W}_{nj}(\alpha,:) = \sum_{\beta\in[k]} U^{\alpha,\beta}(j,:)/k$ is a good estimate of $W_{nj}(\alpha,:)$.

Suppose that $\eta(\mathcal{W}, \Theta) \geq \eta$, once we obtain the estimates $\hat{W}_{ij}$, the last step is to form a graph by keeping the edge (i, j) that satisfies $\max_{a,b} |\hat{W}_{ij}(a, b)| \geq \eta/2$. The pseudocode of the above algorithm is given in Algorithm 6.

Algorithm 6: Learning a pairwise graphical model via $\ell_{2,1}$-constrained logistic regression

Input: alphabet size k; N i.i.d. samples $\{z^1, \cdots, z^N\}$, where $z^m \in [k]^n$ for $m \in [N]$; an upper bound on $\lambda(\mathcal{W}, \Theta) \leq \lambda$; a lower bound on $\eta(\mathcal{W}, \Theta) \geq \eta > 0$.

Output: $\hat{W}_{ij} \in \mathbb{R}^{k \times k}$ for all $i \neq j \in [n]$; an undirected graph $\hat{G}$ on n nodes.

for $i \leftarrow 1$ **to** n **do**
  for *each pair* $\alpha \neq \beta \in [k]$ **do**
    $S \leftarrow \{z^m, m \in [N] : z_i^m \in \{\alpha, \beta\}\}$.
    for $z^t \in S$ **do**
      $x^t \leftarrow \text{OneHotEncode}([z_{-i}^t, 1])$.
      $y^t \leftarrow 1$ if $z_i^t = \alpha$; $y^t \leftarrow -1$ if $z_i^t = \beta$.
    end
    $w^{\alpha,\beta} \leftarrow \arg\min_{w \in \mathbb{R}^{n \times k}} \frac{1}{|S|} \sum_{t=1}^{|S|} \ln(1 + e^{-y^t \langle w, x^t \rangle})$
      s.t. $\|w\|_{2,1} \leq 2\lambda\sqrt{k}$.
    Define $U^{\alpha,\beta} \in \mathbb{R}^{n \times k}$ by centering the first $n-1$ rows of $w^{\alpha,\beta}$ (see (3.12)).
  end
  for $j \in [n] \backslash i$ *and* $\alpha \in [k]$ **do**
    $\hat{W}_{ij}(\alpha, :) \leftarrow \frac{1}{k} \sum_{\beta \in [k]} U^{\alpha,\beta}(\tilde{j}, :)$, where $\tilde{j} = j$ if $j < i$ and $\tilde{j} = j - 1$ if $j > i$.
  end
end
Form graph $\hat{G}$ on n nodes with edges $\{(i, j) : \max_{a,b} |\hat{W}_{ij}(a, b)| \geq \eta/2, i < j\}$.

Theorem 3.2.3. *Let $\mathcal{D}(\mathcal{W}, \Theta)$ be an n-variable pairwise graphical model distribution with width $\lambda(\mathcal{W}, \Theta) \leq \lambda$. Given $\rho \in (0, 1)$ and $\epsilon > 0$, if the number of*

i.i.d. samples satisfies $N = O(\lambda^2 k^4 \exp(14\lambda) \ln(nk/\rho)/\epsilon^4)$*, then with probability at least* $1 - \rho$*, Algorithm 6 produces* $\hat{W}_{ij} \in \mathbb{R}^{k \times k}$ *that satisfies*

$$|W_{ij}(a,b) - \hat{W}_{ij}(a,b)| \leq \epsilon, \quad \forall i \neq j \in [n],\ \forall a, b \in [k]. \tag{3.15}$$

Corollary 3.2.4. *In the setup of Theorem 3.2.3, suppose that the pairwise graphical model distribution* $\mathcal{D}(\mathcal{W}, \Theta)$ *satisfies* $\eta(\mathcal{W}, \Theta) \geq \eta > 0$*. If we set* $\epsilon < \eta/2$ *in (3.15), i.e., the sample complexity* $N = O(\lambda^2 k^4 \exp(14\lambda) \ln(nk/\rho)/\eta^4)$*, then with probability at least* $1 - \rho$*, Algorithm 6 recovers the dependency graph, i.e.,* $\hat{G} = G$*.*

Remark. The $w^* \in \mathbb{R}^{n \times k}$ matrix defined in (3.10) satisfies $\|w^*\|_{\infty,1} \leq 2\lambda(\mathcal{W}, \Theta)$. This implies that $\|w^*\|_{2,1} \leq 2\lambda(\mathcal{W}, \Theta)\sqrt{k}$ and $\|w^*\|_1 \leq 2\lambda(\mathcal{W}, \Theta)k$. Instead of solving the $\ell_{2,1}$-constrained logistic regression defined in (3.11), we could solve an ℓ_1-constrained logistic regression with $\|w\|_1 \leq 2\lambda(\mathcal{W}, \Theta)k$. However, this will lead to a sample complexity that scales as $\tilde{O}(k^5)$, which is worse than the $\tilde{O}(k^4)$ sample complexity achieved by the $\ell_{2,1}$-constrained logistic regression. The reason why we use the $\ell_{2,1}$ constraint instead of the tighter $\ell_{\infty,1}$ constraint in the algorithm is because our proof relies on a sharp generalization bound for $\ell_{2,1}$-constrained logistic regression (see Lemma B.2.4 in the appendix). It is unclear whether a similar generalization bound exists for the $\ell_{\infty,1}$ constraint.

3.2.3 Learning Pairwise Models in $\tilde{O}(n^2)$ Time

Our results so far assume that the ℓ_1-constrained logistic regression (in Algorithm 5) and the $\ell_{2,1}$-constrained logistic regression (in Algorithm 6) can be solved exactly. This would require $\tilde{O}(n^4)$ complexity if an interior-point based method is used (Koh et al., 2007). The goal of this section is to reduce the runtime to $\tilde{O}(n^2)$ via first-order optimization method. Note that $\tilde{O}(n^2)$ is an efficient time complexity for graph recovery over n nodes. Previous structural learning algorithms of Ising models require either $\tilde{O}(n^2)$ complexity (e.g., (Bresler, 2015; Klivans and Meka, 2017)) or a worse complexity (e.g., (Ravikumar et al., 2010; Vuffray et al., 2016) require $\tilde{O}(n^4)$ runtime). We would like to remark that our goal here is not to give the fastest first-order optimization algorithm (see our remark after Theorem 3.2.6). Instead, our contribution is to provably show that it is possible to run Algorithm 5 and Algorithm 6 in $\tilde{O}(n^2)$ time without affecting the original statistical guarantees.

To better exploit the problem structure[4], we use the mirror descent algorithm[5] with a properly chosen distance generating function (aka the mirror

[4]Specifically, for the ℓ_1-constrained logisitic regression defined in (3.4), since the input sample satisifies $\|x\|_\infty = 1$, the loss function is $O(1)$-Lipschitz w.r.t. $\|\cdot\|_1$. Similarly, for the $\ell_{2,1}$-constrained logisitic regression defined in (3.11), the loss function is $O(1)$-Lipschitz w.r.t. $\|\cdot\|_{2,1}$ because the input sample satisfies $\|x\|_{2,\infty} = 1$.

[5]Other approaches include the standard projected gradient descent and the coordinate descent. Their convergence rates depend on either the smoothness or the Lipschitz constant (w.r.t. $\|\cdot\|_2$) of the objective function (Bubeck, 2015). This would lead to a total runtime of $\tilde{O}(n^3)$ for our problem setting. Another option would be the composite gradient descent method, the analysis of which relies on the restricted strong convexity of the objective function (Agarwal et al., 2010). For other variants of mirror descent algorithms, see the remark after Theorem 3.2.6.

map). Following the standard mirror descent setup, we use negative entropy as the mirror map for ℓ_1-constrained logistic regression and a scaled group norm for $\ell_{2,1}$-constrained logistic regression (see Section 5.3.3.2 and Section 5.3.3.3 in (Ben-Tal and Nemirovski, 2013) for more details). The pseudocode is given in Appendix B.8. The main advantage of mirror descent algorithm is that its convergence rate scales *logarithmically* in the dimension (see Lemma B.9.1 in Appendix B.9). Specifically, let $\bar{w}$ be the output after $O(\ln(n)/\gamma^2)$ mirror descent iterations, then $\bar{w}$ satisfies

$$\hat{\mathcal{L}}(\bar{w}) - \hat{\mathcal{L}}(\hat{w}) \leq \gamma, \tag{3.16}$$

where $\hat{\mathcal{L}}(w) = \sum_{i=1}^{N} \ln(1 + e^{-y^i \langle w, x^i \rangle})/N$ is the empirical logistic loss, and $\hat{w}$ is the actual minimizer of $\hat{\mathcal{L}}(w)$. Since each mirror descent update requires $O(nN)$ time, where N is the number of samples and scales as $O(\ln(n))$, and we have to solve n regression problems (one for each variable in $[n]$), the total runtime scales as $\tilde{O}(n^2)$, which is our desired runtime.

There is still one problem left, that is, we have to show that $\|\bar{w} - w^*\|_\infty \leq \epsilon$ (where w^* is the minimizer of the true loss $\mathcal{L}(w) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \ln(1 + e^{-y\langle w, x \rangle})$) in order to conclude that Theorem 3.2.1 and 3.2.3 still hold when using mirror descent algorithms. Since $\hat{\mathcal{L}}(w)$ is not strongly convex, (3.16) alone does not necessarily imply that $\|\bar{w} - \hat{w}\|_\infty$ is small. Our key insight is that in the proof of Theorem 3.2.1 and 3.2.3, the definition of $\hat{w}$ (as a minimizer of $\hat{\mathcal{L}}(w)$) is only used to show that $\hat{\mathcal{L}}(\hat{w}) \leq \hat{\mathcal{L}}(w^*)$ (see inequality (b) of (B.10) in Appendix B.2). It is then possible to replace this step with (3.16) in the original proof, and

prove that Theorem 3.2.1 and 3.2.3 still hold as long as γ is small enough (see (B.41) in Appendix B.9).

Our key results in this section are Theorem 3.2.5 and Theorem 3.2.6, which show that Algorithm 5 and Algorithm 6 can run in $\tilde{O}(n^2)$ time without affecting the original statistical guarantees.

Theorem 3.2.5. *In the setup of Theorem 3.2.1, suppose that the ℓ_1-constrained logistic regression in Algorithm 5 is optimized by the mirror descent method (Algorithm 8) given in Appendix B.8. Given $\rho \in (0,1)$ and $\epsilon > 0$, if the number of mirror descent iterations satisfies $T = O(\lambda^2 \exp(12\lambda)\ln(n)/\epsilon^4)$, and the number of i.i.d. samples satisfies $N = O(\lambda^2 \exp(12\lambda)\ln(n/\rho)/\epsilon^4)$, then (3.6) still holds with probability at least $1-\rho$. The total time complexity of Algorithm 5 is $O(TNn^2)$.*

Theorem 3.2.6. *In the setup of Theorem 3.2.3, suppose that the $\ell_{2,1}$-constrained logistic regression in Algorithm 6 is optimized by the mirror descent method (Algorithm 9) given in Appendix B.8. Given $\rho \in (0,1)$ and $\epsilon > 0$, if the number of mirror descent iterations satisfies $T = O(\lambda^2 k^3 \exp(12\lambda)\ln(n)/\epsilon^4)$, and the number of i.i.d. samples satisfies $N = O(\lambda^2 k^4 \exp(14\lambda)\ln(nk/\rho)/\epsilon^4)$, then (3.15) still holds with probability at least $1-\rho$. The total time complexity of Algorithm 6 is $O(TNn^2k^2)$.*

Remark. It is possible to improve the time complexity given in Theorem 3.2.5 and 3.2.6 (especially the dependence on ϵ and λ), by using stochastic or accelerated versions of mirror descent algorithms (instead of the batch

version given in Appendix B.8). In fact, the Sparsitron algorithm proposed by Klivans and Meka (2017) can be seen as an online mirror descent algorithm for optimizing the ℓ_1-constrained logistic regression (see Algorithm 8 in Appendix B.8). Furthermore, Algorithm 5 and 6 can be parallelized as the regression problem is defined separately for each variable.

3.3 Analysis

3.3.1 Proof Outline

We give a proof outline for Theorem 3.2.1. The proof of Theorem 3.2.3 follows a similar outline. Let D be a distribution over $\{-1,1\}^n \times \{-1,1\}$, where $(x,y) \sim D$ satisfies $\mathbb{P}[y=1|x] = \sigma(\langle w^*, x\rangle)$. Let $\mathcal{L}(w) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \ln(1+e^{-y\langle w,x\rangle})$ and $\hat{\mathcal{L}}(w) = \sum_{i=1}^{N} \ln(1+e^{-y^i\langle w,x^i\rangle})/N$ be the expected and empirical logistic loss. Suppose $\|w^*\|_1 \leq 2\lambda$. Let $\hat{w} \in \arg\min_w \hat{\mathcal{L}}(w)$ s.t. $\|w\|_1 \leq 2\lambda$. Our goal is to prove that $\|\hat{w} - w^*\|_\infty$ is small when the samples are constructed from an Ising model distribution. Our proof can be summarized in three steps:

1. If the number of samples satisfies $N = O(\lambda^2 \ln(n/\rho)/\gamma^2)$, then $\mathcal{L}(\hat{w}) - \mathcal{L}(w^*) \leq O(\gamma)$. This is obtained using a sharp generalization bound when $\|w\|_1 \leq 2\lambda$ and $\|x\|_\infty \leq 1$ (see Lemma B.2.1 in Appendix B.2).

2. For any w, we show that $\mathcal{L}(w) - \mathcal{L}(w^*) \geq \mathbb{E}_x[\sigma(\langle w, x\rangle) - \sigma(\langle w^*, x\rangle)]^2$ (see Lemma B.2.3 and Lemma B.2.2 in Appendix B.2). Hence, Step 1 implies that $\mathbb{E}_x[\sigma(\langle \hat{w}, x\rangle) - \sigma(\langle w^*, x\rangle)]^2 \leq O(\gamma)$ (see Lemma 3.3.1 in the next subsection).

3. We now use a result from (Klivans and Meka, 2017) (see Lemma 3.3.5 in the next subsection), which says that if the samples are from an Ising model and if $\gamma = O(\epsilon^2 \exp(-6\lambda))$, then $\mathbb{E}_x[\sigma(\langle \hat{w}, x\rangle) - \sigma(\langle w^*, x\rangle)]^2 \leq O(\gamma)$ implies that $\|\hat{w} - w^*\|_\infty \leq \epsilon$. The required number of samples is $N = O(\lambda^2 \ln(n/\rho)/\gamma^2) = O(\lambda^2 \exp(12\lambda) \ln(n/\rho)/\epsilon^4)$.

For the general setting with non-binary alphabet (i.e., Theorem 3.2.3), the proof is similar to that of Theorem 3.2.1. The main difference is that we need to use a sharp generalization bound when $\|w\|_{2,1} \leq 2\lambda\sqrt{k}$ and $\|x\|_{2,\infty} \leq 1$ (see Lemma B.2.4 in Appendix B.2). This would give us Lemma 3.3.2 (instead of Lemma 3.3.1 for the Ising models). The last step is to use Lemma 3.3.6 to bound the infinity norm between the two weight matrices.

3.3.2 Supporting Lemmas

Lemma 3.3.1 and Lemma 3.3.2 are the key results in our proof. They essentially say that given enough samples, solving the corresponding constrained logistic regression problem will provide a prediction $\sigma(\langle \hat{w}, x\rangle)$ close to the true $\sigma(\langle w^*, x\rangle)$ in terms of their expected squared distance.

Lemma 3.3.1. *Let $\mathcal{D}$ be a distribution on $\{-1,1\}^n \times \{-1,1\}$ where for $(X,Y) \sim \mathcal{D}$, $\mathbb{P}[Y = 1|X = x] = \sigma(\langle w^*, x\rangle)$. We assume that $\|w^*\|_1 \leq 2\lambda$ for a known $\lambda \geq 0$. Given N i.i.d. samples $\{(x^i, y^i)\}_{i=1}^N$, let $\hat{w}$ be any minimizer of the following ℓ_1-constrained logistic regression problem:*

$$\hat{w} \in \arg\min_{w \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^{N} \ln(1 + e^{-y^i \langle w, x^i\rangle}) \quad \text{s.t. } \|w\|_1 \leq 2\lambda. \tag{3.17}$$

Given $\rho \in (0,1)$ and $\epsilon > 0$, if the number of samples satisfies

$$N = O(\lambda^2 \ln(n/\rho)/\epsilon^2), \tag{3.18}$$

then with probability at least $1-\rho$ over the samples, $\mathbb{E}_{(x,y)\sim\mathcal{D}}[(\sigma(\langle w^*, x\rangle) - \sigma(\langle \hat{w}, x\rangle))^2] \leq \epsilon$.

Lemma 3.3.2. *Let $\mathcal{D}$ be a distribution on $\mathcal{X} \times \{-1,1\}$, where $\mathcal{X} = \{x \in \{0,1\}^{n\times k} : \|x\|_{2,\infty} \leq 1\}$. Furthermore, $(X,Y) \sim \mathcal{D}$ satisfies $\mathbb{P}[Y=1|X=x] = \sigma(\langle w^*, x\rangle)$, where $w^* \in \mathbb{R}^{n\times k}$. We assume that $\|w^*\|_{2,1} \leq 2\lambda\sqrt{k}$ for a known $\lambda \geq 0$. Given N i.i.d. samples $\{(x^i, y^i)\}_{i=1}^N$ from $\mathcal{D}$, let $\hat{w}$ be any minimizer of the following $\ell_{2,1}$-constrained logistic regression problem:*

$$\hat{w} \in \operatorname*{arg\,min}_{w\in\mathbb{R}^{n\times k}} \frac{1}{N}\sum_{i=1}^{N} \ln(1 + e^{-y^i\langle w, x^i\rangle}) \quad \text{s.t. } \|w\|_{2,1} \leq 2\lambda\sqrt{k}. \tag{3.19}$$

Given $\rho \in (0,1)$ and $\epsilon > 0$, if the number of samples satisfies

$$N = O(\lambda^2 k(\ln(n/\rho))/\epsilon^2), \tag{3.20}$$

then with probability at least $1-\rho$ over the samples, $\mathbb{E}_{(x,y)\sim\mathcal{D}}[(\sigma(\langle w^*, x\rangle) - \sigma(\langle \hat{w}, x\rangle))^2] \leq \epsilon$.

The proofs of Lemma 3.3.1 and Lemma 3.3.2 are given in Appendix B.2. Note that in the setup of both lemmas, we form a pair of dual norms for x and w, e.g., $\|x\|_{2,\infty}$ and $\|w\|_{2,1}$ in Lemma 3.3.2, and $\|x\|_\infty$ and $\|w\|_1$ in Lemma 3.3.1. This duality allows us to use a sharp generalization bound with a sample complexity that scales logarithmic in the dimension (see Lemma B.2.1 and Lemma B.2.4 in Appendix B.2).

Definition 3.3.1 defines a δ-unbiased distribution. This notion of δ-unbiasedness is proposed by Klivans and Meka (2017).

Definition 3.3.1. Let S be the alphabet set, e.g., $S = \{-1, 1\}$ for Ising model and $S = [k]$ for an alphabet of size k. A distribution $\mathcal{D}$ on S^n is δ-unbiased if for $X \sim \mathcal{D}$, any $i \in [n]$, and any assignment $x \in S^{n-1}$ to X_{-i}, $\min_{\alpha \in S}(\mathbb{P}[X_i = \alpha | X_{-i} = x]) \geq \delta$.

For a δ-unbiased distribution, any of its marginal distribution is also δ-unbiased (see Lemma 3.3.3).

Lemma 3.3.3. *Let $\mathcal{D}$ be a δ-unbiased distribution on S^n, where S is the alphabet set. For $X \sim \mathcal{D}$, any $i \in [n]$, the distribution of X_{-i} is also δ-unbiased.*

Lemma 3.3.4 describes the δ-unbiased property of graphical models. This property has been used in the previous papers (e.g., (Klivans and Meka, 2017; Bresler, 2015)).

Lemma 3.3.4. *Let $\mathcal{D}(\mathcal{W}, \Theta)$ be a pairwise graphical model distribution with alphabet size k and width $\lambda(\mathcal{W}, \Theta)$. Then $\mathcal{D}(\mathcal{W}, \Theta)$ is δ-unbiased with $\delta = e^{-2\lambda(\mathcal{W},\Theta)}/k$. Specifically, an Ising model distribution $\mathcal{D}(A, \theta)$ is $e^{-2\lambda(A,\theta)}/2$-unbiased.*

In Lemma 3.3.1 and Lemma 3.3.2, we give a sample complexity bound for achieving a small ℓ_2 error between $\sigma(\langle \hat{w}, x \rangle)$ and $\sigma(\langle w^*, x \rangle)$. The following

two lemmas show that if the sample distribution is δ-unbiased, then a small ℓ_2 error implies a small distance between $\hat{w}$ and w^*.

Lemma 3.3.5. *Let $\mathcal{D}$ be a δ-unbiased distribution on $\{-1,1\}^n$. Suppose that for two vectors $u, w \in \mathbb{R}^n$ and $\theta', \theta'' \in \mathbb{R}$, $\mathbb{E}_{X\sim\mathcal{D}}[(\sigma(\langle w, X\rangle + \theta') - \sigma(\langle u, X\rangle + \theta''))^2] \le \epsilon$, where $\epsilon < \delta e^{-2\|w\|_1 - 2|\theta'| - 6}$. Then $\|w-u\|_\infty \le O(1)\cdot e^{\|w\|_1+|\theta'|}\cdot\sqrt{\epsilon/\delta}$.*

Lemma 3.3.6. *Let $\mathcal{D}$ be a δ-unbiased distribution on $[k]^n$. For $X \sim \mathcal{D}$, let $\tilde{X} \in \{0,1\}^{n\times k}$ be the one-hot encoded X. Let $u, w \in \mathbb{R}^{n\times k}$ be two matrices satisfying $\sum_a u(i,a) = 0$ and $\sum_a w(i,a) = 0$, for $i \in [n]$. Suppose that for some u, w and $\theta', \theta'' \in \mathbb{R}$, we have $\mathbb{E}_{X\sim\mathcal{D}}[(\sigma(\left\langle w, \tilde{X}\right\rangle + \theta') - \sigma(\left\langle u, \tilde{X}\right\rangle + \theta''))^2] \le \epsilon$, where $\epsilon < \delta e^{-2\|w\|_{\infty,1} - 2|\theta'| - 6}$. Then[6] $\|w-u\|_\infty \le O(1)\cdot e^{\|w\|_{\infty,1}+|\theta'|}\cdot\sqrt{\epsilon/\delta}$.*

The proofs of Lemma 3.3.5 and Lemma 3.3.6 can be found in (Klivans and Meka, 2017) (see Claim 8.6 and Lemma 4.3 in their paper). We give a slightly different proof of these two lemmas in Appendix B.5.

3.3.3 Proof Sketches

We provide proof sketches for Theorem 3.2.1 and Theorem 3.2.3 using the supporting lemmas. The detailed proof can be found in Appendix B.6 and B.7.

[6] For a matrix w, we define $\|w\|_\infty = \max_{ij}|w(i,j)|$. Note that this is different from the induced matrix norm.

Proof sketch of Theorem 3.2.1. Without loss of generality, let us consider the n-th variable. Let $Z \sim \mathcal{D}(A, \theta)$, and $X = [Z_1, Z_2, \cdots, Z_{n-1}, 1] \in \{-1, 1\}^n$. By Fact 2 and Lemma 3.3.1, if $N = O(\lambda^2 \ln(n/\rho)/\gamma^2)$, then $\mathbb{E}_X[(\sigma(\langle w^*, X\rangle) - \sigma(\langle \hat{w}, X\rangle))^2] \leq \gamma$ with probability at least $1 - \rho/n$. By Lemma 3.3.4 and Lemma 3.3.3, Z_{-n} is δ-unbiased with $\delta = e^{-2\lambda}/2$. We can then apply Lemma 3.3.5 to show that if $N = O(\lambda^2 \exp(12\lambda) \ln(n/\rho)/\epsilon^4)$, then $\max_{j\in[n]} |A_{nj} - \hat{A}_{nj}| \leq \epsilon$ with probability at least $1 - \rho/n$. Theorem 3.2.1 then follows by a union bound over all n variables.

Proof sketch of Theorem 3.2.3. Let us again consider the n-th variable since the proof is the same for all other variables. As described before, the key step is to show that (3.13) holds. Now fix a pair of $\alpha \neq \beta \in [k]$, let $N^{\alpha,\beta}$ be the number of samples such that the n-th variable is either α or β. By Fact 3 and Lemma 3.3.2, if $N^{\alpha,\beta} = O(\lambda^2 k \ln(n/\rho')/\gamma^2)$, then with probability at least $1 - \rho'$, the matrix $U^{\alpha,\beta} \in \mathbb{R}^{n\times k}$ satisfies $\mathbb{E}_x[(\sigma(\langle w^*, x\rangle) - \sigma(\langle U^{\alpha,\beta}, x\rangle))^2] \leq \gamma$, where $w^* \in \mathbb{R}^{n\times k}$ is defined in (3.10). By Lemma 3.3.6 and Lemma 3.3.4, if $N^{\alpha,\beta} = O(\lambda^2 k^3 \exp(12\lambda) \ln(n/\rho'))/\epsilon^4)$, then with probability at least $1 - \rho'$, $|W_{nj}(\alpha, b) - W_{nj}(\beta, b) - U^{\alpha,\beta}(j, b)| \leq \epsilon$, $\forall j \in [n-1]$, $\forall b \in [k]$. Since $\mathcal{D}(\mathcal{W}, \Theta)$ is δ-unbiased with $\delta = e^{-2\lambda}/k$, in order to have $N^{\alpha,\beta}$ samples for a given (α, β) pair, we need the total number of samples to satisfy $N = O(N^{\alpha,\beta}/\delta)$. Theorem 3.2.3 then follows by setting $\rho' = \rho/(nk^2)$ and taking a union bound over all (α, β) pairs and all n variables.

3.4 Experiments

In all the experiments, we assume that the external field is zero. Sampling is done via exactly computing the distribution.

3.4.1 Learning Ising Models

In Figure 3.1 we construct a diamond-shape graph and show that the incoherence value at Node 1 becomes bigger than 1 (and hence violates the incoherence condition in (Ravikumar et al., 2010)) when we increase the graph size n and edge weight a. We then run 100 times of Algorithm 5 and plot the fraction of runs that exactly recovers the underlying graph structure. In each run we generate a different set of samples. The result shown in Figure 3.1 is consistent with our analysis and also indicates that our conditions for graph recovery are weaker than those in (Ravikumar et al., 2010).

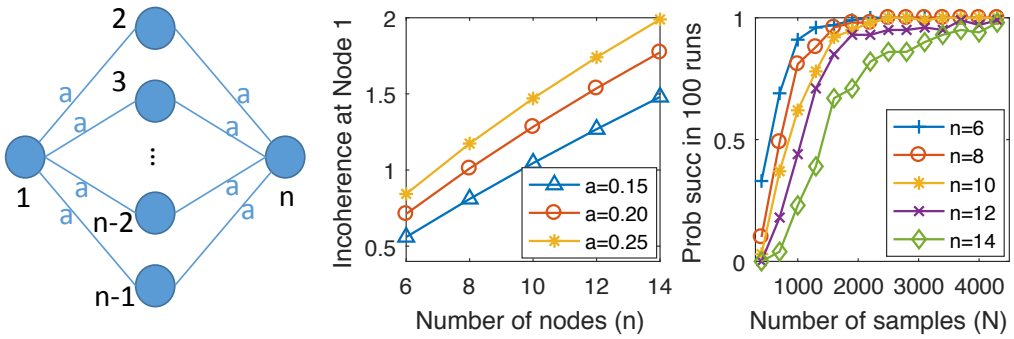


Figure 3.1: **Left**: The graph structure used in this simulation. It has n nodes and $2(n-2)$ edges. Every edge has the same weight $a > 0$. **Middle**: Incoherence value at Node 1. **Right**: We simulate 100 runs of Algorithm 5 for the diamond graph with edge weight $a = 0.2$.

3.4.2 Learning General Pairwise Graphical Models

We compare our algorithm (Algorithm 6) with the Sparsitron algorithm in (Klivans and Meka, 2017) on a two-dimensional 3-by-3 grid (shown in Figure 3.2). We experiment two alphabet sizes: $k = 4, 6$. For each value of k, we simulate both algorithms 100 runs, and in each run we generate random W_{ij} matrices with entries ± 0.2. As shown in the Figure 3.2, our algorithm requires fewer samples for successfully recovering the graphs. More details about this experiment can be found in Appendix B.10.

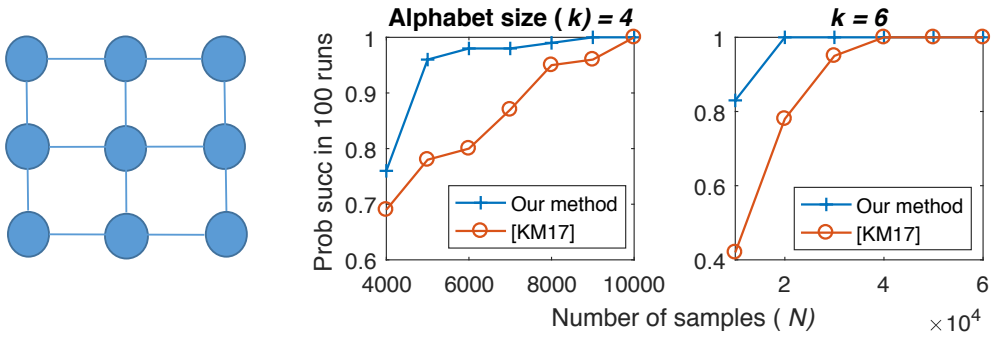


Figure 3.2: **Left**: A two-dimensional 3-by-3 grid graph used in the simulation. **Middle and right**: Our algorithm needs fewer samples than the Sparsitron algorithm for graph recovery.

3.5 Conclusion

The main contribution of this project is to theoretically prove the state-of-the-art performance of an existing and popular algorithm (i.e., properly regularized logistic regression), in a setting where alternative algorithms are being proposed. Specifically, we have shown that the $\ell_{2,1}$-constrained logistic

regression can recover the Markov graph of any discrete pairwise graphical model from i.i.d. samples. For the Ising model, it reduces to the ℓ_1-constrained logistic regression. This algorithm has better sample complexity than the previous state-of-the-art result (k^4 versus k^5), and can run in $\tilde{O}(n^2)$ time. One interesting direction for future work is to see if the $1/\eta^4$ dependency in the sample complexity can be improved.

Another interesting direction is to consider MRFs with higher-order interactions. Intuitively, it should not be difficult to prove that ℓ_1-constrained logistic regression can recover the structure of binary t-wise MRFs. One can prove it by combining results from Section 7 of (Klivans and Meka, 2017) and the following fact: the Sparsitron algorithm can be viewed as an online mirror descent algorithm that approximately solves an ℓ_1-constrained logistic regression. This observation is actually the starting point of this project. For higher-order MRFs with non-binary alphabet, we conjecture that similar result can be proved for group-sparse regularized logistic regression.

Finally, it is interesting to see a large-scale empirical evaluation of different structural learning algorithms. The biggest challenge with large-scale simulation is that efficiently sampling from large graphical models is difficult. In our experiments, the samples are generated as follows: 1) We first *exactly* compute the probability distribution defined by a graphical model with n variables and alphabet size k; 2) We then sample from this probability distribution. Because the distribution contains k^n probabilities, the above sampling procedure is only possible for small n and k. When n is large (e.g.,

$n \sim 100$), exactly computing the probability distribution is impossible. In that case, Gibbs sampling needs to be used, but its mixing time can be very large (Bento and Montanari, 2009).

Chapter 4

Learning Compressed Sensing Measurement Matrix

4.1 Introduction

Assume we have some unknown data vector $x \in \mathbb{R}^d$. We can observe only a few ($m < d$) linear equations of its entries and we would like to design these projections by creating a measurement matrix $A \in \mathbb{R}^{m \times d}$ such that the projections $y = Ax$ allow exact (or near-exact) recovery of the original vector $x \in \mathbb{R}^d$.

If $d > m$, this is an ill-posed problem in general: we are observing m linear equations with d unknowns, so any vector x' on the subspace $Ax' = y$ satisfies our observed measurements. In this high-dimensional regime, the only hope is to make a structural assumption on x, so that unique reconstruction is possible. A natural approach is to assume that the data vector is sparse. The problem of designing measurement matrices and reconstruction algorithms that recover sparse vectors from linear observations is called Compressed Sensing

Chapter 4 is based on material from (Wu et al., 2019b). The author of this dissertation is the leading author of (Wu et al., 2019b), and contributed to the idea, the analysis, the implementation and experiments, and the writing of the paper. Source code can be found at https://github.com/wushanshan/L1AE.

(CS), Sparse Approximation or Sparse Recovery Theory (Donoho, 2006; Candès et al., 2006).

A natural way to recover is to search for the sparsest solution that satisfies the linear measurements:

$$\arg\min_{x' \in \mathbb{R}^d} \|x'\|_0 \quad \text{s.t. } Ax' = y. \tag{4.1}$$

Unfortunately this problem is NP-hard and for that reason the ℓ_0 norm is relaxed into an ℓ_1-norm minimization[1]

$$D(A, y) := \arg\min_{x' \in \mathbb{R}^d} \|x'\|_1 \quad \text{s.t. } Ax' = y. \tag{4.2}$$

Remarkably, if the measurement matrix A satisfies some properties (e.g. Restricted Isometry Property (RIP) (Candès, 2008) or the nullspace condition (NSP) (Rauhut, 2010)) it is possible to prove that the ℓ_1 minimization in (4.2) produces the same output as the intractable ℓ_0 minimization in (4.1). However, it is NP-hard to test whether a matrix satisfies RIP (Bandeira et al., 2013).

In this project, we are interested in vectors that are not only sparse but have *additional structure* in their support. This extra structure may not be known or a-priori specified. We propose a data-driven algorithm that *learns a good linear measurement matrix A from data samples*. Our linear measurements are subsequently decoded with the ℓ_1-minimization in (4.2) to estimate the unknown vector x.

[1] Other examples are greedy (Tropp and Gilbert, 2007), and iterative algorithms, e.g., CoSaMP (Needell and Tropp, 2009), IHT (Blumensath and Davies, 2009), and AMP (Donoho et al., 2009).

Many real-world sparse datasets have additional structures beyond simple sparsity. For example, in a demographic dataset with (one-hot encoded) categorical features, a person's income level may be related to their education. Similarly, in a text dataset with bag-of-words representation, it is much more likely for two related words (e.g., football and game) to appear in the same document than two unrelated words (e.g., football and biology). A third example is that in a genome dataset, certain groups of gene features may be correlated. In this project, the goal is to *learn* a measurement matrix A to leverage such additional structure.

Our method is an autoencoder for sparse data, with a linear encoder (the measurement matrix) and a complex non-linear decoder that solves an optimization problem. The latent code is the measurement $y \in \mathbb{R}^m$ which forms the bottleneck of the autoencoder that makes the representation interesting. A popular data-driven dimensionality reduction method is Principal Components Analysis (PCA) (see e.g., (Hotelling, 1933; Boutsidis et al., 2015; Wu et al., 2016; Li et al., 2017) and the references therein). PCA is also an autoencoder where both the encoder and decoder are linear and learned from samples. Given a data matrix $X \in \mathbb{R}^{n \times d}$ (each row is a sample), PCA projects each data point $x \in \mathbb{R}^d$ onto the subspace spanned by the top right-singular vectors of X. As is well-known, PCA provides the lowest mean-squared error when used with a linear decoder. However, when the data is sparse, non-linear recovery algorithms like (4.2) can yield significantly better recovery performance.

In this project, we focus on learning a linear encoder for sparse data.

Compared to non-linear embedding methods such as kernel PCA (Mika et al., 1999), a linear method enjoys two broad advantages: 1) it is easy to compute, as it only needs a matrix-vector multiplication; 2) it is easy to interpret, as every column of the encoding matrix can be viewed as a feature embedding. Interestingly, Arora et al. (2018) recently observe that the pre-trained word embeddings such as GloVe and word2vec (Mikolov et al., 2013; Pennington et al., 2014) form a good measurement matrix for text data. Those measurement matrices, when used with ℓ_1-minimization, need fewer measurements than the random matrices to achieve near-perfect recovery.

Given a sparse dataset that has additional (but unknown) structure, our goal is to learn a good measurement matrix A, when the recovery algorithm is the ℓ_1-minimization in (4.2). More formally, given n sparse samples $x_1, x_2, \cdots, x_n \in \mathbb{R}^d$, our problem of finding the best A can be formulated as

$$\min_{A \in \mathbb{R}^{m \times d}} f(A), \text{ where } f(A) := \sum_{i=1}^{n} \|x_i - D(A, Ax_i)\|_2^2. \tag{4.3}$$

Here $D(\cdot, \cdot)$ is the ℓ_1 decoder defined in (4.2). Unfortunately, there is no easy way to compute the gradient of $f(A)$ with respect to A, due to the optimization problem defining $D(\cdot, \cdot)$. Our main insight, which will be elaborated on in Section 4.3.1, is that **replacing the ℓ_1-minimization with a T-step projected subgradient update of it**, results in gradients being (approximately) computable. In particular, consider the approximate function $\tilde{f}(A) : \mathbb{R}^{m \times d} \mapsto \mathbb{R}$

defined as

$$\begin{aligned} \tilde{f}(A) := & \quad \sum_{i=1}^{n} \|x_i - \hat{x}_i\|_2^2, \quad \text{where} \\ \hat{x}_i = & \quad T\text{-step projected subgradient of} \\ & \quad D(A, Ax_i), \text{ for } i = 1, \cdots, n. \end{aligned} \tag{4.4}$$

As we will show, now it is *possible* to compute the gradients of $\tilde{f}(A)$ with respect to A. This is idea is sometimes called *unrolling* and has appeared in various other applications as we discuss in the related work section. To the best of our knowledge, we are the first to use unrolling to learn a measurement matrix for compressed sensing.

4.1.1 Our Contributions

- We design a novel autoencoder, called ℓ_1-AE , to learn an efficient and compressed representation for structured sparse vectors. Our autoencoder is easy to train and has only two tuning hyper-parameters associated with the network architecture: the encoding dimension m and the network depth T. The architecture of ℓ_1-AE is inspired by the projected subgradient method of solving the ℓ_1-minimization in (4.2). While the exact projected subgradient method requires computing the pseudoinverse, we circumvent this by observing that it is possible to replace the expensive pseudoinverse operation by a simple transpose (see Lemma 4.3.1).

- The most surprising result in this project is that we can learn a linear encoder using an unfolded T-step projected subgradient decoder and the learned measurement matrix *still* performs well for the original ℓ_1-

minimization decoder. We empirically compare 10 algorithms over 6 sparse datasets (3 synthetic and 3 real). As shown in Figure 4.2 and Figure 4.3, using the measurement matrix learned from our autoencoder, we can compress the sparse vectors (in the test set) to a lower dimension (by a factor of 1.1-3x) than random Gaussian matrices while still being able to *perfectly* recover the original sparse vectors (see also Table 4.3). This demonstrates the superior ability of our autoencoder in learning and adapting to the additional structures in the given data.

- Although our autoencoder is specifically designed for ℓ_1-minimization decoder, the learned measurement matrix also performs well (and can perform even better) with the model-based decoder (Baraniuk et al., 2010) (Figure 4.4). This further demonstrates the benefit of learning a measurement matrix from data. As a baseline algorithm, we slightly modify the original model-based CoSaMP algorithm by adding a positivity constraint without changing the theoretical guarantee (Appendix C.3), which could be of independent interest.

- Besides the application in compressed sensing, one interesting direction for furture research is to use the proposed autoencoder ℓ_1-AE in other supervised tasks. We illustrate a potential application of ℓ_1-AE in extreme multi-label learning. We show that ℓ_1-AE can be used to learn label embeddings for multi-label datasets. We compare the resulted method with one of the state-of-the-art embedding-based methods SLEEC (Bhatia

et al., 2015) over two benchmark datasets. Our method is able to achieve better or comparable precision scores than SLEEC (see Table 4.5).

4.1.2 Notation

We use the uppercase letters to denote matrices and the lowercase letters to denote vectors and scalars. Let $A^\dagger$ denote the Moore-Penrose inverse of matrix A, and A^T denote the transpose. The ℓ_p-norm ($p > 0$) of a vector $x \in \mathbb{R}^d$ is defined as $\|x\|_p = (\sum_{i=1}^d x_i^p)^{1/p}$, where x_i is the i-th coordinate. We use I to denote an identity matrix.

4.2 Related Work

We briefly review the relevant work, and highlight the differences compared to our work.

Model-based compressed sensing (CS). Model-based CS (Baraniuk et al., 2010; Hegde et al., 2014) extends the conventional compressed sensing theory by considering more realistic structured models than simple sparsity. It requires to know the sparsity structure a priori, which is not always possible in practice. Our approach, by contrast, does not require a priori knowledge about the sparsity structure.

Learning-based measurement design. Most theoretical results in CS are based on random measurement matrices. There are a few approaches proposed to learn a measurement matrix from training data. One approach is to learn a near-isometric embedding that preserves pairwise distance (Hegde

et al., 2015; Bah et al., 2013). This approach usually requires computing the pairwise difference vectors, and hence is computationally expensive if both the number of training samples and the dimension are large (which is the setting that we are interested in). Another approach restricts the form of the measurement matrix, e.g., matrices formed by a subset of rows of a given basis matrix. The learning problem then becomes selecting the best subset of rows (Baldassarre et al., 2016; Li and Cevher, 2016; Gözcü et al., 2018). In Figure 4.2 and Figure 4.3, we compare our method with the learning-based subsampling method proposed in (Baldassarre et al., 2016), and show that our method needs fewer measurements to recover the original sparse vector.

Adaptive CS. In adaptive CS (Seeger and Nickisch, 2011; Arias-Castro et al., 2013; Malloy and Nowak, 2014), the new measurement is designed based on the previous measurements in order to maximize the the gain of new information. This is in contrast to our setting, where we are given a set of training samples. Our goal is to learn a good measurement matrix to leverage additional structure in the given samples.

Dictionary learning. Dictionary learning (Aharon et al., 2006; Mairal et al., 2009) is the problem of learning an overcomplete set of basis vectors from data so that every datapoint (presumably dense) can be represented as a sparse linear combination of the basis vectors. By contrast, we focus on learning a good measurement matrix for structured sparse data.

Sparse coding. The goal of sparse coding (Olshausen and Field, 1996; Donoho and Elad, 2003) is to find the sparse representation of a dense vector,

given a fixed family of basis vectors (aka a dictionary). Training a deep neural network to compute the sparse codes becomes popular recently (Gregor and LeCun, 2010; Sprechmann et al., 2015; Wang et al., 2016). Several recent papers (see, e.g., (Kulkarni et al., 2016; Xin et al., 2016; Shi et al., 2017; Jin et al., 2017; Mardani et al., 2017; Mousavi et al., 2017, 2019, 2015; Mousavi and Baraniuk, 2017; He et al., 2017; Zhang and Ghanem, 2018; Lohit et al., 2018) and the references therein) proposes new convolutional neural network architectures for image reconstruction from low-dimensional measurements. Note that some architectures such as (Lohit et al., 2018; Shi et al., 2017; Mousavi et al., 2015, 2017, 2019) also have an image sensing component, and hence the overall architecture becomes an autoencoder. Sparse coding is different from our problem, because we focus on learning a good measurement/encoding matrix rather than learning a good recovery/decoding algorithm.

Autoencoders. An autoencoder is a popular neural network architecture for unsupervised learning. It has applications in dimensionality reduction (Hinton and Salakhutdinov, 2006), pre-training (Erhan et al., 2010), image compression and recovery (Lohit et al., 2018; Mousavi et al., 2015, 2017, 2019), denoising (Vincent et al., 2010), and generative modeling (Kingma and Welling, 2014). In this project we design a novel autoencoder ℓ_1-AE to learn a compressed sensing measurement matrix for the ℓ_1 decoder. We focus on high-dimensional sparse (non-image) data such as the categorical data and bag-of-words data (see Table 4.1).

Unrolling. The idea of unfolding an iterative algorithm (e.g., gradient descent of an optimization problem) into a neural network structure is a natural way of making the algorithm trainable (see, e.g., (Gregor and LeCun, 2010; Hershey et al., 2014; Sprechmann et al., 2015; Xin et al., 2016; Wang et al., 2016; Shi et al., 2017; Jin et al., 2017; Mardani et al., 2017; He et al., 2017; Zhang and Ghanem, 2018) and references therein). The main difference between the previous papers and our problem is that, the previous papers seek a trained neural network as a replacement of the original optimization-based algorithm, while here we design an autoencoder based on the unrolling idea, and after training, we show that the learned measurement matrix still performs well with the *original* ℓ_1-minimization decoder.

Extreme multi-label learning (XML). The goal of XML is to learn a classifier to identify (for each datapoint) a subset of relevant labels from a extreme large label set. Different approaches have been proposed for XML, e.g., embedding-based (Bhatia et al., 2015; Yu et al., 2014; Mineiro and Karampatziakis, 2015; Tagami, 2017), tree-based (Prabhu and Varma, 2014; Jain et al., 2016; Jasinska et al., 2016; Prabhu et al., 2018a), 1-vs-all (Prabhu et al., 2018b; Babbar and Schölkopf, 2017; Yen et al., 2016, 2017; Niculescu-Mizil and Abbasnejad, 2017; Hariharan et al., 2012), and deep learning (Jernite et al., 2017; Liu et al., 2017). Here we focus on the embedding-based approach. In Section 4.5 we show that the proposed autoencoder can be used to learn label embeddings for multi-label datasets and the resulted method achieves better or comparable precision scores as SLEEC (Bhatia et al., 2015) over two

benchmark datasets.

4.3 Algorithm

Our goal is to learn a measurement matrix A from the given sparse vectors. This is done via training a novel autoencoder, called ℓ_1-AE. In this section, we will describe the key idea behind the design of ℓ_1-AE. Note that we focus on the vectors that are sparse in the standard basis and also nonnegative[2]. This is a natural setting for many real-world datasets, e.g., categorical data and bag-of-words data.

4.3.1 Intuition

Our design is strongly motivated by the projected subgradient method used to solve an ℓ_1-minimization problem. Consider the following ℓ_1-minimization problem:

$$\min_{x' \in \mathbb{R}^d} \|x'\|_1 \quad \text{s.t. } Ax' = y, \tag{4.5}$$

where $A \in \mathbb{R}^{m \times d}$ and $y \in \mathbb{R}^m$. We assume that $m < d$ and that A has rank m. One approach[3] to solving (4.5) is the projected subgradient method. The update is given by

$$x^{(t+1)} = \Pi(x^{(t)} - \alpha_t g^{(t)}), \text{ where } g^{(t)} = \text{sign}(x^{(t)}) \tag{4.6}$$

[2]Extending our method to more general settings is left for future research.
[3]Another approach is via linear programming.

where Π denotes the (Euclidean) projection onto the convex set $\{x' : Ax' = y\}$, $g^{(t)}$ is the sign function, i.e., the subgradient of the objective function $\|\cdot\|_1$ at $x^{(t)}$, and α_t is the step size at the t-th iteration. Since A has full row rank, Π has a closed-form solution given by

$$\Pi(z) = \arg \min_h \|h - z\|_2^2 \quad \text{s.t. } Ah = y \tag{4.7}$$

$$= z + \arg \min_{h'} \|h'\|_2^2 \quad \text{s.t. } Ah' = y - Az \tag{4.8}$$

$$= z + A^\dagger(y - Az), \tag{4.9}$$

where $A^\dagger = A^T(AA^T)^{-1}$ is the Moore-Penrose inverse of matrix A. Substituting (4.9) into (4.6), and using the fact that $Ax^{(t)} = y$, we get the following update equation

$$x^{(t+1)} = x^{(t)} - \alpha_t(I - A^\dagger A)\text{sign}(x^{(t)}), \tag{4.10}$$

where I is the identity matrix. We use $x^{(1)} = A^\dagger y$ (which satisfies the constraint $Ax' = y$) as the starting point.

As mentioned in the Introduction, our main idea is to replace the solution of an ℓ_1 decoder given in (4.5) by a T-step projected subgradient update given in (4.10). One technical difficulty[4] in simulating (4.10) using neural networks is backpropagating through the pesudoinverse $A^\dagger$. Fortunately, Lemma 4.3.1 shows that it is possible to replace $A^\dagger$ by A^T.

Lemma 4.3.1. *For any vector $x \in \mathbb{R}^d$, and any matrix $A \in \mathbb{R}^{m\times d}$ $(m < d)$ with rank m, there exists an $\tilde{A} \in \mathbb{R}^{m\times d}$ with all singular values being ones, such*

[4]One approach is to replace $A^\dagger$ by a trainable matrix $B \in \mathbb{R}^{d\times m}$. This approach performs worse than ours (see Figure 4.5).

that the following two ℓ_1-minimization problems have the same solution:

$$P_1: \quad \min_{x' \in \mathbb{R}^d} \|x'\|_1 \quad \text{s.t. } Ax' = Ax. \tag{4.11}$$

$$P_2: \quad \min_{x' \in \mathbb{R}^d} \|x'\|_1 \quad \text{s.t. } \tilde{A}x' = \tilde{A}x. \tag{4.12}$$

Furthermore, the projected subgradient update of P_2 is

$$x^{(t+1)} = x^{(t)} - \alpha_t(I - \tilde{A}^T\tilde{A})\text{sign}(x^{(t)}), \quad x^{(1)} = \tilde{A}^T\tilde{A}x. \tag{4.13}$$

A natural choice for $\tilde{A}$ is $U(AA^T)^{-1/2}A$, where $U \in \mathbb{R}^{m\times m}$ can be any unitary matrix.

Lemma 1 essentially says that: 1) Instead of searching over *all* matrices (of size m-by-d with rank m), it is enough to search over a subset of matrices $\tilde{A}$, whose singular values are all ones. This is because A and $\tilde{A}$ has the same recovery performance for ℓ_1-minimization (this is true as long as $\tilde{A}$ and A have the same null space). 2) The key benefit of searching over matrices with singular values being all ones is that the corresponding projected subgradient update has a simpler form: the annoying pesudoinverse term $A^\dagger$ in (4.10) is replaced by a simple matrix transpose A^T in (4.13).

As we will show in the next section, our decoder is designed to simulate (4.13) instead of (4.10): the only difference is that the pesudoinverse term $A^\dagger$ is replaced by matrix transpose A^T. Ideally we should train our ℓ_1-AE by enforcing the constraint that the matrices have singular values being ones. In practice, we do not enforce that constraint during training. We empirically observe that the learned measurement matrix A is not far from the constraint set (see Appendix C.4.4).

4.3.2 Network Structure of ℓ_1-AE

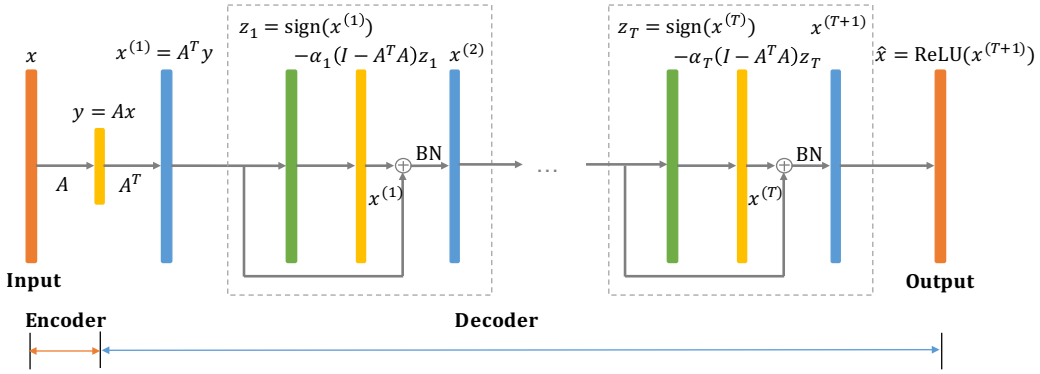


Figure 4.1: Network structure of the proposed autoencoder ℓ_1-AE.

As shown in Figure 4.1, ℓ_1-AE has a simple linear encoder and a non-linear decoder. When a data point $x \in \mathbb{R}^d$ comes, it is encoded as $y = Ax$, where $A \in \mathbb{R}^{m \times d}$ is the encoding matrix that will be learned from data. A decoder is then used to recover the original vector x from its embedding y.

The decoder network of ℓ_1-AE consists of T blocks connected in a feedforward manner: the output vector of the t-th block is the input vector to the $(t+1)$-th block. The network structure inside each block is identical. Let $x^{(1)} = A^T y$. For $t \in \{1, 2, ..., T\}$, if $x^{(t)} \in \mathbb{R}^d$ is the input to the t-th block, then its output vector $x^{(t+1)} \in \mathbb{R}^d$ is

$$x^{(t+1)} = x^{(t)} - \alpha_t (I - A^T A)\text{sign}(x^{(t)}), \tag{4.14}$$

where $\alpha_1, \alpha_2, \cdots, \alpha_T \in \mathbb{R}$ are scalar variables to be learned from data. We empirically observe that regularizing α_t to have the following form[5] $\alpha_t = \beta/t$

[5]This form is sometimes called *square summable but not summable* (Boyd, 2014).

for $t \in \{1, 2, ..., T\}$ improves test accuracy. Here, $\beta \in \mathbb{R}$ is the only scalar variable to be learned from data. We also add a standard batch normalization (BN) layer (Ioffe and Szegedy, 2015) inside each block, because empirically it improves the test accuracy (see Figure 4.5). After T blocks, we use rectified linear units (ReLU) in the last layer[6] to obtain the final output $\hat{x} \in \mathbb{R}^d$: $\hat{x} = \text{ReLU}(x^{(T+1)})$.

It is worth noting that the low-rank structure of the weight matrix $I - A^T A$ in (4.14) is essential for reducing the computational complexity. A fully-connected layer requires a weight matrix of size $d \times d$, which is intractable for large d.

Given n unlabeled training examples $\{x_i\}_{i=1}^n$, we will train an ℓ_1-AE to minimize the average squared ℓ_2 distance between $x \in \mathbb{R}^d$ and $\hat{x} \in \mathbb{R}^d$:

$$\min_{A \in \mathbb{R}^{m \times d},\, \beta \in \mathbb{R}} \quad \frac{1}{n} \sum_{i=1}^{n} \|x_i - \hat{x}_i\|_2^2. \tag{4.15}$$

4.4 Experiments

We implement ℓ_1-AE in Tensorflow 1.4. Our code can be found at https://github.com/wushanshan/L1AE. The goal of this section is to demonstrate that ℓ_1-AE is able to learn a good measurement matrix A for the structured sparse datasets, when we use ℓ_1-minimization for decoding[7].

[6] This makes sense as we focus on nonnegative sparse vectors in this project. Nonnegativity is a natural setting for many real-world sparse datasets, e.g., categorical data and text data.

[7] We use Gurobi (a commercial optimization solver) to solve it.

4.4.1 Datasets and Training

Dataset	d	NNZ	Train/Valid/Test	Description
Synthetic1	1000	10	6000/2000/2000	1-block sparse with block size 10
Synthetic2	1000	10	6000/2000/2000	2-block sparse with block size 5
Synthetic3	1000	10	6000/2000/2000	Power-law structured sparsity
Amazon	15626	9	19661/6554/6554	1-hot encoded categorical data
Wiki10-31K	30398	19	14146/3308/3308	Extreme multi-label data
RCV1	47236	76	13889/4630/4630	Text data with TF-IDF features

Table 4.1: Summary of the datasets. Each data point has dimension d. The third column "NNZ" is the average number of nonzeros in each data point. The fourth column gives the sizes of the training/validation/test sets. The validation set is used for parameter tuning and early stopping.

Synthetic datasets. As shown in Table 4.1, we generate three synthetic datasets: two satisfy the block sparsity model[8] (Baraniuk et al., 2010), and one follows the power-law structured sparsity (The i-th coordinate is nonzero with probability proportional to $1/i$). Each sample is generated as follows: 1) choose a random support set satisfying the sparsity model; 2) set the nonzeros to be uniform in $[0, 1]$.

Real datasets. Our first dataset is from Kaggle "Amazon Employee Access Challenge"[9]. Each training example contains 9 categorical features. We use one-hot encoding to transform each example. Our second dataset Wiki10-31K is a multi-label dataset downloaded from this repository (Bhatia

[8] A signal $x \in \mathbb{R}^d$ is called K-block sparse with block size J if it satisfies: 1) x can be reshaped into a matrix X of size $J \times N$, where $JN = d$; 2) every column of X acts as a group, i.e., the entire column is either zero or nonzero; 3) X has K nonzero columns, and hence x has sparsity KJ.

[9] https://www.kaggle.com/c/amazon-employee-access-challenge

et al., 2017). We only use the label vectors to train our autoencoder. Our third dataset is RCV1 (Lewis et al., 2004), a popular text dataset. We use scikit-learn to fetch the training set and randomly split it into train/validation/test sets.

Training. We use stochastic gradient descent to train the autoencoder. Before training, every sample is normalized to have unit ℓ_2 norm. The parameters are initialized as follows: $A \in \mathbb{R}^{m \times d}$ is a random Gaussian matrix with standard deviation $1/\sqrt{d}$; β is initialized as 1.0. Other hyper-parameters are given in Appendix C.2. A single NVIDIA Quadro P5000 GPU is used in the experiments. We set the decoder depth $T = 10$ for most datasets[10]. Training an ℓ_1-AE can be done in several minutes for small-scale datasets and in around an hour for large-scale datasets.

4.4.2 Baseline Algorithms

We compare 10 algorithms in terms of their recovery performance. The results are shown in Figure 4.2 and Figure 4.3. All the algorithms follow a two-step "encoding + decoding" process.

ℓ_1-AE + ℓ_1-min pos (our algorithm): After training an ℓ_1-AE , we use the encoder matrix A as the measurement matrix. To decode, we use Gurobi (a commercial optimization solver) to solve the following ℓ_1-minimization

[10] Although the subgradient method (Boyd, 2014) has a $O(1/\epsilon^2)$ convergence rate, in practice, we found that a small value of T (e.g., $T = 10$) seemed to be good enough (see Table 4.2).

problem with positivity constraint (denoted as "ℓ_1-min pos"):

$$\min_{x' \in \mathbb{R}^d} \|x'\|_1 \quad \text{s.t. } Ax' = y,\ x' \geq 0. \tag{4.16}$$

Since we focus on nonnegative sparse vectors, adding a positivity constraint improves the recovery performance[11] (see Appendix C.4.3).

Gauss + ℓ_1-min pos / model-based CoSaMP pos: A random Gaussian matrix $G \in \mathbb{R}^{m \times d}$ with i.i.d. $\mathcal{N}(0, 1/m)$ entries is used as the measurement matrix[12]. We experiment with two decoding methods: 1) Solve the optimization problem given in (4.16); 2) Use the model-based CoSaMP algorithm[13] (Algorithm 1 in (Baraniuk et al., 2010)) with an additional positivity constraint (see Appendix C.3).

PCA or PCA + ℓ_1-min pos: We perform truncated singular value decomposition (SVD) on the training set. Let $A \in \mathbb{R}^{m \times d}$ be the top-m singular vectors. For PCA, every vector $x \in \mathbb{R}^d$ in the test set is estimated as $A^T Ax$. We can also use "ℓ_1-min pos" (4.16) as the decoder.

Simple AE or Simple AE + ℓ_1-min pos: We train a simple autoencoder: for an input vector $x \in \mathbb{R}^d$, the output is $\text{ReLU}(B^T Ax) \in \mathbb{R}^d$, where both $B \in \mathbb{R}^{m \times d}$ and $A \in \mathbb{R}^{m \times d}$ are learned from data. We use the same loss

[11] The sufficient and necessary condition (Theorem 3.1 of (Khajehnejad et al., 2011)) for exact recovery using (4.16) is weaker than the the nullspace property (NSP) (Rauhut, 2010) for (4.5).

[12] Additional experiments with random partial Fourier matrices (Haviv and Regev, 2017) can be found in Appendix C.4.2.

[13] Model-based method needs the explicit sparsity model, and hence is not applicable for RCV1, Wiki10-31K, and Synthetic3.

function as our autoencoder. After training, we use the learned A matrix as the measurement matrix. Decoding is performed either by the learned decoder or solving "ℓ_1-min pos" (4.16).

LBCS + ℓ_1-min pos / model-based CoSaMP pos: We implement the learning-based compressive subsampling (LBCS) method in (Baldassarre et al., 2016). The idea is to select a subset (of size m) of coordinates (in the transformed space) that preserves the most energy. We use Gaussian matrix as the basis matrix and "ℓ_1-min pos" as the decoder[14]. Decoding with "model-based CoSaMP pos" is in Figure 4.4.

4-layer AE: We train a standard 4-layer autoencoder (we do not count the input layer), whose encoder network (and decoder network) consists of two feedforward layers with ReLU activation. The dimension of the 1st (and 3rd) layer is tuned based on the performance on the validation set.

4.4.3 Results and Analysis

The experimental results are shown in Figure 4.2 and Figure 4.3. Two performance metrics are compared[15]. The first one is the fraction of exactly recovered test samples. We say that a vector x is exactly recovered by an

[14]We tried four variations of LBCS: two different basis matrices (random Gaussian matrix and DCT matrix), two different decoders (ℓ_1-minimization and linear decoder). The combination of Gaussian and ℓ_1-minimization performs the best (see Appendix C.4.5).

[15]As an iterative algorithm, the CoSaMP algorithm's performance may depend on the halting criterion. In our experiments, we assume that it knows the original vector x (which is not possible in reality), and monitor the error $\|\hat{x} - x\|_2$ after each iteration. In other words, the performance of the CoSaMP algorithm shown in Figure 4.2 and Figure 4.3 is its best performance regardless of the halting criterion.

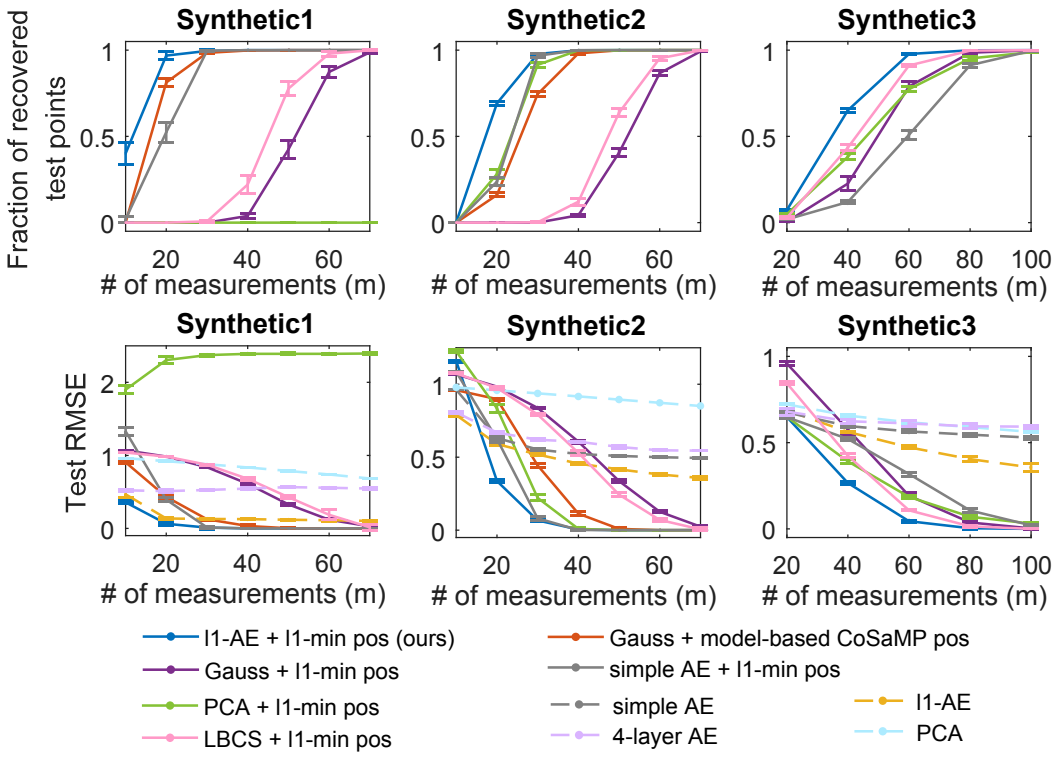


Figure 4.2: Best viewed in color. Recovery performance over three synthetic datasets: fraction of exactly recovered data points in the test set (1st row); reconstruction error (2nd row). We only plot the test RMSE for the following baselines: simple AE, 4-layer AE, ℓ_1-AE, and PCA. This is because they cannot produce a perfect reconstruction $\hat{x}$ that satisfies $\|x - \hat{x}\|_2 \leq 10^{-10}$ (see also Table 4.3). We plot the mean and standard deviation (indicated by the error bars) across 10 randomly generated datasets. Note that model-based CoSaMP decoder is not applicable for the Synthetic3 datasets. Our "ℓ_1-AE + ℓ_1-min pos" gives the best recovery performance across all the three synthetic datasets.

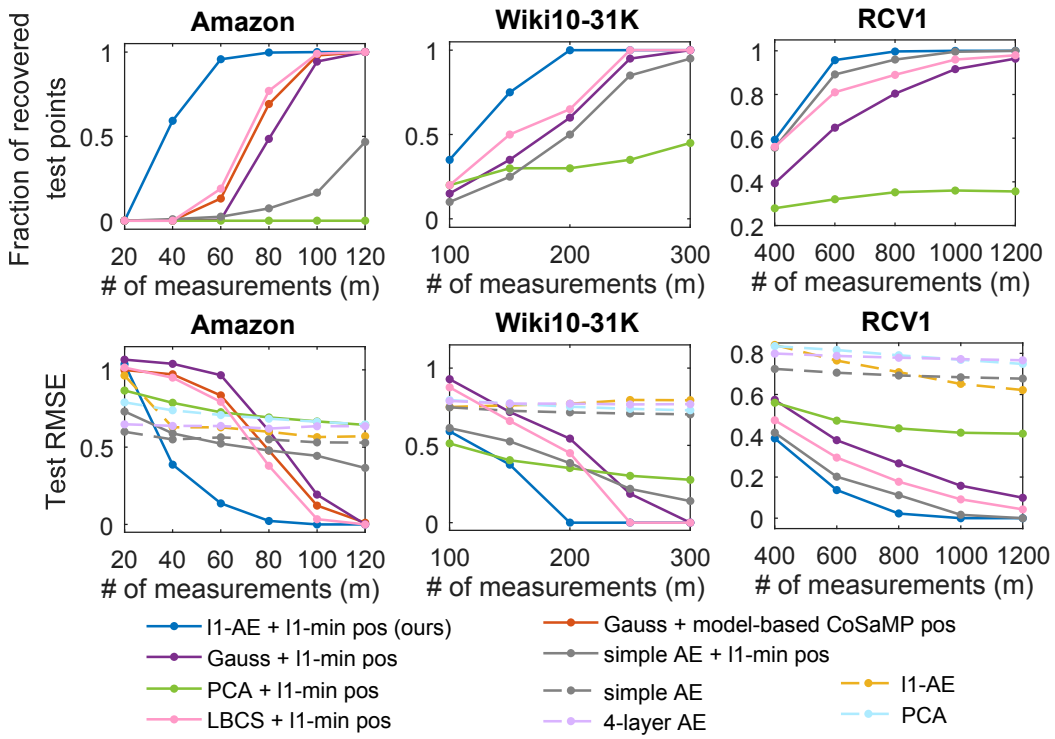


Figure 4.3: Best viewed in color. Recovery performance over three real datasets: fraction of exactly recovered data points in the test set (1st row); reconstruction error (2nd row). Similar to Figure 4.2, we only plot the test RMSE for the following baselines: simple AE, 4-layer AE, ℓ_1-AE, and PCA. This is because they cannot produce a perfect reconstruction $\hat{x}$ that satisfies $\|x - \hat{x}\|_2 \leq 10^{-10}$ (see also Table 4.3). Note that model-based CoSaMP decoder is not applicable for Wiki10-31K and RCV1 datasets. Our "ℓ_1-AE + ℓ_1-min pos" gives the best recovery performance across all the three real datasets.

algorithm if it produces an $\hat{x}$ that satisfies $\|x - \hat{x}\|_2 \leq 10^{-10}$. The second metric is the root mean-squared error (RMSE) over the test set[16]: $\sqrt{\sum_{i=1}^{n} \|x_i - \hat{x}_i\|_2^2 / n}$. We only evaluate the second metric (i.e., test RMSE) for the following baselines: simple AE, 4-layer AE, ℓ_1-AE, and PCA. This is because they cannot produce a perfect reconstruction $\hat{x}$ that satisfies $\|x - \hat{x}\|_2 \leq 10^{-10}$ (see also Table 4.3).

As shown in Figure 4.2 and Figure 4.3, our algorithm "ℓ_1-AE + ℓ_1-min pos" outperforms the other baselines over all datasets. By learning a data-dependent linear encoding matrix, our method requires fewer measurements to achieve perfect recovery.

Learned decoder versus ℓ_1-min decoder. We now compare two methods: ℓ_1-AE and ℓ_1-AE + ℓ_1-min pos. They have a common encoder but different decoders. As shown in Figure 4.2 and Figure 4.3, "ℓ_1-AE + ℓ_1-min pos" almost always gives a smaller RMSE. In fact, as shown in Table 4.3, "ℓ_1-min pos" is able to achieve reconstruction errors in the order of 1e-15, which is impossible for a neural network. The strength of optimization-based decoder over neural network-based decoder has been observed before, e.g., see Figure 1 in (Bora et al., 2017)[17]. Nevertheless, neural network-based decoder usually has an advantage that it can handle nonlinear encoders, for which the corresponding optimization problem may become non-convex and hard to solve exactly.

Model-based decoder versus learned encoder. It is interesting to

[16]Note that in Figure 4.2 and Figure 4.3, test RMSE has similar scale across all datasets, because the vectors are normalized to have unit ℓ_2 norm.

[17]As indicated by Figure 1 in (Bora et al., 2017), LASSO gives better reconstruction than GAN-based method when given enough Gaussian measurements.

Dataset	Synthetic1			Amazon		
# measurements	10	30	50	40	80	120
ℓ_1-AE	0.465	0.129	0.118	0.638	0.599	0.565
ℓ_1-AE+ℓ_1-min pos (ours)	**0.357**	**9.9e-3**	**1.4e-15**	**0.387**	**0.023**	**2.8e-15**

Table 4.2: Comparison of test RMSE for ℓ_1-AE and ℓ_1-AE + ℓ_1-min pos.

# measurements	10	20	30	40
$T = 10$	0.357	0.097	9.9e-3	1.3e-15
$T = 20$	0.293	0.063	1.2e-15	1.3e-15
$T = 30$	0.259	9.6e-16	1.1e-15	1.3e-15

Table 4.3: Test RMSE of our method "ℓ_1-AE + ℓ_1-min pos" on the Synthetic1 dataset: the error decreases as the decoder depth T increases.

see that our algorithm even outperforms model-based method (Baraniuk et al., 2010), even though the model-based decoder has more information about the given data than ℓ_1-minimization decoder. For the Amazon dataset, compared to "Gauss + model-based CoSaMP pos", our method reduces the number of measurements needed for exact recovery by about 2x. This is possible because the model-based decoder only knows that the input vector comes from one-hot encoding, which is a *coarse* model for the underlying sparsity model. By contrast, ℓ_1-AE learns a measurement matrix directly from the given training data.

A natural question to ask is whether the measurement matrix learned by ℓ_1-AE can improve the recovery performance of the model-based decoding algorithm. As shown in Figure 4.4, the recovery performance of "ℓ_1-AE + model-based CoSaMP pos" is better than "Gauss + model-based CoSaMP pos". This further demonstrates the benefit of learning a data-adaptive measurement

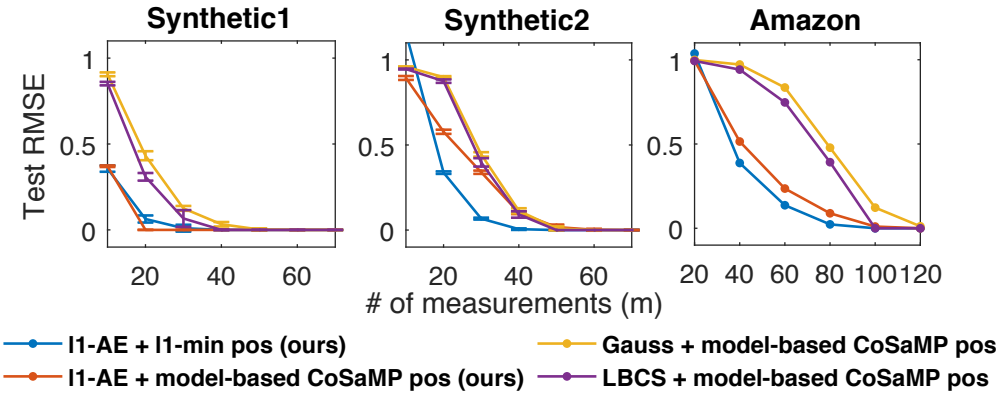


Figure 4.4: Best viewed in color. Although ℓ_1-AE is designed for the ℓ_1-min decoder, the matrix learned from ℓ_1-AE also improves the recovery performance when the decoder is model-based CoSaMP.

matrix. Note that "ℓ_1-AE + model-based CoSaMP pos" does not always outperform "ℓ_1-AE + ℓ_1-min pos", possibly because our autoencoder is designed for the ℓ_1-min decoder rather than the CoSaMP decoder.

Variations of ℓ_1-AE. We now examine how slightly varying the decoder structure would affect the performance. We make the following changes to the decoder structure: 1) remove the Batch Normalization layer; 2) remove the ReLU operation in the last layer; 3) change the nonlinearity from sign to tanh; 4) replace the A^T term in the decoder network by a matrix $B \in \mathbb{R}^{d \times m}$ that is learned from data; 5) use one-layer neural network as the decoder, i.e., set $T = 0$ in ℓ_1-AE . Each change is applied in isolation. As shown in Figure 4.5, our design gives the best recovery performance among all the variations for the Amazon dataset.

Decoder depth of ℓ_1-AE. The decoder depth T is a tuning parameter

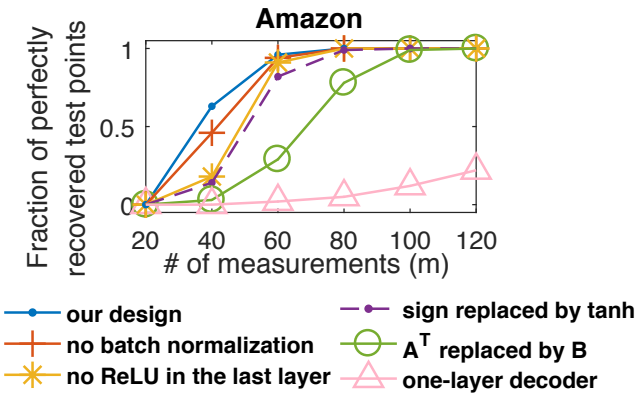


Figure 4.5: Recovery performance of "ℓ_1-AE + ℓ_1-min pos" on the Amazon test set when we slightly change the decoder structure. Each change is applied in isolation.

of ℓ_1-AE . Empirically, the performance of the learned matrix improves as T increases (see Table 4.2). On the other hand, the training time increases as T increases. The parameter T is tuned as follows: we start with $T = 5, 10, ..., T_{\max}$, and stop if the validation performance improvement is smaller than a threshold or if T equals $T_{\max}$. The hyper-parameters used for training ℓ_1-AE are given in Appendix C.2. We set $T = 5$ for Synthetic2 and Synthetic3, $T = 10$ for Synthetic1, RCV1, and Wiki10-31K, and $T = 60$ for Amazon dataset. The autoencoder is trained on a single GPU. Training an ℓ_1-AE takes a few minutes for small datasets and around an hour for large datasets.

4.5 Application in Extreme Multi-Label Learning

We have proposed a novel autoencoder ℓ_1-AE to learn a compressed sensing measurement matrix for high-dimensional sparse data. Besides the

application of ℓ_1-AE in compressed sensing, one interesting direction for future research is to use ℓ_1-AE in other machine learning tasks. Here we illustrate a potential application of ℓ_1-AE in extreme multi-label learning (XML). For each data point, the goal of XML is to predict a subset of relevant labels from a extremely large label set. As a result, the output label vector is high-dimensional, sparse, and nonnegative (with 1's denoting the relevant labels and 0's otherwise).

Many approaches have been proposed for XML(see (Bhatia et al., 2017) for the benchmark algorithms and datasets). Here we focus on the embedding-based approach[18], and one of the state-of-the-art embedding-based approaches[19] is SLEEC (Bhatia et al., 2015). Given n training samples (x_i, y_i), $i = 1, ..., n$, where $x_i \in \mathbb{R}^p$, $y_i \in \mathbb{R}^d$, we use $X \in \mathbb{R}^{p\times n}$ and $Y \in \mathbb{R}^{d\times n}$ to denote the stacked feature matrix and label matrix. SLEEC works in two steps. In Step 1, SLEEC reduces the dimension of the labels Y by learning an low-dimension embedding for each training sample. Let $Z \in \mathbb{R}^{m\times n}$ (where $m < d$) denote the learned embedding matrix. Note that only the Y matrix is used for learning Z in this step. In Step 2, SLEEC learns a linear mapping $V \in \mathbb{R}^{m\times p}$ such that $Z \approx VX$. To predict the label vector for a new sample $x \in \mathbb{R}^p$, SLEEC uses nearest neighbors method: first computes the embedding Vx, identifies a few nearest neighbors (from the training set) in the embedding space, and uses the sum of their label vectors as prediction.

[18] Other approaches include tree-based, 1-vs-all, etc.

[19] AnnexML (Tagami, 2017) is a graph-embedding approach for XML. Some of its techniques (such as better partition of the input data points) can be potentially used with our method.

The method that we propose follows SLEEC's two-step procedure. The main difference is that in Step 1, we train an autoencoder ℓ_1-AE to learn embeddings for the labels Y. Note that in XML, the label vectors Y are high-dimensional, sparse, and nonnegative. Let $A \in \mathbb{R}^{m\times d}$ be the learned measurement matrix for Y, then the embedding matrix is $Z = AY$. In Step 2, we use the same subroutine as SLEEC to learn a linear mapping from X to Z. To predict the label vector for a new sample, we compared three methods in our experiments: 1) use the nearest neighbor method (same as SLEEC); 2) use the decoder of the trained ℓ_1-AE (which maps from the embedding space to label space); 3) use an average of the label vectors obtained from 1) and 2). The three methods are denoted as "ℓ_1-AE 1/2/3" in Table 4.5.

In Table 4.5, we compare the precision score P@1 over two benchmark datasets shown in Table 4.4. The second row is the number of models combined in the ensemble. According to the XML benchmark website (Bhatia et al., 2017), SLEEC achieves a precision score 0.7926 for EURLex-4K and 0.8588 for Wiki10-31K, which are consistent with what we obtained by running their code (after combining 5 models in the ensemble). The embedding dimensions are $m = 100$ for EURLex-4K and $m = 75$ for Wiki10-31K. We set $T = 10$ for ℓ_1-AE. As shown in Table 4.5, our method has higher score than SLEEC for EURLex-4K. For Wiki10-31K, a single model of our method has higher score than SLEEC. When 3 or 5 models are ensembled, our method has comparable precision score to SLEEC. More results can be found in Appendix C.4.6.

Dataset	Feature dimension	Label dimension	Train/Test size	# labels per point
EURLex-4K	5000	3993	15539/3809	5.31
Wiki10-31K	101938	30938	14146/6616	18.64

Table 4.4: Summary of two XML benchmark datasets.

Dataset	EURLex-4K			Wiki10-31K		
# models in the ensemble	1	3	5	1	3	5
SLEEC	0.7600	0.7900	0.7944	0.8356	0.8603	0.8600
ℓ_1-AE 1	0.7655	0.7928	0.7931	0.8529	0.8564	0.8597
ℓ_1-AE 2	0.7949	0.8033	0.8070	0.8560	0.8579	0.8581
ℓ_1-AE 3	**0.8062**	**0.8151**	**0.8136**	**0.8617**	**0.8640**	**0.8630**

Table 4.5: Comparison of the P@1 scores.

4.6 Conclusion

Combining ideas from compressed sensing, convex optimization and deep learning, we proposed a novel unsupervised learning framework for high-dimensional sparse data. The proposed autoencoder ℓ_1-AE is able to learn an efficient measurement matrix by adapting to the sparsity structure of the given data. The learned measurement matrices can be subsequently used in other machine learning tasks such as extreme multi-label learning. We expect that the learned ℓ_1-AE can lead to useful representations in various supervised learning pipelines, for datasets that are well represented by large sparse vectors. Investigating the relation between the training data and the learned matrix (see Appendix C.4.1 for a toy example) is an interesting direction for future research.

Chapter 5

Single Pass PCA of Matrix Products

5.1 Introduction

Given two large matrices A and B we study the problem of finding a low rank approximation of their product $A^T B$, using only one pass over the matrix elements. This problem has many applications in machine learning and statistics. For example, if $A = B$, then this general problem reduces to Principal Component Analysis (PCA). Another example is a low rank approximation of a co-occurrence matrix from large logs, e.g., A may be a user-by-query matrix and B may be a user-by-ad matrix, so $A^T B$ computes the joint counts for each query-ad pair. The matrices A and B can also be two large bag-of-word matrices. For this case, each entry of $A^T B$ is the number of times a pair of words co-occurred together. As a fourth example, $A^T B$ can be a cross-covariance matrix between two sets of variables, e.g., A and B may be genotype and phenotype data collected on the same set of observations. A low rank approximation of the product matrix is useful for Canonical Correlation Analysis (CCA) (Chen et al.,

Chapter 5 is based on material from (Wu et al., 2016; Wu and Lindgren, 2016). The author of this dissertation is the leading author of (Wu et al., 2016; Wu and Lindgren, 2016), and contributed to the idea, the analysis, the implementation and experiments, and the writing of the paper. Source code can be found at https://github.com/wushanshan/MatrixProductPCA.

2012). For all these examples, $A^T B$ captures pairwise variable interactions and a low rank approximation is a way to efficiently represent the significant pairwise interactions in sub-quadratic space.

Let A and B be matrices of size $d \times n$ ($d \gg n$) assumed too large to fit in main memory. To obtain a rank-r approximation of $A^T B$, a naive way is to compute $A^T B$ first, and then perform truncated singular value decomposition (SVD) of $A^T B$. This algorithm needs $O(n^2 d)$ time and $O(n^2)$ memory to compute the product, followed by an SVD of the $n \times n$ matrix. An alternative option is to directly run power method on $A^T B$ without explicitly computing the product. Such an algorithm will need to access the data matrices A and B multiple times and the disk IO overhead for loading the matrices to memory multiple times will be the major performance bottleneck.

For this reason, a number of recent papers introduce randomized algorithms that require only a few passes over the data, approximately linear memory, and also provide spectral norm guarantees. The key step in these algorithms is to compute a smaller representation of data. This can be achieved by two different methods: (1) dimensionality reduction, i.e., matrix sketching (Sarlos, 2006; Clarkson and Woodruff, 2013; Magen and Zouzias, 2011; Cohen et al., 2016); (2) random sampling (Drineas et al., 2006a; Bhojanapalli et al., 2015). The recent result of Cohen et al. (2016) provides the strongest spectral norm guarantee of the former. They show that a sketch size of $O(\tilde{r}/\epsilon^2)$ suffices for the sketched matrices $\widetilde{A}^T \widetilde{B}$ to achieve a spectral error of ϵ, where $\tilde{r}$ is the maximum stable rank of A and B. Note that $\widetilde{A}^T \widetilde{B}$ is not the desired

rank-r approximation of A^TB. On the other hand, the recent sampling method proposed by Bhojanapalli et al. (2015) gives very good performance guarantees. Bhojanapalli et al. (2015) consider entrywise sampling based on column norms, followed by a matrix completion step to compute low rank approximations. There is also a lot of interesting work on streaming PCA (Balsubramani et al., 2013; Mitliagkas et al., 2013; Boutsidis et al., 2015; Shamir, 2015), but none can be directly applied to the general case when A is different from B (see an example in Figure 5.5(c) which shows that PCA on the individual matrices may not give information about their product).

Despite the significant volume of the prior work (see Section 5.1.2 for more related work), there is no method that computes a rank-r approximation of A^TB when the entries of A and B are streaming in a single pass[1]. Bhojanapalli et al. (2015) consider a two-pass algorithm which computes column norms in the first pass and uses them to sample in a second pass over the matrix elements. In this project, we combine ideas from the sketching and sampling literature to obtain the first algorithm that requires only a single pass over the data.

5.1.1 Our Contributions

- We propose a new algorithm SMP-PCA (which stands for Streaming Matrix Product PCA) that computes a rank-r approximation of A^TB

[1]One straightforward idea is to sketch each matrix individually and perform SVD on the product of the sketches. We compare against that scheme and show that we can perform arbitrarily better using our rescaled JL embedding.

using one pass over the data. Existing two-pass algorithms such as (Bhojanapalli et al., 2015) typically have longer runtime than our algorithm (see Figure 5.4(a)). We also compare our algorithm with the simple idea that first sketches A and B separately and then performs SVD on the product of their sketches. We show that our algorithm achieves better accuracy if the column vectors of A and B come from a cone (see Figures 5.3, 5.5(b), 5.4(b)).

- The central idea of our algorithm is a novel *rescaled JL embedding* that preserves the norm information during dimensionality reduction. This allows us to get a better estimator of the dot products between high dimensional vectors compared to previous sketching approaches. We explain the benefit compared to a naive JL embedding in Section 5.3. We believe it may be of more general interest beyond low rank matrix approximations.

- We implement SMP-PCA in Apache Spark and perform several distributed experiments on synthetic and real datasets. Our distributed implementation uses several design innovations described in Section 5.4. Our experiments show that we improve by approximately a factor of $2\times$ in running time compared to the previous state of the art and scale gracefully as the cluster size increases. Our source code can be found at `https://github.com/wushanshan/MatrixProductPCA`.

- In addition to better performance, our algorithm offers another advantage:

It is possible to compute low-rank approximations to $A^T B$ even when the entries of the two matrices arrive in some arbitrary order (as would be the case in streaming logs). We can therefore discover significant correlations even when the original datasets cannot be stored, for example due to storage or privacy limitations.

5.1.2 Related Work

Approximate matrix multiplication: In the seminal work, Drineas et al. (2006a) give a randomized algorithm which samples a few rows of A and B and computes the approximate product. The distribution depends on the row norms of the matrices and the algorithm achieves an additive error proportional to $||A||_F||B||_F$. Later, Sarlos (2006) proposes a sketching based algorithm, which computes sketched matrices and then outputs their product. The analysis for this algorithm is then improved by Clarkson and Woodruff (2009). All of these results compare the error in the Frobenius norm $||A^T B - \tilde{A}^T \tilde{B}||_F$.

For spectral norm bound of the form $||A^T B - C||_2 \leq \epsilon ||A||_2 ||B||_2$, the authors in (Sarlos, 2006; Clarkson and Woodruff, 2013) show that the sketch size needs to satisfy $O(r/\epsilon^2)$, where $r = \text{rank}(A) + \text{rank}(B)$. This dependence on rank is later improved to stable rank in (Magen and Zouzias, 2011), but at the cost of a weaker dependence on ϵ. Recently, Cohen et al. (2016) further improve the dependence on ϵ and give a bound of $O(\tilde{r}/\epsilon^2)$, where $\tilde{r}$ is the maximum stable rank. Our algorithm SMP-PCA also uses sketching for dimensionality

reduction, but in addition to sketching, we also have a novel rescaling procedure, which provide a better estimator than just using the sketched matrices, as illustrated in Figure 5.3. Furthermore, the previous sketching-based algorithms do not directly give a matrix with the desired rank. Directly taking SVD on the sketched matrices may give higher error rate than our algorithm, as shown in Figure 5.4(b).

Low rank approximation: Frieze et al. (2004) introduce the problem of computing low rank approximation of a given matrix using only a few passes over the data. They give an algorithm that samples a few rows and columns of the matrix and computes its SVD for low rank approximation. They show that this algorithm achieves an additive error in the Frobenius norm. Later, the authors in (Drineas et al., 2006b; Sarlos, 2006; Har-Peled, 2014; Deshpande and Vempala, 2006) develop algorithms using various sketching techniques such as Gaussian projection, random Hadamard transform, and volume sampling, which achieve a relative error in the Frobenius norm. Improved analysis of these algorithms can be found in (Woolfe et al., 2008; Nguyen et al., 2009; Halko et al., 2011; Boutsidis and Gittens, 2013), where they also provide error guarantees in the spectral norm. More recently, Clarkson and Woodruff (2013) present an algorithm based on subspace embedding which can compute the sketches in the input sparsity time.

Another class of methods uses entrywise sampling instead of sketching to compute low rank approximation. Achlioptas and McSherry (2001) consider an uniform entrywise sampling algorithm followed by SVD to compute low rank

approximation. This gives an additive approximation error. More recently, Bhojanapalli et al. (2015) consider biased entrywise sampling using leverage scores, followed by matrix completion to compute low rank approximation. While this algorithm achieves relative error approximation, it takes two passes over the data.

5.1.3 Notation

Throughout Chapter 5, we use $A(i,j)$ or A_{ij} to denote (i,j) entry for any matrix A. Let A_i and A^j be the i-th column vector and j-th row vector. We use $\|A\|_F$ for Frobenius norm, and $\|A\|_2$ for spectral (or ℓ_2) norm. The optimal rank-r approximation of matrix A is A_r, which can be found by SVD. Given a set $\Omega \subset [n_1] \times [n_2]$ and a matrix $A \in \mathbb{R}^{n_1 \times n_2}$, we define $P_\Omega(A) \in \mathbb{R}^{n_1 \times n_2}$ as the projection of A on Ω, i.e., $P_\Omega(A)(i,j) = A(i,j)$ if $(i,j) \in \Omega$ and 0 otherwise.

5.2 Algorithm

Consider the following problem: given two matrices $A \in \mathbb{R}^{d \times n_1}$ and $B \in \mathbb{R}^{d \times n_2}$ that are stored in disk, find a rank-r approximation of their product $A^T B$. In particular, we are interested in the setting where both A, B and $A^T B$ are too large to fit into memory. This is common for modern large scale machine learning applications. For this setting, we develop a single-pass algorithm SMP-PCA that computes the rank-r approximation without explicitly forming the entire matrix $A^T B$. Our algorithm SMC-PCA (Streaming Matrix Product PCA) is described in Section 5.2.1. SMC-PCA is based on a key idea called

Rescaled JL Embedding, which is analyzed in Section 5.3. The time complexity of SMC-PCA is given in Section 5.2.2.

5.2.1 SMC-PCA

Our algorithm SMP-PCA (Streaming Matrix Product PCA) takes four parameters as input: the desired rank r, number of samples m, sketch size k, and the number of iterations T. As illustrated in Figure 5.1, our algorithm has three main steps: 1) compute sketches and side information in one pass over A and B; 2) given partial information of A and B, estimate *important* entries of A^TB; 3) compute low rank approximation given estimates of a few entries of A^TB. Now we explain each step in detail.

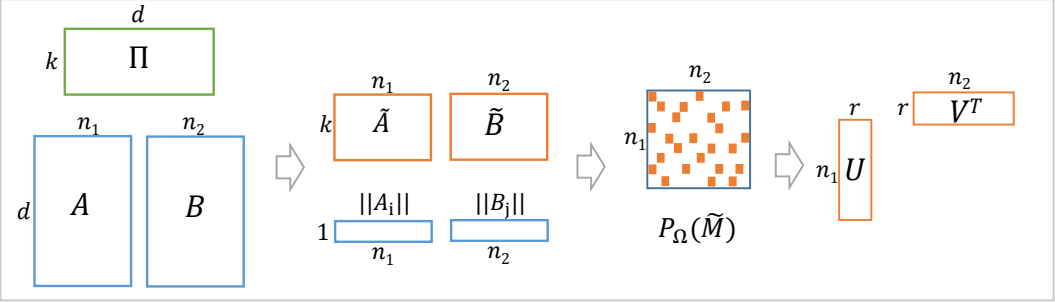


Figure 5.1: An overview of our algorithm. A single pass is performed over the data to produce the sketched matrices $\widetilde{A}$, $\widetilde{B}$ and the column norms $\|A\|_2$, $\|B_j\|_2$, for all $(i,j) \in [n_1] \times [n_2]$. We then compute the sampled matrix $P_\Omega(\widetilde{M})$ through a biased sampling process, where $P_\Omega(\widetilde{M}) = \widetilde{M}(i,j)$ if $(i,j) \in \Omega$ and zero otherwise. Here Ω represents the set of sampled entries. We define $\widetilde{M}$ as an estimator for A^TB, and compute its entry as $\widetilde{M}(i,j) = \|A_i\|_2\|B_j\|_2 \cdot \frac{\widetilde{A}_i^T\widetilde{B}_j}{\|\widetilde{A}_i\|_2\|\widetilde{B}_j\|_2}$. Performing matrix completion on $P_\Omega(\widetilde{M})$ gives the desired rank-r approximation.

Algorithm 7: SMP-PCA: Streaming Matrix Product PCA

Input: $A \in \mathbb{R}^{d\times n_1}$, $B \in \mathbb{R}^{d\times n_2}$, desired rank: r, sketch size: k, number of samples: m, number of iterations: T

Output: $\widehat{U} \in \mathbb{R}^{n_1\times r}$ and $\widehat{V} \in \mathbb{R}^{n_2\times r}$

1 Construct $\Pi \in \mathbb{R}^{k\times d}$, where $\Pi(i,j) \sim \mathcal{N}(0, 1/k)$.

2 Perform a single pass over A and B to obtain: $\widetilde{A} = \Pi A$, $\widetilde{B} = \Pi B$, and column norms $\|A_i\|_2$, $\|B_j\|_2$, $\forall (i,j) \in [n_1] \times [n_2]$.

3 Sample entry $(i,j) \in [n_1] \times [n_2]$ independently with probability $\hat{q}_{ij} = \min\{1, q_{ij}\}$, where q_{ij} is defined in (5.1); maintain a set $\Omega \subset [n_1] \times [n_2]$ which stores all the sampled pairs (i,j).

4 Compute $P_\Omega(\widetilde{M}) \in \mathbb{R}^{n_1\times n_2}$, where $\widetilde{M}(i,j)$ is given in (5.2).

5 $\widehat{U}, \widehat{V} \leftarrow$ WAltMin($P_\Omega(\widetilde{M})$, Ω, r, $\hat{q}$, T) (see Appendix D.1).

Step 1: Compute sketches and side information in one pass over A and B. In this step we compute sketches $\widetilde{A} := \Pi A$ and $\widetilde{B} := \Pi B$, where $\Pi \in \mathbb{R}^{k\times d}$ is a random matrix with entries being i.i.d. $\mathcal{N}(0, 1/k)$ random variables. It is known that Π satisfies an "oblivious Johnson-Lindenstrauss (JL) guarantee" (Johnson and Lindenstrauss, 1984; Sarlos, 2006; Woodruff et al., 2014), which helps preserving the top row spaces of A and B (Clarkson and Woodruff, 2013). Note that any sketching matrix Π that is an oblivious subspace embedding can be considered here, e.g., sparse JL transform and randomized Hadamard transform (see (Cohen et al., 2016) for more discussions).

Besides $\widetilde{A}$ and $\widetilde{B}$, we also compute the ℓ_2 norms for all column vectors, i.e., $\|A_i\|_2$ and $\|B_j\|_2$, for all $(i,j) \in [n_1] \times [n_2]$. We use this additional information to design better estimates of $A^T B$ in the next step, and also to determine *important* entries of $\widetilde{A}^T \widetilde{B}$ to sample. Note that this is the only step that needs one pass over data.

Step 2: Estimate important entries of A^TB by rescaled JL embedding. In this step we use the partial information obtained from the previous step to compute a few important entries of $\widetilde{A}^T\widetilde{B}$.

We sample the (i,j)-th entry of A^TB independently with probability $\hat{q}_{ij} = \min\{1, q_{ij}\}$, where

$$q_{ij} = m \cdot \left(\frac{\|A_i\|_2^2}{2n_2\|A\|_F^2} + \frac{\|B_j\|_2^2}{2n_1\|B\|_F^2}\right). \tag{5.1}$$

Let $\Omega \subset [n_1] \times [n_2]$ be the set of sampled entries (i,j). Since $\mathbb{E}(\sum_{i,j} q_{ij}) = m$, the expected number of sampled entries is roughly m. The special form of q_{ij} ensures that we can draw m samples in $O(n_1 + m\log(n_2))$ time; we show how to do this in Appendix D.2.

The sampling probability q_{ij} given in (5.1) captures important entries of A^TB by giving higher weight to heavy rows and columns. This biased sampling distribution is first proposed by Bhojanapalli et al. (2015). However, their algorithm needs a second pass to compute the sampled entries, while we propose a novel way of estimating the dot products, using information obtained in the first step. Specifically, let $\widetilde{M} \in \mathbb{R}^{n_1 \times n_2}$ be a matrix with its (i,j)-th entry being defined as

$$\widetilde{M}(i,j) = \|A_i\|_2\|B_j\|_2 \cdot \frac{\widetilde{A}_i^T\widetilde{B}_j}{\|\widetilde{A}_i\|_2\|\widetilde{B}_j\|_2}. \tag{5.2}$$

We now explain the intuition behind (5.2). To estimate the (i,j)-th entry of A^TB, the standard approach (Johnson and Lindenstrauss, 1984) is to compute the dot product between their low-dimensional embeddings: $\widetilde{A}_i^T\widetilde{B}_j$. We can

express $\widetilde{A}_i^T \widetilde{B}_j$ as $\|\widetilde{A}_i\|_2 \|\widetilde{B}_j\|_2 \cos\widetilde{\theta}_{ij}$, where $\widetilde{\theta}_{ij}$ is the angle between vectors $\widetilde{A}_i$ and $\widetilde{B}_j$. Since we already know the actual column norms, a better estimator would be $\|A_i\|_2 \|B_j\|_2 \cos\widetilde{\theta}_{ij}$, because it removes the uncertainty that comes from the distorted column norms[2]. Eq. (5.2) is equivalent to computing the dot product between two low-dimensional vectors:

$$\left\langle \frac{\|A_i\|_2}{\|\widetilde{A}_i\|_2} \widetilde{A}_i, \quad \frac{\|B_j\|_2}{\|\widetilde{B}_j\|_2} \widetilde{B}_j \right\rangle. \tag{5.3}$$

The low-dimensional embedding $\frac{\|A_i\|_2}{\|\widetilde{A}_i\|_2} \widetilde{A}_i$ is a rescaled version of the original linear embedding $\widetilde{A}_i$. We call it the **Rescaled JL Embedding**, and formally define it and analyze it in Section 5.3.

Note that the whole matrix $\widetilde{M}$ does not need to be computed and stored. All we need is a subset of its entries: $\widetilde{M}(i,j)$ for $(i,j) \in \Omega$. This matrix is denoted as $P_\Omega(\widetilde{M})$, where $P_\Omega(\widetilde{M})(i,j) = \widetilde{M}(i,j)$ if $(i,j) \in \Omega$ and 0 otherwise.

Step 3: Compute low rank approximation given estimates of few entries of $A^T B$. Finally we compute the low rank approximation of $A^T B$ from the samples using alternating least squares:

$$\min_{U,V \in \mathbb{R}^{n \times r}} \sum_{(i,j) \in \Omega} w_{ij} (e_i^T U V^T e_j - \widetilde{M}(i,j))^2, \tag{5.4}$$

where $w_{ij} = 1/\hat{q}_{ij}$ denotes the weights, and e_i, e_j are standard base vectors. This is a popular technique for low rank recovery and matrix completion

[2]We also tried using the cosine rule for computing the dot product, and another sketching method specifically designed for preserving angles (Boufounos, 2013), but empirically those methods perform worse than our current estimator.

(see (Bhojanapalli et al., 2015) and the references therein). After T iterations, we will get a rank-r approximation of $\widetilde{M}$ presented in the convenient factored form. This subroutine is quite standard, so we defer the details to Appendix D.1.

5.2.2 Time Complexity

We now analyze the computation complexity of SMP-PCA. In Step 1, we compute the sketched matrices of A and B, which requires $O(\text{nnz}(A)k + \text{nnz}(B)k)$ flops. Here $\text{nnz}(\cdot)$ denotes the number of non-zero entries. The main job of Step 2 is to sample a set Ω and calculate the corresponding inner products, which takes $O(m\log(n) + mk)$ flops. Here we define n as $\max\{n_1, n_2\}$ for simplicity. In Step 3, we run alternating least squares on the sampled matrix, which can be completed in $O((mr^2 + nr^3)T)$ flops. Since $m \geq nr$ (Bhojanapalli et al., 2015), the computation complexity of Step 3 is $O(mr^2T)$. Therefore, SMP-PCA has a total computation complexity $O(\text{nnz}(A)k + \text{nnz}(B)k + m\log(n) + mk + mr^2T)$.

5.3 Analysis of Rescaled JL Embedding

In Section 5.2.1, we present *Rescaled JL Embedding*, a new dimensionality reduction map that can potentially better preserve the pairwise geometry. In this section, we compare (both theoretically and empirically) the performance between the rescaled JL embedding and the standard JL embedding. We start

with a formal definition of the two embedding schemes[3].

Definition 5.3.1. Let $G \in \mathbb{R}^{m\times d}$ be a random matrix with every entry being drawn independently from a Gaussian distribution $\mathcal{N}(0, 1/m)$. For any $x \in \mathbb{R}^d$, we define the standard JL embedding $g : \mathbb{R}^d \to \mathbb{R}^m$ and the rescaled JL embedding $f : \mathbb{R}^d \to \mathbb{R}^m$ as

$$g(x) := Gx; \quad f(x) := \frac{Gx}{\|Gx\|_2}\|x\|_2. \tag{5.5}$$

It is easy to check that $\|f(x)\|_2 = \|x\|_2$, i.e., the length of the vector is preserved, which is done by a simple rescaling operation.

5.3.1 Theoretical Guarantees

We now compare the theoretical guarantees of the two embedding schemes defined in Definition 5.5.

Theorem 5.3.1 (see, e.g., Theorem 2.1 in Woodruff et al. (2014))**.** *Let $g : \mathbb{R}^d \to \mathbb{R}^m$ be the standard JL map defined in Definition 5.5. Given two arbitrary vectors $x, y \in \mathbb{R}^d$, for any $\epsilon, \delta \in (0, 1/2)$, if $m = O(\frac{1}{\epsilon^2})$, then with probability at least $3/4$*[4]*,*

$$|\langle g(x), g(y)\rangle - \langle x, y\rangle| \le \epsilon\|x\|_2\|y\|_2. \tag{5.6}$$

[3]We focus on random Gaussian matrices here, but the same idea works for other random matrices.

[4]We focus on constant success probability here as standard techniques can be used to boost the success probability to $1 - \delta$ with an extra multiplicative factor $\ln(1/\delta)$ in the embedding dimension m.

Theorem 5.3.2. *Let $f : \mathbb{R}^d \to \mathbb{R}^m$ be the rescaled JL map defined in Definition 5.5. Given two arbitrary vectors $x, y \in \mathbb{R}^d$, for any $\epsilon, \delta \in (0, 1/2)$, if $m = O(\frac{1}{\epsilon^2})$, then with probability at least $3/4$,*

$$| \langle f(x), f(y) \rangle - \langle x, y \rangle | \leq p(\theta_{x,y}, m) \|x\|_2 \|y\|_2, \tag{5.7}$$

where $\theta_{x,y}$ is the angle between x and y, and $p(\cdot, \cdot)$ is defined in Definition 5.3.2.

Definition 5.3.2. Define $p(\theta, m)$ as a function of an angle $\theta \in [0, \pi]$ and a positive integer m:

$$p(\theta, m) = 2\sqrt{\mathop{\mathbb{E}}_{g,u}[\gamma^2 - 2\cos(\theta)\gamma] + \cos^2(\theta)}, \tag{5.8}$$

where $\frac{u+1}{2} \sim \text{Beta}(\frac{m-1}{2}, \frac{m-1}{2})$ follows Beta-distribution, $g \sim \text{F}(m, m)$ is a random variable with F-distribution, and

$$\gamma = \frac{u\sin(\theta) + \sqrt{g}\cos(\theta)}{\sqrt{g\cos^2(\theta) + \sin^2(\theta) + 2u\sin(\theta)\cos(\theta)\sqrt{g}}}. \tag{5.9}$$

Comparing Theorem 5.3.1 and Theorem 5.3.2, we see that both embedding schemes preserve the dot product information up to an additive error that depends on $\|x\|_2\|y\|_2$. The only difference is that for standard JL embedding, the error term (shown in Theorem 5.3.1) only depends on the embedding dimension, while for rescaled JL embedding, the error term $p(\theta, m)$ (shown in Theorem 5.3.2) is a function of the angle and the embedding dimension.

We are now interested in how $p(\theta, m)$ changes with θ for a fixed m. If $m = \lceil 4/\epsilon^2 \rceil$, then it is easy to check that

$$p(0, m) = p(\pi, m) = 2\sqrt{\mathop{\mathbb{E}}_{g,u}[\cos^2(0) - 2\cos^2(0)] + \cos^2(0)} = 0;$$
$$p(\pi/2, m) = 2\sqrt{\mathop{\mathbb{E}}_{u} u^2} = 2\sqrt{1/m} \leq \epsilon.$$

We plot $p(\theta, 400)$ as a function of θ in Figure 5.2. This function has a bell-shaped curve: it reaches maximum at $\theta = \pi/2$ and minimum at $\theta = 0$ and π. Note that $p(0, m) = p(\pi, m) = 0$ is consistent with our intuition: when $\theta = 0$ (or π), the two vectors x, y (and also $f(x), f(y)$) have the same direction, so their dot product only depends on the norms. More formally,

$$\langle f(x), f(y) \rangle = \|f(x)\|_2 \|f(y)\|_2 = \|x\|_2 \|y\|_2 = \langle x, y \rangle .$$

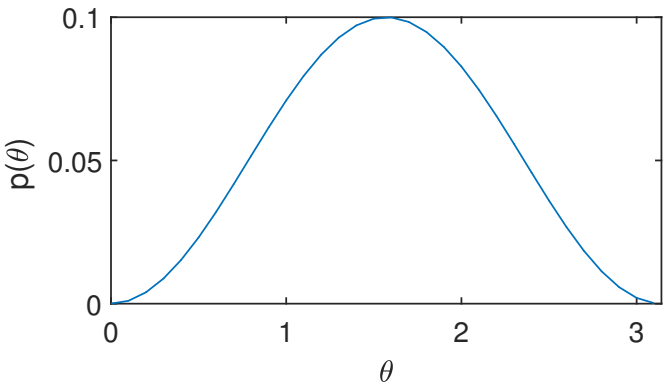


Figure 5.2: We plot $p(\theta, 400)$ as a function of θ. This function has a bell-shaped curve: it reaches maximum at $\theta = \pi/2$ and minimum at $\theta = 0$ and π.

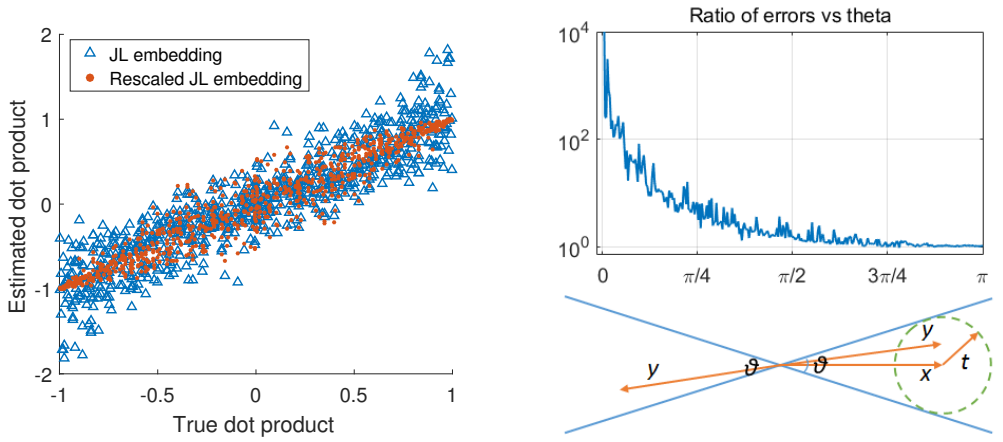
5.3.2 Empirical Comparisons

In Figure 5.3(a), we compare the performance of standard JL embedding and rescaled JL embedding in estimating the dot products. Rescaled JL embedding is formally defined in Definition 5.5. Every data point in Figure 5.3(a) is generated as follows: 1) randomly pick a $\theta \in [0, \pi]$; 2) generate two unit-norm vectors $x, y \in \mathbb{R}^{1000}$ with angle θ; 3) generate a random Gaussian matrix $G \in \mathbb{R}^{10\times 1000}$ with every entry being $\mathcal{N}(0, 0.1)$; 4) compute $\langle x, y\rangle$, $\langle Gx, Gy\rangle$, and $\langle Gx/\|Gx\|_2, Gy/\|Gy\|_2\rangle$.

As shown in Figure 5.3(a), rescaled JL embedding reduces the estimation uncertainty compared to the standard JL embedding. This phenomenon is more prominent when the true dot products are close to ± 1. In the extreme case when $\cos(\theta) = \pm 1$, rescaled JL embedding can perfectly recover the true dot product. This is consistent with our theoretical guarantee given in Theorem 5.3.2.

In Figure 5.3(b), we compare the performance between standard JL and rescaled JL embedding when estimating the product of two matrices. Given a $\theta \in [0, \pi]$, we generate two matrices $A \in \mathbb{R}^{1000\times 1000}$ and $B \in \mathbb{R}^{1000\times 1000}$ such that their columns are unit-norm vectors randomly drawn from a cone with angle θ. We illustrate how to do this in the lower part of Figure 5.3(b) (more details can be found in the caption of Figure 5.3(b)). We then generate a random Gaussian matrix $\Pi \in \mathbb{R}^{10\times 1000}$ with each entry sampled from $\mathcal{N}(0, 0.1)$. We now compute $\widetilde{A} = \Pi A$, $\widetilde{B} = \Pi B$, and $\widetilde{M}$ in (5.2). In the upper part of Figure 5.3(b), we plot the ratio of spectral norm errors $\|A^T B - \widetilde{A}^T \widetilde{B}\|_2 / \|A^T B - \widetilde{M}\|_2$ as a function of

θ. As shown in the figure, $\widetilde{M}$ always outperforms $\widetilde{A}^T\widetilde{B}$ and can be much better when θ approaches zero, which agrees with the trend shown in Figure 5.3(a).



(a) (b)

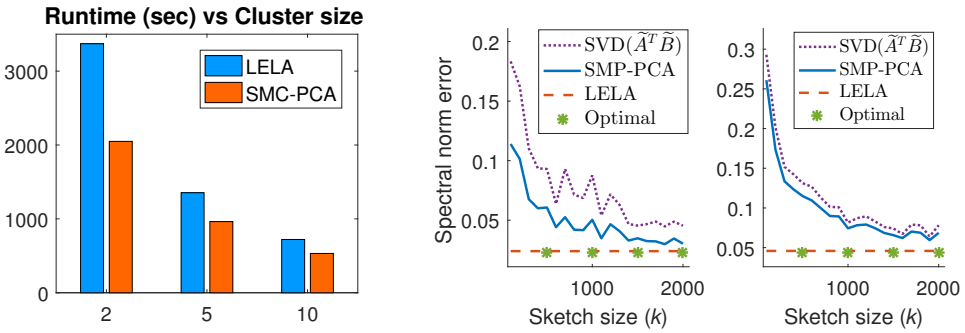
Figure 5.3: (a) Rescaled JL embedding (red dots, mean squared error 0.053) has smaller estimation error than standard JL embedding (blue triangles, mean squared error 0.129). (b) In the lower plot, we illustrate how to construct unit-norm vectors from a cone with angle θ. Let x be a fixed unit-norm vector, and let t be a random Gaussian vector with expected norm $\tan(\theta/2)$, we set y as either $x+t$ or $-(x+t)$ with probability half, and then normalize it. In the upper figure, we plot the ratio of spectral norm errors $\|A^TB - \widetilde{A}^T\widetilde{B}\|_2/\|A^TB - \widetilde{M}\|_2$, when the column vectors of A and B are unit vectors drawn from a cone with angle θ. Clearly, $\widetilde{M}$ has better accuracy than $\widetilde{A}^T\widetilde{B}$ for all possible values of θ, especially when θ is small.

5.4 Experiments

5.4.1 Spark Implementation

We implement our SMP-PCA in Apache Spark 1.6.2 (Zaharia et al., 2012). For the purpose of comparison, we also implement a two-pass algorithm

LELA (Bhojanapalli et al., 2015) in Spark[5]. The matrices A and B are stored as a resilient distributed dataset (RDD) in disk (by setting its `StorageLevel` as `DISK_ONLY`). We implement the two passes of LELA using the `treeAggregate` method. For SMP-PCA, we choose the subsampled randomized Hadamard transform (SRHT) (Tropp, 2011) as the sketching matrix [6]. The biased sampling procedure is performed using binary search (see Appendix D.2 for how to sample m elements in $O(m \log n)$ time). After obtaining the sampled matrix, we run `ALS` (alternating least squares) to get the desired low-rank matrices. Our code can be found at https://github.com/wushanshan/MatrixProductPCA.



(a) (b)

Figure 5.4: (a) Spark-1.6.2 running time on a 150GB dataset. All nodes are m.2xlarge EC2 instances. (b) Spectral norm error achieved by three algorithms over two datasets: SIFT10K (left) and NIPS-BW (right). We observe that SMP-PCA outperforms SVD($\widetilde{A}^T\widetilde{B}$) by a factor of 1.8 for SIFT10K and 1.1 for NIPS-BW. Besides, the error of SMP-PCA keeps decreasing as the sketch size k grows.

[5]To our best knowledge, this the first distributed implementation of LELA.

[6]Compared to Gaussian sketch, SRHT reduces the runtime from $O(ndk)$ to $O(nd \log d)$ and space cost from $O(dk)$ to $O(d)$, while maintains the same quality of the output.

5.4.2 Description of the Datasets

We test our algorithm on synthetic datasets and three real datasets: SIFT10K (Jegou et al., 2011), NIPS-BW (Lichman, 2013), and URL-reputation (Ma et al., 2009). For synthetic data, we generate matrices A and B as GD, where G has entries independently drawn from standard Gaussian distribution, and D is a diagonal matrix with $D_{ii} = 1/i$. SIFT10K is a dataset of 10,000 images. Each image is represented by 128 features. We set A as the image-by-feature matrix. The task here is to compute a low rank approximation of $A^T A$, which is a standard PCA task. The NIPS-BW dataset contains bag-of-words features extracted from 1,500 NIPS papers. We divide the papers into two subsets, and let A and B be the corresponding word-by-paper matrices, so $A^T B$ computes the counts of co-occurred words between two sets of papers. The original URL-reputation dataset has 2.4 million URLs. Each URL is represented by 3.2 million features, and is indicated as malicious or benign. This dataset has been used previously for CCA (Ma et al., 2015). Here we extract two subsets of features, and let A and B be the corresponding URL-by-feature matrices. The goal is to compute a low rank approximation of $A^T B$, the cross-covariance matrix between two subsets of features.

5.4.3 Empirical Evaluations

Sample complexity. In Figure 5.5(a) we present simulation results on a small synthetic data with $n = d = 5,000$ and $r = 5$. We observe that a phase transition occurs when the sample complexity $m = \Theta(nr \log n)$. This agrees

with the experimental results shown in the previous papers (Chen et al., 2015; Bhojanapalli et al., 2015). For the rest experiments presented in this section, unless otherwise specified, we set $r = 5$, $T = 10$, and sampling complexity m as $4nr \log n$.

Dataset	d	n	Algorithm	Sketch size k	Error
Synthetic	100,000	100,000	Optimal	-	0.0271
			LELA	-	0.0274
			SMP-PCA	2,000	0.0280
URL-malicious	792,145	10,000	Optimal	-	0.0163
			LELA	-	0.0182
			SMP-PCA	2,000	0.0188
URL-benign	1,603,985	10,000	Optimal	-	0.0103
			LELA	-	0.0105
			SMP-PCA	2,000	0.0117

Table 5.1: Comparison of spectral norm error over three datasets.

Comparison of SMP-PCA and LELA. We now compare the statistical performance of SMP-PCA and LELA (Bhojanapalli et al., 2015) on three real datasets and one synthetic dataset. As shown in Figure 5.4(b) and Table 5.1, LELA always achieves a smaller spectral norm error than SMP-PCA, which makes sense because LELA takes two passes and hence has more chances exploring the data. Besides, we observe that as the sketch size increases, the error of SMP-PCA keeps decreasing and gets closer to that of LELA.

In Figure 5.4(a) we compare the runtime of SMP-PCA and LELA

using a 150GB synthetic dataset on m3.2xlarge Amazon EC2 instances[7]. The matrices A and B have dimension $n = d = 100,000$. The sketch dimension is set as $k = 2,000$. We observe that the speedup achieved by SMP-PCA is more prominent for small clusters (e.g., 56 mins versus 34 mins on a cluster of size two). This is possibly due to the increasing spark overheads at larger clusters, see (Gittens et al., 2016) for more related discussions.

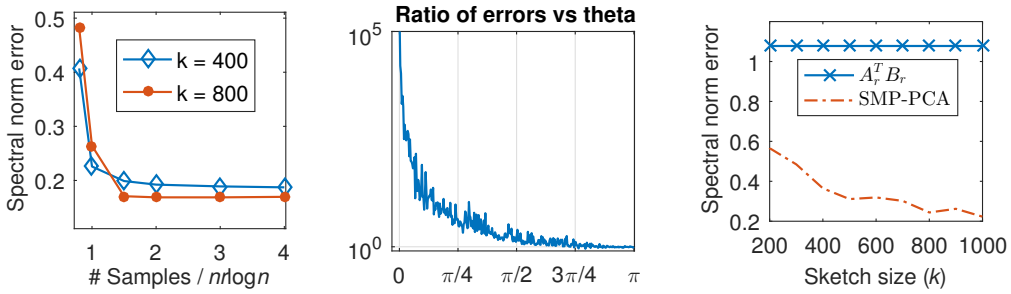
Comparison of SMP-PCA and SVD($\widetilde{A}^T\widetilde{B}$). In Figure 5.5(b) we repeat the experiment in Section 5.2 by generating column vectors of A and B from a cone with angle θ. Here SVD($\widetilde{A}^T\widetilde{B}$) refers to computing SVD on the sketched matrices[8]. We plot the ratio of the spectral norm error of SVD($\widetilde{A}^T\widetilde{B}$) over that of SMP-PCA, as a function of θ. Note that this is different from Figure 5.3(b), as now we take the effect of random sampling and SVD into account. However, the trend in both figures are the same: SMP-PCA always outperforms SVD($\widetilde{A}^T\widetilde{B}$) and can be arbitrarily better as θ goes to zero.

In Figure 5.4(b) we compare SMP-PCA and SVD($\widetilde{A}^T\widetilde{B}$) on two real datasets SIFK10K and NIPS-BW. The y-axis represents spectral norm error, defined as $\|A^TB - \widehat{A^TB}_r\|_2/\|A^TB\|_2$, where $\widehat{A^TB}_r$ is the rank-r approximation found by a specific algorithm. We observe that SMP-PCA outperforms SVD($\widetilde{A}^T\widetilde{B}$) by a factor of 1.8 for SIFT10K and 1.1 for NIPS-BW.

Now we explain why SMP-PCA produces a more accurate result than

[7]Each machine has 8 cores, 30GB memory, and 2×80GB SSD.

[8]This can be done by standard power iteration based method, without explicitly forming the product matrix $\widetilde{A}^T\widetilde{B}$, whose size is too big to fit into memory according to our assumption.



(a) (b) (c)

Figure 5.5: (a) A phase transition occurs when the sample complexity $m = \Theta(nr \log n)$. (b) This figure plots the ratio of spectral norm error of SVD($\widetilde{A}^T\widetilde{B}$) over that of SMP-PCA. The columns of A and B are unit vectors drawn from a cone with angle θ. We see that the ratio of errors scales to infinity as the cone angle shrinks. (c) If the top r left singular vectors of A are orthogonal to those of B, the product $A_r^T B_r$ is a very poor low rank approximation of $A^T B$.

SVD($\widetilde{A}^T\widetilde{B}$). The reasons are twofold. First, our rescaled JL embedding $\widetilde{M}$ is a better estimator for $A^T B$ than $\widetilde{A}^T\widetilde{B}$ (Figure 5.3). Second, the noise due to sampling is relatively small compared to the benefit obtained from $\widetilde{M}$, and hence the final result computed using $P_\Omega(\widetilde{M})$ still outperforms SVD($\widetilde{A}^T\widetilde{B}$).

Comparison of SMP-PCA and $A_r^T B_r$. Let A_r and B_r be the optimal rank-r approximation of A and B, we show that even if one could use existing methods (e.g., algorithms for streaming PCA) to estimate A_r and B_r, their product $A_r^T B_r$ can be a very poor low rank approximation of $A^T B$. This is demonstrated in Figure 5.5(c), where we intentionally make the top r left singular vectors of A orthogonal to those of B.

5.5 Conclusion

We developed a novel one-pass algorithm SMP-PCA that directly computes a low rank approximation of a matrix product, using ideas of matrix sketching and entrywise sampling. As a subroutine of our algorithm, we proposed rescaled JL embedding for estimating entries of $A^T B$, which has smaller error compared to that of standard JL embedding. This we believe can be extended to other applications. We provided a distributed implementation for SMP-PCA in Apache Spark. Compared to algorithms that require two or more passes over the data, our experimental results showed that SMP-PCA gives comparable error at a faster runtime.

Appendices

Appendix A

Appendix for Chapter 2

A.1 Proof of Lemma 2.4.1

We first restate the lemma and then give the proof.

Lemma. *For any $\epsilon \in (0,1)$ and $\delta \in (0,1)$, Algorithm 1 takes $n = \widetilde{O}\left(\frac{1}{\epsilon^2}\ln(\frac{d}{\delta})\right)$ samples from $\mathcal{D}(W^*, b^*)$ (for some non-negative b^*) and outputs $\widehat{b}(i)$ and $\widehat{\Sigma}(i,i)$ for all $i \in [d]$ that satisfy*

$$(1-\epsilon)\|W^*(i,:)\|_2^2 \le \widehat{\Sigma}(i,i) \le (1+\epsilon)\|W^*(i,:)\|_2^2, \quad |\widehat{b}(i) - b^*(i)| \le \epsilon\|W^*(i,:)\|_2$$

with probability at least $1-\delta$.

Proof. For a fixed $i \in [d]$, according to Theorem 1 of Daskalakis et al. (2018), given $\widetilde{O}(\ln(d/\delta)/\epsilon^2)$ truncated samples from $\mathcal{N}(b^*(i), \|W^*(i,:)\|_2^2, \mathbb{R}_{>0})$, the output of Algorithm 2 satisfies (2.9) with probability at least $1-\delta/d$. Since $b^*(i) \ge 0$, a sample $x \sim \mathcal{N}(b^*(i), \|W^*(i,:)\|_2^2)$ satisfies $x > 0$ with probability at least $1/2$. By Hoeffding's inequality, if we take $\widetilde{O}(\ln(d/\delta)/\epsilon^2) + O(\ln(d/\delta)) = \widetilde{O}(\ln(d/\delta)/\epsilon^2)$ samples from $\mathcal{D}(W^*, b^*)$, then we are able to obtain $\widetilde{O}(\ln(d/\delta)/\epsilon^2)$ truncated samples with probability at least $1-\delta/d$. Therefore, if we take $\widetilde{O}(\ln(d/\delta)/\epsilon^2)$ samples from $\mathcal{D}(W^*, b^*)$, for a fixed coordinate $i \in [d]$, the output of Algorithm 1 satisfies (2.9) with probability at least $1-2\delta/d$. Lemma 2.4.1

then follows by taking a union bound over all coordinates in $[d]$ and re-scaling δ to $\delta/2$. $\square$

A.2 Proof of Lemma 2.4.3

We first restate the lemma and then give the proof.

Lemma. *Let $x \sim \mathcal{D}(W^*, b^*)$, where b^* is non-negative. Suppose that $\widehat{b} \in \mathbb{R}^d$ is non-negative and satisfies $|\widehat{b}(i) - b^*(i)| \leq \epsilon \|W^*(i,:)\|_2$ for all $i \in [d]$ and some $\epsilon > 0$. Then for all $i \neq j \in [d]$,*

$$\left| \mathop{\mathbb{P}}_{x}[x(i) > \widehat{b}(i) \text{ and } x(j) > \widehat{b}(j)] - \mathop{\mathbb{P}}_{x}[x(i) > b^*(i) \text{ and } x(j) > b^*(j)] \right| \leq \epsilon.$$

Proof. We first notice that $\widehat{b}$ satisfies

$$\max(0, b^*(i) - \epsilon \|W^*(i,:)\|_2) \leq \widehat{b}(i) \leq b^*(i) + \epsilon \|W^*(i,:)\|_2, \text{ for all } i \in [d]. \quad \text{(A.1)}$$

To prove Lemma 2.4.3, we only need to prove that (2.12) holds when $\widehat{b}$ is substituted by its lower bound as well as the upper bound. We focus on substituting the lower bound here (as the upper bound follows a similar proof). We assume that $\|W^*(i,:)\|_2 \neq 0$ for all $i \in [d]$ (the proof extends

straightforwardly to the setting when this is not true).

$$
\begin{aligned}
&\mathop{\mathbb{P}}_{x}[x(i) > \max(0, b^*(i) - \epsilon\|W^*(i,:)\|_2) \text{ and } x(j) > \max(0, b^*(j) - \epsilon\|W^*(j,:)\|_2)] \\
&\quad - \mathop{\mathbb{P}}_{x}[x(i) > b^*(i) \text{ and } x(j) > b^*(j)] \\
&\overset{(a)}{\leq} \mathop{\mathbb{P}}_{z\sim\mathcal{N}(0,I_k)}[W^*(i,:)^T z > -\epsilon\|W^*(i,:)\|_2 \text{ and } W^*(j,:)^T z > -\epsilon\|W^*(j,:)\|_2] \\
&\qquad - \mathop{\mathbb{P}}_{z\sim\mathcal{N}(0,I_k)}[W^*(i,:)^T z > 0 \text{ and } W^*(j,:)^T z > 0] \\
&= \mathop{\mathbb{P}}_{z\sim\mathcal{N}(0,I_k)}[-\epsilon < \frac{W^*(i,:)^T}{\|W^*(i,:)\|_2} z \leq 0 \text{ and } -\epsilon < \frac{W^*(j,:)^T}{\|W^*(j,:)\|_2} z \leq 0] \\
&\qquad + \mathop{\mathbb{P}}_{z\sim\mathcal{N}(0,I_k)}[-\epsilon < \frac{W^*(i,:)^T}{\|W^*(i,:)\|_2} z \leq 0 \text{ and } \frac{W^*(j,:)^T}{\|W^*(j,:)\|_2} z > 0] \\
&\qquad + \mathop{\mathbb{P}}_{z\sim\mathcal{N}(0,I_k)}[\frac{W^*(i,:)^T}{\|W^*(i,:)\|_2} z > 0 \text{ and } -\epsilon < \frac{W^*(j,:)^T}{\|W^*(j,:)\|_2} z \leq 0] \\
&\leq \mathop{\mathbb{P}}_{z\sim\mathcal{N}(0,I_k)}[-\epsilon < \frac{W^*(i,:)^T}{\|W^*(i,:)\|_2} z \leq 0] + \mathop{\mathbb{P}}_{z\sim\mathcal{N}(0,I_k)}[-\epsilon < \frac{W^*(j,:)^T}{\|W^*(j,:)\|_2} z \leq 0] \\
&\overset{(b)}{\leq} \frac{2}{\sqrt{2\pi}}\epsilon \leq \epsilon.
\end{aligned}
$$

Here (a) is true because $x(i) = \text{ReLU}\left(W^*(i,:)^T z + b^*(i)\right)$ and b^* is non-negative. Inequality (b) is true because $\frac{W^*(i,:)^T}{\|W^*(i,:)\|_2} z$ is a one-dimensional Gaussian distribution $\mathcal{N}(0,1)$ and the probability density of $\mathcal{N}(0,1)$ have value no larger than $1/\sqrt{2\pi}$. □

A.3 Proof of Lemma 2.4.4

We first restate the lemma and then give the proof.

Lemma. *For a fixed pair of $i \neq j \in [d]$, for any $\epsilon, \delta \in (0,1)$, suppose $\widehat{b}$ satisfies the condition in Lemma 2.4.3, given $80\ln(2/\delta)/\epsilon^2$ samples, with probability at least $1-\delta$, $|\cos(\widehat{\theta}_{ij}) - \cos(\theta_{ij})| \leq \epsilon$.*

Proof. For a fixed pair $i \neq j \in [d]$, let $f(x) := \mathbb{1}(x(i) > \widehat{b}(i) \text{ and } x(j) > \widehat{b}(j))$. Since the indicator function is bounded, Hoeffding's inequality implies that if the number of samples $n \geq \ln(2/\delta)/(2\epsilon^2)$, then with probability at least $1 - \delta$,

$$\left| \frac{1}{n} \sum_{m=1}^{n} f(x_m) - \mathop{\mathbb{E}}_{x}[f(x)] \right| \leq \epsilon. \tag{A.2}$$

By Lemma 2.4.3, the above equation implies that

$$\left| \frac{1}{n} \sum_{m=1}^{n} f(x_m) - \mathop{\mathbb{E}}_{x}[\mathbb{1}(x(i) > b^*(i) \text{ and } x(j) > b^*(j))] \right| \leq 2\epsilon. \tag{A.3}$$

By Lemma 2.4.2, we have $|\widehat{\theta}_{ij} - \theta^*_{ij}| \leq 4\pi\epsilon$. Lemma 2.4.4 follows from the fact that $\cos(\cdot)$ has Lipschitz constant 1. Re-scaling ϵ gives the desired sample complexity. □

A.4 Proof of Theorem 2.4.5

We first restate the theorem, and then give the proof.

Theorem. *For any $\epsilon \in (0, 1)$ and $\delta \in (0, 1)$, Algorithm 1 takes $n = \widetilde{O}\left(\frac{1}{\epsilon^2} \ln(\frac{d}{\delta})\right)$ samples from $\mathcal{D}(W^*, b^*)$ (for some non-negative b^*) and outputs $\widehat{\Sigma} \in \mathbb{R}^{d \times d}$ and $\widehat{b} \in \mathbb{R}^d$ that satisfy*

$$\|\widehat{\Sigma} - W^* W^{*T}\|_F \leq \epsilon \|W^*\|_F^2, \quad \|\widehat{b} - b^*\|_2 \leq \epsilon \|W^*\|_F \tag{A.4}$$

with probability at least $1 - \delta$. Algorithm 1 runs in time $\widetilde{O}\left(\frac{d^2}{\epsilon^2} \ln(\frac{d}{\delta})\right)$ and space $\widetilde{O}\left(\frac{d}{\epsilon^2} \ln(\frac{d}{\delta}) + d^2\right)$.

Proof. By Lemma 2.4.1, the first for-loop of Algorithm 1 needs $\widetilde{O}\left(\frac{1}{\epsilon^2}\ln(\frac{d}{\delta})\right)$ samples and outputs $\widehat{\Sigma}(i,i)$ and $\widehat{b}(i)$ that satisfy for all $i \in [d]$,

$$(1-\epsilon)\|W^*(i,:)\|_2^2 \le \widehat{\Sigma}(i,i) \le (1+\epsilon)\|W^*(i,:)\|_2^2, \quad |\widehat{b}(i) - b^*(i)| \le \epsilon\|W^*(i,:)\|_2 \tag{A.5}$$

with probability at least $1-\delta$. Since $\epsilon \in (0,1)$, the above equation implies that

$$(1-\epsilon)\|W^*(i,:)\|_2 \le \sqrt{\widehat{\Sigma}(i,i)} \le (1+\epsilon)\|W^*(i,:)\|_2, \quad \|\widehat{b}-b^*\|_2 \le \epsilon\|W\|_F. \tag{A.6}$$

By Lemma 2.4.4, if $\widehat{b}$ satisfies (A.5), then the second for-loop of Algorithm 1 needs $O(\frac{1}{\epsilon^2}\ln(\frac{d^2}{\delta}))$ samples and outputs $\widehat{\theta}_{ij}$ that satisfies

$$|\cos(\widehat{\theta}_{ij}) - \cos(\theta^*_{ij})| \le \epsilon, \text{ for all } i \ne j \in [d] \tag{A.7}$$

with probability at least $1-\delta$. Combining (A.6) and (A.7) gives that for all $i, j \in [d]$,

$$|\widehat{\Sigma}(i,j) - \langle W^*(i,:), W^*(j,:)\rangle| \le 7\epsilon\|W^*(i,:)\|_2\|W^*(j,:)\|_2 \tag{A.8}$$

with probability at least $1-2\delta$. To see why (A.8) is true, suppose (with loss of generality) that $\cos(\theta_{ij}) \ge 0$, then $\widehat{\Sigma}(i,j)$ can be upper bounded by

$$\begin{aligned}
\widehat{\Sigma}(i,j) &= \sqrt{\widehat{\Sigma}(i,i)\widehat{\Sigma}(j,j)}\cos(\widehat{\theta}_{ij}) \\
&\le (1+\epsilon)^2\|W^*(i,:)\|_2\|W^*(j,:)\|_2(\cos(\theta^*_{ij}) + \epsilon) \\
&= (1+2\epsilon+\epsilon^2)\langle W^*(i,:), W^*(j,:)\rangle + \epsilon(1+\epsilon)^2\|W^*(i,:)\|_2\|W^*(j,:)\|_2 \\
&\le \langle W^*(i,:), W^*(j,:)\rangle + 3\epsilon\langle W^*(i,:), W^*(j,:)\rangle + 4\epsilon\|W^*(i,:)\|_2\|W^*(j,:)\|_2 \\
&\le \langle W^*(i,:), W^*(j,:)\rangle + 7\epsilon\|W^*(i,:)\|_2\|W^*(j,:)\|_2.
\end{aligned} \tag{A.9}$$

The lower bound can be derived in a similar way. Given (A.8), we can bound $\|\widehat{\Sigma} - W^* W^{*T}\|_F$ as

$$\begin{aligned}\|\widehat{\Sigma} - W^* W^{*T}\|_F^2 &= \sum_{i,j\in[d]} \Big(\widehat{\Sigma}(i,j) - \langle W^*(i,:), W^*(j,:)\rangle\Big)^2 \\ &\leq \sum_{i,j\in[d]} 49\epsilon^2 \|W^*(i,:)\|_2^2 \|W^*(j,:)\|_2^2 \\ &\leq 49\epsilon^2 \|W\|_F^2 \sum_{i\in[d]} \|W^*(i,:)\|_2^2 = 49\epsilon^2 \|W^*\|_F^4, \end{aligned} \tag{A.10}$$

which holds with probability at least $1 - 2\delta$. Re-scaling ϵ and δ gives the desired bound in Theorem 2.4.5. The final sample complexity is $\widetilde{O}\left(\frac{1}{\epsilon^2}\ln(\frac{d}{\delta})\right) + O(\frac{1}{\epsilon^2}\ln(\frac{d^2}{\delta})) = \widetilde{O}\left(\frac{1}{\epsilon^2}\ln(\frac{d}{\delta})\right)$.

We now analyze the time complexity. The first for-loop runs in time $O(dn)$, where n is the number of input samples. Note that in Step 3 of Algorithm 3, gradient estimation requires sampling from a truncated normal distribution. This can be done by sampling from a normal distribution until it falls into the truncation set. The probability of hitting a truncation set is lower bounded by a constant (Lemma 7 of (Daskalakis et al., 2018)). The second for-loop of Algorithm 1 runs in time $O(d^2 n)$. The space complexity is determined by the space required to store n samples and the matrix $\widehat{\Sigma} \in \mathbb{R}^{d\times d}$, which is $O(dn + d^2)$. □

A.5 Proof of Corollary 2.4.6

We first restate the corollary and then give the proof.

Corollary. *Suppose that $W^* \in \mathbb{R}^{d\times d}$ is full-rank. Let κ be the condition*

number of W^*W^{*T}. *For any* $\epsilon \in (0, 1/2]$ *and* $\delta \in (0, 1)$, *Algorithm 1 takes* $n = \widetilde{O}\left(\frac{\kappa^2 d^2}{\epsilon^2} \ln(\frac{d}{\delta})\right)$ *samples from* $\mathcal{D}(W^*, b^*)$ *(for some non-negative* b^**) and outputs a distribution* $\mathcal{D}(\widehat{\Sigma}^{1/2}, \widehat{b})$ *that satisfies*

$$\mathrm{TV}\left(\mathcal{D}(\widehat{\Sigma}^{1/2}, \widehat{b}),\ \mathcal{D}(W^*, b^*)\right) \leq \epsilon,$$

with probability at least $1 - \delta$. *Algorithm 1 runs in time* $\widetilde{O}\left(\frac{\kappa^2 d^4}{\epsilon^2} \ln(\frac{d}{\delta})\right)$ *and space* $\widetilde{O}\left(\frac{\kappa^2 d^3}{\epsilon^2} \ln(\frac{d}{\delta})\right)$.

Proof. Let $\Sigma = W^*W^{*T}$. We will prove that given $\widetilde{O}\left(\frac{\kappa^2 d^2}{\epsilon^2} \ln(\frac{d}{\delta})\right)$ samples, the output of Algorithm 1 satisfies

$$\|\Sigma^{-1/2}(\widehat{b} - b^*)\|_2 \leq \epsilon,\ \|\Sigma^{-1/2}\widehat{\Sigma}\Sigma^{-1/2} - I\|_F \leq \epsilon. \tag{A.11}$$

The above implies that the TV distance between $\mathcal{D}(\widehat{\Sigma}^{1/2}, \widehat{b})$ and $\mathcal{D}(W^*, b^*)$ is less than ϵ. To see why, note that

$$\begin{aligned}\mathrm{TV}\left(\mathcal{D}(\widehat{\Sigma}^{1/2}, \widehat{b}),\ \mathcal{D}(W^*, b^*)\right) &\leq \mathrm{TV}\left(\mathcal{N}(\widehat{b}, \widehat{\Sigma}), \mathcal{N}(b^*, \Sigma)\right) \\ &\leq \sqrt{\mathrm{KL}\left(\mathcal{N}(\widehat{b}, \widehat{\Sigma})||\mathcal{N}(b^*, \Sigma)\right)/2}.\end{aligned} \tag{A.12}$$

The first inequality follows from the data processing inequality for f-divergence given by Lemma A.6.3 in Appendix A.6 (see also (Ashtiani et al., 2018, Fact A.5)): $\mathrm{TV}(f(X), f(Y)) \leq \mathrm{TV}(X, Y)$ for any function f and random variables X, Y over the same space. The second inequality follows from the Pinsker's inequality (Tsybakov, 2009, Lemma 2.5). The KL divergence between two Gaussian distributions can be computed as $\mathrm{KL}\left(\mathcal{N}(\widehat{b}, \widehat{\Sigma})||\mathcal{N}(b^*, \Sigma)\right) =$

$$\frac{1}{2}\left(\mathrm{tr}(\Sigma^{-1}\widehat{\Sigma} - I) - \ln(\det(\Sigma^{-1}\widehat{\Sigma})) + \|\Sigma^{-1/2}(\widehat{b} - b^*)\|_2^2\right). \tag{A.13}$$

Let $\lambda_1, ..., \lambda_d$ be the eigenvalues of $\Sigma^{-1}\widehat{\Sigma}$. We have

$$\mathrm{tr}(\Sigma^{-1}\widehat{\Sigma} - I) - \ln(\det(\Sigma^{-1}\widehat{\Sigma})) = \sum_{i=1}^{d}(\lambda_i - 1) - \ln(\Pi_{i=1}^{d}\lambda_i) = \sum_{i=1}^{d}(\lambda_i - 1 - \ln(\lambda_i)). \tag{A.14}$$

Suppose that (A.11) holds with $\epsilon \leq 1/2$, since $\Sigma^{-1/2}\widehat{\Sigma}\Sigma^{-1/2}$ and $\Sigma^{-1}\widehat{\Sigma}$ have the same eigenvalues,

$$\epsilon^2 \geq \|\Sigma^{-1/2}\widehat{\Sigma}\Sigma^{-1/2} - I\|_F^2 = \sum_{i=1}^{d}(\lambda_i - 1)^2 \geq \sum_{i=1}^{d}(\lambda_i - 1 - \ln(\lambda)), \tag{A.15}$$

where the last inequality follows from the fact that $x - 1 - \ln(x) \leq (x-1)^2$ for $x \geq 1/2$. Since $\epsilon \leq 1/2$, we have $(\lambda_i - 1)^2 \leq 1/4$, which implies that $\lambda_i \in [1/2, 3/2]$. Substituting (A.15) into (A.14), and combining (A.13) and (A.12) give that the TV $\left(\mathcal{D}(\widehat{\Sigma}^{1/2}, \widehat{b}),\ \mathcal{D}(W^*, b^*)\right) \leq \epsilon$.

The only thing left is to prove that (A.11) holds. According to Theorem 2.4.5, given $\widetilde{O}\left(\frac{1}{\eta^2}\ln(\frac{d}{\delta})\right)$ samples, we have

$$\|\widehat{\Sigma} - \Sigma\|_F \leq \eta\|W^*\|_F^2, \quad \|\widehat{b} - b^*\|_2 \leq \eta\|W^*\|_F. \tag{A.16}$$

We can bound $\|\Sigma^{-1/2}(\widehat{b} - b^*)\|_2$ and $\|\Sigma^{-1/2}\widehat{\Sigma}\Sigma^{-1/2} - I\|_F$ as

$$\|\Sigma^{-1/2}(\widehat{b} - b^*)\|_2 \leq \|\Sigma^{-1/2}\|_2\|\widehat{b} - b\|_2 \leq \|\Sigma^{-1/2}\|_2\eta\|W^*\|_F \leq \eta\sqrt{\kappa d}.$$

$$\|\Sigma^{-1/2}\widehat{\Sigma}\Sigma^{-1/2} - I\|_F = \|\Sigma^{-1/2}(\widehat{\Sigma} - \Sigma)\Sigma^{-1/2}\|_F \leq \|\Sigma^{-1/2}\|_2^2\|\widehat{\Sigma} - \Sigma\|_F \leq \eta\kappa d.$$

Now setting $\eta = \epsilon/(\kappa d)$ gives (A.11). □

A.6 Proof of Theorem 2.5.1

To establish a lower bound for parameter estimation, the key step is to construct a local packing set such that their parameter distance is large but

their KL divergence is small (and hence it is hard to distinguish them without observing many samples). We remark that our way of constructing this local packing is similar to the one used in proving the minimax rate for Gaussian mean estimation (see, e.g., (Duchi, 2019)), despite the fact that our class of distributions is not Gaussian.

We will start by stating three results in information theory and statistics. Proofs of Lemma A.6.1, A.6.2, and A.6.3 can be found in, e.g., (Duchi, 2019).

Lemma A.6.1. *(Gilbert-Varshamov bound). There is a subset $\mathcal{V}$ of the d-dimensional hypercube $\{0,1\}^d$ of size $|\mathcal{V}| \geq \exp(d/8)$ such that the ℓ_1-distance*

$$\|v - v'\|_1 = \sum_{j=1}^{d} \mathbb{1}(v_j \neq v'_j) \geq d/4, \quad \text{for any } v, v' \in \mathcal{V}. \tag{A.17}$$

Lemma A.6.2. *(Fano's inequality). Let V be a random variable taking values uniformly in the finite set $\mathcal{V}$ with cardinality $|\mathcal{V}| \geq 2$. Conditioned on $V = v$, we draw a sample $X \sim P_v$. The KL divergence of the distributions $\{P_v\}_{v \in \mathcal{V}}$ satisfy*

$$\mathrm{KL}(P_v \parallel P_{v'}) \leq \beta, \quad \text{for any } v, v' \in \mathcal{V}. \tag{A.18}$$

For any Markov chain $V \to X \to \widehat{V}$,

$$\mathbb{P}[\widehat{V} \neq V] \geq 1 - \frac{\beta + \ln(2)}{\ln(|\mathcal{V}|)}. \tag{A.19}$$

Lemma A.6.3. *(Data processing inequality for f-divergence). Let f_1 and f_2 be the distributions of two random variables x_1 and x_2. Let g_1 and g_2 be the distributions of two random variables $T(x_1)$ and $T(x_2)$, where $T(\cdot)$ is any function. For any f-divergence $D_f(\cdot \parallel \cdot)$, we have*

$$D_f(f_1 \parallel f_2) \geq D_f(g_1 \parallel g_2). \tag{A.20}$$

We are now ready to prove Theorem 2.5.1, which is restated below.

Theorem. *Let $\sigma > 0$ be a fixed and known scalar. Let I_d be the identity matrix in $\mathbb{R}^d$. Let $S := \{\mathcal{D}(W, b) : W = \sigma I_d, b \in \mathbb{R}^d \text{ non-negative}\}$ be a class of distributions in $\mathbb{R}^d$. Any algorithm that learns S to satisfy $\|\widehat{b} - b^*\|_2 \leq \epsilon \|W^*\|_F$ with success probability at least 2/3 requires $\Omega(\frac{1}{\epsilon^2})$ samples.*

Proof. Let $\mathcal{V} \subset \{0,1\}^d$ be a finite set satisfying the property in Lemma A.6.1. Given an $\epsilon \in (0,1)$, we can construct a finite set of distributions $\{P_v\}_{v \in V}$ as follows:

$$P_v = \mathcal{D}(\sigma I_d, b_v), \text{ where } b_v = 6\epsilon\sigma v. \tag{A.21}$$

Clearly $\{P_v\}_{v \in V}$ belong to the class of the distributions that we are interested in. Furthermore, they satisfy two properties:

- Property 1: $\|b_v - b_{v'}\|_2 \geq 3\epsilon\sigma\sqrt{d}$ and $|\mathcal{V}| \geq \exp(d/8)$.
- Property 2: $\mathrm{KL}(P_v \parallel P_{v'}) \leq 18 d\epsilon^2$.

Assuming that the above two properties hold, we can use Fano's inequality (Lemma A.6.2) to obtain a sample complexity lower bound for learning $\{P_v\}_{v\in\mathcal{V}}$. Let V be a random variable taking values uniformly in $\mathcal{V}$. Conditioned on $V = v$, we draw n i.i.d. samples $X^n \sim P_v^n$, where P_v^n represents a product distribution of n P_v's. Given X^n, our goal is to output an index $\hat{v} \in \mathcal{V}$. By Lemma A.6.2, any estimator will suffer an estimation error larger than

$$\mathbb{P}[\widehat{V} \neq V] \geq 1 - \frac{18nd\epsilon^2 + \ln(2)}{d/8}, \tag{A.22}$$

which follows from the fact that $|\mathcal{V}| \geq \exp(d/8)$ (Property 1) and $\mathrm{KL}(P_v^n||P_{v'}^n) = n\mathrm{KL}(P_v||P_{v'}) \leq 18nd\epsilon^2$ (Property 2). Eq. (A.22) implies that any estimator that estimates the index correctly with probability at least $2/3$ must observe $\Omega(\frac{1}{\epsilon^2})$ samples. Furthermore, by Property 1, $\|b_v - b_{v'}\|_2 \geq 3\epsilon\sigma\sqrt{d}$, any algorithm that learns S to satisfy $\|\widehat{b} - b^*\|_2 \leq \epsilon\|W^*\|_F = \epsilon\sigma\sqrt{d}$ can be used to estimate $\mathcal{V}$ (we can just choose $\widehat{v} \in \mathcal{V}$ such that $b_{\widehat{v}}$ is closest to $\widehat{b}$). Therefore, any algorithm that learns S to satisfy $\|\widehat{b} - b^*\|_2 \leq \epsilon\|W^*\|_F$ with success probability at least $2/3$ requires $\Omega(\frac{1}{\epsilon^2})$ samples.

The only thing left is to show that Property 1 and 2 hold. Property 1 follows from Lemma A.6.1 and the way we construct P_v. Property 2 is true because of the following two facts.

- Fact 1: The KL-divergence between two Gaussian distributions can be computed as

$$\mathrm{KL}(\mathcal{N}(b_v, \sigma^2 I_d) \parallel \mathcal{N}(b_{v'}, \sigma^2 I_d)) = \frac{\|b_v - b_{v'}\|_2^2}{2\sigma^2} = 18d\epsilon^2. \tag{A.23}$$

- Fact 2: $\mathrm{KL}(P_v \parallel P_{v'}) \le \mathrm{KL}(\mathcal{N}(b_v, \sigma^2 I_d) \parallel \mathcal{N}(b_{v'}, \sigma^2 I_d))$, which follows from Lemma A.6.3 and the fact that KL-divergence is an instance of f-divergence.

□

A.7 Proof of Theorem 2.5.2

We first restate the theorem, and then give the proof.

Theorem. *Let $S := \{\mathcal{D}(W, 0) : W \in \mathbb{R}^{d\times d}$ full rank$\}$ be a set of distributions in $\mathbb{R}^d$. Any algorithm that learns S within total variation distance ϵ and success probability at least 2/3 requires $\Omega(\frac{d}{\epsilon^2})$ samples.*

Proof. Similar to the proof of Theorem 2.5.1, we construct a local packing of S for which their pairwise TV distance is large while their KL-divergence is small. Let $\mathcal{V} \subset \{0,1\}^d$ be a finite set satisfying the property in Lemma A.6.1. Given an $\epsilon \in (0,1)$, define $\lambda = C \cdot \epsilon/\sqrt{d}$, where C is a universal constant to be specified later, we can construct a finite set of distributions $\{P_v\}_{v\in V}$ as follows:

$$P_v = \mathcal{D}(W_v, 0), \text{ where } W_v = I_d + \lambda \cdot \mathrm{diag}(v). \tag{A.24}$$

Here $\mathrm{diag}(\cdot) : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ defines a diagonal matrix. This finite set of distributions satisfies two properties:

- Property 1: $\mathrm{TV}(P_v, P_{v'}) \ge 3\epsilon$ and $|\mathcal{V}| \ge \exp(d/8)$.

- Property 2: $\mathrm{KL}(P_v \parallel P_{v'}) = O(\epsilon^2)$.

Given the above two properties, we can use Fano's inequality (Lemma A.6.2) in a way similar to the proof of Theorem 2.5.1 to conclude that any estimator that identifies P_v from i.i.d. samples with success probability at least $2/3$ must require $\Omega(d/\epsilon^2)$ samples. Since $\mathrm{TV}(P_v, P_{v'}) \geq 3\epsilon$, any algorithm that learns S within TV distance ϵ can be used to estimate $\{P_v\}_{v\in\mathcal{V}}$ (we can just choose P_v that has the smallest TV distance to the output of the algorithm). This implies that any algorithm that learns S within TV distance ϵ with success probability at least $2/3$ requires $\Omega(d/\epsilon^2)$ samples.

The only thing left is to show that the two properties hold for our packing set $\{P_v\}_{v\in\mathcal{V}}$. To prove Property 2, note that

$$\mathrm{KL}(P_v \parallel P_{v'}) \overset{(a)}{\leq} \mathrm{KL}(\mathcal{N}(0, W_v W_v^T) \parallel \mathcal{N}(0, W_{v'} W_{v'}^T)) \overset{(b)}{=} O(\lambda^2 d) = O(\epsilon^2), \tag{A.25}$$

where (a) follows from Lemma A.6.3 and the fact that KL-divergence belongs to f-divergence; (b) follows from exactly computing the KL-divergence between the two Gaussian distributions. Before computing that, we need a few more notations. Specifically, let $S_v = \{i \in [d] : W_v(i,i) = 1 + \lambda\}$ be the set of coordinates that the corresponding diagonal entry of W_v is $1 + \lambda$. We use $S_v - S_{v'} = \{i \in S_v : i \neq S_{v'}\}$ to denote the difference between two sets. For simplicity, we write $\Sigma_v = W_v W_v^T$. Now we can compute the KL-divergence

between the two Gaussian distributions as

$$
\begin{aligned}
&2\mathrm{KL}(\mathcal{N}(0, W_v W_v^T) \parallel \mathcal{N}(0, W_{v'} W_{v'}^T)) \\
&= \mathrm{Tr}\left(\Sigma_{v'}^{-1}\Sigma_v - I_d\right) + \ln\left(\det(\Sigma_{v'})\right) - \ln\left(\det(\Sigma_v)\right) \\
&= |S_v - S_{v'}|\left((1+\lambda)^2 - 1\right) + |S_{v'} - S_v|\left(\frac{1}{(1+\lambda)^2} - 1\right) \\
&\quad + 2|S_{v'}|\ln(1+\lambda) - 2|S_v|\ln(1+\lambda) \\
&\overset{(a)}{\leq} |S_v|\left[(1+\lambda)^2 - 1 - 2\ln(1+\lambda)\right] + |S_{v'}|\left[2\ln(1+\lambda) + \frac{1}{(1+\lambda)^2} - 1\right] \\
&\overset{(b)}{\leq} |S_v|\left(2\lambda + \lambda^2 - \frac{2\lambda}{1+\lambda}\right) + |S_{v'}|\left(2\lambda - \frac{2\lambda+\lambda^2}{(1+\lambda)^2}\right) \\
&= |S_v|\frac{3\lambda^2+\lambda^3}{1+\lambda} + |S_{v'}|\frac{3\lambda^2+2\lambda^3}{(1+\lambda)^2} \\
&\overset{(c)}{=} O(d\lambda^2) = O(\epsilon^2),
\end{aligned}
$$

where (a) follows from $|S_v| \leq |S_v - S_{v'}|$, (b) follows from $\ln(1+x) \leq x$, and (c) is true because $|S_v| \leq d$ and $\lambda \in (0,1)$. Substituting $\lambda = O(\epsilon/\sqrt{d})$ gives the final result.

To prove Property 1, note that $|\mathcal{V}| \leq \exp(d/8)$ directly follows from Lemma A.6.1. The key challenge lies in proving a lower bound for $\mathrm{TV}(P_v, P_{v'})$. Note that the data-processing inequality (i.e., Lemma A.6.3) only implies that $\mathrm{TV}(P_v, P_{v'}) \leq \mathrm{TV}(\mathcal{N}(0, W_v W_v^T), \mathcal{N}(0, W_{v'} W_{v'}^T))$, so we cannot use the TV distance for Gaussian to obtain a lower bound on the TV distance for rectified Gaussian. Our proof strategy instead is to directly compute the TV distance for the specially-constructed $\{P_v\}_{v\in\mathcal{V}}$ (computing the exact TV distance is hard for general rectified Gaussian distributions). Specifically, let $\Sigma_v = W_v W_v^T$, our proof uses the following two facts:

- Fact 1: $\mathrm{TV}(\mathcal{N}(0, \Sigma_v), \mathcal{N}(0, \Sigma_{v'})) \geq 0.01 \|\Sigma_v^{-1}\Sigma_{v'} - I_d\|_F \geq C' \cdot \lambda\sqrt{d}$, where C' is a universal constant.

- Fact 2: Let Q_v be the probability density function of a multivariate normal distribution $\mathcal{N}(0, \Sigma_v)$. Let $\mathbb{R}^d_{>0} = \{x \in \mathbb{R}^d : x > 0 \text{ coordinate-wise}\}$ be the (open) positive orthant. Then

$$\|Q_v - Q_{v'}\|_1 = \int_{\mathbb{R}^d} |Q_v(x) - Q_{v'}(x)| \, \mathrm{d}x = 2^d \int_{\mathbb{R}^d_{>0}} |Q_v(x) - Q_{v'}(x)| \, \mathrm{d}x.$$

The first inequality in Fact 1 follows from (Devroye et al., 2018, Theorem 1.1). The second inequality follows from our definition of Σ_v. Specifically, the diagonal entry of Σ_v is either 1 or $1 + \lambda$. By Lemma A.6.1, we know that Σ_v and $\Sigma_{v'}$ have at least $d/4$ different diagonal entries. Since the total variation distance is symmetric, i.e., $\mathrm{TV}(\mathcal{N}(0, \Sigma_v), \mathcal{N}(0, \Sigma_{v'})) = \mathrm{TV}(\mathcal{N}(0, \Sigma_{v'}), \mathcal{N}(0, \Sigma_v))$, we can w.l.o.g assume that among the diagonal entries that Σ_v is different from $\Sigma_{v'}$, $\Sigma_{v'}$ has more entries with value $1 + \lambda$ than entries with value 1. This then implies that $\|\Sigma_v^{-1}\Sigma_{v'} - I_d\|_F = \Omega(\lambda\sqrt{d})$.

Fact 2 is true because $\mathcal{N}(0, \Sigma_v)$ has zero mean and diagonal covariance matrix, and hence the value of $Q_v(x)$ is invariant to the sign of x's coordinates.

Now we prove a lower bound on $\mathrm{TV}(P_v, P_{v'})$, assuming that are all the d diagonal entries of Σ_v and $\Sigma_{v'}$ are different. Let $\Omega \subseteq [d]$ be any subset of the d coordinates. For any Ω, let $x_\Omega \in \mathbb{R}^{|\Omega|}$ be the sub-vector of $x \in \mathbb{R}^d$ over the coordinates in Ω. Let $\Omega^c = [d] - \Omega$ be its complement. We can re-write $\mathrm{TV}(P_v, P_{v'})$ as a summation of integrals, where each integral is over the space

$A_\Omega = \{x \in \mathbb{R}^d : x_\Omega > 0,\ x_{\Omega^c} = 0\}$:

$$\mathrm{TV}(P_v, P_{v'}) = \|P_v - P_{v'}\|_1 = \sum_{\Omega} \int_{x \in A_\Omega} |P_v(x) - P_{v'}(x)| \, \mathrm{d}x. \tag{A.26}$$

We now give a lower bound for every integral. Let $\Sigma_{v,\Omega} \in \mathbb{R}^{|\Omega| \times |\Omega|}$ be the sub-matrix of Σ_v over the coordinates in Ω. Since Σ_v has zero mean and diagonal covariance matrix, for any $\Omega \subseteq [d]$ and any $x \in A_\Omega$, we have $P_v(x) = (\frac{1}{2})^{|\Omega^c|} P_{v,\Omega}(x_\Omega)$, where $P_{v,\Omega}$ is the probability density function of the normal distribution $\mathcal{N}(0, \Sigma_{v,\Omega})$. By Fact 1 and 2, we have

$$\begin{aligned}\int_{x \in A_\Omega} |P_v(x) - P_{v'}(x)| \, \mathrm{d}x &= (\frac{1}{2})^{|\Omega^c|} \cdot \frac{1}{2^{|\Omega|}} \mathrm{TV}(\mathcal{N}(0, \Sigma_{v,\Omega}), \mathcal{N}(0, \Sigma_{v',\Omega})) \\ &\geq \frac{C' \cdot \lambda \sqrt{|\Omega|}}{2^d}. \end{aligned} \tag{A.27}$$

Combining (A.26) and (A.27) gives

$$\begin{aligned}\mathrm{TV}(P_v, P_{v'}) &\geq \sum_{i=0}^{d} \binom{d}{i} \frac{C' \cdot \lambda \sqrt{i}}{2^d} \\ &\geq \sum_{i=\lfloor d/2 \rfloor}^{d} \binom{d}{i} \frac{C' \cdot \lambda \sqrt{\lfloor d/2 \rfloor}}{2^d} \\ &\overset{(a)}{\geq} \frac{C' \cdot \lambda \sqrt{\lfloor d/2 \rfloor}}{2^d} \frac{1}{2} \sum_{i=0}^{d} \binom{d}{i} \\ &\overset{(b)}{=} \frac{C' \cdot \lambda \sqrt{\lfloor d/2 \rfloor}}{2} \overset{(c)}{=} 3\epsilon, \end{aligned} \tag{A.28}$$

where (a) follows from the fact that $\binom{d}{i} = \binom{d}{d-i}$, (b) is true because $\sum_i \binom{d}{i} = 2^d$, and (c) holds if we choose $\lambda = C \cdot \epsilon / \sqrt{d}$ with a proper constant C.

So far we have proved that $\mathrm{TV}(P_v, P_{v'}) \geq 3\epsilon$ when all the d diagonal entries of Σ_v and $\Sigma_{v'}$ are different. The proof can be easily extended when only

a subset of their diagonal entries are different. Let $\Omega \subset [d]$ be the subset of d diagonal entries that Σ_v and $\Sigma_{v'}$ are different. By Lemma A.6.1, we know that $|\Omega| \geq d/4$. The definition of TV distance gives

$$\begin{aligned}
\mathrm{TV}(P_v, P_{v'}) &= \int_x |P_v(x) - P_{v'}(x)| \, \mathrm{d}x \\
&\overset{(a)}{=} \int_{x^{\Omega^c}} \int_{x^\Omega} |P_{v^\Omega}(x^\Omega) P_{v^{\Omega^c}}(x^{\Omega^c}) - P_{(v')^\Omega}(x^\Omega) P_{(v')^{\Omega^c}}(x^{\Omega^c})| \, \mathrm{d}x^\Omega \, \mathrm{d}x^{\Omega^c} \\
&\overset{(b)}{=} \int_{x^\Omega} |P_{v^\Omega}(x^\Omega) - P_{(v')^\Omega}(x^\Omega)| \, \mathrm{d}x^\Omega \int_{x^{\Omega^c}} P_{v^{\Omega^c}}(x^{\Omega^c}) \, \mathrm{d}x^{\Omega^c} \\
&= \int_{x^\Omega} |P_{v^\Omega}(x^\Omega) - P_{(v')^\Omega}(x^\Omega)| \, \mathrm{d}x^\Omega \\
&= \mathrm{TV}(P_{v^\Omega}, P_{(v')^\Omega}). \qquad \text{(A.29)}
\end{aligned}$$

Here equality (a) uses the fact that P_v and $P_{v'}$ have independent coordinates as Σ_v and $\Sigma_{v'}$ are diagonal matrices. Equality (b) follows from the definition of Ω: the diagonal entries in Ω^c are the same for Σ_v and $\Sigma_{v'}$, and hence, $P_{v^{\Omega^c}}(x^{\Omega^c}) = P_{(v')^{\Omega^c}}(x^{\Omega^c})$.

By (A.29), we have proved that the TV distance between P_v and $P_{v'}$ equals the TV distance between the two distributions over the coordinates in Ω. By definition, $\Sigma_{v^\Omega} \in \mathbb{R}^{|\Omega| \times |\Omega|}$ and $\Sigma_{(v')^\Omega} \in \mathbb{R}^{|\Omega| \times |\Omega|}$ have different diagonal entries, and $|\Omega| \geq d/4$, we can use the same proof in (A.28) to show that $\mathrm{TV}(P_{v^\Omega}, P_{(v')^\Omega}) \geq 3\epsilon$ for small enough λ. $\square$

Appendix B

Appendix for Chapter 3

B.1 Related Work on Learning Ising Models

For the special case of learning Ising models (i.e., binary variables), we compare the sample complexity among different graph recovery algorithms in Table B.1. Note that most of the algorithms listed in this table are only designed for learning Ising models instead of general pairwise graphical models. Hence, they are not presented in Table 3.1.

As mentioned in the Introduction, Ravikumar et al. (2010) consider ℓ_1-regularized logistic regression for learning Ising models in the high-dimensional setting. They require incoherence assumptions that ensure, via conditions on sub-matrices of the Fisher information matrix, that sparse predictors of each node are hard to confuse with a false set. Their analysis obtains significantly better sample complexity compared to what is possible when these extra assumptions are not imposed (see (Bento and Montanari, 2009)). Others have also considered ℓ_1-regularization (Lee et al., 2007; Yuan and Lin, 2007; Banerjee et al., 2008; Jalali et al., 2011; Yang et al., 2012; Aurell and Ekeberg, 2012) for structure learning of Markov random fields but they all require certain assumptions about the graphical model and hence their methods do not work

for general graphical models. The analysis of (Ravikumar et al., 2010) is of essentially the same convex program as this work (except that we have an additional thresholding procedure). The main difference is that they obtain a better sample guarantee but require significantly more restrictive assumptions.

In the general setting with no restrictions on the model, Santhanam and Wainwright (2012) provide an information-theoretic lower bound on the number of samples needed for graph recovery. This lower bound depends logarithmically on n, and exponentially on the width λ, and (somewhat inversely) on the minimum edge weight η. We will find these general broad trends, but with important differences, in the other algorithms as well.

Bresler (2015) provides a greedy algorithm and shows that it can learn with sample complexity that grows logarithmically in n, but *doubly* exponentially in the width λ and also exponentially in $1/\eta$. It is thus suboptimal with respect to its dependence on λ and η.

Vuffray et al. (2016) propose a new convex program (i.e. different from logistic regression), and for this they are able to show a single-exponential dependence on λ. There is also low-order polynomial dependence on λ and $1/\eta$. Note that given λ and η, the degree is bounded by $d \leq \lambda/\eta$ (the equality is achieved when every edge has the same weight and there is no external field). Therefore, their sample complexity can scale as worse as $1/\eta^5$. Later, the same authors (Lokhov et al., 2018) prove a similar result for the ℓ_1-regularized logistic regression using essentially the same proof technique as (Vuffray et al., 2016).

Rigollet and Hütter (2017) analyze the ℓ_1-constrained logistic regression for learning Ising models. Their sample complexity[1] has a better dependence on $1/\eta$ ($1/\eta^4$ vs $1/\eta^5$) than (Lokhov et al., 2018). However, naïvely extending their analysis to the $\ell_{2,1}$-constrained logistic regression will give a sample complexity exponential in the alphabet size[2].

In Chapter 3, we analyze the $\ell_{2,1}$-constrained logistic regression for learning discrete pairwise graphical models with general alphabet. Our proof uses a sharp generalization bound for constrained logistic regression, which is different from (Lokhov et al., 2018; Rigollet and Hütter, 2017). For Ising models (shown in Table B.1), our sample complexity matches that of (Klivans and Meka, 2017). For non-binary pairwise graphical models (shown in Table 3.1), our sample complexity improves the state-of-the-art result.

[1] Lemma 5.21 in (Rigollet and Hütter, 2017) has a typo: The upper bound should depend on $\exp(2\lambda)$. Accordingly, Theorem 5.23 should depend on $\exp(4\lambda)$ rather than $\exp(3\lambda)$.

[2] This is because the Hessian of the population loss has a lower bound that depends on $\exp(-2\lambda\sqrt{k})$ for $\|w\|_{2,1} \leq \lambda\sqrt{k}$ and $\|x\|_{2,\infty} \leq 1$.

Paper	Assumptions	Sample complexity (N)
Information-theoretic lower bound (Santhanam and Wainwright, 2012)	1. Model width $\leq \lambda$, and $\lambda \geq 1$ 2. Degree $\leq d$ 3. Minimum edge weight $\geq \eta > 0$ 4. External field $= 0$	$\max\{\frac{\ln(n)}{2\eta \tanh(\eta)}$, $\frac{d}{8}\ln(\frac{n}{8d})$, $\frac{\exp(\lambda)\ln(nd/4-1)}{4\eta d \exp(\eta)}\}$
ℓ_1-regularized logistic regression (Ravikumar et al., 2010)	Q^* is the Fisher information matrix, S is set of neighbors given a variable. 1. Dependency: $\exists\ C_{\min} > 0$ such that eigenvalues of $Q^*_{SS} \geq C_{\min}$ 2. Incoherence: $\exists\ \alpha \in (0, 1]$ such that $\|Q^*_{S^cS}(Q^*_{SS})^{-1}\|_\infty \leq 1 - \alpha$ 3. Regularization parameter: $\lambda_N \geq \frac{16(2-\alpha)}{\alpha}\sqrt{\frac{\ln(n)}{N}}$ 4. Min edge weight $\geq 10\sqrt{d}\lambda_N / C_{\min}$ 5. External field $= 0$ 6. Success prob. $\geq 1 - 2e^{-O(\lambda_N^2 N)}$	$O(d^3 \ln(n))$
Greedy algorithm (Bresler, 2015)	1. Model width $\leq \lambda$ 2. Degree $\leq d$ 3. Minimum edge weight $\geq \eta > 0$ 4. Probability of success $\geq 1 - \rho$	$O(\exp(\frac{\exp(O(d\lambda))}{\eta^{O(1)}})$ $\cdot \ln(\frac{n}{\rho}))$
Interaction Screening (Vuffray et al., 2016)	1. Model width $\leq \lambda$ 2. Degree $\leq d$ 3. Minimum edge weight $\geq \eta > 0$ 4. Regularization $= 4\sqrt{\frac{\ln(3n^2/\rho)}{N}}$ 5. Probability of success $\geq 1 - \rho$	$O(\max\{d, \frac{1}{\eta^2}\}$ $d^3 \exp(6\lambda) \ln(\frac{n}{\rho}))$
ℓ_1-regularized logistic regression (Lokhov et al., 2018)	1. Model width $\leq \lambda$ 2. Degree $\leq d$ 3. Minimum edge weight $\geq \eta > 0$ 4. Regularization $O(\sqrt{\frac{\ln(n^2/\rho)}{N}})$ 5. Probability of success $\geq 1 - \rho$	$O(\max\{d, \frac{1}{\eta^2}\}$ $d^3 \exp(8\lambda) \ln(\frac{n}{\rho}))$

Continued from previous page		
Paper	Assumptions	Sample complexity (N)
ℓ_1-constrained logistic regression (Rigollet and Hütter, 2017)	1. Model width $\leq \lambda$ 2. Minimum edge weight $\geq \eta > 0$ 3. Probability of success $\geq 1 - \rho$	$O(\frac{\lambda^2 \exp(8\lambda)}{\eta^4} \ln(\frac{n}{\rho}))$
Sparsitron (Klivans and Meka, 2017)	1. Model width $\leq \lambda$ 2. Minimum edge weight $\geq \eta > 0$ 3. Probability of success $\geq 1 - \rho$	$O(\frac{\lambda^2 \exp(12\lambda)}{\eta^4} \ln(\frac{n}{\rho\eta}))$
ℓ_1-constrained logistic regression (Wu et al., 2019c)	1. Model width $\leq \lambda$ 2. Minimum edge weight $\geq \eta > 0$ 3. Probability of success $\geq 1 - \rho$	$O(\frac{\lambda^2 \exp(12\lambda)}{\eta^4} \ln(\frac{n}{\rho}))$

Table B.1: Sample complexity comparison for learning Ising models. The second column lists the assumptions in their analysis. Given λ and η, the degree is bounded by $d \leq \lambda/\eta$, with equality achieved when every edge has the same weight and there is no external field.

B.2 Proof of Lemma 3.3.1 and Lemma 3.3.2

The proof of Lemma 3.3.1 relies on the following lemmas. The first lemma is a generalization error bound for any Lipschitz loss of linear functions with bounded $\|w\|_1$ and $\|x\|_\infty$.

Lemma B.2.1. *(see, e.g., Corollary 4 of (Kakade et al., 2009) and Theorem 26.15 of (Shalev-Shwartz and Ben-David, 2014)) Let $\mathcal{D}$ be a distribution on $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} = \{x \in \mathbb{R}^n : \|x\|_\infty \leq X_\infty\}$, and $\mathcal{Y} = \{-1, 1\}$. Let $\ell : \mathbb{R} \to \mathbb{R}$ be*

a loss function with Lipschitz constant L_ℓ. Define the expected loss $\mathcal{L}(w)$ and the empirical loss $\hat{\mathcal{L}}(w)$ as

$$\mathcal{L}(w) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \ell(y \langle w, x\rangle), \quad \hat{\mathcal{L}}(w) = \frac{1}{N}\sum_{i=1}^{N} \ell(y^i \langle w, x^i\rangle), \tag{B.1}$$

where $\{x^i, y^i\}_{i=1}^N$ are i.i.d. samples from distribution $\mathcal{D}$. Define $\mathcal{W} = \{w \in \mathbb{R}^n : \|w\|_1 \le W_1\}$. Then with probability at least $1-\rho$ over the samples, we have that for all $w \in \mathcal{W}$,

$$\mathcal{L}(w) \le \hat{\mathcal{L}}(w) + 2L_\ell X_\infty W_1 \sqrt{\frac{2\ln(2n)}{N}} + L_\ell X_\infty W_1 \sqrt{\frac{2\ln(2/\rho)}{N}}. \tag{B.2}$$

Lemma B.2.2. *(Pinsker's inequality) Let $D_{KL}(a||b) := a\ln(a/b) + (1-a)\ln((1-a)/(1-b))$ denote the KL-divergence between two Bernoulli distributions $(a, 1-a)$, $(b, 1-b)$ with $a, b \in [0,1]$. Then*

$$(a-b)^2 \le \frac{1}{2} D_{KL}(a||b). \tag{B.3}$$

Lemma B.2.3. *Let $\mathcal{D}$ be a distribution on $\mathcal{X} \times \{-1, 1\}$. For $(X, Y) \sim \mathcal{D}$, $\mathbb{P}[Y = 1|X = x] = \sigma(\langle w^*, x\rangle)$, where $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid function. Let $\mathcal{L}(w)$ be the expected logistic loss:*

$$\begin{aligned}\mathcal{L}(w) &= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \ln(1 + e^{-y\langle w,x\rangle}) \\ &= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} [-\frac{y+1}{2}\ln(\sigma(\langle w, x\rangle)) - \frac{1-y}{2}\ln(1-\sigma(\langle w, x\rangle))].\end{aligned} \tag{B.4}$$

Then for any w, we have

$$\mathcal{L}(w) - \mathcal{L}(w^*) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}[D_{KL}(\sigma(\langle w^*, x\rangle)||\sigma(\langle w, x\rangle))], \tag{B.5}$$

where $D_{KL}(a||b) := a\ln(a/b) + (1-a)\ln((1-a)/(1-b))$ denotes the KL-divergence between two Bernoulli distributions $(a, 1-a)$, $(b, 1-b)$ with $a, b \in [0,1]$.

Proof. Simply plugging in the definition of the expected logistic loss $\mathcal{L}(\cdot)$ gives

$$\begin{aligned}
&\mathcal{L}(w) - \mathcal{L}(w^*) \\
&= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}[-\frac{y+1}{2}\ln(\sigma(\langle w, x\rangle)) - \frac{1-y}{2}\ln(1-\sigma(\langle w, x\rangle))] \\
&\quad + \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}[\frac{y+1}{2}\ln(\sigma(\langle w^*, x\rangle)) + \frac{1-y}{2}\ln(1-\sigma(\langle w^*, x\rangle))] \\
&= \mathop{\mathbb{E}}_{x}\mathop{\mathbb{E}}_{y|x}[-\frac{y+1}{2}\ln(\sigma(\langle w, x\rangle)) - \frac{1-y}{2}\ln(1-\sigma(\langle w, x\rangle))] \\
&\quad + \mathop{\mathbb{E}}_{x}\mathop{\mathbb{E}}_{y|x}[\frac{y+1}{2}\ln(\sigma(\langle w^*, x\rangle)) + \frac{1-y}{2}\ln(1-\sigma(\langle w^*, x\rangle))] \\
&\overset{(a)}{=} \mathop{\mathbb{E}}_{x}[-\sigma(\langle w^*, x\rangle)\ln(\sigma(\langle w, x\rangle)) - (1-\sigma(\langle w^*, x\rangle))\ln(1-\sigma(\langle w, x\rangle))] \\
&\quad + \mathop{\mathbb{E}}_{x}[\sigma(\langle w^*, x\rangle)\ln(\sigma(\langle w^*, x\rangle)) + (1-\sigma(\langle w^*, x\rangle))\ln(1-\sigma(\langle w^*, x\rangle))] \\
&= \mathop{\mathbb{E}}_{x}\left[\sigma(\langle w^*, x\rangle)\ln\left(\frac{\sigma(\langle w^*, x\rangle)}{\sigma(\langle w, x\rangle)}\right) + (1-\sigma(\langle w^*, x\rangle))\ln\left(\frac{1-\sigma(\langle w^*, x\rangle)}{1-\sigma(\langle w, x\rangle)}\right)\right] \\
&= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}[D_{KL}(\sigma(\langle w^*, x\rangle)||\sigma(\langle w, x\rangle))],
\end{aligned}$$

where (a) follows from the fact that

$$E_{y|x}[y] = 1\cdot\mathbb{P}[y=1|x] + (-1)\cdot\mathbb{P}[y=-1|x] = 2\sigma(\langle w^*, x\rangle) - 1.$$

□

We are now ready to prove Lemma 3.3.1 (which is restated below):

Lemma. *Let $\mathcal{D}$ be a distribution on $\{-1,1\}^n \times \{-1,1\}$ where for $(X,Y) \sim \mathcal{D}$, $\mathbb{P}[Y=1|X=x] = \sigma(\langle w^*, x\rangle)$. We assume that $\|w^*\|_1 \le 2\lambda$ for a known $\lambda \ge 0$. Given N i.i.d. samples $\{(x^i, y^i)\}_{i=1}^N$, let $\hat{w}$ be any minimizer of the following ℓ_1-constrained logistic regression problem:*

$$\hat{w} \in \arg\min_{w \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^{N} \ln(1 + e^{-y^i \langle w, x^i \rangle}) \quad \text{s.t. } \|w\|_1 \le 2\lambda. \tag{B.6}$$

Given $\rho \in (0,1)$ and $\epsilon > 0$, suppose that $N = O(\lambda^2(\ln(n/\rho))/\epsilon^2)$, then with probability at least $1-\rho$ over the samples, we have that $\mathbb{E}_{(x,y)\sim\mathcal{D}}[(\sigma(\langle w^*, x\rangle) - \sigma(\langle \hat{w}, x\rangle))^2] \le \epsilon$.

Proof. We first apply Lemma B.2.1 to the setup of Lemma 3.3.1. The loss function $\ell(z) = \ln(1+e^{-z})$ defined above has Lipschitz constant $L_\ell = 1$. The input sample $x \in \{-1,1\}^n$ satisfies $\|x\|_\infty \le 1$. Let $\mathcal{W} = \{w \in \mathbb{R}^{n\times k} : \|w\|_1 \le 2\lambda\}$. According to Lemma B.2.1, with probability at least $1-\rho/2$ over the draw of the training set, we have that for all $w \in \mathcal{W}$,

$$\mathcal{L}(w) \le \hat{\mathcal{L}}(w) + 4\lambda\sqrt{\frac{2\ln(2n)}{N}} + 2\lambda\sqrt{\frac{2\ln(4/\rho)}{N}}. \tag{B.7}$$

where $\mathcal{L}(w) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \ln(1+e^{-y\langle w,x\rangle})$ and $\hat{\mathcal{L}}(w) = \sum_{i=1}^N \ln(1+e^{-y^i\langle w,x^i\rangle})/N$ are the expected loss and empirical loss.

Let $N = C \cdot \lambda^2 \ln(8n/\rho)/\epsilon^2$ for a global constant C, then (B.7) implies that with probability at least $1-\rho/2$,

$$\mathcal{L}(w) \le \hat{\mathcal{L}}(w) + \epsilon, \text{ for all } w \in \mathcal{W}. \tag{B.8}$$

We next prove a concentration result for $\hat{\mathcal{L}}(w^*)$. Here w^* is the true regression vector and is assumed to be fixed. First notice that $\ln(1+e^{-y\langle w^*,x\rangle})$ is bounded because $|y\langle w^*,x\rangle| \le 2\lambda$. Besides, the $\ln(1+e^{-z})$ has Lipschitz 1, so $|\ln(1+e^{-2\lambda}) - \ln(1+e^{2\lambda})| \le 4\lambda$. Hoeffding's inequality gives that $\mathbb{P}[\hat{\mathcal{L}}(w^*) - \mathcal{L}(w^*) \ge t] \le e^{-2Nt^2/(4\lambda)^2}$. Let $N = C' \cdot \lambda^2 \ln(2/\rho)/\epsilon^2$ for a global constant C', then with probability at least $1-\rho/2$ over the samples,

$$\hat{\mathcal{L}}(w^*) \le \mathcal{L}(w^*) + \epsilon. \tag{B.9}$$

Then the following holds with probability at least $1-\rho$:

$$\mathcal{L}(\hat{w}) \overset{(a)}{\le} \hat{\mathcal{L}}(\hat{w}) + \epsilon \overset{(b)}{\le} \hat{\mathcal{L}}(w^*) + \epsilon \overset{(c)}{\le} \mathcal{L}(w^*) + 2\epsilon, \tag{B.10}$$

where (a) follows from (B.8), (b) follows from the fact $\hat{w}$ is the minimizer of $\hat{\mathcal{L}}(w)$, and (c) follows from (B.9).

So far we have shown that $\mathcal{L}(\hat{w}) - \mathcal{L}(w^*) \le 2\epsilon$ with probability at least $1-\rho$. The last step is to lower bound $\mathcal{L}(\hat{w}) - \mathcal{L}(w^*)$ by $\mathbb{E}_{(x,y)\sim\mathcal{D}}(\sigma(\langle w^*,x\rangle) - \sigma(\langle w,x\rangle))^2$ using Lemma B.2.2 and Lemma B.2.3.

$$\begin{aligned}
\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}(\sigma(\langle w^*,x\rangle) - \sigma(\langle w,x\rangle))^2 &\overset{(d)}{\le} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} D_{KL}(\sigma(\langle w^*,x\rangle)||\sigma(\langle w,x\rangle))/2 \\
&\overset{(e)}{=} (\mathcal{L}(\hat{w}) - \mathcal{L}(w^*))/2 \\
&\overset{(f)}{\le} \epsilon,
\end{aligned}$$

where (d) follows from Lemma B.2.2, (e) follows from Lemma B.2.3, and (f) follows from (B.10). Therefore, we have that $\mathbb{E}_{(x,y)\sim\mathcal{D}}(\sigma(\langle w^*,x\rangle) - \sigma(\langle w,x\rangle))^2 \le \epsilon$ with probability at least $1-\rho$, if the number of samples satisfies $N = O(\lambda^2 \ln(n/\rho)/\epsilon^2)$. □

The proof of Lemma 3.3.2 is identical to the proof of Lemma 3.3.1, except that it relies on the following generalization error bound for Lipschitz loss functions with bounded $\ell_{2,1}$-norm.

Lemma B.2.4. *Let $\mathcal{D}$ be a distribution on $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} = \{x \in \mathbb{R}^{n \times k} : \|x\|_{2,\infty} \leq X_{2,\infty}\}$, and $\mathcal{Y} = \{-1, 1\}$. Let $\ell : \mathbb{R} \to \mathbb{R}$ be a loss function with Lipschitz constant L_ℓ. Define the expected loss $\mathcal{L}(w)$ and the empirical loss $\hat{\mathcal{L}}(w)$ as*

$$\mathcal{L}(w) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \ell(y \langle w, x \rangle), \quad \hat{\mathcal{L}}(w) = \frac{1}{N} \sum_{i=1}^{N} \ell(y^i \langle w, x^i \rangle), \tag{B.11}$$

where $\{x^i, y^i\}_{i=1}^N$ are i.i.d. samples from distribution $\mathcal{D}$. Define $\mathcal{W} = \{w \in \mathbb{R}^{n \times k} : \|w\|_{2,1} \leq W_{2,1}\}$. Then with probability at least $1 - \rho$ over the draw of N samples, we have that for all $w \in \mathcal{W}$,

$$\mathcal{L}(w) \leq \hat{\mathcal{L}}(w) + 2L_\ell X_{2,\infty} W_{2,1} \sqrt{\frac{6 \ln(n)}{N}} + L_\ell X_{2,\infty} W_{2,1} \sqrt{\frac{2 \ln(2/\rho)}{N}}. \tag{B.12}$$

Lemma B.2.4 can be readily derived from the existing results. First, notice that the dual norm of $\|\cdot\|_{2,1}$ is $\|\cdot\|_{2,\infty}$. Using Corollary 14 in (Kakade et al., 2012), Theorem 1 in (Kakade et al., 2009), and the fact that $\|w\|_{2,q} \leq \|w\|_{2,1}$ for $q \geq 1$, we conclude that the Rademacher complexity of the function class $\mathcal{F} := \{x \to \langle w, x \rangle : \|w\|_{2,1} \leq W_{2,1}\}$ is at most $X_{2,\infty} W_{2,1} \sqrt{6 \ln(n)/N}$. We can then obtain the standard Rademacher-based generalization bound (see, e.g., (Bartlett and Mendelson, 2002) and Theorem 26.5 in (Shalev-Shwartz and Ben-David, 2014)) for bounded Lipschitz loss functions.

We omit the proof of Lemma 3.3.2 since it is the same as that of Lemma 3.3.1.

B.3 Proof of Lemma 3.3.3

Lemma 3.3.3 is restated below.

Lemma. *Let $\mathcal{D}$ be a δ-unbiased distribution on S^n, where S is the alphabet set. For $X \sim \mathcal{D}$, any $i \in [n]$, the distribution of X_{-i} is also δ-unbiased.*

Proof. For any $j \neq i \in [n]$, any $a \in S$, and any $x \in S^{n-2}$, we have

$$\begin{aligned}
&\mathbb{P}[X_j = a | X_{[n]\setminus\{i,j\}} = x] \\
&= \sum_{b \in S} \mathbb{P}[X_j = a, X_i = b | X_{[n]\setminus\{i,j\}} = x] \\
&= \sum_{b \in S} \mathbb{P}[X_i = b | X_{[n]\setminus\{i,j\}} = x] \cdot \mathbb{P}[X_j = a | X_i = b, X_{[n]\setminus\{i,j\}} = x] \\
&\overset{(a)}{\geq} \delta \sum_{b \in S} \mathbb{P}[X_i = b | X_{[n]\setminus\{i,j\}} = x] \\
&= \delta, \qquad \text{(B.13)}
\end{aligned}$$

where (a) follows from the fact that $X \sim \mathcal{D}$ and $\mathcal{D}$ is a δ-unbiased distribution. Since (B.13) holds for any $j \neq i \in [n]$, any $a \in S$, and any $x \in S^{n-2}$, by definition, the distribution of X_{-i} is δ-unbiased. □

B.4 Proof of Lemma 3.3.4

The lemma is restated below, followed by its proof.

Lemma. *Let $\mathcal{D}(\mathcal{W}, \Theta)$ be a pairwise graphical model distribution with alphabet size k and width $\lambda(\mathcal{W}, \Theta)$. Then $\mathcal{D}(\mathcal{W}, \Theta)$ is δ-unbiased with $\delta = e^{-2\lambda(\mathcal{W},\Theta)}/k$. Specifically, an Ising model distribution $\mathcal{D}(A, \theta)$ is $e^{-2\lambda(A,\theta)}/2$-unbiased.*

Proof. Let $X \sim \mathcal{D}(\mathcal{W}, \Theta)$, and assume that $X \in [k]^n$. For any $i \in [n]$, any $a \in [k]$, and any $x \in [k]^{n-1}$, we have

$$\begin{aligned}\mathbb{P}[X_i = a | X_{-i} = x] &= \frac{\exp(\sum_{j\neq i} W_{ij}(a, x_j) + \theta_i(a))}{\sum_{b\in[k]} \exp(\sum_{j\neq i} W_{ij}(b, x_j) + \theta_i(b))} \\ &= \frac{1}{\sum_{b\in[k]} \exp(\sum_{j\neq i}(W_{ij}(b, x_j) - W_{ij}(a, x_j)) + \theta_i(b) - \theta_i(a))} \\ &\overset{(a)}{\geq} \frac{1}{k \cdot \exp(2\lambda(\mathcal{W}, \Theta))} = e^{-2\lambda(\mathcal{W},\Theta)}/k, \end{aligned} \tag{B.14}$$

where (a) follows from the definition of model width. The lemma then follows (Ising model corresponds to the special case of $k = 2$). □

B.5 Proof of Lemma 3.3.5 and Lemma 3.3.6

The proof relies on the following basic property of the sigmoid function (see Claim 4.2 of (Klivans and Meka, 2017)):

$$|\sigma(a) - \sigma(b)| \geq e^{-|a|-3} \cdot \min(1, |a - b|), \quad \forall a, b \in \mathbb{R}. \tag{B.15}$$

We first prove Lemma 3.3.5 (which is restated below).

Lemma. *Let $\mathcal{D}$ be a δ-unbiased distribution on $\{-1, 1\}^n$. Suppose that for two vectors $u, w \in \mathbb{R}^n$ and $\theta', \theta'' \in \mathbb{R}$, $\mathbb{E}_{X\sim\mathcal{D}}[(\sigma(\langle w, X\rangle + \theta') - \sigma(\langle u, X\rangle + \theta''))^2] \leq \epsilon$, where $\epsilon < \delta e^{-2\|w\|_1 - 2|\theta'| - 6}$. Then $\|w - u\|_\infty \leq O(1) \cdot e^{\|w\|_1 + |\theta'|} \cdot \sqrt{\epsilon/\delta}$.*

Proof. For any $i \in [n]$, any $X \in \{-1,1\}^n$, let $X_i \in \{-1,1\}$ be the i-th variable and $X_{-i} \in \{-1,1\}^{n-1}$ be the $[n]\backslash\{i\}$ variables. Let $X^{i,+} \in \{-1,1\}^n$ (respectively $X^{i,-}$) be the vector obtained from X by setting $X_i = 1$ (respectively $X_i = -1$). Then we have

$$\begin{aligned}
\epsilon &\geq \mathop{\mathbb{E}}_{X\sim\mathcal{D}}[(\sigma(\langle w, X\rangle + \theta') - \sigma(\langle u, X\rangle + \theta''))^2] \\
&= \mathop{\mathbb{E}}_{X_{-i}}\left[\mathop{\mathbb{E}}_{X_i|X_{-i}}(\sigma(\langle w, X\rangle + \theta') - \sigma(\langle u, X\rangle + \theta''))^2\right] \\
&= \mathop{\mathbb{E}}_{X_{-i}}[(\sigma(\langle w, X^{i,+}\rangle + \theta') - \sigma(\langle u, X^{i,+}\rangle + \theta''))^2 \cdot \mathbb{P}[X_i = 1|X_{-i}] \\
&\qquad + (\sigma(\langle w, X^{i,-}\rangle + \theta') - \sigma(\langle u, X^{i,-}\rangle + \theta''))^2 \cdot \mathbb{P}[X_i = -1|X_{-i}]] \\
&\overset{(a)}{\geq} \delta \cdot \mathop{\mathbb{E}}_{X_{-i}}[(\sigma(\langle w, X^{i,+}\rangle + \theta') - \sigma(\langle u, X^{i,+}\rangle + \theta''))^2 \\
&\qquad + (\sigma(\langle w, X^{i,-}\rangle + \theta') - \sigma(\langle u, X^{i,-}\rangle + \theta''))^2] \\
&\overset{(b)}{\geq} \delta e^{-2\|w\|_1 - 2|\theta'| - 6} \cdot \mathop{\mathbb{E}}_{X_{-i}}[\min(1, ((\langle w, X^{i,+}\rangle + \theta') - (\langle u, X^{i,+}\rangle + \theta''))^2) \\
&\qquad\qquad + \min(1, ((\langle w, X^{i,-}\rangle + \theta') - (\langle u, X^{i,-}\rangle + \theta''))^2)] \\
&\overset{(c)}{\geq} \delta e^{-2\|w\|_1 - 2|\theta'| - 6} \cdot \mathop{\mathbb{E}}_{X_{-i}} \min(1, (2w_i - 2u_i)^2/2) \\
&\overset{(d)}{=} \delta e^{-2\|w\|_1 - 2|\theta'| - 6} \cdot \min(1, 2(w_i - u_i)^2).
\end{aligned} \tag{B.16}$$

Here (a) follows from the fact that $\mathcal{D}$ is a δ-unbiased distribution, which implies that $\mathbb{P}[X_i = 1|X_{-i}] \geq \delta$ and $\mathbb{P}[X_i = -1|X_{-i}] \geq \delta$. Inequality (b) is obtained by substituting (B.15). Inequality (c) uses the following fact

$$\min(1, a^2) + \min(1, b^2) \geq \min(1, (a-b)^2/2), \forall a, b \in \mathbb{R}. \tag{B.17}$$

To see why (B.17) holds, note that if both $|a|, |b| \leq 1$, then (B.17) is true since $a^2 + b^2 \geq (a-b)^2/2$. Otherwise, (B.17) is true because the left-hand side is

at least 1 while the right-hand side is at most 1. The last equality (d) follows from that X_{-i} is independent of $\min(1, 2(w_i - u_i)^2)$.

Since $\epsilon < \delta e^{-2\|w\|_1 - 2|\theta'| - 6}$, (B.16) implies that $|w_i - u_i| \le O(1) \cdot e^{\|w\|_1 + |\theta'|} \cdot \sqrt{\epsilon/\delta}$. Because (B.16) holds for any $i \in [n]$, we have that $\|w - u\|_\infty \le O(1) \cdot e^{\|w\|_1 + |\theta'|} \cdot \sqrt{\epsilon/\delta}$. □

We now prove Lemma 3.3.6 (which is restated below).

Lemma. *Let $\mathcal{D}$ be a δ-unbiased distribution on $[k]^n$. For $X \sim \mathcal{D}$, let $\tilde{X} \in \{0,1\}^{n \times k}$ be the one-hot encoded X. Let $u, w \in \mathbb{R}^{n \times k}$ be two matrices satisfying $\sum_j u(i,j) = 0$ and $\sum_j w(i,j) = 0$ for $i \in [n]$. Suppose that for some u, w and $\theta', \theta'' \in \mathbb{R}$, we have $\mathbb{E}_{X \sim \mathcal{D}}[(\sigma(\left\langle w, \tilde{X} \right\rangle + \theta') - \sigma(\left\langle u, \tilde{X} \right\rangle + \theta''))^2] \le \epsilon$, where $\epsilon < \delta e^{-2\|w\|_{\infty,1} - 2|\theta'| - 6}$. Then $\|w - u\|_\infty \le O(1) \cdot e^{\|w\|_{\infty,1} + |\theta'|} \cdot \sqrt{\epsilon/\delta}$.*

Proof. Fix an $i \in [n]$ and $a \ne b \in [k]$. Let $X^{i,a} \in [k]^n$ (respectively $X^{i,b}$) be the vector obtained from X by setting $X_i = a$ (respectively $X_i = b$). Let

$\tilde{X}^{i,a} \in \{0,1\}^{n\times k}$ be the one-hot encoding of $X^{i,a} \in [k]^n$. Then we have

$$\begin{aligned}
\epsilon &\geq \mathop{\mathbb{E}}_{X\sim\mathcal{D}}[(\sigma(\left\langle w, \tilde{X}\right\rangle + \theta') - \sigma(\left\langle u, \tilde{X}\right\rangle + \theta''))^2] \\
&= \mathop{\mathbb{E}}_{X_{-i}}\left[\mathop{\mathbb{E}}_{X_i|X_{-i}}(\sigma(\left\langle w, \tilde{X}\right\rangle + \theta') - \sigma(\left\langle u, \tilde{X}\right\rangle + \theta''))^2\right] \\
&\geq \mathop{\mathbb{E}}_{X_{-i}}[(\sigma(\left\langle w, \tilde{X}^{i,a}\right\rangle + \theta') - \sigma(\left\langle u, \tilde{X}^{i,a}\right\rangle + \theta''))^2 \cdot \mathbb{P}[X_i = a|X_{-i}] \\
&\qquad + (\sigma(\left\langle w, \tilde{X}^{i,b}\right\rangle + \theta') - \sigma(\left\langle u, \tilde{X}^{i,b}\right\rangle + \theta''))^2 \cdot \mathbb{P}[X_i = b|X_{-i}]] \\
&\overset{(a)}{\geq} \delta e^{-2\|w\|_{\infty,1}-2|\theta'|-6} \cdot \mathop{\mathbb{E}}_{X_{-i}}[\min(1, ((\left\langle w, \tilde{X}^{i,a}\right\rangle + \theta') - (\left\langle u, \tilde{X}^{i,a}\right\rangle + \theta''))^2) \\
&\qquad\qquad + \min(1, ((\left\langle w, \tilde{X}^{i,b}\right\rangle + \theta') - (\left\langle u, \tilde{X}^{i,b}\right\rangle + \theta''))^2)] \\
&\overset{(b)}{\geq} \delta e^{-2\|w\|_{\infty,1}-2|\theta'|-6} \cdot \mathop{\mathbb{E}}_{X_{-i}} \min(1, ((w(i,a) - w(i,b)) - (u(i,a) - u(i,b)))^2/2) \\
&= \delta e^{-2\|w\|_{\infty,1}-2|\theta'|-6} \min(1, ((w(i,a) - w(i,b)) - (u(i,a) - u(i,b)))^2/2)
\end{aligned} \tag{B.18}$$

Here (a) follows from that $\mathcal{D}$ is a δ-unbiased distribution and (B.15). Inequality (b) follows from (B.17). Because $\epsilon < \delta e^{-2\|w\|_{\infty,1}-2|\theta'|-6}$, (B.18) implies that

$$(w(i,a) - w(i,b)) - (u(i,a) - u(i,b)) \leq O(1) \cdot e^{\|w\|_{\infty,1}+|\theta'|} \cdot \sqrt{\epsilon/\delta}. \tag{B.19}$$

$$(u(i,a) - u(i,b)) - (w(i,a) - w(i,b)) \leq O(1) \cdot e^{\|w\|_{\infty,1}+|\theta'|} \cdot \sqrt{\epsilon/\delta}. \tag{B.20}$$

Since (B.19) and (B.20) hold for any $a \neq b \in [k]$, we can sum over $b \in [k]$ and use the fact that $\sum_j u(i,j) = 0$ and $\sum_j w(i,j) = 0$ to get

$$\begin{aligned}
w(i,a) - u(i,a) &= \frac{1}{k}\sum_b (w(i,a) - w(i,b)) - (u(i,a) - u(i,b)) \\
&\leq O(1) \cdot e^{\|w\|_{\infty,1}+|\theta'|} \cdot \sqrt{\epsilon/\delta}. \\
u(i,a) - w(i,a) &= \frac{1}{k}\sum_b (u(i,a) - u(i,b)) - (w(i,a) - w(i,b)) \\
&\leq O(1) \cdot e^{\|w\|_{\infty,1}+|\theta'|} \cdot \sqrt{\epsilon/\delta}.
\end{aligned}$$

Therefore, we have $|w(i,a) - u(i,a)| \leq O(1) \cdot e^{\|w\|_{\infty,1}+|\theta'|} \cdot \sqrt{\epsilon/\delta}$, for any $i \in [n]$ and $a \in [k]$. □

B.6 Proof of Theorem 3.2.1

We first restate Theorem 3.2.1 and then give the proof.

Theorem. *Let $\mathcal{D}(A,\theta)$ be an unknown n-variable Ising model distribution with dependency graph G. Suppose that the $\mathcal{D}(A,\theta)$ has width $\lambda(A,\theta) \leq \lambda$. Given $\rho \in (0,1)$ and $\epsilon > 0$, if the number of i.i.d. samples satisfies $N = O(\lambda^2 \exp(12\lambda) \ln(n/\rho)/\epsilon^4)$, then with probability at least $1-\rho$, Algorithm 5 produces $\hat{A}$ that satisfies*

$$\max_{i,j\in[n]} |A_{ij} - \hat{A}_{ij}| \leq \epsilon. \tag{B.21}$$

Proof. For ease of notation, we consider the n-th variable. The goal is to prove that Algorithm 5 is able to recover the n-th row of the true weight matrix A. Specifically, we will show that if the number samples satisfies $N = O(\lambda^2 \exp(O(\lambda)) \ln(n/\rho)/\epsilon^4)$, then with probability as least $1-\rho/n$,

$$\max_{j\in[n]} |A_{nj} - \hat{A}_{nj}| \leq \epsilon. \tag{B.22}$$

We then use a union bound to conclude that with probability as least $1-\rho$, $\max_{i,j\in[n]} |A_{ij} - \hat{A}_{ij}| \leq \epsilon$.

Let $Z \sim \mathcal{D}(A,\theta)$, $X = [Z_{-n}, 1] = [Z_1, Z_2, \cdots, Z_{n-1}, 1] \in \{-1,1\}^n$, and $Y = Z_n \in \{-1,1\}$. By Fact 2, $\mathbb{P}[Y = 1|X = x] = \sigma(\langle w^*, x\rangle)$, where

$w^* = 2[A_{n1}, \cdots, A_{n(n-1)}, \theta_n]$. Further, $\|w^*\|_1 \leq 2\lambda$. Let $\hat{w}$ be the solution of the ℓ_1-constrained logistic regression problem defined in (3.4).

By Lemma 3.3.1, if the number of samples satisfies $N = O(\lambda^2 \ln(n/\rho)/\gamma^2)$, then with probability at least $1 - \rho/n$, we have

$$\mathop{\mathbb{E}}_{X}[(\sigma(\langle w^*, X\rangle) - \sigma(\langle \hat{w}, X\rangle))^2] \leq \gamma. \tag{B.23}$$

By Lemma 3.3.4, $Z_{-n} \in \{-1, 1\}^{n-1}$ is δ-unbiased (Definition 3.3.1) with $\delta = e^{-2\lambda}/2$. By Lemma 3.3.5, if $\gamma < C_1 \delta e^{-4\lambda}$ for some constant $C_1 > 0$, then (B.23) implies that

$$\|w^*_{1:(n-1)} - \hat{w}_{1:(n-1)}\|_\infty \leq O(1) \cdot e^{2\lambda} \cdot \sqrt{\gamma/\delta}. \tag{B.24}$$

Note that $w^*_{1:(n-1)} = 2[A_{n1}, \cdots, A_{n(n-1)}]$ and $\hat{w}_{1:(n-1)} = 2[\hat{A}_{n1}, \cdots, \hat{A}_{n(n-1)}]$. Let $\gamma = C_2 \delta e^{-4\lambda} \epsilon^2$ for some constant $C_2 > 0$ and $\epsilon \in (0, 1)$, (B.24) then implies that

$$\max_{j \in [n]} |A_{nj} - \hat{A}_{nj}| \leq \epsilon. \tag{B.25}$$

The number of samples needed is $N = O(\lambda^2 \ln(n/\rho)/\gamma^2) = O(\lambda^2 e^{12\lambda} \ln(n/\rho)/\epsilon^4)$.

We have proved that (B.22) holds with probability at least $1 - \rho/n$. Using a union bound over all n variables gives that with probability as least $1 - \rho$, $\max_{i,j \in [n]} |A_{ij} - \hat{A}_{ij}| \leq \epsilon$. □

B.7 Proof of Theorem 3.2.3

The following lemma will be used in the proof.

Lemma B.7.1. *Let $Z \sim \mathcal{D}$, where $\mathcal{D}$ is a δ-unbiased distribution on $[k]^n$. Given $\alpha \neq \beta \in [k]$, conditioned on $Z_n \in \{\alpha, \beta\}$, $Z_{-n} \in [k]^{n-1}$ is also δ-unbiased.*

Proof. For any $i \in [n-1]$, $a \in [k]$, and $x \in [k]^{n-2}$, we have

$$\begin{aligned}
&\mathbb{P}[Z_i = a | Z_{[n]\setminus\{i,n\}} = x, Z_n \in \{\alpha, \beta\}] \\
&= \frac{\mathbb{P}[Z_i = a, Z_{[n]\setminus\{i,n\}} = x, Z_n = \alpha] + \mathbb{P}[Z_i = a, Z_{[n]\setminus\{i,n\}} = x, Z_n = \beta]}{\mathbb{P}[Z_{[n]\setminus\{i,n\}} = x, Z_n = \alpha] + \mathbb{P}[Z_{[n]\setminus\{i,n\}} = x, Z_n = \beta]} \\
&\overset{(a)}{\geq} \min\left(\frac{\mathbb{P}[Z_i = a, Z_{[n]\setminus\{i,n\}} = x, Z_n = \alpha]}{\mathbb{P}[Z_{[n]\setminus\{i,n\}} = x, Z_n = \alpha]}, \frac{\mathbb{P}[Z_i = a, Z_{[n]\setminus\{i,n\}} = x, Z_n = \beta]}{\mathbb{P}[Z_{[n]\setminus\{i,n\}} = x, Z_n = \beta]}\right) \\
&= \min(\mathbb{P}[Z_i = a | Z_{[n]\setminus\{i,n\}} = x, Z_n = \alpha], \mathbb{P}[Z_i = a | Z_{[n]\setminus\{i,n\}} = x, Z_n = \beta]) \\
&\overset{(b)}{\geq} \delta.
\end{aligned} \tag{B.26}$$

where (a) follows from the fact that $(a+b)/(c+d) \geq \min(a/c, b/d)$ for $a, b, c, d > 0$, (b) follows from the fact that Z is δ-unbiased. □

Now we are ready to prove Theorem 3.2.3, which is restated below.

Theorem. *Let $\mathcal{D}(\mathcal{W}, \Theta)$ be an n-variable pairwise graphical model distribution with width $\lambda(\mathcal{W}, \Theta) \leq \lambda$ and alphabet size k. Given $\rho \in (0,1)$ and $\epsilon > 0$, if the number of i.i.d. samples satisfies $N = O(\lambda^2 k^4 \exp(14\lambda) \ln(nk/\rho)/\epsilon^4)$, then with probability at least $1-\rho$, Algorithm 6 produces $\hat{W}_{ij} \in \mathbb{R}^{k \times k}$ that satisfies*

$$|W_{ij}(a,b) - \hat{W}_{ij}(a,b)| \leq \epsilon, \quad \forall i \neq j \in [n],\ \forall a, b \in [k]. \tag{B.27}$$

Proof. To ease notation, let us consider the n-th variable (i.e., set $i = n$ inside the first "for" loop of Algorithm 6). The proof directly applies to other variables. We will prove the following result: if the number of samples

$N = O(\lambda^2 k^4 \exp(14\lambda) \ln(nk/\rho)/\epsilon^4)$, then with probability at least $1 - \rho/n$, the $U^{\alpha,\beta} \in \mathbb{R}^{n\times k}$ matrices produced by Algorithm 6 satisfies

$$|W_{nj}(\alpha, b) - W_{nj}(\beta, b) - U^{\alpha,\beta}(j, b)| \le \epsilon, \quad \forall j \in [n-1],\ \forall \alpha, \beta, b \in [k]. \tag{B.28}$$

Suppose that (B.28) holds, summing over $\beta \in [k]$ and using the fact that $\sum_\beta W_{nj}(\beta, b) = 0$ gives

$$|W_{nj}(\alpha, b) - \frac{1}{k} \sum_{\beta\in[k]} U^{\alpha,\beta}(j, b)| \le \epsilon, \quad \forall j \in [n-1],\ \forall \alpha, b \in [k]. \tag{B.29}$$

Theorem 3.2.3 then follows by taking a union bound over the n variables.

The only thing left is to prove (B.28). Now fix a pair of $\alpha, \beta \in [k]$, let $N^{\alpha,\beta}$ be the number of samples such that the n-th variable is either α or β. By Lemma 3.3.2 and Fact 3, if $N^{\alpha,\beta} = O(\lambda^2 k \ln(n/\rho')/\gamma^2)$, then with probability at least $1-\rho'$, the minimizer of the $\ell_{2,1}$ constrained logistic regression $w^{\alpha,\beta} \in \mathbb{R}^{n\times k}$ satisfies

$$\mathop{\mathbb{E}}_{X}[(\sigma(\langle w^*, X\rangle) - \sigma(\langle w^{\alpha,\beta}, X\rangle))^2] \le \gamma. \tag{B.30}$$

Recall that $X \in \{0,1\}^{n\times k}$ is the one-hot encoding of the vector $[Z_{-n}, 1] \in [k]^n$, where $Z \sim \mathcal{D}(\mathcal{W}, \Theta)$ and $Z_n \in \{\alpha, \beta\}$. Besides, $w^* \in \mathbb{R}^{n\times k}$ satisfies

$$w^*(j, :) = W_{nj}(\alpha, :) - W_{nj}(\beta, :),\ \forall j \in [n-1];$$
$$w^*(n, :) = [\theta_n(\alpha) - \theta_n(\beta), 0, \cdots, 0].$$

Let $U^{\alpha,\beta} \in \mathbb{R}^{n\times k}$ be formed by centering the first $n-1$ rows of $w^{\alpha,\beta}$. Since each row of X is a standard basis vector (i.e., all 0's except a single 1), $\langle U^{\alpha,\beta}, X\rangle = \langle w^{\alpha,\beta}, X\rangle$. Hence, (B.30) implies

$$\mathop{\mathbb{E}}_{X}[(\sigma(\langle w^*, X\rangle) - \sigma(\langle U^{\alpha,\beta}, X\rangle))^2] \le \gamma. \tag{B.31}$$

By Lemma 3.3.4, we know that $Z \sim \mathcal{D}(\mathcal{W}, \Theta)$ is δ-unbiased with $\delta = e^{-2\lambda}/k$. By Lemma B.7.1, conditioned on $Z_n \in \{\alpha, \beta\}$, Z_{-n} is also δ-unbiased. Hence, the condition of Lemma 3.3.6 holds. Applying Lemma 3.3.6 to (B.31), we get that if $N^{\alpha,\beta} = O(\lambda^2 k^3 \exp(12\lambda) \ln(n/\rho'))/\epsilon^4)$, the following holds with probability at least $1 - \rho'$:

$$|W_{nj}(\alpha, b) - W_{nj}(\beta, b) - U^{\alpha,\beta}(j, b)| \leq \epsilon, \ \forall j \in [n-1], \ \forall b \in [k]. \tag{B.32}$$

So far we have proved that (B.28) holds for a fixed (α, β) pair. This requires that $N^{\alpha,\beta} = O(\lambda^2 k^3 \exp(12\lambda) \ln(n/\rho'))/\epsilon^4)$. Recall that $N^{\alpha,\beta}$ is the number of samples that the n-th variable takes α or β. We next derive the number of total samples needed in order to have $N^{\alpha,\beta}$ samples for a given (α, β) pair. Since $\mathcal{D}(\mathcal{W}, \Theta)$ is δ-unbiased with $\delta = e^{-2\lambda(\mathcal{W},\Theta)}/k$, for $Z \sim \mathcal{D}(\mathcal{W}, \Theta)$, we have $\mathbb{P}[Z_n \in \{\alpha, \beta\} | Z_{-n}] \geq 2\delta$, and hence $\mathbb{P}[Z_n \in \{\alpha, \beta\}] \geq 2\delta$. By the Chernoff bound, if the total number of samples satisfies $N = O(N^{\alpha,\beta}/\delta + \log(1/\rho'')/\delta)$, then with probability at least $1 - \rho''$, we have $N^{\alpha,\beta}$ samples for a given (α, β) pair.

To ensure that (B.32) holds for all (α, β) pairs with probability at least $1 - \rho/n$, we can set $\rho' = \rho/(nk^2)$ and $\rho'' = \rho/(nk^2)$ and take a union bound over all (α, β) pairs. The total number of samples required is $N = O(\lambda^2 k^4 \exp(14\lambda) \ln(nk/\rho)/\epsilon^4)$.

We have shown that (B.28) holds for the n-th variable with probability at least $1 - \rho/n$. By the discussion at the beginning of the proof, Theorem 3.2.3 then follows by a union bound over the n variables. □

B.8 Mirror Descent Algorithms for Constrained Logistic Regression

Algorithm 8 gives a mirror descent algorithm for the following ℓ_1-constrained logistic regression:

$$\min_{w\in\mathbb{R}^n} \frac{1}{N}\sum_{i=1}^{N} \ln(1+e^{-y^i\langle w,x^i\rangle}) \qquad \text{s.t. } \|w\|_1 \le W_1. \tag{B.33}$$

We use the doubling trick to expand the dimension and re-scale the samples (Step 2 in Algorithm 8). Now the original problem becomes a logistic regression problem over a probability simplex: $\Delta_{2n+1} = \{w \in \mathbb{R}^{2n+1} : \sum_{i=1}^{2n+1} w_i = 1, w_i \ge 0, \forall i \in [2n+1]\}$.

$$\min_{w\in\Delta_{2n+1}} \frac{1}{N}\sum_{i=1}^{N} -\hat{y}^i \ln(\sigma(\langle w,\hat{x}^i\rangle)) - (1-\hat{y}^i)\ln(1-\sigma(\langle w,\hat{x}^i\rangle)), \tag{B.34}$$

where $(\hat{x}^i,\hat{y}^i) \in \mathbb{R}^{2n+1}\times\{0,1\}$. In Step 4-11 of Algorithm 8, we follow the standard simplex setup for mirror descent algorithm (see Section 5.3.3.2 of (Ben-Tal and Nemirovski, 2013)). Specifically, the negative entropy is used as the distance generating function (aka the mirror map). The projection step (Step 9) can be done by a simple ℓ_1 normalization operation. After that, we transform the solution back to the original space (Step 12).

Algorithm 9 gives a mirror descent algorithm for the $\ell_{2,1}$-constrained logistic regression:

$$\min_{w\in\mathbb{R}^{n\times k}} \frac{1}{N}\sum_{i=1}^{N} \ln(1+e^{-y^i\langle w,x^i\rangle}) \qquad \text{s.t. } \|w\|_{2,1} \le W_{2,1}. \tag{B.35}$$

Algorithm 8: Mirror descent algorithm for ℓ_1-constrained logistic regression

Input: $\{(x^i, y^i)\}_{i=1}^N$ where $x^i \in \{-1, 1\}^n$, $y^i \in \{-1, 1\}$; constraint on the ℓ_1 norm $W_1 \in \mathbb{R}_+$; number of iterations T.

Output: $\bar{w} \in \mathbb{R}^n$.

for *sample* $i \leftarrow 1$ **to** N **do**
  // Form samples $(\hat{x}^i, \hat{y}^i) \in \mathbb{R}^{2n+1} \times \{0, 1\}$.
  $\hat{x}^i \leftarrow [x^i, -x^i, 0] \cdot W_1, \quad \hat{y}^i \leftarrow (y^i + 1)/2$
end
// Initialize w as the uniform distribution.
$w^1 \leftarrow [\frac{1}{2n+1}, \frac{1}{2n+1}, \cdots, \frac{1}{2n+1}] \in \mathbb{R}^{2n+1}$
$\gamma \leftarrow \frac{1}{2W_1}\sqrt{\frac{2\ln(2n+1)}{T}}$ // Set the step size.
for *iteration* $t \leftarrow 1$ **to** T **do**
  $g^t \leftarrow \frac{1}{N}\sum_{i=1}^N (\sigma(\langle w^t, \hat{x}^i \rangle) - \hat{y}^i)\hat{x}^i$ // Compute the gradient.
  // Coordinate-wise update.
  $w_i^{t+1} \leftarrow w_i^t \exp(-\gamma g_i^t)$, for $i \in [2n+1]$
  $w^{t+1} \leftarrow w^{t+1}/\|w^{t+1}\|_1$ // Projection step.
end
$\bar{w} \leftarrow \sum_{t=1}^T w^t/T$ // Aggregate the updates.
// Transform $\bar{w}$ back to $\mathbb{R}^n$ and the actual scale.
$\bar{w} \leftarrow (\bar{w}_{1:n} - \bar{w}_{(n+1):2n}) \cdot W_1$

For simplicity, we assume that $n \geq 3$[3]. We then follow Section 5.3.3.3 of (Ben-Tal and Nemirovski, 2013) to use the following function as the mirror map $\Phi : \mathbb{R}^{n\times k} \to \mathbb{R}$:

$$\Phi(w) = \frac{e\ln(n)}{p}\|w\|_{2,p}^p, \quad p = 1 + 1/\ln(n). \tag{B.36}$$

The update step (Step 8) can be computed efficiently in $O(nk)$ time, see the discussion in Section 5.3.3.3 of (Ben-Tal and Nemirovski, 2013) for more details.

Algorithm 9: Mirror descent algorithm for $\ell_{2,1}$-constrained logistic regression

Input: $\{(x^i, y^i)\}_{i=1}^N$ where $x^i \in \{0,1\}^{n\times k}$, $y^i \in \{-1, 1\}$; constraint on the $\ell_{2,1}$ norm $W_{2,1} \in \mathbb{R}_+$; number of iterations T.

Output: $\bar{w} \in \mathbb{R}^{n\times k}$.

1 **for** *sample* $i \leftarrow 1$ **to** N **do**

  `// Form samples` $(\hat{x}^i, \hat{y}^i) \in \mathbb{R}^{n\times k} \times \{0, 1\}$.

2 $\hat{x}^i \leftarrow x^i \cdot W_{2,1}, \quad \hat{y}^i \leftarrow (y^i + 1)/2$

3 **end**

`// Initialize` w `as a constant matrix.`

4 $w^1 \leftarrow [\frac{1}{n\sqrt{k}}, \frac{1}{n\sqrt{k}}, \cdots, \frac{1}{n\sqrt{k}}] \in \mathbb{R}^{n\times k}$

5 $\gamma \leftarrow \frac{1}{2W_{2,1}}\sqrt{\frac{e\ln(n)}{T}}$ `// Set the step size.`

6 **for** *iteration* $t \leftarrow 1$ **to** T **do**

7 $g^t \leftarrow \frac{1}{N}\sum_{i=1}^N(\sigma(\langle w^t, \hat{x}^i\rangle) - \hat{y}^i)\hat{x}^i$ `// Compute the gradient.`

  `//` $\Phi(w)$ `is given in (B.36).`

8 $w^{t+1} \leftarrow \arg\min_{\|w\|_{2,1}\leq 1} \Phi(w) - \langle \nabla\Phi(w^t) - \gamma g^t, w\rangle$

9 **end**

10 $\bar{w} \leftarrow (\sum_{t=1}^T w^t/T) \cdot W_{21}$ `// Aggregate the updates.`

[3]For $n \leq 2$, we need to switch to a different mirror map, see Section 5.3.3.3 of (Ben-Tal and Nemirovski, 2013) for more details.

B.9 Proof of Theorem 3.2.5 and Theorem 3.2.6

Lemma B.9.1. *Let* $\hat{\mathcal{L}}(w) = \frac{1}{N}\sum_{i=1}^{N} \ln(1+e^{-y^i\langle w,x^i\rangle})$ *be the empirical loss. Let* $\hat{w}$ *be a minimizer of the ERM defined in (B.33). The output* $\bar{w}$ *of Algorithm 8 satisfies*

$$\hat{\mathcal{L}}(\bar{w}) - \hat{\mathcal{L}}(\hat{w}) \leq 2W_1\sqrt{\frac{2\ln(2n+1)}{T}}. \tag{B.37}$$

Similarly, let $\hat{w}$ *be a minimizer of the ERM defined in (B.35). Then the output* $\bar{w}$ *of Algorithm 9 satisfies*

$$\hat{\mathcal{L}}(\bar{w}) - \hat{\mathcal{L}}(\hat{w}) \leq O(1)\cdot W_{2,1}\sqrt{\frac{\ln(n)}{T}}. \tag{B.38}$$

Lemma B.9.1 follows from the standard convergence result for mirror descent algorithm (see, e.g., Theorem 4.2 of (Bubeck, 2015)), and the fact that the gradient g^t in Step 6 of Algorithm 8 satisfies $\|g^t\|_\infty \leq 2W_1$ (reps. the gradient g^t in Step 6 of Algorithm 9 satisfies $\|g^t\|_\infty \leq 2W_{2,1}$). This implies that the objective function after rescaling the samples is $2W_1$-Lipschitz w.r.t. $\|\cdot\|_1$ (reps. $2W_{2,1}$-Lipschitz w.r.t. $\|\cdot\|_{2,1}$).

We are now ready to prove Theorem 3.2.5, which is restated below.

Theorem. *In the setup of Theorem 3.2.1, suppose that the* ℓ_1*-constrained logistic regression in Algorithm 5 is optimized by the mirror descent method (Algorithm 8) given in Appendix B.8. Given* $\rho \in (0,1)$ *and* $\epsilon > 0$*, if the number of mirror descent iterations satisfies* $T = O(\lambda^2\exp(12\lambda)\ln(n)/\epsilon^4$*, and the number of i.i.d. samples satisfies* $N = O(\lambda^2\exp(12\lambda)\ln(n/\rho)/\epsilon^4)$*, then*

(3.6) still holds with probability at least $1-\rho$*. The total run-time of Algorithm 5 is* $O(TNn^2)$*.*

Proof. We first note that in the proof of Theorem 3.2.1, we only use $\hat{w}$ in order to apply the result from Lemma 3.3.1. In the proof of Lemma 3.3.1 (given in Appendix B.2), there is only one place where we use the definition of $\hat{w}$: the inequality (b) in (B.10). As a result, if we can show that (B.10) still holds after replacing $\hat{w}$ by $\bar{w}$, i.e.,

$$\mathcal{L}(\bar{w}) \leq \mathcal{L}(w^*) + O(\gamma), \tag{B.39}$$

then Lemma 3.3.1 would still hold, and so is Theorem 3.2.1.

By Lemma B.9.1, if the number of iterations $T = O(W_1^2 \ln(n)/\gamma^2)$, then

$$\hat{\mathcal{L}}(\bar{w}) - \hat{\mathcal{L}}(\hat{w}) \leq \gamma. \tag{B.40}$$

As a result, we have

$$\mathcal{L}(\bar{w}) \overset{(a)}{\leq} \hat{\mathcal{L}}(\bar{w}) + \gamma \overset{(b)}{\leq} \hat{\mathcal{L}}(\hat{w}) + 2\gamma \overset{(c)}{\leq} \hat{\mathcal{L}}(w^*) + 2\gamma \overset{(d)}{\leq} \mathcal{L}(w^*) + 3\gamma, \tag{B.41}$$

where (a) follows from (B.8), (b) follows from (B.40), (c) follows from the fact that $\hat{w}$ is the minimizer of $\hat{\mathcal{L}}(w)$, and (d) follows from (B.9). The number of mirror descent iterations needed for (B.39) to hold is $T = O(W_1^2 \ln(n)/\gamma^2)$. In the proof of Theorem 3.2.1, we need to set $\gamma = O(1)\epsilon^2 \exp(-6\lambda)$ (see the proof following (B.24)), so the number of mirror descent iterations needed is $T = O(\lambda^2 \exp(12\lambda) \ln(n)/\epsilon^4)$.

To analyze the runtime of Algorithm 5, note that for *each variable* in $[n]$, transforming the samples takes $O(N)$ time, solving the ℓ_1-constrained logisitic regression via Algorithm 8 takes $O(TNn)$ time, and updating the edge weight estimate takes $O(n)$ time. Forming the graph $\hat{G}$ over n nodes takes $O(n^2)$ time. The total runtime is $O(TNn^2)$. □

The proof of Theorem 3.2.6 is identical to that of Theorem 3.2.5 and is omitted here. The key step is to show that (B.39) holds after replacing $\hat{w}$ by $\bar{w}$. This can be done by using the convergence result in Lemma B.9.1 and applying the same logic in (B.41). The runtime of Algorithm 6 can be analyzed in the same way as above. The $\ell_{2,1}$-constrained logistic regression dominates the total runtime. It requires $O(TN^{\alpha,\beta}nk)$ time for each pair (α, β) and each variable in $[n]$, where $N^{\alpha,\beta}$ is the subset of samples that a given variable takes either α or β. Since $N \geq kN^{\alpha,\beta}$, the total runtime is $O(TNn^2k^2)$.

B.10 More Experimental Results

We compare our algorithm (Algorithm 6) with the Sparsitron algorithm in (Klivans and Meka, 2017) on a two-dimensional 3-by-3 grid (shown in Figure 3.2). We experiment three alphabet sizes: $k = 2, 4, 6$. For each value of k, we simulate both algorithms 100 runs, and in each run we generate the W_{ij} matrices with entries ± 0.2. To ensure that each row (as well as each column) of W_{ij} is centered (i.e., zero mean), we will randomly choose W_{ij} between two options: as an example of $k = 2$, $W_{ij} = [0.2, -0.2; -0.2, 0.2]$ or

$W_{ij} = [-0.2, 0.2; 0.2, -0.2]$. The external field is zero. Sampling is done via exactly computing the distribution. The Sparsitron algorithm requires two sets of samples: 1) to learn a set of candidate weights; 2) to select the best candidate. We use $\max\{200, 0.01 \cdot N\}$ samples for the second part. We plot the estimation error $\max_{ij} \|W_{ij} - \hat{W}_{ij}\|_\infty$ and the fraction of successful runs (i.e., runs that exactly recover the graph) in Figure B.1. Compared to the Sparsitron algorithm, our algorithm requires fewer samples for successfully recovery.

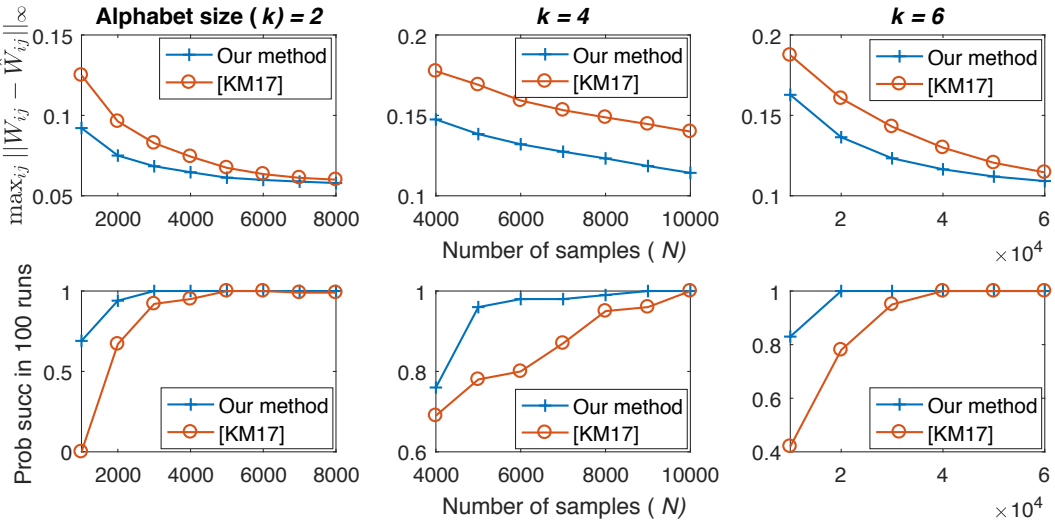


Figure B.1: Comparison of our algorithm and the Sparsitron algorithm in (Klivans and Meka, 2017) on a two-dimensional 3-by-3 grid. Top row shows the average of the estimation error $\max_{ij} \|W_{ij} - \hat{W}_{ij}\|_\infty$. Bottom row plots the faction of successful runs (i.e., runs that exactly recover the graph). Each column corresponds to an alphabet size: $k = 2, 4, 6$. Our algorithm needs fewer samples than the Sparsitron algorithm for graph recovery.

Appendix C

Appendix for Chapter 4

C.1 Proof of Lemma 4.3.1

For convenience, we re-state Lemma 4.3.1 here and then give the proof.

Lemma. *For any vector $x \in \mathbb{R}^d$, and any matrix $A \in \mathbb{R}^{m\times d}$ $(m < d)$ with rank m, there exists an $\tilde{A} \in \mathbb{R}^{m\times d}$ with all singular values being ones, such that the following two ℓ_1-norm minimization problems have the same solution:*

$$P_1: \quad \min_{x'\in\mathbb{R}^d} \|x'\|_1 \quad \text{s.t. } Ax' = Ax. \tag{C.1}$$

$$P_2: \quad \min_{x'\in\mathbb{R}^d} \|x'\|_1 \quad \text{s.t. } \tilde{A}x' = \tilde{A}x. \tag{C.2}$$

Furthermore, the projected subgradient update of P_2 is given as

$$x^{(t+1)} = x^{(t)} - \alpha_t(I - \tilde{A}^T\tilde{A})\text{sign}(x^{(t)}), \quad x^{(1)} = \tilde{A}^T\tilde{A}x.$$

A natural choice for $\tilde{A}$ is $U(AA^T)^{-1/2}A$, where $U \in \mathbb{R}^{m\times m}$ can be any unitary matrix.

Proof. To prove that P_1 and P_2 give the same solution, it suffices to show that their constraint sets are equal, i.e.,

$$\{x : Ax = Az\} = \{x : \tilde{A}x = \tilde{A}z\}. \tag{C.3}$$

Since $\{x : Ax = Az\} = \{z + v : v \in \text{null}(A)\}$ and $\{x : \tilde{A}x = \tilde{A}z\} = \{z + v : v \in \text{null}(\tilde{A})\}$, it then suffices to show that A and $\tilde{A}$ have the same nullspace:

$$\text{null}(A) = \text{null}(\tilde{A}). \tag{C.4}$$

If v satisfies $Av = 0$, then $U(AA^T)^{-1/2}Av = 0$, which implies $\tilde{A}v = 0$. Conversely, we suppose that $\tilde{A}v = 0$. Since U is unitary, $AA^T \in \mathbb{R}^{m\times m}$ is full-rank, $(AA^T)^{(1/2)}U^T\tilde{A}v = 0$, which implies that $Av = 0$. Therefore, (C.4) holds.

The projected subgradient of P_2 has the following update

$$x^{(t+1)} = x^{(t)} - \alpha_t(I - \tilde{A}^T(\tilde{A}\tilde{A}^T)^{-1}\tilde{A})\text{sign}(x^{(t)}), \tag{C.5}$$

$$x^{(1)} = \tilde{A}^T(\tilde{A}\tilde{A}^T)^{-1}\tilde{A}z \tag{C.6}$$

Since $\tilde{A} = U(AA^T)^{-1/2}A$, we have

$$\begin{aligned}\tilde{A}\tilde{A}^T &= U(AA^T)^{-1/2}AA^T(AA^T)^{-1/2}U^T \\ &= U(AA^T)^{-1/2}(AA^T)^{1/2}(AA^T)^{1/2}(AA^T)^{-1/2}U^T \\ &= I. \end{aligned} \tag{C.7}$$

Substituting (C.7) into (C.6) gives the desired recursion:

$$x^{(t+1)} = x^{(t)} - \alpha_t(I - \tilde{A}^T\tilde{A})\text{sign}(x^{(t)}), \quad x^{(1)} = \tilde{A}^T\tilde{A}z.$$

□

C.2 Training Parameters

Table C.1 lists the parameters used to train ℓ_1-AE in our experiments. We explain the parameters as follows.

Dataset	Depth	Batch size	Learning rate	N_{max}	$N_{\text{validation}}$	$N_{\text{no improve}}$
Toy	10	128	0.01	2e4	10	5
Synthetic1	10	128	0.01	2e4	10	5
Synthetic2	5	128	0.01	2e4	10	1
Synthetic3	5	128	0.01	2e4	10	1
Amazon	60	256	0.01	2e4	1	1
Wiki10-31K	10	256	0.001	5e3	10	1
RCV1	10	256	0.001	1e3	1	50

Table C.1: Hyper-parameters used to train our autoencoder.

- Depth: The number of blocks in the decoder, indicated by T in Figure 4.1.
- Batch size: The number of training samples in a batch.
- Learning rate: The learning rate for SGD.
- N_{max}: Maximum number of training epochs.
- $N_{\text{validation}}$: Validation error is computed every $N_{\text{validation}}$ epochs. This is used for early-stopping.
- $N_{\text{no improve}}$: Training is stopped if the validation error does not improve for $N_{\text{no improve}} * N_{\text{validation}}$ epochs.

C.3 Model-based CoSaMP with Additional Positivity Constraint

The CoSaMP algorithm (Needell and Tropp, 2009) is a simple iterative and greedy algorithm used to recover a K-sparse vector from the linear measurements. The model-based CoSaMP algorithm (Algorithm 1 of (Baraniuk

et al., 2010)) is a modification of the CoSaMP algorithm. It uses the prior knowledge about the support of the K-sparse vector, which is assumed to follow a predefined *structured sparsity model*. In this section we slightly modify the model-based CoSaMP algorithm to ensure that the output vector follows the given sparsity model and is also *nonnegative*.

To present the pseudocode, we need a few definitions. We begin with a formal definition for the structured sparsity model $\mathcal{M}_K$ and the sparse approximation algorithm $\mathbb{M}$. For a vector $x \in \mathbb{R}^d$, let $x|_\Omega \in \mathbb{R}^{|\Omega|}$ be entries of x in the index set $\Omega \in [d]$. Let $\Omega^C = [d] - \Omega$ be the complement of set Ω.

Definition C.3.1 ((Baraniuk et al., 2010))**.** A structured sparsity model $\mathcal{M}_K$ is defined as the union of m_K canonical K-dimensional subspaces

$$\mathcal{M}_K = \bigcup_{m=1}^{m_K} \mathcal{X}_m \quad \text{s.t. } \mathcal{X}_m = \{x : x|_{\Omega_m} \in \mathbb{R}^K, x|_{\Omega_m^C} = 0\}, \tag{C.8}$$

where $\{\Omega_1, ..., \Omega_{m_K}\}$ is the set containing all allowed supports, with $|\Omega_m| = K$ for each $m = 1, ..., m_K$, and each subspace $\mathcal{X}_m$ contains all signals x with $\text{supp}(x) \subset \Omega_m$.

We define $\mathbb{M}(x, K)$ as the algorithm that obtains the best K-term structured sparse approximation of x in the union of subspaces $\mathcal{M}_K$:

$$\mathbb{M}(x, K) = \arg \min_{\bar{x} \in \mathcal{M}_K} \|x - \bar{x}\|_2. \tag{C.9}$$

We next define an enlarged set of subspaces $\mathcal{M}_K^B$ and the associated sparse approximation algorithm.

Definition C.3.2 ((Baraniuk et al., 2010)). The B-order sum for the set $\mathcal{M}_K$, with $B > 1$ an integer, is defined as

$$\mathcal{M}_K^B = \left\{ \sum_{r=1}^{B} x^{(r)}, \quad \text{with } x^{(r)} \in \mathcal{M}_K \right\}. \tag{C.10}$$

We define $\mathbb{M}_B(x, K)$ as the algorithm that obtains the best approximation of x in the union of subspaces $\mathcal{M}_K^B$:

$$\mathbb{M}_B(x, K) = \arg \min_{\bar{x} \in \mathcal{M}_K^B} \|x - \bar{x}\|_2. \tag{C.11}$$

Algorithm 10 presents the model-based CoSaMP with positivity constraint. Comparing Algorithm 10 with the original model-based CoSaMP algorithm (Algorithm 1 of (Baraniuk et al., 2010)), we note that the only different is that Algorithm 10 has an extra step (Step 6). In Step 6 we take a ReLU operation on b to ensure that $\hat{x}_i$ is always nonnegative after Step 7.

We now show that Algorithm 10 has the same performance guarantee as the original model-based CoSaMP algorithm for structured sparse signals. Speficially, we will show that Theorem 4 of (Baraniuk et al., 2010) also applies to Algorithm 10. In (Baraniuk et al., 2010), the proof of Theorem 4 is based on six lemmas (Appendix D), among which the only lemma that is related to Step 6-7 is Lemma 6. It then suffices to prove that this lemma is also true for Algorithm 10 under the constraint that the true vector x is nonnegative.

Lemma (Prunning). *The pruned approximation* $\hat{x}_i = \mathbb{M}(\hat{b}, K)$ *is such that*

$$\|x - \hat{x}_i\|_2 \leq 2\|x - b\|_2. \tag{C.12}$$

Algorithm 10: Model-based CoSaMP with positivity constraint

Input: measurement matrix A, measurements y, structured sparse approximation algorithm $\mathbb{M}$.

Output: K-sparse approximation $\hat{x}$ to the true signal x, which is assumed to be nonnegative.

$\hat{x}_0 = 0$, $r = y$; $i = 0$. `// Initialization`

while *halting criterion false* **do**

  $i \leftarrow i + 1$

  $e \leftarrow A^T r$ `// Form signal residual estimate`

  `// Prune residual estimate according to structure`

  $\Omega \leftarrow \text{supp}(\mathbb{M}_2(e, K))$

  $T \leftarrow \Omega \cup \text{supp}(\hat{x}_{i-1})$ `// Merge support`

  `// Form signal estimate by least-squares`

  $b|_T \leftarrow A_T^{\dagger} y$, $b|_{T^C} \leftarrow 0$

  $\hat{b} = \max\{0, b\}$ `// Set the negative entries to be zero`

  `// Prune signal estimate according to structure`

  $\hat{x}_i \leftarrow \mathbb{M}(\hat{b}, K)$

  $r \leftarrow y - A\hat{x}_i$ `// Update measurement residual`

end

$\hat{x} \leftarrow \hat{x}_i$ `// Return the result`

Proof. Since $\hat{x}_i$ is the K-best approximation of $\hat{b}$ in $\mathcal{M}_K$, and $x \in \mathcal{M}_K$, we have

$$\|x - \hat{x}_i\|_2 \leq \|x - \hat{b}\|_2 + \|\hat{b} - \hat{x}_i\|_2 \leq 2\|x - \hat{b}\|_2 \leq 2\|x - b\|_2, \tag{C.13}$$

where the last inequality follows from that $\hat{b} = \max\{0, b\}$, and $x \geq 0$. □

The above lemma matches Lemma 6, which is used to prove Theorem 4 in (Baraniuk et al., 2010). Since the other lemmas (i.e., Lemma 1-5 in Appendix D of (Baraniuk et al., 2010)) still hold for Algorithm 10, we conclude that the performance guarantee for structured sparse signals (i.e., Theorem 4 of (Baraniuk et al., 2010)) is also true for Algorithm 10.

In Figure C.1, we compare the recovery performance of two decoding algorithms: 1) model-based CoSaMP algorithm (Algorithm 1 of (Baraniuk et al., 2010)) and 2) model-based CoSaMP algorithm with positivity constraint (indicated by "Model-based CoSaMP pos" in Figure C.1). We use random Gaussian matrices as the measurement matrices. Since our sparse datasets are all nonnegative, adding the positivity constraint to the decoding algorithm is able to improve the recovery performance.

C.4 Additional Experimental Results

C.4.1 A toy experiment

We use a simple example to illustrate that the measurement matrix learned from our autoencoder is adapted to the training samples. The toy dataset is generated as follows: each vector $x \in \mathbb{R}^{100}$ has 5 nonzeros randomly

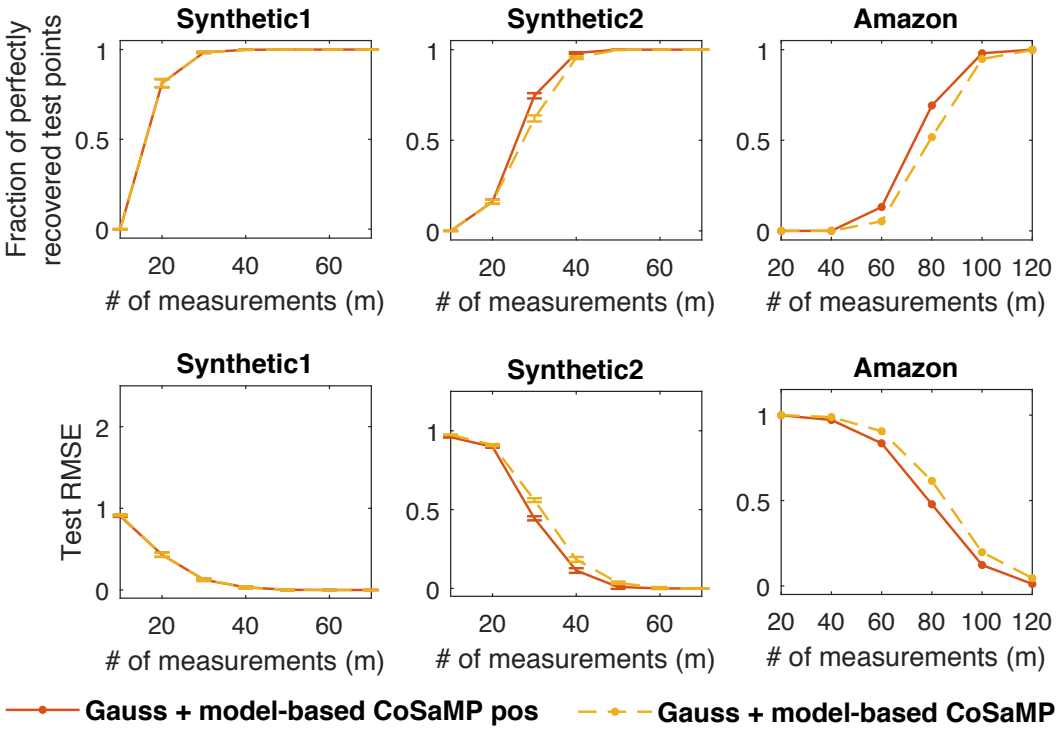


Figure C.1: Incorporating the positivity constraint to the model-based CoSaMP algorithm improves its recovery performance.

located in the first 20 dimensions; the nonzeros are random values between [0,1]. We train ℓ_1-AE on a training set with 6000 samples. The parameters are $T = 10$, $m = 10$, and learning rate 0.01. A validation set with 2000 samples is used for early-stopping.After training, we plot the matrix A in Figure C.2. The entries with large values are concentrated in the first 20 dimensions. This agrees with the specific structure in the toy dataset.

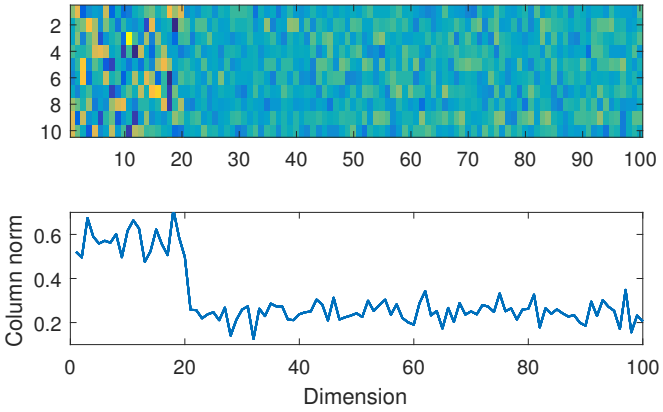


Figure C.2: Visualization of the learned matrix $A \in \mathbb{R}^{10\times 100}$ on the toy dataset: a color map of the matrix (upper), the column-wise ℓ_2 norm (lower). Every sample in the toy dataset has 5 nonzeros, located randomly in the first 20 dimensions.

C.4.2 Random Partial Fourier Matrices

Figure C.3 is a counterpart of Figure 4.2 and Figure 4.3. The only difference is that in Figure C.3 we use random partial Fourier matrices in place of random Gaussian matrices. A random $M \times N$ partial Fourier matrix is obtained by choosing M rows uniformly and independently with replacement from the $N \times N$ discrete Fourier transform (DFT) matrix. We then scale each entry to have absolute value $1/\sqrt{M}$ (Haviv and Regev, 2017). Because the DFT matrix is complex, to obtain m *real* measurements, we draw $m/2$ random rows from a DFT matrix to form the partial Fourier matrix.

A random partial Fourier matrix is a Vandermonde matrix. According to (Donoho and Tanner, 2005), one can exactly recover a k-sparse nonnegative vector from $2k$ measurements using a Vandermonde matrix (Donoho and

Tanner, 2005). However, the Vandermonde matrices are numerically unstable in practice (Pan, 2016), which is consistent with our empirical observation. Comparing Figure C.3 with Figure 4.2 and Figure 4.3, we see that the recovery performance of a random partial Fourier matrix has larger variance than that of a random Gaussian matrix.

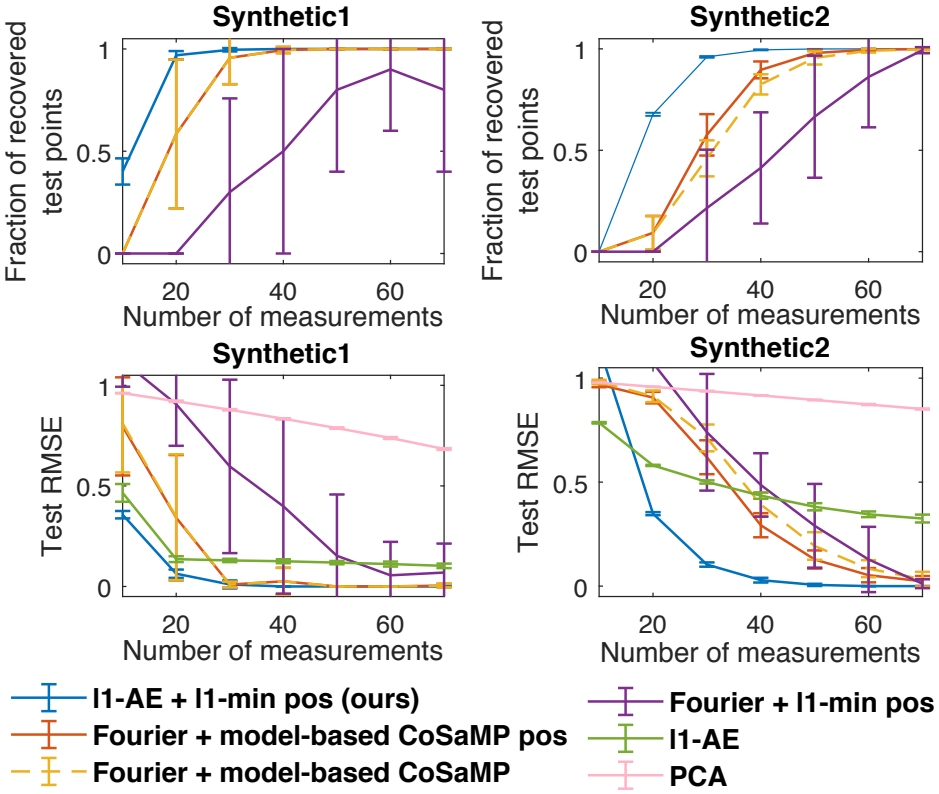


Figure C.3: Recovery performance of random partial Fourier matrices. Best viewed in color. Similar to Figure 4.2 and Figure 4.3, the error bars represent the standard deviation across 10 randomly generated datasets. We see that the recovery performance of a random partial Fourier matrix (shown in this figure) has a larger variance than that of a random Gaussian matrix (shown in Figure 4.2 and Figure 4.3).

C.4.3 ℓ_1-minimization with Positivity Constraint

We compare the recovery performance between solving an ℓ_1-min (4.5) and an ℓ_1-min with positivity constraint (4.16). The results are shown in Figure C.4. We experiment with two measurement matrices: 1) the one obtained from training our autoencoder, and 2) random Gaussian matrices. As shown in Figure C.4, adding a positivity constraint to the ℓ_1-minimization improves the recovery performance for nonnegative input vectors.

C.4.4 Singular Values of the Learned Measurement Matrices

We have shown that the measurement matrix obtained from training our autoencoder is able to capture the sparsity structure of the training data. We are now interested in looking at those data-dependent measurement matrices more closely. Table C.2 shows that those matrices have singular values close to one. Recall that in Section 4.3.1 we show that matrices with all singular values being ones have a simple form for the projected subgradient update (4.13). Our decoder is designed based on this simple update rule. Although we do not explicitly enforce this constraint during training, Table C.2 indicates that the learned matrices are not far from the constraint set.

C.4.5 Additional Experiments of LBCS

We experimented with four variations of LBCS: two different basis matrices (random Gaussian matrix and DCT matrix), two different decoders (ℓ_1-minimization and linear decoder). As shown in Figure C.5, the combination

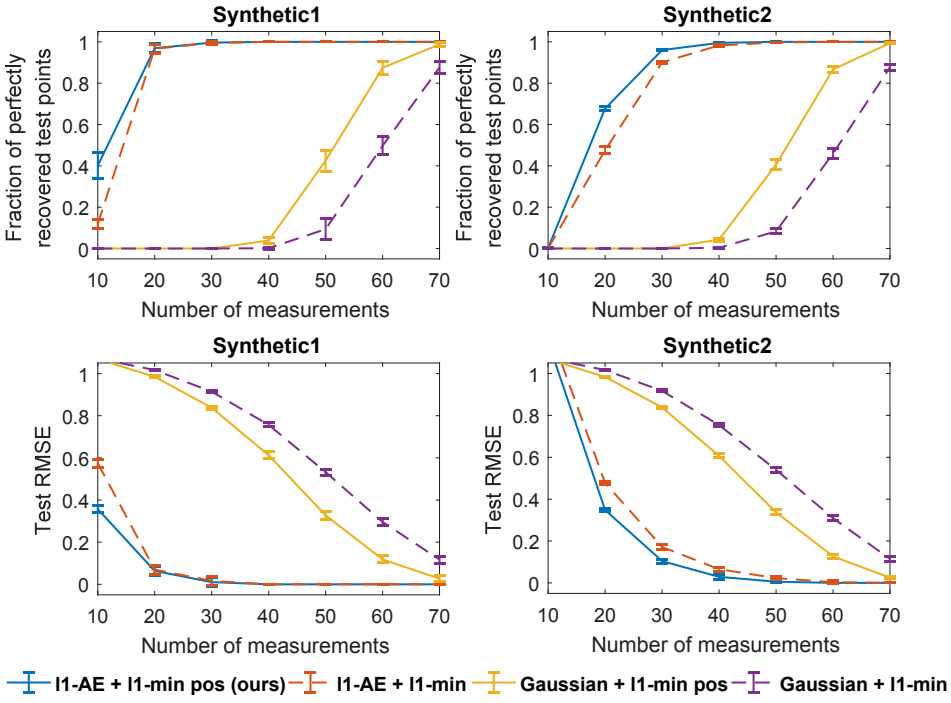


Figure C.4: A comparison of the recovery performance between ℓ_1-min (4.5) and the ℓ_1-min with positivity constraint (4.16). The sparse recovery performance is measured on the test set. Best viewed in color. We plot the mean and standard deviation (indicated by the error bars) across 10 randomly generated datasets. Adding a positivity constraint to the ℓ_1-minimization gives better recovery performance than a vanilla ℓ_1-minimization.

Dataset	σ_{largest}	σ_{smallest}
Synthetic1	1.117 ± 0.003	0.789 ± 0.214
Synthetic2	1.113 ± 0.006	0.929 ± 0.259
Synthetic3	1.162 ± 0.014	0.927 ± 0.141
Amazon	1.040 ± 0.021	0.804 ± 0.039
Wiki10-31K	1.097 ± 0.003	0.899 ± 0.044
RCV1	1.063 ± 0.016	0.784 ± 0.034

Table C.2: Range of the singular values of the measurement matrices $A \in \mathbb{R}^{m \times d}$ obtained from training ℓ_1-AE . The mean and standard deviation is computed by varying the number of m (i.e., the "number of measurements" in Figure 4.2 and Figure 4.3).

of Gaussian and ℓ_1-minimization performs the best.

C.4.6 Precision Score Comparisons for XML

Table C.3 compares the precision scores (P@1, P@3, P@5) over two benchmark datasets. For SLEEC, the precision scores we obtained by running their code (and combining 5 models in the ensemble) are consistent with those reported in the benchmark website (Bhatia et al., 2017). Compared to SLEEC, our method (which learns label embeddings via training an autoencoder ℓ_1-AE) is able to achieve better or comparable precision scores. For our method, we have experimented with three prediction approaches (denoted as "ℓ_1-AE 1/2/3" in Table C.3): 1) use the nearest neighbor method (same as SLEEC); 2) use the decoder of the trained ℓ_1-AE (which maps from the embedding space to label space); 3) use an average of the label vectors obtained from 1) and 2). As indicated in Table C.3, the third prediction approach performs the best.

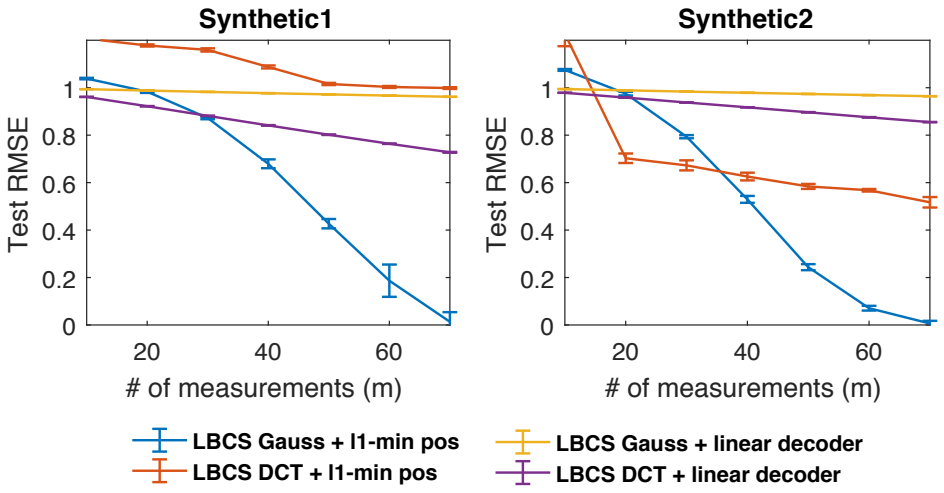


Figure C.5: We compare four variations of the LBCS method proposed in (Baldassarre et al., 2016; Li and Cevher, 2016): two basis matrices (random Gaussian and DCT matrix); two decoders (ℓ_1-minimization and linear decoding). The combination of "Gaussian + ℓ_1-minimization" performs the best. Best viewed in color. For each method, we plot the mean and standard deviation (indicated by the error bars) across 10 randomly generated datasets.

Dataset	EURLex-4K			Wiki10-31K		
# models in the ensemble	1	3	5	1	3	5
SLEEC	0.7600	0.7900	0.7944	0.8356	0.8603	0.8600
ℓ_1-AE 1	0.7655	0.7928	0.7931	0.8529	0.8564	0.8597
ℓ_1-AE 2	0.7949	0.8033	0.8070	0.8560	0.8579	0.8583
ℓ_1-AE 3	**0.8062**	**0.8151**	**0.8136**	**0.8617**	**0.8640**	**0.8630**

Dataset	EURLex-4K			Wiki10-31K		
# models in the ensemble	1	3	5	1	3	5
SLEEC	0.6116	0.6403	0.6444	0.7046	0.7304	0.7357
ℓ_1-AE 1	0.6094	0.6347	0.6360	0.7230	0.7298	0.7323
ℓ_1-AE 2	0.6284	0.6489	0.6575	0.7262	0.7293	0.7296
ℓ_1-AE 3	**0.6500**	**0.6671**	**0.6693**	**0.7361**	**0.7367**	**0.7373**

Dataset	EURLex-4K			Wiki10-31K		
# models in the ensemble	1	3	5	1	3	5
SLEEC	0.4965	0.5214	0.5275	0.5979	0.6286	0.6311
ℓ_1-AE 1	0.4966	0.5154	0.5209	0.6135	0.6198	0.6230
ℓ_1-AE 2	0.5053	0.5315	0.5421	0.6175	0.6245	0.6268
ℓ_1-AE 3	**0.5353**	**0.5515**	**0.5549**	**0.6290**	**0.6322**	**0.6341**

Table C.3: Comparison of the precision scores: P@1, P@3, P@5.

Appendix D

Appendix for Chapter 5

D.1 Weighted Alternating Minimization

Algorithm 11 provides the pseudocode of the weighted alternating minimization algorithm (WAltMin). Alternating minimization is a standard procedure for matrix completion (Jain et al., 2013). WAltMin is a weighted version of this procedure. The weights (i.e., the w's defined in Step 2 of Algorithm 11) are used to compensate the sampling bias. Here we follow the notations in Subroutine 2 of (Bhojanapalli et al., 2015) to illustrate WAltMin.

Recall that we use $P_\Omega(A)$ to denote a matrix with entries sampled from the set Ω: $P_\Omega(A)(i,j) = A(i,j)$ if $(i,j) \in \Omega$, and $P_\Omega(A)(i,j) = 0$ otherwise. We now define $R_\Omega(A) = w \bigodot P_\Omega(A)$ as the Hadamard product (i.e., element-wise multiplication) between the weight matrix w and $P_\Omega(A)$: $R_\Omega(A)(i,j) = w(i,j) \cdot P_\Omega(A)(i,j)$ for all (i,j). Similarly we define the matrix $R_\Omega^{1/2}(A)$ as $R_\Omega^{1/2}(A)(i,j) = \sqrt{w(i,j)} \cdot P_\Omega(A)(i,j)$ for $(i,j) \in \Omega$ and 0 otherwise. The weight matrix is defined in Step 2 of Algorithm 11.

In Step 4, we use the sample-splitting technique (Jain et al., 2013) to obtain $2T+1$ random partitions of Ω (every sample belongs to any one of the partitions with equal probability). The algorithm then proceed in two steps:

initialization (Step 5-7) and weighted alternating minimization (Step 8-11). In the initialization part, we compute SVD of the matrix $R_\Omega(\widetilde{M})$ to obtain its top-r left singular vectors $U^{(0)}$. We then use trim $U^{(0)}$ to obtain $\widehat{U}^{(0)}$: for row i of $U^{(0)}$, if its row norm is larger than $4\|A_i\|/\|A\|_F$, make the entire row to be zero; let the resulted matrix be $\tilde{U}^{(0)}$; we then compute its column space as $\widehat{U}^{(0)}$. The goal of Step 8-11 is to solve the following non-convex problem via alternating minimization:

$$\min_{U,V} \sum_{(i,j)\in\Omega} w_{ij}(e_i^T U V^T e_j - \widetilde{M}(i,j))^2, \tag{D.1}$$

where e_i, e_j are the standard basis vectors. After running T iterations, the algorithm outputs $\widehat{U}^{(T)}$ and $\widehat{V}^{(T)}$, which are a rank-r approximation of $\widetilde{M}$ presented in the factored form.

D.2 Sampling

We describe a way to sample m elements in $O(m\log(n))$ time using distribution q_{ij} defined in Eq. (5.1). Naively one can compute all the n^2 entries of $\min\{q_{ij}, 1\}$ and toss a coin for each entry, which takes $O(n^2)$ time. Instead of this binomial sampling we can switch to row wise multinomial sampling. For this, first estimate the expected number of samples per row $m_i = m(\frac{||A_i||^2}{2||A||_F^2} + \frac{1}{2n})$. Now sample m_1 entries from row 1 according to the multinomial distribution,

$$\widetilde{q}_{1j} = \frac{m}{m_1} \cdot (\frac{||A_1||^2}{2n||A||_F^2} + \frac{||B_j||^2}{2n||B||_F^2}) = \frac{\frac{||A_1||^2}{2n||A||_F^2} + \frac{||B_j||^2}{2n||B||_F^2}}{\frac{||A_i||^2}{2||A||_F^2} + \frac{1}{2n}}.$$

Note that $\sum_j \widetilde{q}_{1j} = 1$. To sample from this distribution, we can generate a random number in the interval $[0, 1]$, and then locate the corresponding column

Algorithm 11: WAltMin (Subroutine 2 of (Bhojanapalli et al., 2015))

Input: $P_\Omega(\widetilde{M}) \in \mathbb{R}^{n_1 \times n_2}$, Ω, r, $\hat{q}$, and T.

Output: $\widehat{U}^{(T)} \in \mathbb{R}^{n_1 \times r}$ and $\widehat{V}^{(T)} \in \mathbb{R}^{n_2 \times r}$.

for $i \leftarrow 1$ **to** n_1, $j \leftarrow 1$ **to** n_2 **do**

  $w_{ij} = 1/\hat{q}_{ij}$ if $\hat{q}_{ij} > 0$; otherwise $w_{ij} = 0$.

end

Divide Ω in $2T + 1$ equal uniformly random subsets, i.e., $\Omega = \{\Omega_0, \ldots, \Omega_{2T}\}$.

$R_{\Omega_0}(\widetilde{M}) = w \bigodot P_{\Omega_0}(\widetilde{M})$.

$U^{(0)}\Sigma^{(0)}(V^{(0)})^T = \mathrm{SVD}(R_{\Omega_0}(\widetilde{M}), r)$.

Trim $U^{(0)}$ to get $\widehat{U}^{(0)}$.

for $t \leftarrow 0$ **to** $T - 1$ **do**

  $\widehat{V}^{(t+1)} = \arg\min_V \|R^{1/2}_{\Omega_{2t+1}}(\widetilde{M} - \widehat{U}^{(t)}V^T)\|_F^2$;

  $\widehat{U}^{(t+1)} = \arg\min_U \|R^{1/2}_{\Omega_{2t+2}}(\widetilde{M} - U(\widehat{V}^{(t+1)})^T)\|_F^2$.

end

index by binary searching over the cumulative distribution function (CDF) of $\widetilde{q}_{1j}$. This takes $O(n)$ time for setting up the distribution and $O(m_1 \log(n))$ time to sample. For subsequent row i, we only need $O(m_i \log(n))$ time to sample m_i entries. This is because for binary search to work, only $O(m_i \log(n))$ entries of the CDF vector needs to be computed and checked. Note that the specific form of $\widetilde{q}_{ij}$ ensures that its CDF entries can be updated in an efficient way (since we only need to update the linear shift and scale). Hence, sampling m elements takes a total $O(m \log(n))$ time. Furthermore, the error in this model is bounded up to a factor of 2 of the error achieved by the Binomial model (Candès and Recht, 2009; Kannan et al., 2014).

D.3 Proof of Theorem 5.3.2

Our proof will use the following lemma.

Lemma D.3.1. *Let $x, y \in \mathbb{R}^k$ be two vectors that are independently uniformly distributed on the unit sphere S^{k-1}. Let $u = \langle x, y \rangle$, then $(u+1)/2 \sim$* Beta$(\frac{k-1}{2}, \frac{k-1}{2})$.

Proof. Without loss of generality, we assume that $x = [1, 0, ..., 0]$ and $y = [u, y(2), ..., y(k)]$. According to the coarea formula (Nicolaescu, 2018, Example 9.1.10), we have for $0 \le a \le b \le 1$,

$$\mathbb{P}[a \le u \le b] = \sigma_{k-2} \int_a^b (1 - t^2)^{(k-3)/2} \, \mathrm{d}t,$$

where $\sigma_m \in \mathbb{R}$ denotes the area of the unit m-dimensional sphere S^m. The

probability density function of u is

$$\mathbb{P}[u = x] = \sigma_{k-2} \frac{\mathrm{d}}{\mathrm{d}x}\bigg|_{h=0} \int_x^{x+h} (1-t^2)^{(k-3)/2}\,\mathrm{d}t \propto (1-x^2)^{(k-3)/2}.$$

The last equation implies that

$$\mathbb{P}\left[\frac{u+1}{2} = x\right] \propto x^{(k-3)/2}(1-x)^{(k-3)/2}.$$

Hence, $\frac{u+1}{2}$ has Beta($\frac{k-1}{2}, \frac{k-1}{2}$) distribution. □

Now we are ready to prove Theorem 5.3.2, which is restated below.

Theorem. *Let $f : \mathbb{R}^d \to \mathbb{R}^m$ be the rescaled JL map defined in Definition 5.5. Given two arbitrary vectors $x, y \in \mathbb{R}^d$, for any $\epsilon, \delta \in (0, 1/2)$, if $m = O(\frac{1}{\epsilon^2})$, then with probability at least $3/4$,*

$$|\,\langle f(x), f(y)\rangle - \langle x, y\rangle\,| \le p(\theta_{x,y}, m)\|x\|_2\|y\|_2, \tag{D.2}$$

where $\theta_{x,y}$ is the angle between x and y, and $p(\cdot,\cdot)$ is defined in Definition 5.3.2.

Proof. We will show that

$$\mathbb{E}[(\langle f(x), f(y)\rangle - \langle x, y\rangle)^2] = p(\theta_{x,y}, m)^2\|x\|_2^2\|y\|_2^2/4. \tag{D.3}$$

If (D.3) holds, then applying Markov's inequality gives the desired bound:

$$\begin{aligned}\mathbb{P}(|\,\langle f(x), f(y)\rangle - \langle x, y\rangle\,| \ge p(\theta_{x,y}, m)\|x\|_2\|y\|_2) &\le \frac{\mathbb{E}[(\langle f(x), f(y)\rangle - \langle x, y\rangle)^2]}{p(\theta_{x,y}, m)^2\|x\|_2^2\|y\|_2^2} \\ &\le \frac{1}{4}. \end{aligned} \tag{D.4}$$

To prove (D.3), note that

1. Since $\langle f(x), f(y)\rangle = \left\langle f(\frac{x}{\|x\|_2}), f(\frac{y}{\|y\|_2})\right\rangle \|x\|_2\|y\|_2$, without loss of generality we can assume $\|x\|_2 = \|y\|_2 = 1$ and prove that

$$\mathbb{E}[(\langle f(x), f(y)\rangle - \langle x, y\rangle)^2] = p(\theta_{x,y}, m)^2/4.$$

2. The random Gaussian matrix G will project x and y on a random subspace, so without loss of generality we can assume that $x = e_1$ and $y = e_1\cos(\theta_{x,y}) + e_2\sin(\theta_{x,y})$, where e_1 and e_2 are the standard basis vectors in $\mathbb{R}^d$.

Let $g_1, g_2 \in \mathbb{R}^k$ be the first two columns of G. Let $u = \left\langle \frac{g_1}{\|g_1\|_2}, \frac{g_2}{\|g_2\|_2}\right\rangle$, then according to Lemma D.3.1, $\frac{u+1}{2} \sim \text{Beta}(\frac{m-1}{2}, \frac{m-1}{2})$ is a random variable with Beta-distribution. Since $x = e_1$ and $y = e_1\cos(\theta_{x,y}) + e_2\sin(\theta_{x,y})$, we can write $\langle Gx, Gy\rangle$ as

$$\begin{aligned}\langle Gx, Gy\rangle &= \langle g_1, g_1\cos(\theta_{x,y}) + g_2\sin(\theta_{x,y})\rangle \\ &= \|g_1\|_2^2\cos(\theta_{x,y}) + \|g_1\|_2\|g_2\|_2 u\sin(\theta_{x,y}).\end{aligned}$$

Besides, we have $\|Gx\|_2 = \|g_1\|_2$ and

$$\begin{aligned}\|Gy\|_2^2 &= \|g_1\cos(\theta_{x,y}) + g_2\sin(\theta_{x,y})\|_2^2 \\ &= \|g_1\|_2^2\cos^2(\theta_{x,y}) + 2\|g_1\|_2\|g_2\|_2 u\cos(\theta_{x,y})\sin(\theta_{x,y}) + \|g_2\|_2^2\sin^2(\theta_{x,y}).\end{aligned}$$

Let $g = \frac{\|g_1\|_2^2}{\|g_2\|_2^2}$, then $g \sim \text{F}(m, m)$ is a random variable with F-distribution. We can write $\langle f(x), f(y)\rangle$ as a function of two random variables g and u:

$$\begin{aligned}\langle f(x), f(y)\rangle &= \frac{\langle Gx, Gy\rangle}{\|Gx\|_2\|Gy\|_2} \\ &= \frac{\sqrt{g}\cos(\theta_{x,y}) + u\sin(\theta_{x,y})}{\sqrt{g\cos^2(\theta_{x,y}) + \sin^2(\theta_{x,y}) + 2u\sqrt{g}\sin(\theta_{x,y})\cos(\theta_{x,y})}}.\end{aligned}$$

Let $\gamma = \langle f(x), f(y) \rangle$. The last step is to write $\mathbb{E}[(\langle f(x), f(y) \rangle - \langle x, y \rangle)^2]$ as an expectation over the two random variables g and u:

$$\begin{aligned}
&\mathbb{E}[(\langle f(x), f(y) \rangle - \langle x, y \rangle)^2] \\
&= \mathbb{E}[\langle f(x), f(y) \rangle^2] - 2\cos(\theta_{x,y})\, \mathbb{E}[\langle f(x), f(y) \rangle] + \cos^2(\theta_{x,y}) \\
&= \mathop{\mathbb{E}}_{g,u} \gamma^2 - 2\cos(\theta_{x,y}) \mathop{\mathbb{E}}_{g,u} \gamma + \cos^2(\theta_{x,y}) \\
&= p(\theta_{x,y}, m)^2/4,
\end{aligned}$$

where the last inequality follows from the definition of $p(\theta, m)$ given in Definition 5.3.2. We have proved (D.3) when x and y are unit-norm vectors. As shown in (D.4), applying the Markov's inequality gives Theorem 5.3.2. □

Bibliography

Achlioptas, D. and McSherry, F. (2001). Fast computation of low rank matrix approximations. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing (STOC)*, pages 611–618. ACM.

Agarwal, A., Negahban, S., and Wainwright, M. J. (2010). Fast global convergence rates of gradient methods for high-dimensional statistical recovery. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 37–45.

Aharon, M., Elad, M., and Bruckstein, A. (2006). k-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322.

Arias-Castro, E., Candès, E. J., and Davenport, M. A. (2013). On the fundamental limits of adaptive sensing. *IEEE Transactions on Information Theory*, 59(1):472–481.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Arora, S., Ge, R., Kannan, R., and Moitra, A. (2012). Computing a nonnegative matrix factorization–provably. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 145–162. ACM.

Arora, S., Khodak, M., Nikunj, S., and Vodrahalli, K. (2018). A compressed sensing view of unsupervised text embeddings, bag-of-n-grams, and lstms. In *International Conference on Learning Representations (ICLR)*.

Ashtiani, H., Ben-David, S., Harvey, N., Liaw, C., Mehrabian, A., and Plan, Y. (2018). Nearly tight sample complexity bounds for learning mixtures of gaussians via sample compression schemes. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3412–3421.

Aurell, E. and Ekeberg, M. (2012). Inverse ising inference using all the data. *Physical review letters*, 108(9):090201.

Babbar, R. and Schölkopf, B. (2017). Dismec: distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 721–729. ACM.

Bah, B., Sadeghian, A., and Cevher, V. (2013). Energy-aware adaptive bi-lipschitz embeddings. In *Proceedings of the 10th International Conference on Sampling Theory and Applications*.

Baldassarre, L., Li, Y.-H., Scarlett, J., Gözcü, B., Bogunovic, I., and Cevher, V. (2016). Learning-based compressive subsampling. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):809–822.

Balsubramani, A., Dasgupta, S., and Freund, Y. (2013). The fast convergence

of incremental pca. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3174–3182.

Bandeira, A. S., Dobriban, E., Mixon, D. G., and Sawin, W. F. (2013). Certifying the restricted isometry property is hard. *IEEE Transactions on Information Theory*, 59(6):3448–3450.

Banerjee, O., Ghaoui, L. E., and d'Aspremont, A. (2008). Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine learning research*, 9(Mar):485–516.

Baraniuk, R. G., Cevher, V., Duarte, M. F., and Hegde, C. (2010). Model-based compressive sensing. *IEEE Transactions on Information Theory*, 56(4):1982–2001.

Bartlett, P. L. and Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482.

Ben-Tal, A. and Nemirovski, A. (Fall 2013). Lectures on modern convex optimization. https://www2.isye.gatech.edu/~nemirovs/Lect_ModConvOpt.pdf.

Bento, J. and Montanari, A. (2009). Which graphical models are difficult to learn? In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1303–1311.

Bhatia, K., Dahiya, K., Jain, H., Prabhu, Y., and Varma, M. (2017). The extreme classification repository: Multi-label datasets and code. http://manikvarma.org/downloads/XC/XMLRepository.html.

Bhatia, K., Jain, H., Kar, P., Varma, M., and Jain, P. (2015). Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 730–738.

Bhojanapalli, S., Jain, P., and Sanghavi, S. (2015). Tighter low-rank approximation via sampling the leveraged element. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 902–920. SIAM.

Blumensath, T. and Davies, M. E. (2009). Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274.

Bora, A., Jalal, A., Price, E., and Dimakis, A. G. (2017). Compressed sensing using generative models. In *International Conference on Machine Learning (ICML)*, pages 537–546.

Boufounos, P. T. (2013). Angle-preserving quantized phase embeddings. In *SPIE Optical Engineering+ Applications*. International Society for Optics and Photonics.

Boutsidis, C., Garber, D., Karnin, Z., and Liberty, E. (2015). Online principal

components analysis. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 887–901.

Boutsidis, C. and Gittens, A. (2013). Improved matrix algorithms via the subsampled randomized hadamard transform. *SIAM Journal on Matrix Analysis and Applications*, 34(3):1301–1340.

Boyd, S. (2014). Subgradient methods. *Notes for EE364b, Stanford University, Spring 2013–14*.

Bresler, G. (2015). Efficiently learning ising models on arbitrary graphs. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing (STOC)*, pages 771–782. ACM.

Bubeck, S. (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357.

Candès, E. J. (2008). The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9-10):589–592.

Candès, E. J. and Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772.

Candès, E. J., Romberg, J., and Tao, T. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509.

Chen, X., Liu, H., and Carbonell, J. G. (2012). Structured sparse canonical correlation analysis. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 199–207.

Chen, Y., Bhojanapalli, S., Sanghavi, S., and Ward, R. (2015). Completing any low-rank matrix, provably. *The Journal of Machine Learning Research*, 16(1):2999–3034.

Cho, Y. and Saul, L. K. (2009). Kernel methods for deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 342–350.

Choi, M. J., Lim, J. J., Torralba, A., and Willsky, A. S. (2010). Exploiting hierarchical context on a large database of object categories. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 129–136. IEEE.

Clarkson, K. L. and Woodruff, D. P. (2009). Numerical linear algebra in the streaming model. In *Proceedings of the forty-first annual ACM symposium on Theory of computing (STOC)*, pages 205–214. ACM.

Clarkson, K. L. and Woodruff, D. P. (2013). Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing (STOC)*, pages 81–90. ACM.

Cohen, M. B., Nelson, J., and Woodruff, D. P. (2016). Optimal approximate matrix product in terms of stable rank. In *43rd International Colloquium on*

Automata, Languages, and Programming (ICALP 2016). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Daskalakis, C., Gouleakis, T., Tzamos, C., and Zampetakis, M. (2018). Efficient statistics, in high dimensions, from truncated samples. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 639–649. IEEE.

Deshpande, A. and Vempala, S. (2006). Adaptive sampling and fast low-rank matrix approximation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 292–303. Springer.

Devroye, L., Mehrabian, A., and Reddad, T. (2018). The total variation distance between high-dimensional gaussians. *arXiv preprint arXiv:1810.08693*.

Donoho, D. and Stodden, V. (2004). When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1141–1148.

Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306.

Donoho, D. L. and Elad, M. (2003). Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202.

Donoho, D. L., Maleki, A., and Montanari, A. (2009). Message-passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences*, 106(45):18914–18919.

Donoho, D. L. and Tanner, J. (2005). Sparse nonnegative solution of underdetermined linear equations by linear programming. *Proceedings of the National Academy of Sciences of the United States of America*, 102(27):9446–9451.

Drineas, P., Kannan, R., and Mahoney, M. W. (2006a). Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, 36(1):158–183.

Drineas, P., Mahoney, M. W., and Muthukrishnan, S. (2006b). Subspace sampling and relative-error matrix approximation: Column-based methods. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 316–326. Springer.

Duchi, J. (March 13, 2019). Lecture notes for statistics 311/electrical engineering 377. https://stanford.edu/class/stats311/lecture-notes.pdf.

Eagle, N., Pentland, A. S., and Lazer, D. (2009). Inferring friendship network structure by using mobile phone data. *Proceedings of the national academy of sciences*, 106(36):15274–15278.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

Frieze, A., Kannan, R., and Vempala, S. (2004). Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041.

Ge, R., Kuditipudi, R., Li, Z., and Wang, X. (2019). Learning two-layer neural networks with symmetric inputs. In *International Conference on Learning Representations (ICLR)*.

Gittens, A., Devarakonda, A., Racah, E., Ringenburg, M., Gerhardt, L., Kottalam, J., Liu, J., Maschhoff, K., Canon, S., Chhugani, J., et al. (2016). Matrix factorizations at scale: A comparison of scientific data analytics in spark and c+ mpi using three case studies. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 204–213. IEEE.

Goel, S., Klivans, A., and Meka, R. (2018). Learning one convolutional layer with overlapping patches. In *International Conference on Machine Learning (ICML)*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems (NeurIPS)*, pages 2672–2680.

Gözcü, B., Mahabadi, R. K., Li, Y.-H., Ilıcak, E., Çukur, T., Scarlett, J., and Cevher, V. (2018). Learning-based compressive mri. *IEEE Transactions on Medical Imaging*, 37(6):1394–1406.

Grathwohl, W., Chen, R. T., Betterncourt, J., Sutskever, I., and Duvenaud, D. (2019). Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations (ICLR)*.

Gregor, K. and LeCun, Y. (2010). Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 399–406.

Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

Hamilton, L., Koehler, F., and Moitra, A. (2017). Information theoretic properties of markov random fields, and their algorithmic applications. In *Advances in Neural Information Processing Systems*, pages 2463–2472.

Har-Peled, S. (2014). Low rank matrix approximation in linear time. *arXiv preprint arXiv:1410.8802*.

Hariharan, B., Vishwanathan, S., and Varma, M. (2012). Efficient max-margin multi-label classification with applications to zero-shot learning. *Machine learning*, 88(1-2):127–155.

Harva, M. and Kabán, A. (2007). Variational learning for rectified factor analysis. *Signal Processing*, 87(3):509–527.

Haviv, I. and Regev, O. (2017). The restricted isometry property of subsampled fourier matrices. In *Geometric Aspects of Functional Analysis*, pages 163–179. Springer.

He, H., Xin, B., Ikehata, S., and Wipf, D. (2017). From bayesian sparsity to gated recurrent nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5560–5570.

Hegde, C., Indyk, P., and Schmidt, L. (2014). Approximation-tolerant model-based compressive sensing. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 1544–1561. SIAM.

Hegde, C., Sankaranarayanan, A. C., Yin, W., and Baraniuk, R. G. (2015). Numax: A convex approach for learning near-isometric linear embeddings. *IEEE Transactions on Signal Processing*, 63(22):6109–6121.

Hershey, J. R., Roux, J. L., and Weninger, F. (2014). Deep unfolding: Model-based inspiration of novel deep architectures. *arXiv preprint arXiv:1409.2574*.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 448–456.

Jain, H., Prabhu, Y., and Varma, M. (2016). Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 935–944. ACM.

Jain, P., Netrapalli, P., and Sanghavi, S. (2013). Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing (STOC)*, pages 665–674. ACM.

Jalali, A., Ravikumar, P., Vasuki, V., and Sanghavi, S. (2011). On learning discrete graphical models using group-sparse regularization. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 378–387.

Jasinska, K., Dembczynski, K., Busa-Fekete, R., Pfannschmidt, K., Klerx, T., and Hullermeier, E. (2016). Extreme f-measure maximization using sparse probability estimates. In *International Conference on Machine Learning (ICML)*, pages 1435–1444.

Jegou, H., Douze, M., and Schmid, C. (2011). Product quantization for nearest neighbor search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(1):117–128.

Jernite, Y., Choromanska, A., and Sontag, D. (2017). Simultaneous learning of trees and representations for extreme classification and density estimation. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1665–1674.

Jin, K. H., McCann, M. T., Froustey, E., and Unser, M. (2017). Deep convolutional neural network for inverse problems in imaging. *IEEE Transactions on Image Processing*, 26(9):4509–4522.

Johnson, W. B. and Lindenstrauss, J. (1984). Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics*, 26:189–206.

Kakade, S. M., Shalev-Shwartz, S., and Tewari, A. (2012). Regularization techniques for learning with matrices. *Journal of Machine Learning Research*, 13(Jun):1865–1890.

Kakade, S. M., Sridharan, K., and Tewari, A. (2009). On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. In *Advances in neural information processing systems (NeurIPS)*, pages 793–800.

Kannan, R., Vempala, S. S., and Woodruff, D. P. (2014). Principal component analysis and higher correlations for distributed data. In *Proceedings of The 27th Conference on Learning Theory (COLT)*, pages 1040–1057.

Khajehnejad, M. A., Dimakis, A. G., Xu, W., and Hassibi, B. (2011). Sparse recovery of nonnegative signals with minimal expansion. *IEEE Transactions on Signal Processing*, 59(1):196–208.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *The International Conference on Learning Representations (ICLR)*.

Klivans, A. R. and Meka, R. (2017). Learning graphical models using multiplicative weights. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 343–354. IEEE.

Koh, K., Kim, S.-J., and Boyd, S. (2007). An interior-point method for large-scale ℓ_1-regularized logistic regression. *Journal of Machine learning research*, 8(Jul):1519–1555.

Kulkarni, K., Lohit, S., Turaga, P., Kerviche, R., and Ashok, A. (2016). Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 449–458.

Lee, S.-I., Ganapathi, V., and Koller, D. (2007). Efficient structure learning of markov networks using l_1-regularization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 817–824.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Li, C. J., Wang, M., and Zhang, T. (2017). Diffusion approximations for online principal component estimation and global convergence. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Li, Y. and Yuan, Y. (2017). Convergence analysis of two-layer neural networks

with relu activation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 597–607.

Li, Y.-H. and Cevher, V. (2016). Learning data triage: linear decoding works for compressive mri. In *ICASSP*.

Lichman, M. (2013). UCI machine learning repository. http://archive.ics.uci.edu/ml.

Liu, J., Chang, W.-C., Wu, Y., and Yang, Y. (2017). Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124. ACM.

Lohit, S., Kulkarni, K., Kerviche, R., Turaga, P., and Ashok, A. (2018). Convolutional neural networks for non-iterative reconstruction of compressively sensed images. *IEEE Transactions on Computational Imaging*.

Lokhov, A. Y., Vuffray, M., Misra, S., and Chertkov, M. (2018). Optimal structure and parameter learning of ising models. *Science advances*, 4(3):e1700791.

Ma, J., Saul, L. K., Savage, S., and Voelker, G. M. (2009). Identifying suspicious urls: an application of large-scale online learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 681–688. ACM.

Ma, Z., Lu, Y., and Foster, D. (2015). Finding linear structure in large datasets with scalable canonical correlation analysis. In *International Conference on Machine Learning (ICML)*, pages 169–178.

Magen, A. and Zouzias, A. (2011). Low rank matrix-valued chernoff bounds and approximate matrix multiplication. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1422–1436. SIAM.

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2009). Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 689–696. ACM.

Malloy, M. L. and Nowak, R. D. (2014). Near-optimal adaptive compressed sensing. *IEEE Transactions on Information Theory*, 60(7):4001–4012.

Marbach, D., Costello, J. C., Küffner, R., Vega, N. M., Prill, R. J., Camacho, D. M., Allison, K. R., Consortium, T. D., Kellis, M., Collins, J. J., and Stolovitzky, G. (2012). Wisdom of crowds for robust gene network inference. *Nature methods*, 9(8):796.

Mardani, M., Monajemi, H., Papyan, V., Vasanawala, S., Donoho, D., and Pauly, J. (2017). Recurrent generative adversarial networks for proximal learning and automated compressive image recovery. *arXiv preprint arXiv:1711.10046*.

Mazumdar, A. and Rawat, A. S. (2019). Learning and recovery in the relu model. In *Proceedings of 57th Annual Allerton Conference on Communication, Control, and Computing, 2019*.

Mika, S., Schölkopf, B., Smola, A. J., Müller, K.-R., Scholz, M., and Rätsch, G. (1999). Kernel pca and de-noising in feature spaces. In *Advances in neural information processing systems (NeurIPS)*, pages 536–542.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems (NeurIPS)*, pages 3111–3119.

Mineiro, P. and Karampatziakis, N. (2015). Fast label embeddings via randomized linear algebra. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 37–51. Springer.

Mitliagkas, I., Caramanis, C., and Jain, P. (2013). Memory limited, streaming pca. In *Advances in Neural Information Processing Systems*, pages 2886–2894.

Mousavi, A. and Baraniuk, R. G. (2017). Learning to invert: Signal recovery via deep convolutional networks. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Mousavi, A., Dasarathy, G., and Baraniuk, R. G. (2017). Deepcodec: Adaptive sensing and recovery via deep convolutional neural networks. In *55th Annual Allerton Conference on Communication, Control and Computing*.

Mousavi, A., Dasarathy, G., and Baraniuk, R. G. (2019). A data-driven and distributed approach to sparse signal representation and recovery. In *International Conference on Learning Representations (ICLR)*.

Mousavi, A., Patel, A. B., and Baraniuk, R. G. (2015). A deep learning approach to structured signal recovery. In *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*, pages 1336–1343. IEEE.

Needell, D. and Tropp, J. A. (2009). Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321.

Negahban, S. N., Ravikumar, P., Wainwright, M. J., and Yu, B. (2012). A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. *Statistical Science*, 27(4):538–557.

Nguyen, N. H., Do, T. T., and Tran, T. D. (2009). A fast and efficient algorithm for low-rank approximation of a matrix. In *Proceedings of the 41st annual ACM symposium on Theory of computing (STOC)*, pages 215–224. ACM.

Nguyen, T. V., Wong, R. K., and Hegde, C. (2018). Autoencoders learn generative linear models. *arXiv preprint arXiv:1806.00572*.

Nicolaescu, L. I. (September 9, 2018). Lectures on the geometry of manifolds. https://www3.nd.edu/~lnicolae/Lectures.pdf.

Niculescu-Mizil, A. and Abbasnejad, E. (2017). Label filters for large scale multilabel classification. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1448–1457.

Olshausen, B. A. and Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607.

Pan, V. Y. (2016). How bad are vandermonde matrices? *SIAM Journal on Matrix Analysis and Applications*, 37(2):676–694.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Prabhu, Y., Kag, A., Gopinath, S., Dahiya, K., Harsola, S., Agrawal, R., and Varma, M. (2018a). Extreme multi-label learning with label features for warm-start tagging, ranking & recommendation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 441–449. ACM.

Prabhu, Y., Kag, A., Harsola, S., Agrawal, R., and Varma, M. (2018b). Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 993–1002.

Prabhu, Y. and Varma, M. (2014). Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 263–272. ACM.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Rauhut, H. (2010). Compressive sensing and structured random matrices. *Theoretical Foundations and Numerical Methods for Sparse Recovery*, 9:1–92.

Ravikumar, P., Wainwright, M. J., and Lafferty, J. D. (2010). High-dimensional ising model selection using ℓ_1-regularized logistic regression. *The Annals of Statistics*, 38(3):1287–1319.

Rigollet, P. and Hütter, J.-C. (Spring 2017). Lectures notes on high dimensional statistics. http://www-math.mit.edu/~rigollet/PDFs/RigNotes17.pdf.

Santhanam, N. P. and Wainwright, M. J. (2012). Information-theoretic limits of selecting binary graphical models in high dimensions. *IEEE Transactions on Information Theory*, 58(7):4117–4134.

Sarlos, T. (2006). Improved approximation algorithms for large matrices via random projections. In *IEEE 47th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 143–152. IEEE.

Seeger, M. W. and Nickisch, H. (2011). Large scale bayesian inference and experimental design for sparse linear models. *SIAM Journal on Imaging Sciences*, 4(1):166–199.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge University Press.

Shamir, O. (2015). A stochastic pca and svd algorithm with an exponential convergence rate. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 144–152.

Shi, W., Jiang, F., Zhang, S., and Zhao, D. (2017). Deep networks for compressed image sensing. In *Multimedia and Expo (ICME), 2017 IEEE International Conference on*, pages 877–882. IEEE.

Soltanolkotabi, M. (2017). Learning relus via gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2007–2017.

Sprechmann, P., Bronstein, A. M., and Sapiro, G. (2015). Learning efficient sparse and low rank models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1821–1833.

Tagami, Y. (2017). Annexml: Approximate nearest neighbor search for extreme multi-label classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 455–464. ACM.

Tropp, J. A. (2011). Improved analysis of the subsampled randomized hadamard transform. *Advances in Adaptive Data Analysis*, pages 115–126.

Tropp, J. A. and Gilbert, A. C. (2007). Signal recovery from partial information via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666.

Tsybakov, A. B. (2009). *Introduction to nonparametric estimation*. Springer.

Van Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *International Conference on Machine Learning (ICML)*, pages 1747–1756.

Vavasis, S. A. (2009). On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408.

Von Neumann, J. (1949). On rings of operators. reduction theory. *Annals of Mathematics*, pages 401–485.

Vuffray, M., Misra, S., Lokhov, A., and Chertkov, M. (2016). Interaction screening: Efficient and sample-optimal learning of ising models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2595–2603.

Vuffray, M., Misra, S., and Lokhov, A. Y. (2019). Efficient learning of discrete graphical models. *arXiv preprint arXiv:1902.00600*.

Wainwright, M. J. (2019). *High-dimensional statistics: A non-asymptotic viewpoint*. Cambridge University Press.

Wang, Z., Ling, Q., and Huang, T. (2016). Learning deep l0 encoders. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Williamson, D. P. and Shmoys, D. B. (2011). *The design of approximation algorithms*. Cambridge University Press.

Woodruff, D. P. et al. (2014). Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157.

Woolfe, F., Liberty, E., Rokhlin, V., and Tygert, M. (2008). A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25(3):335–366.

Wu, S., Bhojanapalli, S., Sanghavi, S., and Dimakis, A. G. (2016). Single pass pca of matrix products. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2585–2593.

Wu, S., Dimakis, A. G., and Sanghavi, S. (2019a). Learning distributions generated by one-layer relu networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Wu, S., Dimakis, A. G., Sanghavi, S., Yu, F. X., Holtmann-Rice, D., Storcheus, D., Rostamizadeh, A., and Kumar, S. (2019b). Learning a compressed sensing measurement matrix via gradient unrolling. In *International Conference on Machine Learning (ICML)*, pages 6828–6839.

Wu, S. and Lindgren, E. (2016). Rescaled JL embedding. *Course Project Report for 2016 Fall CS 395T Sublinear Algorithm.*

Wu, S., Sanghavi, S., and Dimakis, A. G. (2019c). Sparse logistic regression learns all discrete pairwise graphical models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Xin, B., Wang, Y., Gao, W., Wipf, D., and Wang, B. (2016). Maximal sparsity with deep networks? In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4340–4348.

Yang, E., Allen, G., Liu, Z., and Ravikumar, P. K. (2012). Graphical models via generalized linear models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1358–1366.

Yen, I. E., Huang, X., Dai, W., Ravikumar, P., Dhillon, I., and Xing, E. (2017). Ppdsparse: A parallel primal-dual sparse method for extreme classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 545–553. ACM.

Yen, I. E.-H., Huang, X., Ravikumar, P., Zhong, K., and Dhillon, I. (2016). Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *International Conference on Machine Learning (ICML)*, pages 3069–3077.

Yu, H.-F., Jain, P., Kar, P., and Dhillon, I. (2014). Large-scale multi-label learning with missing labels. In *International Conference on Machine Learning (ICML)*, pages 593–601.

Yuan, M. and Lin, Y. (2007). Model selection and estimation in the gaussian graphical model. *Biometrika*, 94(1):19–35.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., and Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*.

Zhang, J. and Ghanem, B. (2018). Ista-net: Interpretable optimization-inspired deep network for image compressive sensing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1828–1837.

Zhong, K., Song, Z., Jain, P., Bartlett, P. L., and Dhillon, I. S. (2017). Recovery guarantees for one-hidden-layer neural networks. In *International Conference on Machine Learning (ICML)*, pages 4140–4149.

Vita

Shanshan Wu is a Ph.D. candidate in the Department of Electrical and Computer Engineering of the University of Texas at Austin. She received the B.S. degree in Electrical and Computer Engineering from Shanghai Jiao Tong University in 2011, and the M.S. degree in Electronics Science and Technology from Shanghai Jiao Tong University in 2014. She worked as Applied Scientist Intern at Amazon, East Palo Alto in Spring 2017, and as Software Engineer Intern at Google Research, New York City in Summer 2017 and 2018. Her research interests are in the broad areas of large-scale machine learning and optimization.

Permanent address: wushanshan0701@gmail.com

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.