2019

Provable algorithms for nonlinear models in machine learning and signal processing

Mohammadreza Soltani
Iowa State University

Provable algorithms for nonlinear models in machine learning and signal processing

by

Mohammadreza Soltani

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Electrical Engineering (Communications and Signal Processing)

Iowa State University

Ames, Iowa

2019

DEDICATION

I would like to thank a special group of people that are the real reasons for my continued good fortune and dedicate this thesis to them: my supportive, kind, and patient dad and mom, Arshad and Rokhsareh, my lovely grandmother, Tooran, my great brothers, Amirreza and Hamidreza, and last but not least to my beautiful girlfriend, Mahdiyeh without whose support I would not have been able to complete this work and any success of mine would be impossible without their constant love and encouragement. I would also like to thank my friends and colleagues during my Ph.D.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Chinmay Hegde for his guidance, patience and support throughout this research and the writing of this thesis. His insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would also like to thank my committee members for their efforts and contributions to this work: Prof. Zhengdao Wang, Prof. Namrata Vaswani, Prof. Jin Tian, and Prof. Srikanta Tirthapura. I would additionally like to thank Prof. Hamid Sharif for his guidance throughout the initial stages of my graduate career and for his support, time, and consideration. I am very grateful to him for his valuable advices which helped me to overcome numerous obstacles I have been facing through my graduate study in the U.S. Furthermore, I want to give my special thank to Technicolor AI Research Lab, to Prof. Jean Bolot, director of the research lab, and Dr. Swayambhoo Jain, the researcher in Technicolor who provided an internship opportunity for me. The research conducted during the writing of this thesis was supported in part by grants CCF-1566281 and CCF-1750920 awarded by the National Science Foundation. Also, chapter 3 of this thesis is related to the project I have accomplished in Technicolor AI Research Lab from May 2018 to Dec 2018.

ABSTRACT

In numerous signal processing and machine learning applications, the problem of signal recovery from a limited number of nonlinear observations is of special interest.

These problems also called inverse problem have recently received attention in signal processing, machine learning, and high-dimensional statistics. In high-dimensional setting, the inverse problems are inherently ill-posed as the number of measurements is typically less than the number of dimensions. As a result, one needs to assume some structures on the underlying signal such as sparsity, structured sparsity, low-rank and so on. In addition, having a nonlinear map from the signal space to the measurement space may add more challenges to the problem. For instance, the assumption on the nonlinear function such as known/unknown, invertibility, smoothness, even/odd, and so on can change the tractability of the problem dramatically. The nonlinear inverse problems are also a special interest in the context of neural network and deep learning as each layer can be cast as an instance of the inverse problem. As a result, understanding of an inverse problem can serve as a building block for more general and complex networks. In this thesis, we study various aspects of such inverse problems with focusing on the underlying signal structure, the compression modes, the nonlinear map from signal space to measurement space, and the connection of the inverse problems to the analysis of some class of neural networks. In this regard, we try to answer statistical properties and computational limits of the proposed methods, and compare them to the state-of-the-art approaches.

First, we start with the superposition signal model in which the underlying signal is assumed to be the superposition of two components with sparse representation (i.e., their support is arbitrary sparse) in some specific domains. Initially, we assume that the nonlinear function also called link function is not known. Then, the goal is defined as recovering the components of the superposition signal from the nonlinear observation model. This problem which is called signal demixing is of

special importance in several applications ranging from astronomy to computer vision. Our first contribution is a simple, fast algorithm that recovers the component signals from the nonlinear measurements. We support our algorithm with rigorous theoretical analysis and provide upper bounds on the estimation error as well as the sample complexity of demixing the components (up to a scalar ambiguity). Next, we remove the assumption on the link function and studied the same problem when the link function is known and monotonic, but the observation is corrupted by some additive noise. We proposed an algorithm under this setup for recovery of the components of the superposition signal, and derive nearly-tight upper bounds on the sample complexity of the algorithm to achieve stable recovery of the components. Moreover, we showed that the algorithm enjoys a linear convergence rate. Chapter 1 includes this part.

In chapter 2, we target two assumptions made in the first chapter: the first assumption which is concerned about the underlying signal model considers the case that the constituent components have arbitrary sparse representations in some incoherent domains. While having arbitrary sparse support can be a good way of modeling of many natural signals, it is just a simple and not realistic assumption. Many real signals such as natural images show some specific structure on their support. That is, when they are represented in a specific domain, their support comprises non-zero coefficients which are grouped or classified in a specific pattern. For instance, it is well-known that many natural images show so-called *tree sparsity* structure when they are represented in the wavelet domains. This motivates us to study other signal models in the context of our demixing problem introduced in chapter 1. In particular, we study certain families of structured sparsity models in the constituent components and propose a method which provably recovers the components given (nearly) $\mathcal{O}(s)$ samples where s denotes the sparsity level of the underlying components. This strictly improves upon previous nonlinear demixing techniques and asymptotically matches the best possible sample complexity. The second assumption we made in the first chapter is about having a smooth monotonic nonlinear map for the case of known link function. In chapter 2, we go beyond this assumption, and we study the bigger class of nonlinear link functions and consider the demixing problem from a limited number of nonlinear observations where this nonlinearity is due to either periodic function or aperiodic one. For both of these considerations, we propose new robust algorithms and equip them with statistical analysis.

In chapter 3, we continue our investigation about choosing a proper underlying signal model in the demixing framework. In the first two chapters, our methods for modeling the underlying signals were based on a *hard-coded* approach. That is, we assume some prior knowledge in the signal domain and exploit the structure of this prior in designing efficient algorithms. However, many real signals including natural images have a more complicated structure than just simple sparsity (arbitrary or structured). Towards choosing a proper structure, some research directions try to automate the process of choosing prior knowledge on the underlying signal by learning them through a lot of training samples. Given the success of deep learning for approximating the distribution of complex signals, in chapter 3, we apply deep learning techniques to model the low-dimension structure of the constituent components and consequently, estimating these components from their superposition. As illustrated through extensive numerical experiments, we show that this approach is able to learn the structure of the constituent components in our demixing problem. Our approach in this chapter is empirical, and we defer more theoretical investigation of the proposed method as our future work.

In chapter 4, we study another low-dimension signal model. In particular, we focus on the common low-rank matrix model as our underlying structure. In this case, our interest quantity to estimate (recover) is a low-rank matrix. In this regard, we focus on studying of optimizing a convex function over the set of matrices, subject to rank constraints. Recently, different algorithms have been proposed for the low-rank matrix estimation problem. However, existing first-order methods for solving such problems either are too slow to converge, or require multiple invocations of singular value decompositions.

On the other hand, factorization-based non-convex algorithms, while being much faster, and has a provable guarantee, require stringent assumptions on the condition number of the optimum. Here, we provide a novel algorithmic framework that achieves the best of both worlds: as fast as factorization methods, while requiring no dependency on the condition number. We instantiate our general framework for three important and practical applications; nonlinear affine rank min-

imization (NLARM), Logistic PCA, and precision matrix estimation (PME) in the probabilistic graphical model. We then derive explicit bounds on the sample complexity as well as the running time of our approach and show that it achieves the best possible bounds for both cases. We also provide an extensive range of experimental results for all of these applications.

Finally, we extend our understanding of nonlinear models to the problem of learning neural network in chapter 5. In particular, we shift gear to study the problem of (provably) learning the weights of a two-layer neural network with quadratic activations (sometimes called shallow networks). Our shallow network comprises of the input layer, one hidden layer, and the output layer with a single neuron. We focus on the under-parametrized regime where the number of neurons in the hidden layer is (much) smaller than the dimension of the input. Our approach uses a lifting trick, which enables us to borrow algorithmic ideas from low-rank matrix estimation (fourth chapter). In this context, we propose three novel, non-convex training algorithms which do not need any extra tuning parameters other than the number of hidden neurons. We support our algorithms with rigorous theoretical analysis and show that the proposed algorithms enjoy linear convergence, fast running time per iteration, and near-optimal sample complexity. We complement our theoretical results with several numerical experiments.

While we have tried to be consistent in the mathematical notations throughout this thesis, each chapter should be treated independently regarding some mathematical notation. Hence, we have provided the notations being used in each chapter to prevent any possible confusion.

CHAPTER 1. FAST ALGORITHMS FOR DEMIXING SPARSE SIGNALS FROM NONLINEAR OBSERVATIONS

In this chapter, we study the problem of *demixing* a pair of sparse signals from nonlinear observations of their superposition. Mathematically, we consider a nonlinear signal observation model, $y_i = g(a_i^T x) + e_i, \ i = 1, \ldots, m$, where $x = \Phi w + \Psi z$ denotes the superposition signal, Φ and Ψ are orthonormal bases in $\mathbb{R}^n$, and $w, z \in \mathbb{R}^n$ are sparse coefficient vectors of the constituent signals. In this chapter, we assume that the support of these vectors are arbitrary sparse. In the next chapters, we will consider other structured patterns. Further, we assume that the observations are corrupted by a subgaussian additive noise. Within this model, g represents a nonlinear *link* function, and $a_i \in \mathbb{R}^n$ is the i-th row of the measurement matrix, $A \in \mathbb{R}^{m \times n}$. Problems of this nature arise in several applications ranging from astronomy, computer vision, and machine learning. We make some concrete algorithmic progress for the above demixing problem. Specifically, we consider two scenarios: (i) the case when the demixing procedure has no knowledge of the link function, and (ii) the case when the demixing algorithm has perfect knowledge of the link function. In both cases, we provide fast algorithms for recovery of the constituents w and z from the observations. Moreover, we support these algorithms with a rigorous theoretical analysis, and derive (nearly) tight upper bounds on the sample complexity of the proposed algorithms for achieving stable recovery of the component signals. Our analysis also shows that the running time of our algorithms is essentially as good as the best possible.

We also provide a range of numerical simulations to illustrate the performance of the proposed algorithms on both real and synthetic signals and images. Our simulations show the superior performance of our algorithms compared to existing methods for demixing signals and images based on convex optimization. In particular, our proposed methods yield demonstrably better sample complexities as well as improved running times, thereby enabling their applicability to large-scale problems.

1.1 Introduction

1.1.1 Setup

In numerous signal processing applications, the problem of *demixing* is of special interest. In simple terms, demixing involves disentangling two (or more) constituent signals from observations of their linear superposition. Formally, consider a discrete-time signal $x \in \mathbb{R}^n$ that can be expressed as the superposition of two signals:

$$x = \Phi w + \Psi z \,,$$

where Φ and Ψ are orthonormal bases of $\mathbb{R}^n$, and $w, z \in \mathbb{R}^n$ are the corresponding basis coefficients. The goal of signal demixing, in this context, is to reliably recover the constituent signals (equivalently, their basis representations w and z) from the superposition signal x.

Demixing suffers from a fundamental *identifiability* issue since the number of unknowns ($2n$) is greater than the number of observations (n). This is easy to see: suppose for simplicity that $\Phi = \Psi = I_n$, the canonical basis of $\mathbb{R}^n$, and therefore, $x = w + z$. Now, suppose that both w and z have only one nonzero entry in the first coordinate. Then, there is an infinite number of w and z that are consistent with the observations x, and any hope of recovering the true components is lost. Therefore, for the demixing problem to have an identifiable solution, one inevitably has to assume some type of *incoherence* between the constituent signals (or more specifically, between the corresponding bases Φ and Ψ) (Elad et al., 2005; Donoho et al., 2006). Such an incoherence assumption certifies that the components are sufficiently "distinct" and that the recovery problem is well-posed. Please see Section 1.3 for a formal definition of incoherence.

However, even if we assume that the signal components are sufficiently incoherent, demixing poses additional challenges under stringent observation models.

Suppose, now, that we only have access to undersampled linear measurements of the signal, i.e., we record:

$$y = Ax\,, \tag{1.1}$$

where $A \in \mathbb{R}^{m\times n}$ denotes the measurement operator and where $m < n$. In this scenario, the demixing problem is further confounded by the fact that A possesses a nontrivial null space. In this case, it might seem impossible to recover the components x and z since A possesses a nontrivial null space. Once again, this problem is highly ill-posed and further structural assumptions on the constituent signals are necessary. Under-determined problems of this kind have recently received significant attention in signal processing, machine learning, and high-dimensional statistics. In particular, the emergent field of *compressive sensing* (Candès, 2006; Donoho, 2006) shows that it is indeed possible to exactly reconstruct the underlying signals under certain assumptions on x, provided the measurement operator is designed carefully. This intuition has enabled the design of a wide range of efficient architectures for signal acquisition and processing (Wakin et al., 2006; Mishali and Eldar, 2010).

In this chapter, we address an *even* more challenging question in the demixing context. Mathematically, we consider a *noisy, nonlinear* signal observation model, formulated as follows:

$$y_i = g(\langle a_i, \Phi w + \Psi z\rangle) + e_i,\ \ i = 1, \ldots, m\,. \tag{1.2}$$

Here, as before, the superposition signal is modeled as $x = \Phi w + \Psi z$. Each observation is generated by the composition of a linear functional of the signal $\langle a_i, x\rangle$, with a (scalar) nonlinear function g. Here, g is sometimes called a *link* or *transfer* function, and a_i denotes the i^{th} row of a linear measurement matrix $A \in \mathbb{R}^{m\times n}$. For full generality, in (1.2) we assume that each observation y_i is corrupted by additive noise; the noiseless case is realized by setting $e_i = 0$. We will exclusively consider the "measurement-poor" regime where the number of observations m is much smaller than the ambient dimension n.

For all the reasons detailed above, the problem of recovering the coefficient vectors w and z from the measurements y seems daunting. Therefore, we make some structural assumptions. Particularly, we assume that w and z are *s-sparse* (i.e., they contain no more than s nonzero entries). Further, we will assume perfect knowledge of the bases Φ and Ψ, and the measurement matrix A. The noise vector $e \in \mathbb{R}^m$ is assumed to be stochastic, zero mean, and bounded. Under these assumptions, we will see that it is indeed possible to stably recover the coefficient vectors, with a number of observations that is proportional to the sparsity level s, as opposed to the ambient dimension n.

The nonlinear link function g plays a crucial role in our algorithm development and analysis. In signal processing applications, such nonlinearities may arise due to imperfections caused during a measurement process, or inherent limitations of the measurement system, or due to quantization or calibration errors. We discuss such practical implications more in detail below. On an abstract level, we consider two distinct scenarios. In the first scenario, the link function may be non-smooth, non-invertible, or even unknown to the recovery procedure. This is the more challenging case, but we will show that recovery of the components is possible even without knowledge of g. In the second scenario; the link function is a known, smooth, and strictly monotonic function. This is the somewhat simpler case, and we will see that this leads to significant improvements in recovery performance both in terms of theory and practice.

1.1.2 Our Contributions

We specifically make some concrete algorithmic progress in the demixing problem under nonlinear observations. In particular, we study the following scenarios depending on certain additional assumptions made on (1.2):

1. **Unknown** g. We first consider the (arguably, more general) scenario where the nonlinear link function g may be non-smooth, non-invertible, or even unknown. In this setting, we do not explicitly model the additive noise term in (1.2). For such settings, we introduce a novel demixing algorithm that is non-iterative, does not require explicit knowledge of the link function g, and produces an estimate of the signal components. We call this algorithm OneShot to emphasize its non-iterative nature. It is assumed that OneShot possess oracle knowledge of the measurement matrix A, and orthonormal bases Φ and Ψ.

We supplement our proposed algorithm with a rigorous theoretical analysis and derive upper bounds on the *sample complexity* of demixing with nonlinear observations. In particular, we prove that the sample complexity of ONESHOT to achieve an estimation error κ is given by $m = \mathcal{O}(\frac{1}{\kappa^2} s \log \frac{n}{s})$ provided that the entries of the measurement matrix are i.i.d. standard normal random variables.

2. **Known** g. Next, we consider the case where the nonlinear link function g is known, smooth, and monotonic. In this setting, the additive noise term in (1.2) is assumed to be bounded either absolutely, or with high probability. For such (arguably, milder) settings, we provide an iterative algorithm for demixing of the constituent signals in (1.2) given the nonlinear observations y. We call this algorithm DEMIXING WITH HARD THRESHOLDING, or DHT for short. In addition to knowledge of g, we assume that DHT possesses oracle knowledge of A, Φ, and Ψ.

   Within this scenario, we also analyze two special sub-cases:

   Case 2a: Isotropic measurements. We assume that the measurement vectors a_i are independent, isotropic random vectors that are incoherent with the bases Φ and Ψ. This assumption is more general than the i.i.d. standard normal assumption on the measurement matrix made in the first scenario, and is applicable to a wider range of measurement models. For this case, we show that the sample complexity of DHT is upper-bounded by $m = \mathcal{O}(s \text{ polylog } n)$, independent of the estimation error κ.

   Case 2b: Subgaussian measurements. we assume that the rows of the matrix A are independent subgaussian isotropic random vectors. This is also a generalization of the i.i.d. standard normal assumption made above, but more restrictive than Case 2a. In this setting, we obtain somewhat better sample complexity. More precisely, we show that the sample complexity of DHT is $m = \mathcal{O}(s \log \frac{n}{s})$ for sample complexity, matching the best known sample complexity bounds for recovering a superposition of s-sparse signals from *linear* observations (Hegde and Baraniuk, 2012b,a).

Table 1.1: Summary of our contributions, and comparison with existing methods for the concrete case where Φ is the identity and Ψ is the DCT basis. Here, s denotes the sparsity level of the components, n denotes the ambient dimension, m denotes the number of samples, and κ denotes estimation error.

Algorithms	Sample complexity	Running time	Measurements	Link function
LASSO	$\mathcal{O}(\frac{s}{\kappa^2}\log\frac{n}{s})$	poly(n)	Gaussian	unknown
ONESHOT	$\mathcal{O}(\frac{s}{\kappa^2}\log\frac{n}{s})$	$\mathcal{O}(mn)$	Gaussian	unknown
DHT	$\mathcal{O}(s \text{ polylog } n)$	$\mathcal{O}(mn\log\frac{1}{\kappa})$	Isotropic rows	known
DHT	$\mathcal{O}(s\log\frac{n}{s})$	$\mathcal{O}(mn\log\frac{1}{\kappa})$	Subgaussian	known

In both the above cases, the underlying assumption is that the bases Φ and Ψ are sufficiently incoherent, and that the sparsity level s is small relative to the ambient dimension n. In this regime, we show that DHT exhibits a *linear* rate of convergence, and therefore the computational complexity of DHT is only a logarithmic factor higher than ONESHOT. Table 1.1 provides a summary of the above contributions for the specific case where Φ is the identity (canonical) basis and Ψ is the discrete cosine transform (DCT) basis, and places them in the context of the existing literature on some nonlinear recovery methods (Plan et al., 2014; Thrampoulidis et al., 2015; Plan and Vershynin, 2016). We stress that these previous works do not explicitly consider the demixing problem, but in principle the algorithms of (Plan et al., 2014; Thrampoulidis et al., 2015; Plan and Vershynin, 2016) can be extended to the demixing setting as well.

1.1.3 Techniques

At a high level, our recovery algorithms are based on the now-classical method of *greedy* iterative thresholding. In both methods, the idea is to first form a proxy of the signal components, followed by hard thresholding to promote sparsity of the final estimates of the coefficient vectors w and z. The key distinguishing factor from existing methods is that the greedy thresholding procedures used to estimate w and z are *deliberately myopic*, in the sense that each thresholding step operates *as if the other component did not exist at all.* Despite this apparent shortcoming, we are still able to derive bounds on recovery performance when the signal components are sufficiently incoherent.

Our first algorithm, OneShot, is based on the recent, pioneering approach of (Plan et al., 2014), which describes a simple (but effective) method to estimate a high-dimensional signal from unknown nonlinear observations. Our first main contribution of this chapter is to extend this idea to the nonlinear demixing problem, and to precisely characterize the role of incoherence in the recovery process. Indeed, a variation of the approach of (Plan et al., 2014) (described in Section 1.5) can be used to solve the nonlinear demixing problem as stated above, with a similar two-step method of first forming a proxy, and then performing a convex estimation procedure (such as the LASSO (Tibshirani, 1996)) to produce the final signal estimates. However, as we show below in our analysis and experiments, OneShot offers superior performance to this approach. The analysis of OneShot is based on a geometric argument, and leverages the *Gaussian mean width* for the set of sparse vectors, which is a statistical measure of complexity of a set of points in a given space.

While OneShot is simple and effective, one can potentially do much better if the link function g were available at the time of recovery. Our second algorithm, DHT, leverages precisely this intuition. First, we formulate our nonlinear demixing problem in terms of an optimization problem with respect to a specially-defined loss function that depends on the nonlinearity g. Next, for solving the proposed optimization problem, we propose an iterative method to solve the optimization problem, up to an additive approximation factor. Each iteration with DHT involves a proxy calculation formed by computing the gradient of the loss function, followed by (myopic) projection onto the constraint sets. Again, somewhat interestingly, this method can be shown to be *linearly convergent*, and therefore only incurs a small (logarithmic) overhead in terms of running time. The analysis of DHT is based on bounding certain parameters of the loss function known as the restricted strong convexity (RSC) and restricted strong smoothness (RSS) constants.[1]

[1]Quantifying algorithm performance by bounding RSC and RSC constants of a given loss function are quite widespread in the machine learning literature (Negahban et al., 2011; Bahmani et al., 2013b; Yuan et al., 2014a; Jain et al., 2014), but have not studied in the context of signal demixing.

Finally, we provide a wide range of simulations to verify empirically our claims both on synthetic and real data. We first compare the performance of OneShot with the convex optimization method of (Plan et al., 2014) for nonlinear demixing via a series of phase transition diagrams. Our simulation results show that OneShot outperforms this convex method significantly in both demixing efficiency as well as running time, and consequently makes it an attractive choice in large-scale problems. However, as discussed below, the absence of knowledge of the link function induces an inevitable scale ambiguity in the final estimation[2]. For situations where we know the link function precisely, our simulation results show that DHT offers much better statistical performance compared to OneShot, and is even able to recover the scale of the signal components explicitly. We also provide simulation results on real-world natural images and astronomical data to demonstrate robustness of our approaches.

1.2 Applications and Prior Art

Demixing problems of various flavors have been long studied in research areas spanning signal processing, statistics, and physics, and we only present a small subset of relevant related work. In particular, demixing methods have been the focus of significant research over the fifteen years, dating back at least to (Chen et al., 1998). The work of Elad et al. (Elad et al., 2005) and Bobin et al. (Bobin et al., 2007) posed the demixing problem as an instance of *morphological components analysis* (MCA), and formalized the observation model (1.1). Specifically, these approaches posed the recovery problem in terms of a convex optimization procedure, such as the LASSO (Tibshirani, 1996). The work of Pope et al. (Studer et al., 2012) analyzed somewhat more general conditions under which stable demixing could be achieved.

More recently, the work of (McCoy and Tropp, 2014) showed a curious phase transition behavior in the performance of the convex optimization methods. Specifically, they demonstrated a sharp statistical characterization of the achievable and non-achievable parameters for which successful demixing of the signal components can be achieved. Moreover, they extended the demixing problem

[2]Indeed, following the discussion in (Plan et al., 2014), any demixing algorithm that does not leverage knowledge of g is susceptible to such a scale ambiguity.

to a large variety of signal structures beyond sparsity via the use of general *atomic norms* in place of the ℓ_1-norm in the above optimization. See (McCoy et al., 2014) for an in-depth discussion of atomic norms, their statistical and geometric properties, and their applications to demixing.

Approaches for (linear) demixing has also considered a variety of signal models beyond sparsity. The robust PCA problem (Candès et al., 2011; Chandrasekaran et al., 2009, 2011) involves the separation of low-rank and sparse matrices from their sum. This idea has been used in several applications ranging from video surveillance to sensor network monitoring. In machine learning applications, the separation of low-rank and sparse matrices has been used for latent variable model selection (Chandrasekaran et al., 2010) as well as the robust alignment of multiple occluded images (Peng et al., 2012). Another type of signal model is the *low-dimensional manifold* model. In (Hegde and Baraniuk, 2012b,a), the authors proposed a greedy iterative method for demixing signals, arising from a mixture of known low-dimensional manifolds by iterative projections onto the component manifolds.

The problem of signal demixing from linear measurements belongs to a class of linear inverse problems that underpin *compressive sensing* (Candès, 2006; Donoho, 2006); see (Foucart and Rauhut,) for an excellent introduction. There, the overarching goal is to recover signals from (possibly randomized) linear measurements of the form (1.1). More recently, it has been shown that compressive sensing techniques can also be extended to inverse problems where the available observations are manifestly *nonlinear*. For instance, in 1-bit compressive sensing (Boufounos and Baraniuk, 2008; Plan and Vershynin, 2013a) the linear measurements of a given signal are quantized in the extreme fashion such that the measurements are binary (± 1) and only comprise the sign of the linear observation. Therefore, the amplitude of the signal is completely discarded by the quantization operator. Another class of such nonlinear recovery techniques can be applied to the classical signal processing problem of phase retrieval (Candes et al., 2015) which is somewhat more challenging than 1-bit compressive sensing. In this problem, the phase information of the signal measurements may be irrecovably lost and we have only access to the amplitude information of the signal (Candes et al., 2015). Therefore, the recovery task here is to retrieve the phase information of the signal from random observations. Other related works include approaches for recovering low-rank matrices from nonlinear observations (Davenport et al., 2014; Ganti et al., 2015a). We mention in passing that inverse problems involving nonlinear observations have also long been studied in the statistical learning theory literature; see (Kalai and Sastry, 2009; Kakade et al., 2011; Ganti et al., 2015b; Yi et al., 2015) for recent work in this area. Analogous to our scenarios above, these works consider both known as well as unknown link functions; these two classes of approaches are respectively dubbed as Generalized Linear Models (GLM) learning methods and Single Index Model (SIM) learning methods.

For our algorithmic development, we build upon a recent line of efficient, iterative methods for signal estimation in high dimensions (Beck and Eldar, 2013; Bahmani et al., 2013b; Yuan et al., 2014a; Plan et al., 2014; Jain et al., 2014; Yang et al., 2015). The basic idea is to pose the recovery as a (non-convex) optimization problem in which an objective function is minimized over the set of s-sparse vectors. Essentially, these algorithms are based on well-known iterative thresholding methods proposed in the context of sparse recovery and compressive sensing (Blumensath and Davies, 2009; Needell and Tropp, 2009a). The analysis of these methods heavily depends on the assumption that the objective function satisfies certain (restricted) regularity conditions; see Sections 1.3 and 1.7 for details. Crucially, we adopt the approach of (Negahban et al., 2011), which introduces the concept of the restricted strong convexity (RSC) and restricted strong smoothness (RSS) constants of a loss function. Bounding these constants in terms of problem parameters n and s, as well as the level of incoherence in the components, enables explicit characterization of both sample complexity and convergence rates.

1.3 Preliminaries

In this section, we introduce some notation and key definitions. Throughout this chapter, $\|.\|_p$ denotes the ℓ_p-norm of a vector in $\mathbb{R}^n$, and $\|A\|$ denotes the spectral norm of the matrix $A \in \mathbb{R}^{m\times n}$. Let Φ and Ψ be orthonormal bases of $\mathbb{R}^n$.

Define the set of sparse vectors in the bases Φ and Ψ as follows:

$$K_1 = \{\Phi a \mid \|a\|_0 \leq s_1\},$$

$$K_2 = \{\Psi a \mid \|a\|_0 \leq s_2\},$$

and define $K = \{a \mid \|a\|_0 \leq s\}$. We use B_2^n to denote the unit ℓ_2 ball. Whenever we use the notation $t = [w; z]$, the vector t is comprised by stacking column vectors w and z.

In order to bound the sample complexity of our proposed algorithms, we will need some concepts from high-dimensional geometry. First, we define a statistical measure of complexity of a set of signals, following (Plan et al., 2014).

Definition 1.1. *(Local gaussian mean width.) For a given set $K \in \mathbb{R}^n$, the local gaussian mean width (or simply, local mean width) is defined as follows $\forall\ t > 0$:*

$$W_t(K) = \mathbb{E} \sup_{x,y \in K, \|x-y\|_2 \leq t} \langle g, x - y\rangle.$$

where $g \sim \mathcal{N}(0, I_{n\times n})$.

Next, we define the notion of a *polar norm* with respect to a given subset Q of the signal space:

Definition 1.2. *(Polar norm.) For a given $x \in \mathbb{R}^n$ and a subset of $Q \in \mathbb{R}^n$, the polar norm with respect to Q is defined as follows:*

$$\|x\|_{Q^o} = \sup_{u \in Q} \langle x, u\rangle.$$

Furthermore, for a given subset of $Q \in \mathbb{R}^n$, we define $Q_t = (Q - Q) \cap tB_2^n$. Since Q_t is a symmetric set, one can show that the polar norm with respect to Q_t defines a semi-norm. Next, we use the following standard notions from random matrix theory (Vershynin, 2010):

Definition 1.3. *(Subgaussian random variable.) A random variable X is called subgaussian if it satisfies the following:*

$$\mathbb{E} \exp\left(\frac{cX^2}{\|X\|_{\psi_2}^2}\right) \leq 2,$$

where $c > 0$ is an absolute constant and $\|X\|_{\psi_2}$ denotes the ψ_2-norm which is defined as follows:

$$\|X\|_{\psi_2} = \sup_{p \geq 1} \frac{1}{\sqrt{p}} (\mathbb{E}|X|^p)^{\frac{1}{p}}.$$

Definition 1.4. *(Isotropic random vectors.) A random vector-valued variable $v \in \mathbb{R}^n$ is said to be isotropic if $\mathbb{E}vv^T = I_{n\times n}$.*

In order to analyze the computational aspects of our proposed algorithms (in particular, DHT), we will need the following definition from (Negahban et al., 2011):

Definition 1.5. *A loss function f satisfies Restricted Strong Convexity/Smoothness (RSC/RSS) if:*

$$m_{4s} \leq \|\nabla^2_{\xi} f(t)\| \leq M_{4s},$$

where $\xi = \mathrm{supp}(t_1) \cup \mathrm{supp}(t_2)$, for all $\|t_i\|_0 \leq 2s$ and $i = 1, 2$. Also, m_{4s} and M_{4s} are (respectively) called the RSC and RSS constants. Here $\nabla^2_{\xi} f(t)$ denotes a $4s \times 4s$ sub-matrix of the Hessian matrix, $\nabla^2 f(t)$, comprised of row/column indices in ξ.

As discussed earlier, the underlying assumption in all demixing problems of the form (1.6) is that the constituent bases are sufficiently *incoherent* as per the following definition:

Definition 1.6. *(ε-incoherence.) The orthonormal bases Φ and Ψ are said to be ε-incoherent if:*

$$\varepsilon = \sup_{\substack{\|u\|_0 \leq s,\ \|v\|_0 \leq s \\ \|u\|_2 = 1,\ \|v\|_2 = 1}} |\langle \Phi u, \Psi v \rangle|. \tag{1.3}$$

The parameter ε is related to the so-called mutual coherence parameter of a matrix. Indeed, if we consider the (overcomplete) dictionary $\Gamma = [\Phi\ \Psi]$, then the mutual coherence of Γ is given by $\gamma = \max_{i \neq j} |(\Gamma^T \Gamma)_{ij}|$. Moreover, one can show that $\varepsilon \leq s\gamma$ (Foucart and Rauhut,).

We now formally establish our *signal model.* Consider a signal $x \in \mathbb{R}^n$ that is the superposition of a pair of sparse vectors in different bases, i.e.,

$$x = \Phi w + \Psi z\,, \tag{1.4}$$

where $\Phi, \Psi \in \mathbb{R}^{n\times n}$ are orthonormal bases, and $w, z \in \mathbb{R}^n$ such that $\|w\|_0 \leq s$, and $\|z\|_0 \leq s$. We define the following quantities:

$$\bar{x} = \frac{\Phi\bar{w} + \Psi\bar{z}}{\|\Phi\bar{w} + \Psi\bar{z}\|_2} = \alpha(\Phi\bar{w} + \Psi\bar{z}), \tag{1.5}$$

where $\alpha = \frac{1}{\|\Phi\bar{w}+\Psi\bar{z}\|_2}$, $\bar{w} = \frac{w}{\|w\|_2}$, $\bar{z} = \frac{z}{\|z\|_2}$. Also, define the coefficient vector, $t = [w; z] \in \mathbb{R}^{2n}$. as the vector obtaining by stacking the individual coefficient vectors w and z of the component signals.

We now state our *measurement model*. Consider the nonlinear observation model:

$$y_i = g(a_i^T x) + e_i, \; i = 1 \ldots m, \tag{1.6}$$

where $x \in \mathbb{R}^n$ is the superposition signal given in (1.4), and $g : \mathbb{R} \mapsto \mathbb{R}$ represents a nonlinear link function. We denote $g(x)$ as the derivative of $\Theta(x)$, i.e., $\Theta'(x) = g(x)$. As mentioned above, depending on the knowledge of the link function g, we consider two scenarios:

1. In the first scenario, the nonlinear link function may be non-smooth, non-invertible, or even unknown. In this setting, we assume the noiseless observation model, i.e., $y = g(Ax)$. In addition, we assume that the measurement matrix is populated by i.i.d. unit normal random variables.

2. In this setup, g represents a known nonlinear, differentiable, and strictly monotonic function. Further, in this scenario, we assume that the observation y_i is corrupted by a subgaussian additive noise with $\|e_i\|_{\psi_2} \leq \tau$ for $i = 1, \ldots, m$. We also assume that the additive noise has zero mean and independent from a_i, i.e., $\mathbb{E}(e_i) = 0$ for $i = 1, \ldots, m$. In addition, we assume that the measurement matrix consists of either (2a) isotropic random vectors that are incoherent with Φ and Ψ, or (2b) populated with subgaussian random variables.

We highlight some additional clarifications for the second case. In particular, we make the following :

Assumption 1.7. *There exist nonnegative* $l_1, l_2 > 0$ (*resp., nonpositive parameters* $l_1, l_2 < 0$) *such that* $0 < l_1 \leq g'(x) \leq l_2$ (*resp.* $l_1 \leq g'(x) \leq l_2 < 0$).

In words, the derivative of the link function is strictly bounded either within a positive interval or within a negative interval. In this chapter, we focus on the case when $0 < l_1 \leq g'(x) \leq l_2$. The analysis of the complementary case is similar.

The lower bound on $g'(x)$ guarantees that the function g is a monotonic function, i.e., if $x_1 < x_2$ then $g(x_1) < g(x_2)$. Moreover, the upper bound on $g'(x)$ guarantees that the function g is *Lipschitz* with constant l_2. Such assumptions are common in the nonlinear recovery literature (Negahban et al., 2011; Yang et al., 2015).[3]

In Case 2a, the vectors a_i (i.e., the rows of A) are independent isotropic random vectors. For this case, in addition to incoherence between the component bases, we also need to define a measure of *cross*-coherence between the measurement matrix A and the dictionary Γ. The following notion of cross-coherence was introduced in the early literature of compressive sensing (Candes and Romberg, 2007):

Definition 1.8. *(Cross-coherence.) The cross-coherence parameter between the measurement matrix A and the dictionary $\Gamma = [\Phi\ \Psi]$ is defined as follows:*

$$\vartheta = \max_{i,j} \frac{a_i^T \Gamma_j}{\|a_i\|_2}, \tag{1.7}$$

where a_i and Γ_j denote the i^{th} row of the measurement matrix A and the j^{th} column of the dictionary Γ.

The cross-coherence assumption implies that $\left\|a_i^T \Gamma_\xi\right\|_\infty \leq \vartheta$ for $i = 1, \ldots, m$, where Γ_ξ denotes the restriction of the columns of the dictionary to set $\xi \subseteq [2n]$, with $|\xi| \leq 4s$ such that $2s$ columns are selected from each basis Φ and Ψ.

1.4 Algorithms and Theoretical Results

Having defined the above quantities, we now present our main results. As per the previous section, we study two distinct scenarios:

[3] Using the monotonicity property of g that arises from Assumption 1.7, one might be tempted to simply apply the inverse of the link function on the measurements y_i in (1.6) convert the nonlinear demixing problem to the more amenable case of linear demixing, and then use any algorithm (e.g., (Hegde and Baraniuk, 2012a)) for recovery of the constituent signals. However, this naïve way could result in a large error in the estimation of the components, particularly in the presence of the noise e_i in (1.6). This issue has been also considered in (Yang et al., 2015) for generic nonlinear recovery both from a theoretical as well as empirical standpoint.

Algorithm 1.1 OneShot

Inputs: Basis matrices Φ and Ψ, measurement matrix A, measurements y, sparsity level s.
Outputs: Estimates $\widehat{x} = \Phi\widehat{w} + \Psi\widehat{z}$, $\widehat{w} \in K_1$, $\widehat{z} \in K_2$

```
$\widehat{x}_{\text{lin}} \leftarrow \frac{1}{m} A^T y$        {form linear estimator}
$b_1 \leftarrow \Phi^* \widehat{x}_{\text{lin}}$        {forming first proxy}
$\widehat{w} \leftarrow \mathcal{P}_s(b_1)$        {sparse projection}
$b_2 \leftarrow \Psi^* \widehat{x}_{\text{lin}}$        {forming second proxy}
$\widehat{z} \leftarrow \mathcal{P}_s(b_2)$        {sparse projection}
$\widehat{x} \leftarrow \Phi\widehat{w} + \Psi\widehat{z}$        {Estimating $\widehat{x}$}
```

1.4.1 When the link function g is unknown

Recall that we wish to recover components w and z given the nonlinear measurements y and the matrix A. Here and below, for simplicity we assume that the sparsity levels s_1 and s_2, specifying the sets K_1 and K_2, are equal, i.e., $s_1 = s_2 = s$. The algorithm (and analysis) effortlessly extends to the case of unequal sparsity levels. Our proposed algorithm, that we call OneShot, is described in pseudocode form below as Algorithm 1.1.

The mechanism of OneShot is simple, and *deliberately* myopic. At a high level, OneShot first constructs a *linear estimator* of the target superposition signal, denoted by $\widehat{x}_{\text{lin}} = \frac{1}{m} A^T y$. Then, it performs independent projections of $\widehat{x}_{\text{lin}}$ onto the constraint sets K_1 and K_2. Finally, it combines these two projections to obtain the final estimate of the target superposition signal.

In the above description of OneShot, we have used the following *projection* operators:

$$\widehat{w} = \mathcal{P}_s(\Phi^* \widehat{x}_{\text{lin}}), \quad \hat{z} = \mathcal{P}_s(\Psi^* \widehat{x}_{\text{lin}}).$$

Here, $\mathcal{P}_s$ denotes the projection onto the set of (canonical) s-sparse signals K and can be implemented by hard thresholding, i.e., any procedure that retains the s largest coefficients of a vector (in terms of absolute value) and sets the others to zero[4]. Ties between coefficients are broken arbitrarily. Observe that OneShot is *not* an iterative algorithm, and this in fact enables us to achieve a fast running time.

[4]The typical way is to sort the coefficients by magnitude and retain the s largest entries, but other methods such as randomized selection can also be used.

We now provide a rigorous performance analysis of OneShot. Our proofs follow the geometric approach provided in (Plan et al., 2014), specialized to the demixing problem. In particular, we derive an upper bound on the estimation error of the *component* signals w and z, modulo scaling factors. In our proofs, we use the following result from (Plan et al., 2014), restated here for completeness. Please see appendix for the proof of all theoretical results.

Lemma 1.9. *(Quality of linear estimator). Given the model in Equation (1.4), the linear estimator, $\widehat{x}_{lin}$, is an unbiased estimator of $\bar{x}$ (defined in (1.5)) up to constants. That is, $\mathbb{E}(\widehat{x}_{lin}) = \mu\bar{x}$ and: $\mathbb{E}\|\widehat{x}_{lin} - \mu\bar{x}\|_2^2 = \frac{1}{m}[\sigma^2 + \eta^2(n-1)]$, where $\mu = \mathbb{E}(y_1\langle a_1, \bar{x}\rangle)$, $\sigma^2 = Var(y_1\langle a_1, \bar{x}\rangle)$, $\eta^2 = \mathbb{E}(y_1^2)$.*

We now state our first main theoretical result, with the full proof provided below in Section 1.7.

Theorem 1.10. *Let $y \in \mathbb{R}^m$ be the set of measurements generated using a nonlinear function g that satisfies the conditions of Lemma (4.9) in (Plan et al., 2014)[5]. Let $A \in \mathbb{R}^{m\times n}$ be a random matrix with i.i.d. standard normal entries. Also, let $\Phi, \Psi \in \mathbb{R}^{n\times n}$ are bases with $\varepsilon \leq 0.65$, where ε is as defined in Def. 2.1. If we use* OneShot *to recover estimates of w and z (modulo a scaling) described in equations (1.4) and (1.5), then the estimation error for w (similarly, z) satisfies the following upper bound in expectation $\forall \rho > 0$:*

$$\mathbb{E}\|\widehat{w} - \mu\alpha w\|_2 \leq \rho + \frac{2}{\sqrt{m}}\left(4\sigma + \eta\frac{W_\rho(K)}{\rho}\right) + 8\mu\varepsilon\,. \tag{1.8}$$

The constant 0.65 is chosen for convenience and can be strengthened. The authors of (Plan and Vershynin, 2013b; Plan et al., 2014) provide upper bounds on the local mean width $W_\rho(K)$ of the set of s-sparse vectors. In particular, for any $\rho > 0$ they show that $W_\rho(K) \leq C\rho\sqrt{s\log(2n/s)}$ for some absolute constant C. By plugging in this bound and letting $\rho \to 0$, we can combine components $\widehat{w}$ and $\widehat{z}$ which gives the following:

Corollary 1.11. *With the same assumptions as Theorem 1.10, the error of nonlinear estimation incurred by the final output $\widehat{x}$ satisfies the upper bound:*

$$\mathbb{E}\|\widehat{x} - \mu\bar{x}\|_2 \leq \frac{4}{\sqrt{m}}\left(4\sigma + C\eta\sqrt{s\log(2n/s)}\right) + 16\mu\varepsilon. \tag{1.9}$$

[5] *Based on this lemma, the nonlinear function g is odd, nondecreasing, and sub-multiplicative on $\mathbb{R}^+$.*

Corollary 1.12. *(Example quantitative result). The constants σ, η, μ depend on the nature of the nonlinear function f, and are often rather mild. For example, if $f(x) = \text{sign}(x)$, then we may substitute*

$$\mu = \sqrt{\frac{2}{\pi}} \approx 0.8, \qquad \sigma^2 = 1 - \frac{2}{\pi} \approx 0.6, \qquad \eta^2 = 1,$$

in the above statement. Hence, the bound in (1.9) becomes:

$$\mathbb{E}\|\widehat{x} - \mu\bar{x}\|_2 \leq \frac{4}{\sqrt{m}}\left(3.1 + C\sqrt{s\log(2n/s)}\right) + 13\varepsilon\,. \tag{1.10}$$

Proof. Using Lemma 1.9, $\mu = \mathbb{E}(y_i\langle a_i, \bar{x}\rangle)$ where $y_i = \text{sign}(\langle a_i, x\rangle)$. Since $a_i \sim \mathcal{N}(0, I)$ and $\bar{x}$ has unit norm, $\langle a_i, \bar{x}\rangle \sim \mathcal{N}(0,1)$. Thus, $\mu = \mathbb{E}|g| = \sqrt{\frac{2}{\pi}}$ where $g \sim \mathcal{N}(0, I)$. Moreover, we can write $\sigma^2 = \mathbb{E}(|g|^2) - \mu^2 = 1 - \frac{2}{\pi}$. Here, we have used the fact that $|g|^2$ obeys the χ_1^2 distribution with mean 1. Finally, $\eta^2 = \mathbb{E}(y_1^2) = 1$. □

In contrast with demixing algorithms for traditional (linear) observation models, our estimated signal $\widehat{x}$ outputting from OneShot can differ from the true signal x by a scale factor. Next, suppose we fix $\kappa > 0$ as a small constant, and suppose that the incoherence parameter $\varepsilon = c\kappa$ for some constant c, and that the number of measurements scales as:

$$m = \mathcal{O}\left(\frac{s}{\kappa^2}\log\frac{n}{s}\right). \tag{1.11}$$

Then, the (expected) estimation error $\|\widehat{x} - \mu\bar{x}\| \leq \mathcal{O}(\kappa)$. In other words, the *sample complexity* of OneShot is given by $m = \mathcal{O}(\frac{1}{\kappa^2}s\log(n/s))$, which resembles results for the linear observation case (Hegde and Baraniuk, 2012a; Plan et al., 2014)[6].

We observe that the estimation error in (1.9) is upper-bounded by $\mathcal{O}(\varepsilon)$. This is meaningful only when $\varepsilon \ll 1$, or when $s\gamma \ll 1$. Per the Welch Bound (Foucart and Rauhut,), the mutual coherence γ satisfies $\gamma \geq 1/\sqrt{n}$. Therefore, Theorem 1.10 provides non-trivial results only when $s = o(\sqrt{n})$. This is consistent with the *square-root bottleneck* that is often observed in demixing problems; see (Tropp, 2008) for detailed discussions.

[6]Here, we use the term "sample-complexity" as the number of measurements required by a given algorithm to achieve an estimation error κ. However, we must mention that algorithms for the linear observation model are able to achieve stronger sample complexity bounds that are independent of κ.

The above theorem obtains a bound on the expected value of the estimation error. We can derive a similar upper bound that holds with high probability. In this theorem, we assume that the *measurements* y_i for $i = 1, 2, \ldots, m$ have a *sub-gaussian* distribution (according to Def. 1.3). We obtain the following result, with full proof deferred to Section 1.7.

Theorem 1.13. *(High-probability version of Thm. 1.10.) Let $y \in \mathcal{R}^m$ be a set of measurements with a sub-gaussian distribution. Assume that $A \in \mathbb{R}^{m\times n}$ is a random matrix with i.i.d standard normal entries. Also, assume that $\Phi, \Psi \in \mathbb{R}^{n\times n}$ are two bases with incoherence $\varepsilon \leq 0.65$ as in Definition 2.1. Let $0 \leq s' \leq \sqrt{m}$. If we use* OneShot *to recover w and z (up to a scaling) described in (1.4) and (1.5), then the estimation error of the output of* OneShot *satisfies the following:*

$$\|\widehat{x} - \mu\bar{x}\|_2 \leq \frac{4\eta}{\sqrt{m}}\left(3s' + C'\sqrt{s\log\frac{2n}{s}}\right) + 16\mu\varepsilon, \tag{1.12}$$

with probability at least $1 - 4\exp(-\frac{cs'^2\eta^4}{\|y_1\|_{\psi_2}^4})$ where $C', c > 0$ are absolute constants. The coefficients μ, σ, and η are given in Lemma 1.9. Here, $\|y_1\|_{\psi_2}$ denotes the ψ_2-norm of the first measurement y_1 (Definition 1.3).

In Theorem 1.13, we stated the tail probability bound of the estimation error for the superposition signal, x. Similar to Theorem 1.10, we can derive a completely analogous tail probability bound in terms of the constituent signals w and z.

1.4.2 When the link function g is known

The advantages of OneShot is that it enables fast demixing, and can handle even unknown, non-differentiable link functions. But its primary weakness is that the sparse components are recovered only up to an arbitrary scale factor. This can lead to high estimation errors in practice, and this can be unsatisfactory in applications. Moreover, even for reliable recovery up to a scale factor, its sample complexity is inversely dependent on the estimation error. To solve these problems, we propose a different, iterative algorithm for recovering the signal components. Here, the main difference is that the algorithm is assumed to possess (perfect) knowledge of the nonlinear link function, g. Recall that we define $\Gamma = [\Phi \ \Psi]$ and $t = [w; z] \in \mathbb{R}^{2n}$. First, we formulate our demixing problem as the minimization of a special loss function $F(t)$:

$$\begin{aligned} &\min_{t \in \mathbb{R}^{2n}} \quad F(t) = \frac{1}{m} \sum_{i=1}^{m} \Theta(a_i^T \Gamma t) - y_i a_i^T \Gamma t \\ &\text{s. t.} \quad \|t\|_0 \leq 2s. \end{aligned} \tag{1.13}$$

Observe that the loss function $F(t)$ is *not* the typical squared-error function commonly encountered in statistics and signal processing applications. In contrast, it heavily depends on the nonlinear link function g (via its integral Θ). Instead, such loss functions are usually used in GLM and SIM estimation in the statistics literature (Negahban et al., 2011). In fact, the objective function in (2.3) can be considered as the *sample* version of the problem:

$$\min_{t \in \mathbb{R}^{2n}} \ \mathbb{E}(\Theta(a^T \Gamma t) - y a^T \Gamma t),$$

where a, y and Γ satisfies the model (1.6). It is not hard to show that the solution of this problem satisfies $\mathbb{E}(y_i | a_i) = g(a_i^T \Gamma t)$. We note that the gradient of the loss function can be calculated in closed form:

$$\begin{aligned} \nabla F(t) &= \frac{1}{m} \sum_{i=1}^{m} \Gamma^T a_i g(a_i^T \Gamma t) - y_i \Gamma^T a_i, \\ &= \frac{1}{m} \Gamma^T A^T (g(A \Gamma t) - y). \end{aligned} \tag{1.14}$$

We now propose an *iterative* algorithm for solving (2.3) that we call it Demixing with Hard Thresholding (DHT). The method is detailed in Algorithm 1.2. At a high level, DHT iteratively refines its estimates of the constituent signals w, z (and the superposition signal x). At any given iteration, it constructs the gradient using (1.14). Next, it updates the current estimate according to the gradient update being determined in Algorithm 1.2. Then, it performs hard thresholding using the operator $\mathcal{P}_{2s}$ to obtain the new estimate of the components w and z. This procedure is repeated until a stopping criterion is met. See Section 1.5 for the choice of stopping criterion and other details. We mention that the initialization step in Algorithm 1.2 is arbitrary and can be

Algorithm 1.2 Demixing with Hard Thresholding (DHT)

```
Inputs: Bases $\Phi$ and $\Psi$, measurement matrix $A$, link function $g$, measurements $y$, sparsity level $s$, step size $\eta'$.
Outputs: Estimates $\widehat{x} = \Phi\widehat{w} + \Psi\widehat{z}$, $\widehat{w}$, $\widehat{z}$
Initialization:
$\left(x^0, w^0, z^0\right) \leftarrow$ ARBITRARY INITIALIZATION
$k \leftarrow 0$
while $k \leq N$ do
    $t^k \leftarrow [w^k; z^k]$                              {forming constituent vector}
    $t_1^k \leftarrow \frac{1}{m}\Phi^T A^T(g(Ax^k) - y)$
    $t_2^k \leftarrow \frac{1}{m}\Psi^T A^T(g(Ax^k) - y)$
    $\nabla F^k \leftarrow [t_1^k; t_2^k]$                   {forming gradient}
    $\tilde{t}^k = t^k - \eta' \nabla F^k$                   {gradient update}
    $[w^k; z^k] \leftarrow \mathcal{P}_{2s}\left(\tilde{t}^k\right)$   {sparse projection}
    $x^k \leftarrow \Phi w^k + \Psi z^k$                     {estimating $\widehat{x}$}
    $k \leftarrow k + 1$
end while
Return: $(\widehat{w}, \widehat{z}) \leftarrow \left(w^N, z^N\right)$
```

implemented (for example) by running OneShot and obtaining initial points $\left(x^0, w^0, z^0\right)$. We use this initialization in our simulation results.

Implicitly, we have again assumed that both component vectors w and z are s-sparse; however, as above we mention that Algorithm 1.2 and the corresponding analysis easily extend to differing levels of sparsity in the two components. In Algorithm 1.2, $\mathcal{P}_{2s}$ denotes the projection of vector $\tilde{t}^k \in \mathbb{R}^{2n}$ on the set of $2s$ sparse vectors, again implemented via hard thresholding.

We now provide our second main theoretical result, supporting the convergence analysis of DHT. In particular, we derive an upper bound on the estimation error of the constituent vector t (and therefore, the component signals w, z). The proofs of Theorems 1.14, 1.15 and 1.16 are deferred to section 1.7.

Theorem 1.14. *(Performance of* DHT*) Consider the measurement model* (1.6) *with all the assumptions mentioned for the second scenario in Section 1.3. Suppose that the corresponding objective function F satisfies the RSS/RSC properties with constants M_{6s} and m_{6s} on the set J_k with $|J_k| \leq 6s$ (k denotes the k^{th} iteration) such that $1 \leq \frac{M_{6s}}{m_{6s}} \leq \frac{2}{\sqrt{3}}$. Choose a step size parameter η'*

with $\frac{0.5}{M_{6s}} < \eta' < \frac{1.5}{m_{6s}}$. *Then,* DHT *outputs a sequence of estimates* $t^k = [w^k; z^k]$ *that satisfies the following upper bound (in expectation) for* $k \geq 1$*:*

$$\|t^{k+1} - t^*\|_2 \leq (2q)^k \|t^0 - t^*\|_2 + C\tau\sqrt{\frac{s}{m}}, \tag{1.15}$$

where $q = \sqrt{1 + \eta'^2 M_{6s}^2 - 2\eta' m_{6s}}$ *and* $C > 0$ *is a constant that depends on the step size* η' *and the convergence rate* q*. Also,* $t^* = [w; z]$ *where* w *and* z *are the true (unknown) vectors in model* (1.2)*.*

Equation (2.4) indicates that Algorithm 1.2 (DHT) enjoys a linear rate of convergence. In particular, for the noiseless case $\tau = 0$, this implies that Alg. 1.2 returns a solution with accuracy κ after $N = \mathcal{O}(\log \frac{\|t^0 - t\|_2}{\kappa})$ iterations. The proof of Theorem 1.14 leverages the fact that the objective function $F(t)$ in (2.3) satisfies the RSC/RSS conditions specified in Definition 2.2. Please refer to Section 1.7 for a more detailed discussion. Moreover, we observe that in contrast with OneShot, DHT can recover the components w and z without any ambiguity in scaling factor, as depicted in the bound (2.4). We also verify this observation empirically in our simulation results in Section 1.5.

Echoing our discussion in Section 1.3, we consider two different models for the measurement matrix A and derive upper bounds on the sample complexity of DHT corresponding to each case. First, we present the sample complexity of Alg. 1.2 when the measurements are chosen to be isotropic random vectors, corresponding to Case (2a) described in the introduction:

Theorem 1.15. *(Sample complexity when the rows of* A *are isotropic.) Suppose that the rows of* A *are independent isotropic random vectors. In order to achieve the requisite RSS/RSC properties of Theorem 1.14, the number of samples needs to scale as:*

$$m = \mathcal{O}(s \log n \log^2 s \log(s \log n)),$$

provided that the bases Φ *and* Ψ *are incoherent enough.*

The sample complexity mentioned in Theorem 1.15 incurs an extra (possibly parasitic) poly-logarithmic factor relative to the sample complexity of OneShot, stated in (1.11). However, the drawback of OneShot is that the sample complexity depends inversely on the estimation error κ, and therefore a very small target error would incur a high overhead in terms of number of samples.

Removing all the extra logarithmic factors remains an open problem in general (although some improvements can be obtained using the method of (Cheraghchi et al., 2013)). However, if we assume additional structure in the measurement matrix A, we can decrease the sample complexity even further. This corresponds to Case 2b.

Theorem 1.16. *(Sample complexity when the elements of A are subgaussian.) Assume that all assumptions and definitions in Theorem 1.14 holds except that the rows of matrix A are independent subgaussian isotropic random vectors. Then, in order to achieve the requisite RSS/RSC properties of Theorem 1.14, the number of samples needs to scale as:*

$$m = \mathcal{O}\left(s \log \frac{n}{s}\right),$$

provided that the bases Φ and Ψ are incoherent enough.

The leading big-Oh constant in the expression for m in Theorems 1.15 and 1.16 is somewhat complicated, and hides the dependence on the incoherence parameter ε, the mutual coherence ϑ, the RSC/RSS constants, and the growth parameters of the link function l_1 and l_2. Please see section 1.7 for more details.

In Theorem 1.14, we expressed the upper bounds on the estimation error in terms of the constituent vector, t. It is easy to translate these results in terms of the component vectors w and z using the triangle inequality:

$$\max\{\|w^0 - w^*\|_2, \|z^0 - z^*\|_2\} \leq \|t^0 - t^*\|_2 \leq \|w^0 - w^*\|_2 + \|z^0 - z^*\|_2.$$

See Section 1.7 for proofs and futher details.

1.5 Experimental Results

In this section, we provide a range of numerical experiments for our proposed algorithms based on synthetic and real data. We compare the performance of OneShot and DHT with a LASSO-type technique for demixing, as well as a heuristic version of OneShot based on soft thresholding (inspired by the approach proposed in (Yang et al., 2000)). We call these methods *Nonlinear* *convex demixing with LASSO* or (NLCDLASSO), and *Demixing with Soft Thresholding* or DST, respectively. Before describing our simulation results, we briefly describe these two methods.

NLCDLASSO is a heuristic method motivated by (Plan et al., 2014), although it was not explicitly developed in the demixing context. Using our notation from Section 1.3 and 1.4, NLCDLASSO solves the following convex problem:

$$\begin{aligned} \min_{z,w} \quad & \left\|\widehat{x}_{\text{lin}} - [\Phi\ \Psi][w;z]\right\|_2 \\ \text{subject to} \quad & \|w\|_1 \leq \sqrt{s}, \quad \|z\|_1 \leq \sqrt{s}. \end{aligned} \tag{1.16}$$

Here, $\widehat{x}_{\text{lin}}$ denotes the proxy of x (equal to $\frac{1}{m}A^Ty$) and s denotes the sparsity level of signals w and z in basis Φ and Ψ, respectively. The constraints in problem (1.16) are convex penalties reflecting the knowledge that w and z are s-sparse and have unit ℓ_2-norm (since the nonlinearity is unknown, we have a scale ambiguity, and therefore w.l.o.g. we can assume that the underlying signals lie in the unit ball). The outputs of this algorithm are the estimates $\widehat{w}$, $\widehat{x}$, and $\widehat{x} = \Phi\widehat{w} + \Psi\widehat{z}$.

To solve the optimization problem in (1.16), we have used the SPGL1 solver (van den Berg and Friedlander, 2008, 2007). This solver can handle large scale problems, which is the scenario that we have used in our experimental evaluations. We impose the joint constraint $\|t\|_1 = \|[w;z]\|_1 \leq 2\sqrt{s}$ which is a slight relaxation of the constraints in 1.16. The upper-bound of $\sqrt{s}$ in the constraints is a worst-case criterion; therefore, for a fairer comparison, we also include simulation results with the constraint $\|t\|_1 \leq \varrho$, where ϱ has been tuned to the best of our ability.

On the other hand, DST solves the optimization problem (2.3) via a convex relaxation of the sparsity constraint. In other words, this method attempts to solve the following relaxed version of the problem (2.3):

$$\min_t \ \frac{1}{m}\sum_{i=1}^m \Theta(a_i^T\Gamma t) - y_i a_i^T\Gamma t + \beta'\|t\|_1, \tag{1.17}$$

where $\|t\|_1$ represents l_1-norm of the constituent vector t and $\beta' > 0$ denotes the tuning parameter. The solution of this problem at iteration k is given by soft thresholding operator as follows:

$$t^{k+1} = S_{\beta'\eta'}(t^k - \eta'\nabla F(t^k)),$$

where η' denotes the step size, and the soft thresholding operator, $S_\lambda(.)$ is given by:

$$S_\lambda(y) = \begin{cases} y-\lambda\,, & \text{if } y>\lambda \\ 0\,, & \text{if } |y|\le\lambda \\ y+\lambda\,, & \text{if } y<-\lambda. \end{cases}$$

Both OneShot and NlcdLASSO do not assume knowledge of the link function, and consequently return a solution up to a scalar ambiguity. Therefore, to compare performance across algorithms, we use the (scale-invariant) *cosine similarity* between the original superposition signal x and the output of a given algorithm $\widehat{x}$ defined as:

$$\cos(x,\widehat{x}) = \frac{x^T\widehat{x}}{\|x\|_2\|\widehat{x}\|_2}.$$

1.5.1 Synthetic Data

As discussed above, for successful recovery we require the constituent signals to be sufficiently incoherent. To achieve this, we choose Φ to be the 1D Haar wavelets basis, and Ψ to be the noiselet basis[7]. For the measurement operator A, we choose a partial DFT matrix. Such matrices are known to have similar recovery performance as random Gaussian matrices, but enable fast numerical operations (Candès et al., 2006). Also, we present our experiments based on both non-smooth as well as differentiable link functions. For the non-smooth case, we choose $g(x) = \text{sign}(x)$; here, we only present recovery results using OneShot and NlcdLASSO since in our analysis DHT and DST can only handle smooth link functions.

The results of our first experiment are shown in Figure 1.1(a) and Figure 1.1(b). The test signal is generated as follows: set length $n = 2^{20}$, and generate the vectors w and z by randomly selecting a signal support with s nonzero elements, and populating the nonzero entries with random ± 1 coefficients. The plot illustrates the performance of Oneshot and NlcdLASSO measured by the cosine similarity for different choices of sparsity level s, where the nonlinear link function is set to $g(x) = \text{sign(x)}$ and we have used both $\|t\|_1 \le 2\sqrt{s}$ and $\|t\|_1 \le \varrho$ constraints. The

[7]These bases are known to be maximally incoherent relative to each other (Coifman et al., 2001)

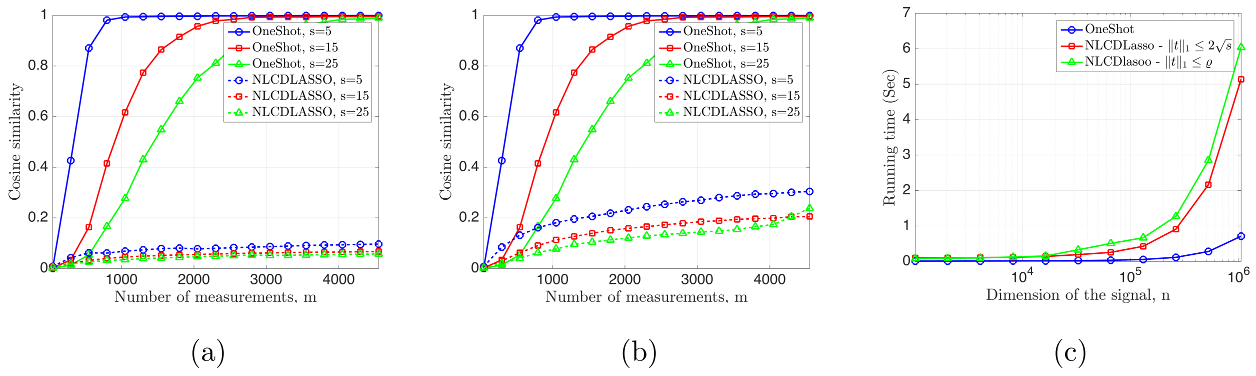


Figure 1.1: Performance of ONESHOT and NLCDLASSO according to the COSINE SIMILARITY for different choices of sparsity level s for $g(x) = sign(x)$. (a) NLCDLASSO with $\|t\|_1 \leq 2\sqrt{s}$. (b) NLCDLASSO with $\|t\|_1 \leq \varrho$. (c) Comparison of running times of ONESHOT with NLCDLASSO.

horizontal axis denotes an increasing number of measurements. Each data point in the plot is obtained by conducting a Monte Carlo experiment in which a new random measurement matrix A is generated, recording the cosine similarity between the true signal x and the reconstructed estimate and averaging over 20 trials.

As we can see, notably, the performance of NLCDLASSO is worse than ONESHOT for any fixed choice of m and s no matter what upper bound we use on t. Even when the number of measurements is high (for example, at $m = 4550$ in plot (b)), we see that ONESHOT outperforms NLCDLASSO by a significant degree. In this case, NLCDLASSO is at least 70% worse in terms of signal estimation quality, while ONESHOT recovers the (normalized) signal perfectly. This result indicates the inefficiency of NLCDLASSO for nonlinear demixing.

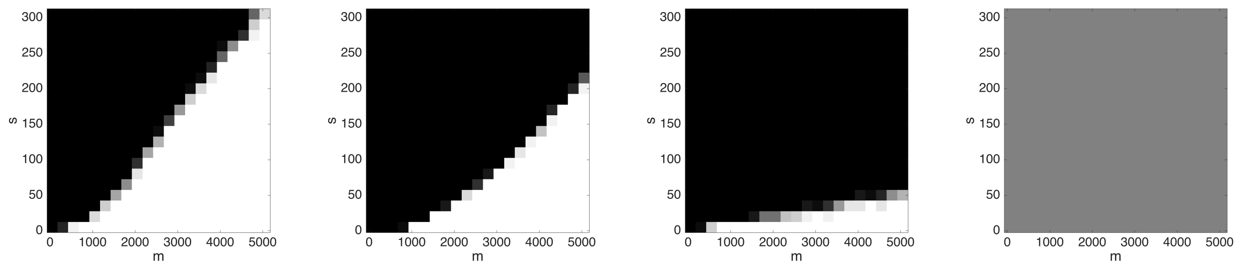
Next, we contrast the running time of both algorithms, illustrated in Figure 1.1(c). In this experiment, we measure the wall-clock running time of the two recovery algorithms (ONESHOT and NLCDLASSO), by varying signal size x from $n = 2^{10}$ to $n = 2^{20}$. Here, we set $m = 500$, $s = 5$, and the number of Monte Carlo trials to 20. Also, the nonlinear link function is considered as $g(x) = \text{sign}(x)$. As we can see from the plot, ONESHOT is at least 6 times faster than NLCDLASSO when the size of signal equals to 2^{20}. Overall, ONESHOT is efficient even for large-scale nonlinear demixing problems. We mention that in the above setup, the main computational costs incurred in OneShot involve a matrix-vector multiplication followed by a thresholding step, both of which can be performed in time that is *nearly-linear* in terms of the signal length n for certain choices of A, Φ, Ψ

Next, we turn to differentiable link functions. In this case, we generate the constituent signal coefficient vectors, w, z with $n = 2^{16}$, and compare performance of the four above algorithms. The nonlinear link function is chosen to be $g(x) = 2x + \sin(x)$; it is easy to check that the derivative of this function is strictly bounded between $l_1 = 1$ and $l_2 = 3$. The maximal number of iterations for both DHT and DST is set to to 1000 with an early stopping criterion if convergence is detected. The step size is hard to estimate in practice, and therefore is chosen by manual tuning such that both DHT and DST obtain the best respective performance.

Figure 1.2 illustrates the performance of the four algorithms in terms of *phase transition* plots, following (McCoy and Tropp, 2014). In these plots, we varied both the sparsity level s and the number of measurements m. For each pair (s, m), as above we randomly generate the test superposition signal by choosing both the support and coefficients of x at random, as well as the measurement matrix. We repeat this experiment over 20 Monte Carlo trials. We calculate the empirical probability of successful recovery as the number of trials in which the output cosine similarity is greater than 0.99. Pixel intensities in each figure are normalized to lie between 0 and 1, indicating the probability of successful recovery.

As we observe in Fig. 1.2, DHT has the best performance among the different methods, and in particular, outperforms both the convex-relaxation based methods. The closest algorithm to DHT in terms of the signal recovery is DST, while the LASSO-based method fails to recover the superposition signal x (and consequently the constituent signals w and z). The improvements over OneShot are to be expected since as discussed before, this algorithm does not leverage the knowledge of the link function g and is not iterative.

In Fig. 1.3, we fix the sparsity level $s = 50$ and plot the probability of recovery of different algorithms with a varying number of measurements. The number of Monte Carlo trials is set



(a) DHT (b) DST (c) OneShot (d) NlcdLASSO

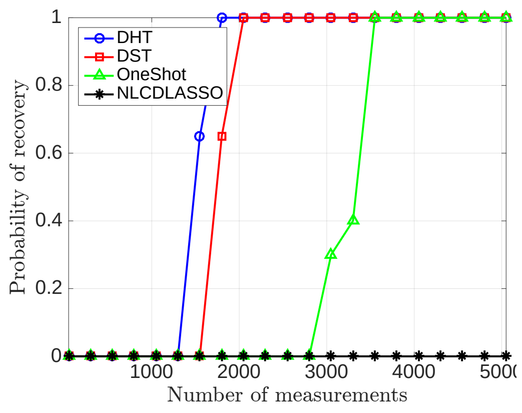
Figure 1.2: Phase transition plots of various algorithms for solving the demixing problem (1.6) as a function of sparsity level s and number of measurements m with cosine similarity as the criterion. Dimension of the signals $n = 2^{16}$.

to 20 and the empirical probability of successful recovery is defined as the number of trials in which the output cosine similarity is greater than 0.95. The nonlinear link function is set to be $g(x) = 2x + \sin(x)$ for figure (a) and $g(x) = \frac{1}{1+e^{-x}}$ for figure (b). As we can see, DHT has the best performance, while NlcdLASSO for figure (a) and OneShot, and NlcdLASSO for figure (b) cannot recover the superposition signal even with the maximum number of measurements.

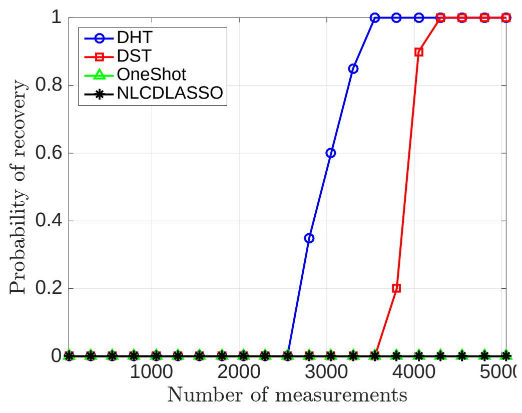
1.5.2 Real Data

In this section, we provide representative results on real-world 2D image data using OneShot and NlcdLASSO for non-smooth link function given by $g(x) = \text{sign}(x)$. In addition, we illustrate results for all four algorithms using smooth $g(x) = \frac{1-e^{-x}}{1+e^{-x}}$ as our link function.

We begin with a 256×256 test image. First, we obtain its 2D Haar wavelet decomposition and retain the $s = 500$ largest coefficients, denoted by the s-sparse vector w. Then, we reconstruct the image based on these largest coefficients, denoted by $\widehat{x} = \Phi w$. Similar to the synthetic case, we generate a noise component in our superposition model based on 500 noiselet coefficients z. In addition, we consider a parameter which controls the strength of the noiselet component contributing to the superposition model. We set this parameter to 0.1. Therefore, our test image x is given by $x = \Phi w + 0.1\Psi z$.



(a)



(b)

Figure 1.3: Probability of recovery for four algorithms; DHT, STM, ONESHOT, and NLCDLASSO. Sparsity level is set to $s = 50$ and dimension of the signals equal to $n = 2^{16}$. (a) $g(x) = 2x + \sin(x)$, (b) $g(x) = \frac{1}{1+e^{-x}}$.

Figure 1.4 illustrates both the true and the reconstructed images x and $\widehat{x}$ using ONESHOT and NLCDLASSO. The number of measurements is set to 35000 (using subsampled Fourier matrix with $m = 35000$ rows). From visual inspection we see that the reconstructed image, $\widehat{x}$, using ONESHOT is better than the reconstructed image by NLCDLASSO. Quantitatively, we also calculate Peak signal-to-noise-ratio (PSNR) of the reconstructed images using both algorithms relative to the test image, x. We obtain PSNR of 19.8335 dB using ONESHOT, and a PSNR of 17.9092 dB using NLCDLASSO, again illustrating the better performance of ONESHOT compared to NLCDLASSO.

Next, we show our results using a differentiable link function. For this experiment, we consider an astronomical image illustrated in Fig. 1.5. This image includes two components; the "stars" component, which can be considered to be sparse in the identity basis (Φ), and the "galaxy" component which are sparse when they are expressed in the discrete cosine transform basis (Ψ). The superposition image $x = \Phi w + \Psi z$ is observed using a subsampled Fourier matrix with $m =$ 15000 rows multiplied with a diagonal matrix with random ± 1 entries (Krahmer and Ward, 2011). Further, each measurement is nonlinearly transformed by applying the (shifted) logistic function

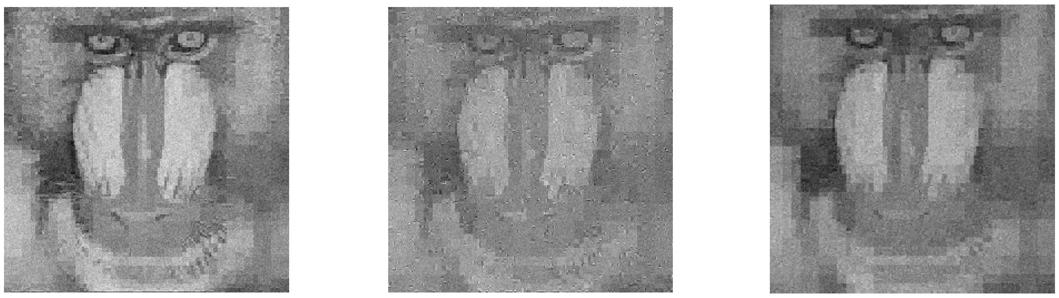


Figure 1.4: Comparison of OneShot and NlcdLASSO for real 2D image data from nonlinear under-sampled observations. Parameters: $n = 256 \times 256, s = 500, m = 35000, g(x) = sign(x)$.

$g(x) = \frac{1}{2}\frac{1-e^{-x}}{1+e^{-x}}$ as the link function. In the recovery procedure using DHT, we set the number of iterations to 1000 and step size η' to 150000. As is visually evident, our proposed DHT method is able to reliably recover the component signals.

1.6 Conclusion

In this chapter, we consider the problem of demixing sparse signals from their nonlinear measurements. We specifically study the more challenging scenario where only a limited number of nonlinear measurements of the superposition signal are available. As our primary contribution, we propose two fast algorithms for recovery of the constituent signals, and support these algorithms with the rigorous theoretical analysis to derive nearly-tight upper bounds on their sample complexity for achieving stable demixing.

We anticipate that the problem of demixing signals from nonlinear observations can be used in several different practical applications. As future work, we intend to extend our methods to more general signal models (including rank-sparsity models for matrix valued data), as well as robust recovery under more general nonlinear observation models.

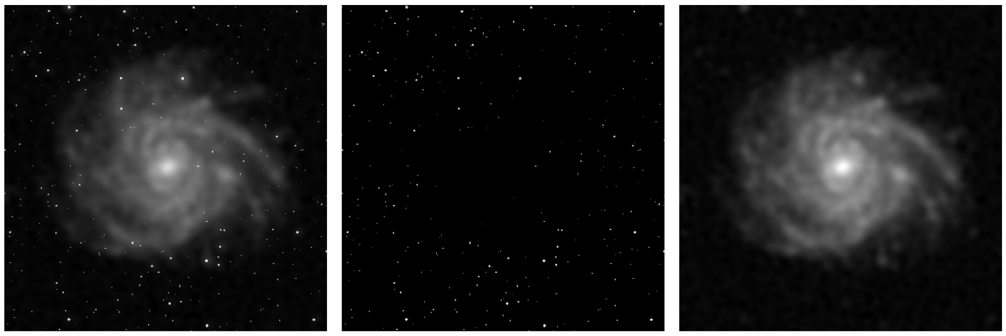

(a) Original x (b) $\Phi(\widehat{w})$ $\Psi(\widehat{z})$

Figure 1.5: Demixing a real 2-dimensional image from nonlinear observations with DHT. Parameters: $n = 512 \times 512, s = 1000, m = 15000, g(x) = \frac{1}{2}\frac{1-e^{-x}}{1+e^{-x}}$. Image credits: NASA, McCoy et al. (2014).

1.7 Appendix. Overview

In this section, we derive the proofs of our theoretical results stated in Section 1.4.

1.7.1 Appendix A. Analysis of OneShot

Our analysis mostly follows the techniques of (Plan et al., 2014). However, several additional complications in the proof arise due to the structure of the demixing problem. As a precursor, we need the following lemma from geometric functional analysis, restated from (Plan et al., 2014).

Lemma 1.17. *Assume K is a closed star-shaped set. Then for $u \in K$, and $a \in \mathbb{R}^n$, one has the following result $\forall\ t > 0$:*

$$\|\mathcal{P}_K(a) - u\|_2 \le \max\left(t, \frac{2}{t}\|a - u\|_{K_t^o}\right). \tag{1.18}$$

We also use the following result of (Plan et al., 2014).

Claim 1.18. *(Orthogonal decomposition of a_i.) Suppose we decompose the rows of A, a_i, as:*

$$a_i = \langle a_i, \bar{x}\rangle\bar{x} + b_i, \tag{1.19}$$

where $b_i \in \mathbb{R}^n$ is orthogonal to $\bar{x}$. Then we have $b_i \sim \mathcal{N}(0, I_{x^\perp})$ since $a_i \sim \mathcal{N}(0, I)$. Also, $I_{x^\perp} = I - \bar{x}\bar{x}^T$. Moreover, the measurements y_i in equation (1.6) and the orthogonal component b_i are statistically independent.

Proof of Theorem 1.10. Observe that the magnitude of the signal x may be lost due to the action of the nonlinear measurement function f (such as the sign$(\cdot)$ function). Therefore, our recovered signal $\widehat{x}$ approximates the true signal modulo a scaling factor. Indeed, for μ defined in Lemma 1.9, we have:

$$\begin{aligned}
\|\widehat{x} - \mu\bar{x}\|_2 &= \|\Phi\widehat{w} + \Psi\widehat{z} - \alpha\mu\Phi w - \alpha\mu\Psi z\|_2 \\
&\leq \|\Phi\|\|\widehat{w} - \mu\alpha w\|_2 + \|\Psi\|\|\widehat{z} - \mu\alpha z\|_2 \\
&\leq (\rho + \frac{2}{\rho}\|\Phi^*\widehat{x}_{\text{lin}} - \mu\alpha w\|_{K_\rho^o}) + (\rho + \frac{2}{\rho}\|\Psi^*\widehat{x}_{\text{lin}} - \mu\alpha z\|_{K_\rho^o}). \qquad (1.20)
\end{aligned}$$

The equality comes from the definition of $\bar{x}$. The first inequality results from an application of the triangle inequality and the definition of the operator norm of a matrix, while the second inequality follows from Lemma 1.17. Now, it suffices to derive a bound on the first term in the above expression (since a similar bound will hold for the second term). This proves the first part of Theorem 1.10. We have:

$$\begin{aligned}
\|\Phi^*\widehat{x}_{\text{lin}} - \mu\alpha w\|_{K_\rho^o} &= \|\Phi^*\frac{1}{m}\Sigma_i(y_i\langle a_i, \bar{x}\rangle\bar{x} + y_i b_i) - \mu\alpha w\|_{K_\rho^o} \\
&\leq \|\Phi^*\frac{1}{m}\Sigma_i(y_i\langle a_i, \bar{x}\rangle\bar{x}) - \mu\alpha w\|_{K_\rho^o} + \|\Phi^*\frac{1}{m}\Sigma_i y_i b_i\|_{K_\rho^o} \\
&\leq \underbrace{\|\Phi^*\frac{1}{m}\Sigma_i(y_i\langle a_i, \bar{x}\rangle\bar{x}) - \mu\Phi^*\bar{x}\|_{K_\rho^o}}_{S_1} + \underbrace{\|\mu\alpha\Phi^*\Psi z\|_{K_\rho^o}}_{S_2} \\
&\quad + \underbrace{\|\Phi^*\frac{1}{m}\Sigma_i y_i b_i\|_{K_\rho^o}}_{S_3}. \qquad (1.21)
\end{aligned}$$

The first equality follows from Claim 1.18, while the second and third inequalities result from the triangle inequality. Also:

$$\begin{aligned}
S_1 &= \|\Phi^* \frac{1}{m}\Sigma_i(y_i\langle a_i, \bar{x}\rangle \bar{x}) - \mu\Phi^*\bar{x}\|_{K_\rho^o} \\
&= \|(\frac{1}{m}\Sigma_i(y_i\langle a_i, \bar{x}\rangle - \mu))\Phi^*\bar{x}\|_{K_\rho^o} \\
&= |\frac{1}{m}\Sigma_i(y_i\langle a_i, \bar{x}\rangle - \mu)|\|\Phi^*\bar{x}\|_{K_\rho^o}. \\
&\implies \mathbb{E}(S_1^2) = \mathbb{E}(|\frac{1}{m}\Sigma_i(y_i\langle a_i, \bar{x}\rangle - \mu)|^2\|\Phi^*\bar{x}\|^2_{K_\rho^o}).
\end{aligned}$$

Define $\gamma_i \triangleq y_i\langle a_i, \bar{x}\rangle - \mu_i$. Then,

$$\begin{aligned}
\mathbb{E}(|\frac{1}{m}\Sigma_i(y_i\langle a_i, \bar{x}\rangle - \mu)|^2) &= \mathbb{E}(\frac{1}{m^2}(\Sigma_i\gamma_i)^2) \\
&= \mathbb{E}(\frac{1}{m^2}(\sum_{i=}^{m}\gamma_i^2 + \Sigma_{i\neq j}\gamma_i\gamma_j)) \\
&= \frac{1}{m^2}(\sum_{i=1}^{m}\mathbb{E}\gamma_i^2) = \frac{1}{m}\mathbb{E}\gamma_1^2 = \frac{\sigma^2}{m},
\end{aligned}$$

where σ^2 has been defined in Lemma 1.9. The third and last equalities follow from the fact that the y_i's are independent and identically distributed. Now, we bound $\|\Phi^*\bar{x}\|^2_{K_\rho^o}$ as follows::

$$\begin{aligned}
\|\Phi^*\bar{x}\|_{K_\rho^o} &= \sup_{u\in(K-K)\cap\rho B_n^2}\langle\Phi^*\bar{x}, u\rangle \\
&= \rho \sup_{\substack{v_1\in\frac{1}{\rho}K, v_2\in\frac{1}{\rho}K \\ \|v_i\|_2\leq 1, i=1,2}}\langle\Phi^*\bar{x}, v_1 - v_2\rangle \\
&\leq 2\rho \sup_{\substack{\|a\|_0\leq s \\ \|a\|_2\leq 1}}|\langle\Phi^*\bar{x}, a\rangle| \\
&\leq 2\rho(\sup_{\substack{\|a\|_0\leq s \\ \|a\|_2\leq 1}}|\langle\alpha w, a\rangle| + \sup_{\substack{\|a\|_0\leq s \\ \|a\|_2\leq 1}}|\langle\alpha\Phi^*\Psi z, a\rangle|) \\
&\leq 2\rho(\alpha\|w\|_2 + \sup_{\substack{\|a\|_0\leq s \\ \|a\|_2\leq 1}}|\langle\alpha\Psi z, \Phi a\rangle|) \leq 2\alpha\rho(\|w\|_2 + \|z\|_2\varepsilon).
\end{aligned}$$

The second inequality follows from (1.4) and the triangle inequality. The last inequality is resulted from an application of the Cauchy-Schwarz inequality and the definition of ε. As a result, we have:

$$\implies \mathbb{E}(S_1^2) \leq 4\frac{\alpha^2\rho^2\sigma^2}{m}(\|w\|_2 + \|z\|_2\varepsilon)^2. \tag{1.22}$$

Similarly we can bound S_2 as follows:

$$\begin{aligned}\mathbb{E}(S_2) = \mathbb{E}(\|\mu\alpha\Phi^*\Psi z\|_{K_\rho^o}) &= \mathbb{E}(|\mu\alpha|\|\Phi^*\Psi z\|_{K_\rho^o}) \\ &= |\mu\alpha|\|\Phi^*\Psi z\|_{K_\rho^o} = |\mu\alpha| \sup_{u\in(K-K)\cap\rho B_n^2} \langle \Psi z, \Phi u\rangle \\ &= |\mu\alpha|\rho \sup_{\substack{v_1\in\frac{1}{\rho}K, v_2\in\frac{1}{\rho}K \\ \|v_i\|_2\le 1, i=1,2}} \langle \Psi z, \Phi(v_1 - v_2)\rangle \le 2\mu\alpha\rho\|z\|_2\varepsilon. \qquad (1.23)\end{aligned}$$

Finally, we give the bound for S_3. Define $L \triangleq \frac{1}{m}\Sigma_i y_i b_i$. Then, we get: $\mathbb{E}(S_3) = \mathbb{E}\|\Phi^*\frac{1}{m}\Sigma_i y_i b_i\|_{K_\rho^o} = \mathbb{E}\|\Phi^* L\|_{K_\rho^o}$. Our goal is to bound $\mathbb{E}\|\Phi^* L\|_{K_\rho^o}$. Since y_i and b_i are independent random variables (as per Claim 1.18), we can use the law of conditional covariance and the law of iterated expectation. That is, we first condition on y_i, and then take expectation with respect to b_i. By conditioning on y_i, we have $L \sim \mathcal{N}(0, \beta^2 I_{x^\perp})$ where $I_{x^\perp} = I - \bar{x}\bar{x}^T$ is the covariance of vector b_i according to claim 1.18 and $\beta^2 = \frac{1}{m^2}\Sigma_i y_i^2$. Define $g_{x^\perp} \sim \mathcal{N}(0, I_{x^\perp})$. Therefore, $L \sim \beta g_{x^\perp}$ (L equivalent to $g_{x^\perp}$ in distribution). Putting everything together, we get:

$$\mathbb{E}(S_3) = \mathbb{E}\|\Phi^* L\|_{K_\rho^o} = \mathbb{E}\|\Phi^*\beta g_{x^\perp}\|_{K_\rho^o} = \beta\mathbb{E}\|\Phi^* g_{x^\perp}\|_{K_\rho^o}.$$

We need to extend the support of distribution of $g_{x^\perp}$ and consequently L from $x^\perp$ to $\mathbb{R}^n$. This follows from (Plan et al., 2014):

Claim 1.19. *Let g_E be a random vector which is distributed as $\mathcal{N}(0, I_E)$. Also, assume that $\Gamma : \mathbb{R}^n \to \mathbb{R}$ is a convex function. Then, for any subspace E of $\mathbb{R}^n$ such that $E \subseteq F$, we have:*

$$\mathbb{E}(\Gamma(g_E)) \le \mathbb{E}(\Gamma(g_F)).$$

Hence, we can orthogonally decompose $\mathbb{R}^n$ as $\mathbb{R}^n = D \oplus C$ where D is a subspace supporting $x^\perp$ and C is the orthogonal subspace onto it. Thus, $g_{\mathbb{R}^n} = g_D + g_C$ in distribution such that $g_D \sim \mathcal{N}(0, I_D)$, $g_C \sim \mathcal{N}(0, I_C)$. Also, $\|.\|_{K_\rho^o}$ is a convex function since it is a semi-norm. Hence,

$$\begin{aligned}\mathbb{E}_D\|\Phi^* g_D\|_{K_\rho^o} &= \mathbb{E}_D\|\Phi^* g_D + \mathbb{E}_C(g_C)\|_{K_\rho^o} \\ &= \mathbb{E}_D\|\mathbb{E}_{C|D}(\Phi^* g_D + g_c)\|_{K_\rho^o} \\ &\le \mathbb{E}_D\mathbb{E}_{C|D}\|\Phi^*(g_D + g_C)\|_{K_\rho^o} = \mathbb{E}\|\Phi^* g_{\mathbb{R}^n}\|_{K_\rho^o}.\end{aligned}$$

The first inequality follows from Jensen's inequality, while the second inequality follows from the law of iterated expectation. Therefore, we get:

$$\begin{aligned}\mathbb{E}\|\Phi^* L\|_{K_\rho^o} = \mathbb{E}\|\Phi^*\beta g_{x^\perp}\|_{K_\rho^o} = \beta\mathbb{E}\|\Phi^* g_{x^\perp}\|_{K_\rho^o} \\ \leq \beta\mathbb{E}\|\Phi^* g_{\mathbb{R}^n}\|_{K_\rho^o} = \beta \sup_{u\in(K-K)\cap\rho B_n^2} \langle \Phi^* g_{\mathbb{R}^n}, u\rangle = \beta W_\rho(K).\end{aligned}$$

The last equality follows from the fact that $\Phi^* g_{\mathbb{R}^n} \sim \mathcal{N}(0, I)$. The final step is to take an expectation with respect to y_i, giving us a bound as $\mathbb{E}(S_3) = \mathbb{E}\|\Phi^* L\|_{K_\rho^o} \leq \mathbb{E}(\beta)W_\rho(K) \leq \sqrt{\mathbb{E}(\beta^2)}W_\rho(K)$, where $\beta^2 = \frac{1}{m^2}\sum_{i=1}^m y_i^2$. Hence,

$$\mathbb{E}(S_3) \leq \frac{\eta}{\sqrt{m}} W_\rho(K)\,. \tag{1.24}$$

Putting together the results from (1.22), (1.23), and (1.24):

$$\begin{aligned}\mathbb{E}(\|\Phi^*\widehat{x}_{\text{lin}} - \mu\alpha w\|_{K_\rho^o}) &\leq \mathbb{E}(S_1) + \mathbb{E}(S_2) + \mathbb{E}(S_3) \\ &\leq \sqrt{\mathbb{E}(S_1)} + \mathbb{E}(S_2) + \mathbb{E}(S_3) \\ &\leq \frac{2\alpha\rho\sigma}{\sqrt{m}}\left(\|w\|_2 + \|z\|_2\varepsilon\right) + 2\mu\alpha\rho\|z\|_2\varepsilon + \frac{\eta}{\sqrt{m}}W_\rho(K).\end{aligned}$$

Therefore, we obtain:

$$\begin{aligned}\mathbb{E}\|\widehat{w} - \mu\alpha w\|_2 &\leq \rho + \frac{2}{\rho}\mathbb{E}(\|\Phi^*\widehat{x}_{lin} - \mu\alpha w\|_{K_\rho^o}) \\ &\leq \rho + \frac{4\alpha\sigma}{\sqrt{m}}\left(\|w\|_2 + \|z\|_2\varepsilon\right) + 4\mu\alpha\|z\|_2\varepsilon + \frac{2\eta}{\rho\sqrt{m}}W_\rho(K).\end{aligned} \tag{1.25}$$

Recall that $\alpha = \frac{1}{\|\Phi w + \Psi z\|_2}$, hence:

$$\begin{aligned}\|\Phi w + \Psi z\|_2^2 &\geq \|\Phi w\|_2^2 + \|\Psi z\|_2^2 - 2|\langle \Phi w, \Psi z\rangle| \\ &\geq \|w\|_2^2 + \|z\|_2^2 - 2\|w\|_2\|z\|_2\varepsilon,\end{aligned}$$

or,
$$\alpha \leq \frac{1}{\sqrt{\|w\|_2^2 + \|z\|_2^2 - 2\|w\|_2\|z\|_2\varepsilon}}.$$

$$\begin{aligned}\mathbb{E}\|\widehat{w} - \mu\alpha w\|_2 \leq \rho + \frac{4\sigma}{\sqrt{m}}\left(\frac{\|w\|_2 + \|z\|_2\varepsilon}{\sqrt{\|w\|_2^2 + \|z\|_2^2 - 2\|w\|_2\|z\|_2\varepsilon}}\right) \\ + 4\mu\left(\frac{\|z\|_2\varepsilon}{\sqrt{\|w\|_2^2 + \|z\|_2^2 - 2\|w\|_2\|z\|_2\varepsilon}}\right) + \frac{2\eta}{\rho\sqrt{m}}W_\rho(K).\end{aligned}$$

Now, let $d = \frac{\|z\|_2}{\|w\|_2}$. Hence:

$$\mathbb{E}\|\widehat{w} - \mu\alpha w\|_2 \leq \rho + \frac{4\sigma}{\sqrt{m}}\left(\frac{1+d\varepsilon}{\sqrt{1+d^2-2d\varepsilon}}\right) + 4\mu\left(\frac{d}{\sqrt{1+d^2-2d\varepsilon}}\right)\varepsilon + \frac{2\eta}{\rho\sqrt{m}}W_\rho(K). \qquad (1.26)$$

It is easy to see that $\frac{1+d\varepsilon}{\sqrt{1+d^2-2d\varepsilon}} \leq 2$ and $\frac{d}{\sqrt{1+d^2-2d\varepsilon}} \leq 2$ provided that $\varepsilon \leq 0.65$ (here, the constant 2 is just selected for convenience). Now, by plugging these bounds in (1.26), we obtain the desired result in Theorem 1.10. However, K is a closed star-shaped set (the set of s-sparse signals), and therefore $W_\rho(K) = \rho W_1(K)$ (Plan et al., 2014). Now using (1.20), we obtain:

$$\mathbb{E}\|\widehat{x} - \mu\bar{x}\|_2 \leq 2\rho + \frac{4}{\sqrt{m}}\left(4\sigma + \eta\frac{W_\rho(K)}{\rho}\right) + 16\mu\varepsilon.$$

We can use Lemma 2.3 in (Plan and Vershynin, 2013b) and plug in $W_\rho(K) \leq C\rho\sqrt{s\log(2n/s)}$. Using the above bound on α and by letting $\rho \to 0$, we get:

$$\mathbb{E}\|\widehat{x} - \mu\bar{x}\|_2 \leq \frac{4}{\sqrt{m}}\left(4\sigma + C\eta\sqrt{s\log(2n/s)}\right) + 16\mu\varepsilon, \qquad (1.27)$$

where $C > 0$ is an absolute constant. This completes the proof of Corollary 1.11. □

We now prove the high-probability version of the main theorem. As a precursor, we need a few preliminary definitions and lemmas:

Definition 1.20. *(Subexponential random variable.) A random variable X is subexponential if it satisfies the following relation:*

$$\mathbb{E}\exp\left(\frac{cX}{\|X\|_{\psi_1}}\right) \leq 2,$$

where $c > 0$ is an absolute constant. Here, $\|X\|_{\psi_1}$ denotes the ψ_1-norm, defined as follows:

$$\|X\|_{\psi_1} = \sup_{p\geq 1}\frac{1}{p}(\mathbb{E}|X|^p)^{\frac{1}{p}}.$$

We should mention that there are other definitions for subexponential random variables (also for subGaussian defined in Definition 1.3). Please see (Vershynin, 2010) for a detailed treatment.

Lemma 1.21. *Let X and Y be two subgaussian random variables. Then, XY is a subexponential random variable.*

Proof. According to the definition of the ψ_2-norm, we have:

$$(\mathbb{E}|XY|^p)^{\frac{1}{p}} = (\mathbb{E}|X|^p|Y|^p)^{\frac{1}{p}} \leq \left((\mathbb{E}|X|^{2p})^{\frac{1}{2p}}(\mathbb{E}|Y|^{2p})^{\frac{1}{2p}}\right) \leq \sqrt{2}p\|X\|_{\psi_2}\|Y\|_{\psi_2}, \tag{1.28}$$

where the first inequality results from Cauchy-schwarz inequality, and the last inequality is followed by the subgaussian assumption on X and Y. This shows that the random variable XY is subexponential random variable according to Definition 5.14. □

Lemma 1.22. *(Gaussian concentration inequality) See (Vershynin, 2010; Ledoux and Talagrand, 2013). Let $(G_x)_{x\in T}$ be a centered gaussian process indexed by a finite set T. Then $\forall t > 0$:*

$$\mathbb{P}(\sup_{x\in T} G_x \geq \mathbb{E}\sup_{x\in T} G_x + t)) \leq \exp\left(-\frac{t^2}{2\sigma^2}\right)$$

where $\sigma^2 = \sup_{x\in T}\mathbb{E}G_x^2 < \infty$.

Lemma 1.23. *(Bernstein-type inequality for random variables) (Vershynin, 2010). Let $X_1, X_2, \ldots, X_n$ be independent sub-exponential random variables with zero-mean. Also, assume that $K = \max_i \|X_i\|_{\psi_1}$. Then, for any vector $a \in \mathbb{R}^n$ and every $t \geq 0$, we have:*

$$\mathbb{P}(|\Sigma_i a_i X_i| \geq t) \leq 2\exp\left(-c\min\left\{\frac{t^2}{K^2\|a\|_2^2}, \frac{t}{K\|a\|_\infty}\right\}\right).$$

where $c > 0$ is an absolute constant.

Proof of Theorem 1.13. We follow the proof given in (Plan et al., 2014). Let $\beta = \frac{s'}{2\sqrt{m}}$ for $0 < s' < \sqrt{m}$ where m denotes the number of measurements. In (1.21), we saw that $\forall\rho > 0$:

$$\|\widehat{x} - \mu\bar{x}\|_2 \leq 2(\rho + \frac{2}{\rho}(S_1 + S_2 + S_3)). \tag{1.29}$$

We attempt to bound each term S_1, S_2, and S_3 with high probability, and then use a union bound to obtain the desired result.

For S_1, we have:

$$S_1 \leq |\frac{1}{m}\Sigma_i(y_i\langle a_i, \bar{x}\rangle - \mu)|\|\Phi^*\bar{x}\|_{K_t^o}.$$

We note that y_i is a sub-gaussian random variable (by assumption) and $\langle a_i, \bar{x}\rangle$ is a standard normal random variable. Hence, by Lemma 1.21, $y_i\langle a_i, \bar{x}\rangle$ is a sub-exponential random variable. Also,

$y_i\langle a_i, \bar{x}\rangle$ for $i = 1, 2, \ldots, m$ are independent sub-exponential random variables that can be centered by subtracting their mean μ. Now, we can apply Lemma 1.23 on $|\frac{1}{m}\Sigma_i(y_i\langle a_i, \bar{x}\rangle - \mu)|$. Therefore:

$$\mathbb{P}(|\frac{1}{m}\Sigma_i(y_i\langle a_i, \bar{x}\rangle - \mu))| \geq \eta\beta) \leq 2\exp\left(-\frac{c\beta^2\eta^2 m}{\|y_1\|_{\psi_2}^2}\right).$$

Here, η and μ are as defined in 1.9. Using the bound on $\|\Phi^*\bar{x}\|_{K_t^o}$, we have:

$$S_1 \leq 2\alpha\eta\beta\rho(\|w\|_2 + \|z\|_2\varepsilon), \tag{1.30}$$

with probability at least $1 - 2\exp(-\frac{c\beta^2\eta^2 m}{\|y_1\|_{\psi_2}^2})$ where $c > 0$ is some constant.

For S_2 we have:

$$S_2 \leq 2\mu\alpha\rho\|z\|_2\varepsilon, \tag{1.31}$$

with probability 1 since S_2 is a deterministic quantity.

For S_3 we have:

$$S_3 \leq \|\Phi^*\frac{1}{m}\Sigma_i y_i b_i\|_{K_\rho^o}.$$

To obtain a tail bound for S_3, we are using the following:

$$S_3 \leq \frac{1}{m}(\Sigma_i y_i^2)^{1/2}\|\Phi^* g\|_{K_\rho^o}.$$

We need to invoke the Bernstein Inequality (Lemma 1.23) for sub-exponential random variables $(y_i^2 - \eta^2)$ for $i = 1, 2, \ldots, m$ which are zero mean subexponential random variables in order to bound $\frac{1}{m}(\Sigma_i y_i^2)^{1/2}$. we have $\left|\frac{1}{m}\Sigma_i(y_i^2 - \eta^2)\right| \leq 3\eta^2$ with high probability $1 - 2\exp(-\frac{cm\eta^4}{\|y_1\|_{\psi_2}^4})$.

Next, we upper-bound $\|\Phi^* g\|$ (where $g \sim \mathcal{N}(0, I)$) with high probability. Since Φ is an orthogonal matrix, we have that $\Phi^* g \sim \mathcal{N}(0, I)$. Hence, we can use the Gaussian concentration inequality to bound $\Phi^* g$ as mentioned in Lemma 1.22. Putting these pieces together, we have:

$$S_3 \leq \frac{2\eta}{\sqrt{m}}\left(W_\rho(K) + \rho\beta\sqrt{m}\right), \tag{1.32}$$

with probability at least $1 - 2\exp(-\frac{cm\eta^4}{\|y_1\|_{\psi_2}^4}) - \exp(c\beta^2 m)$. Here, $W_\rho(K)$ denotes the local mean width for the set K_1 defining in Definition 1.1.

Now, combining (1.29), (1.30), (1.31), and (1.32) together with the union bound, we obtain:

$$\begin{aligned}\|\widehat{x} - \mu\bar{x}\|_2 &\leq \frac{4\alpha\eta s'(\|w\|_2 + \|z\|_2\varepsilon)}{\sqrt{m}} + 8\mu\alpha\|z\|_2\varepsilon + \frac{C\eta}{\sqrt{m}}\sqrt{s\log\frac{2n}{s}} + 4\frac{\eta s'}{\sqrt{m}}, \\ &\leq \frac{4\eta}{\sqrt{m}}\left(3s' + C'\sqrt{s\log\frac{2n}{s}}\right) + 16\mu\varepsilon, \qquad (1.33)\end{aligned}$$

with probability at least $1 - 4\exp(-\frac{cs'^2\eta^4}{\|y_1\|_{\psi_2}^4})$. The second inequality is due to the bounds being used in (1.26) provided that $\varepsilon \leq 0.65$. Also, $C, C', c > 0$ are absolute constants. Here, we have again used the well-known bound on the local mean width of the set of sparse vectors (for example, see Lemma 2.3 of (Plan and Vershynin, 2013b)). This completes the proof. □

1.7.2 Appendix B. Analysis of DHT

Our analysis of DHT occurs in two stages. First, we define a loss function $F(t)$ that depends on the nonlinear link function g and the measurement matrix A. We first assume that $F(t)$ satisfies certain regularity conditions (restricted strong convexity/smoothness), and use this to prove algorithm convergence. The proof of Theorem 1.14 follows the proof of convergence of the iterative hard thresholding (IHT) algorithm in the linear case (Blumensath and Davies, 2009), and is more closely related to the work of (Yuan et al., 2014a) who extended it to the nonlinear setting. Our derivation here differs from these previous works in our specific notion of restricted strong convexity/smoothness, and is relatively more concise. Later, we will prove that the RSS/RSC assumptions on the loss function indeed are valid, given a sufficient number of samples that obey certain measurement models. We assume a variety of measurement models including isotropic row measurements as well as subgaussian measurements. To our knowledge, these derivations of sample complexity are novel.

First, we state the definitions for restricted strong convexity and restricted strong smoothness, abbreviated as *RSC* and *RSS*. The RSC and RSS was first proposed by (Negahban et al., 2011; Raskutti et al., 2010); also, see (Bahmani et al., 2013b).

Definition 1.24. *A function f satisfies the RSC and RSS conditions if one of the following equivalent definitions is satisfied for all t_1, t_2 such that $\|t_1\|_0 \leq 2s$ and $\|t_2\|_0 \leq 2s$:*

$$\frac{m_{4s}}{2}\|t_2-t_1\|_2^2 \le f(t_2)-f(t_1)-\langle \nabla f(t_1), t_2-t_1\rangle \le \frac{M_{4s}}{2}\|t_2-t_1\|_2^2, \tag{1.34}$$

$$m_{4s}\|t_2-t_1\|_2^2 \le \langle \nabla f(t_2)-\nabla f(t_1), t_2-t_1\rangle \le M_{4s}\|t_2-t_1\|_2^2, \tag{1.35}$$

$$m_{4s} \le \|\nabla_\xi^2 f(t)\| \le M_{4s}, \tag{1.36}$$

$$m_{4s}\|t_2-t_1\|_2 \le \|\nabla_\xi f(t_2)-\nabla_\xi f(t_1)\|_2 \le M_{4s}\|t_2-t_1\|_2, \tag{1.37}$$

where $\xi = supp(t_1) \cup supp(t_2)$, $|\xi| \le 4s$. Moreover, m_{4s} and M_{4s} are called the RSC-constant and RSS-constant, respectively. We note that $\nabla_\xi f(t)$ denotes the gradient f restricted to set ξ. In addition, $\nabla_\xi^2 f(t)$ is a $4s \times 4s$ sub-matrix of the Hessian matrix $\nabla^2 f(t)$ comprised of row/column indices indexed by ξ.

Proof. (Equivalence of Eqs. (1.34), (1.35), (1.36), (1.37)). The proof of above equivalent definitions only needs some elementary arguments and we state them here for completeness. If we assume that (1.34) is given, then by exchanging t_1 and t_2 in (1.34), we have:

$$\frac{m_{4s}}{2}\|t_1-t_2\|_2^2 \le f(t_1)-f(t_2)-\langle \nabla f(t_2), t_1-t_2\rangle \le \frac{M_{4s}}{2}\|t_1-t_2\|_2^2, \tag{1.38}$$

by adding (1.38) with (1.34), inequality in (1.35) is resulted. Now, assume that (1.35) is given. Then we can set $t_2 = t_1 + \Delta(t_2-t_1)$ in (1.35) and then letting $\Delta \to 0$ results (1.36) according to the definition of second derivative. Next, if we assume that (1.36) is given, then we can invoke the *mean value theorem* (McLeod, 1965) for twice-differentiable vector-valued multivariate functions:

$$\nabla_\xi f(t_2) - \nabla_\xi f(t_1) = \int_0^1 P_\xi^T \nabla^2 f(ct_2 + (1-c)t_1)(t_2-t_1)dt.$$

where $c > 0$ and P_ξ denotes the identity matrix which its columns is restricted to set ξ with $\|\xi\|_0 \le 2s$. It follows that:

$$\begin{aligned}\big\|\nabla_\xi f(t_2) - \nabla_\xi f(t_1)\big\| &\le \int_0^1 \big\|P_\xi^T \nabla^2 f(ct_2 + (1-c)t_1)(t_2-t_1)\big\|dt\\ &\le M_{4s}\|(t_2-t_1)\|.\end{aligned}$$

where the last inequality follows by (1.36). Similarly, we can establish the lower bound in (1.37) by invoking the Cauchy Schwartz inequality in (1.35).

Finally, suppose that (1.36) holds. We can establish (1.34) by performing a Taylor expansion of $f(t)$. For upper bound in (1.34) and some $0 \leq c \leq 1$, we have:

$$\begin{aligned} f(t_2) &\leq f(t_1) - \langle \nabla f(t_1), t_2 - t_1 \rangle + \frac{1}{2}(t_2 - t_1)^T \nabla_\xi^2 f(ct_2 + (1-c)t_1)(t_2 - t_1) \\ &\leq f(t_1) - \langle \nabla f(t_1), t_2 - t_1 \rangle + \frac{M_{4s}}{2} \|t_1 - t_2\|_2^2. \end{aligned}$$

The lower bound in (1.34) also follows similarly. □

We now give a proof that DHT enjoys the linear convergence, as stated in Theorem 1.14. Recall that as opposed to the commonly used least-squares loss function, we instead define a special objective function:

$$F(t) = \frac{1}{m}\sum_{i=1}^{m} \Theta(a_i^T \Gamma t) - y_i a_i^T \Gamma t,$$

where $\Gamma = [\Phi \ \ \Psi]$, $t = [w; z] \in \mathbb{R}^{2n}$, and $\Theta'(x) = g(x)$. The gradient and Hessian of the objective function are given as follows:

$$\nabla F(t) = \frac{1}{m}\sum_{i=1}^{m} \Gamma^T a_i g(a_i^T \Gamma t) - y_i \Gamma^T a_i\,, \tag{1.39}$$

$$\nabla^2 F(t) = \frac{1}{m}\sum_{i=1}^{m} \Gamma^T a_i a_i^T \Gamma g'(a_i^T \Gamma t)\,. \tag{1.40}$$

We start with the projection step in Algorithm 1.2. In what follows, the superscript k denotes the k-th iteration. Let $t^{k+1} = [t_1^k; t_2^k] \in \mathbb{R}^{2n}$ be the constituent vector as the k^{th} iteration. Hence,

$$t^{k+1} = \mathcal{P}_{2s}\left(t^k - \eta' \nabla F(t^k)\right),$$

where η' denotes the step size in Algorithm 1.2 and $\mathcal{P}_{2s}(.)$ denotes the hard thresholding operation. Furthermore, $\nabla F(t^k)$ is the gradient of the objective function at iteration k. Moreover, we define sets S^k, S^{k+1}, S^* as follows, each of whose cardinalities is no greater than $2s$:

$$\text{supp}(t^k) = S^k,\ \text{supp}(t^{k+1}) = S^{k+1},\ \text{supp}(t^*) = S^*.$$

Moreover, define $S^k \cup S^{k+1} \cup S^* = J_k = J$ such that $|J| \leq 6s$. Define $b_k = t^k - \eta' \nabla_J F(t^k)$. Then,

$$\|t^{k+1} - t^*\|_2 \leq \|t^{k+1} - b\|_2 + \|b - t^*\|_2 \leq 2\|b - t^*\|_2, \tag{1.41}$$

where $t^* = [t_1^*; t_2^*] \in \mathbb{R}^{2n}$ such that $\|t^*\|_0 \leq 2s$ is the solution of the optimization problem in (2.3). The last inequality follows since t^{k+1} is generated by taking the $2s$ largest entries of $t^k - \eta' \nabla F(t^k)$; by definition of J (Please note that J depends on k, i.e., $J := J_k$), t^{k+1} also has the minimum Euclidean distance to b_k over all vectors with cardinality $2s$. Moreover:

$$\begin{aligned} \|b_k - t^*\|_2 &= \|t^k - \eta' \nabla_J F(t^k) - t^*\|_2 \\ &\leq \|t^k - t^* - \eta' \left(\nabla_J F(t^k) - \nabla_J F(t^*) \right) \|_2 + \eta' \|\nabla_J F(t^*)\|_2. \end{aligned} \tag{1.42}$$

Now, by invoking RSC and RSS in the Definition 4.1, we have:

$$\|t^k - t^* - \eta' \left(\nabla_J F(t^k) - \nabla_J F(t^*) \right) \|_2^2 \leq (1 + {\eta'}^2 M_{6s}^2 - 2\eta' m_{6s}) \|t^k - t^*\|_2^2,$$

where M_{6s} and m_{6s} denote the RSC and RSS constants. The above inequality follows by the upper bound of (1.37) and the lower bound of (1.35) in Definition 4.1 with the restriction set ξ chosen as J. Now let $q = \sqrt{1 + {\eta'}^2 M_{6s}^2 - 2\eta' m_{6s}}$. By (1.41) and (1.42), we have:

$$\|t^{k+1} - t^*\|_2 \leq 2q \|t^k - t^*\|_2 + 2\eta' \|\nabla_J F(t^*)\|_2. \tag{1.43}$$

In order for the algorithm to exhibit linear convergence, we need to have $2q < 1$. That is,

$${\eta'}^2 M_{6s}^2 - 2\eta' m_{6s} + \frac{3}{4} < 0.$$

By solving this quadratic inequality with respect to η', we obtain that η', m_{6s}, and M_{6s} should satisfy

$$1 \leq \frac{M_{6s}}{m_{6s}} \leq \frac{2}{\sqrt{3}}, \quad \frac{0.5}{M_{6s}} < \eta' < \frac{1.5}{m_{6s}}.$$

The bound in (1.43) shows that after enough iterations the first term vanishes and the quality of estimation depends on the vanishing speed of the second term, $2\eta' \|\nabla_J F(t^*)\|_2$ that is determined by the number of measurements. To bound the gradient in second term, $\|\nabla_J F(t^*)\|_2$, we need the following lemma:

Lemma 1.25. *(Khintchine inequality (Vershynin, 2010).) Let X_i be a finite number of independent and zero mean subgaussian random variables with unit variance. Assume that $\|X_i\|_{\psi_2} \leq r$. Then, for any real b_i and $p \geq 2$:*

$$\left(\sum_i b_i^2\right)^{\frac{1}{2}} \leq \left(\mathbb{E}|\sum_i b_i X_i|^p\right)^{\frac{1}{p}} \leq Cr\sqrt{p}\left(\sum_i b_i^2\right)^{\frac{1}{2}}.$$

Recall that our measurement model is given by:

$$y_i = g(a_i^T \Gamma t) + e_i, \quad i = 1, \dots, m.$$

As mentioned above, we assume that e_i represents the additive subgaussian noise with $\|e_i\|_{\psi_2} \leq \tau$ for $i = 1 \dots m$.

We leverage the Khintchine inequality to bound $\mathbb{E}\|\nabla_J F(t^*)\|_2$ under the subgaussian assumption on e_i. Denoting by $(\nabla_J F(t^*))_k$ as the k^{th} entry of the gradient (restricted to set J), from the Khintchine inequality, and for each $k = 1, \dots, |J|$, we have:

$$\begin{aligned}
\left(\mathbb{E}\left|(\nabla_J F(t^*))_k\right|^2\right)^{\frac{1}{2}} &\overset{r_1}{=} \left(\mathbb{E}\left(\frac{1}{m}\sum_{i=1}^m (\Gamma_J)_k^T a_i e_i\right)^2\right)^{\frac{1}{2}} \\
&\overset{r_2}{\leq} \frac{1}{m}\mathbb{E}\left(C\tau\sqrt{2}\left(\sum_{i=1}^m \left((\Gamma_J)_k^T a_i\right)^2\right)^{\frac{1}{2}}\right) \\
&\leq \frac{1}{m}C\tau\sqrt{2}\left(\sum_{i=1}^m (\Gamma_J)_k^T \mathbb{E}\left(a_i a_i^T\right)(\Gamma_J)_k\right)^{\frac{1}{2}} \\
&\overset{r_3}{=} \frac{C\tau\sqrt{2}}{\sqrt{m}},
\end{aligned} \tag{1.44}$$

where Γ_J denotes the restriction of the columns of the dictionary to set J with $|J| \leq 6s$ such that $3s$ of the columns are selected from each basis of the dictionary. Here, r_1 follows from (1.39), r_2 follows from the Khintchine inequality with $p = 2$ and the fact that e_i are independent from a_i. Finally, r_3 holds since the rows of A are assumed to be isotropic random vectors. Now, we can bound $\mathbb{E}\|\nabla_J F(t^*)\|_2$ as follows:

$$\mathbb{E}\|\nabla_J F(t^*)\|_2 \leq \sqrt{\mathbb{E}\|\nabla_J F(t^k)\|_2^2} \leq C'\tau\sqrt{\frac{s}{m}}, \tag{1.45}$$

where $C' > 0$ is an absolute constant and the last inequality is followed by (1.44) and the fact that $\|J\|_0 \leq 6s$.

Proof of Theorem 1.14. By using induction on (1.43), taking expectations, and finally using the bound stated in (1.45), we obtain the desired bound in Theorem 1.14 as follows:

$$\begin{aligned} \|t^{k+1} - t^*\|_2 &\leq (2q)^k \|t^0 - t^*\|_2 + \frac{2\eta'}{1-2q}\|\nabla_J F(t^*)\|_2 \\ &\leq (2q)^k \mathbb{E}\|t^0 - t^*\|_2 + C\tau\sqrt{\frac{s}{m}}, \end{aligned} \tag{1.46}$$

where $C > 0$ is a constant which depends only on the step size, η' and q. Also, t^0 denotes the initial value for the constituent vector, t. In addition, in the noiseless case ($\tau = 0$), if we denote κ as the desired accuracy for solving optimization problem (2.3), then the number of iterations to achieve the accuracy κ is given by $N = \mathcal{O}(\log \frac{\|t^0 - t^*\|_2}{\kappa})$. □

In the above convergence analysis of DHT, we assumed that objective function in (2.3), $F(t)$ satisfies the RSC/RSS conditions. In this section, we validate this assumption via the proofs for Theorems 1.15 and 1.16. As discussed above, we separately analyze two cases.

1.7.2.1 Case (a): isotropic rows of A

We first consider the case where the rows of the measurement matrix A are sampled from an isotropic probability distribution in $\mathbb{R}^n$. Specifically, we make the following assumptions on A:

1. the rows of A are independent isotropic vectors. That is, $\mathbb{E}a_i a_i^T = I_{n\times n}$ for $i = 1\ldots m$.
2. $\|a_i^T \Gamma_\xi\|_\infty \leq \vartheta$ for $i = 1\ldots m$.

Remark 1.26. *Assumption 2 is unavoidable in our analysis, and indeed this is one of the cases where our derivation differs from existing proofs. The condition $||a_i^T\Gamma_\xi||_\infty \leq \vartheta$ requires that all entries in $A\Gamma_\xi$ are bounded by some number ϑ. In other words, ϑ captures the cross-coherence between the measurement matrix, A and the dictionary $\Gamma_\xi = [\Phi\ \Psi]_\xi$ and controls the interaction* *between these two matrices. Without this assumption, one can construct a counter-example with the Hessian of the objective to be zero with high probability (for instance, consider partial DFT matrix as the measurement matrix A and $\Gamma_\xi = [I \ \Psi]_\xi$ with Ψ being the inverse DFT basis).*

Modifying (1.40), we define the *restricted* Hessian matrix as a $4s \times 4s$ sub-matrix of the Hessian matrix:

$$\nabla_\xi^2 F(t) = \frac{1}{m}\sum_{i=1}^{m} \Gamma_\xi^T a_i a_i^T \Gamma_\xi g'(a_i^T \Gamma t), \quad \|\xi\|_0 \leq 4s. \tag{1.47}$$

Here, Γ_ξ is the restriction of the columns of the dictionary $\Gamma = [\Phi \ \Psi]$ with respect to set ξ, such that $2s$ columns are selected from each basis. Let $S_i = \Gamma_\xi^T a_i a_i^T \Gamma_\xi g'(a_i^T \Gamma t), i = 1 \ldots m$. As per our assumption in Section 1.3, the derivative of the link function, $g(x)$ satisfies $0 < l_1 \leq g'(x) \leq l_2$. By this assumption, it is guaranteed that $\lambda_{\min}(S_i) \geq 0, i = 1 \ldots m$; this follows since $\Gamma_\xi^T a_i a_i^T \Gamma_\xi$ is a positive semidefinite matrix and $g' > 0$, we have $\lambda_{\min}(S_i) = \lambda_{\min}(\Gamma_\xi^T a_i a_i^T \Gamma_\xi) g' \geq 0$.

Let $\Lambda_{\max} = \max\limits_{\xi} \lambda_{\max}(\nabla_\xi^2 F(t))$ and $\Lambda_{\min} = \min\limits_{\xi} \lambda_{\min}(\nabla_\xi^2 F(t))$ where $\lambda_{\min}$ and $\lambda_{\max}$ denote the minimum and maximum eigenvalues of the restricted Hessian matrix. Furthermore, let U be any index set with $|U| \leq 6s$ such that $\xi \subseteq U$. We have:

$$l_1 \min_U \lambda_{\min}\left(\frac{1}{m}\sum_{i=1}^{m} \Gamma_U^T a_i a_i^T \Gamma_U\right) \leq \Lambda_{\min} \leq \Lambda_{\max} \leq l_2 \max_U \lambda_{\max}\left(\frac{1}{m}\sum_{i=1}^{m} \Gamma_U^T a_i a_i^T \Gamma_U\right).$$

Here, Γ_U is the restriction of the columns of Γ with respect to a set U such that $3s$ columns is selected from each basis. By taking expectations, we obtain:

$$l_1 \mathbb{E} \min_U \lambda_{\min}\left(\frac{1}{m}\sum_{i=1}^{m} \Gamma_U^T a_i a_i^T \Gamma_U\right) \leq \mathbb{E}\Lambda_{\min} \leq \mathbb{E}\Lambda_{\max} \leq l_2 \mathbb{E} \max_U \lambda_{\max}\left(\frac{1}{m}\sum_{i=1}^{m} \Gamma_U^T a_i a_i^T \Gamma_U\right). \tag{1.48}$$

Inequality in (1.48) shows that for proving RSC and RSS, we need to bound the expectation of the maximum and minimum eigenvalues of $\frac{1}{m}\sum_{i=1}^{m} \Gamma_\xi^T a_i a_i^T \Gamma_U$ over sets U with $|U| \leq 6s$. We should mention that (1.48) establishes RSC/RSS constants in expectation. One can establish RSC/RSS in tail probability using results in (Rudelson and Vershynin, 2008; Ledoux and Talagrand, 2013).

As our main tool for bounding the RSC/RSS constants, we use the *uniform Rudelson's inequality* (Vershynin, 2010; Rudelson and Vershynin, 2008).

Lemma 1.27. *(Uniform Rudelson's inequality) Let x_i be vectors in $\mathbb{R}^n$ for $i = 1, \ldots, m$ and $m \leq n$. Also assume that the entries of x_i's are bounded by ϑ, that is, $\|x_i\|_\infty \leq \vartheta$. Let h_i denote independent Bernoulli random variables (with parameter 1/2) for $i = 1 \ldots m$. Then for every set $\Omega \subseteq [n]$, we have:*

$$\mathbb{E} \max_{|\Omega| \leq n} \left\| \sum_{i=1}^{m} h_i (x_i)_\Omega (x_i)_\Omega^T \right\| \leq C_\vartheta l \sqrt{|\Omega|} \max_{|\Omega| \leq n} \left\| \sum_{i=1}^{m} (x_i)_\Omega (x_i)_\Omega^T \right\|^{\frac{1}{2}}, \tag{1.49}$$

where $(x_i)_\Omega$ denotes the restriction of x_i to Ω, $l = \log(|\Omega|)\sqrt{\log m}\sqrt{\log n}$, and C_ϑ denotes the dependency of C only on ϑ.

Before using the above result, we need to restate the uniform version of the standard symmetrization technique (Lemma 5.70 in (Vershynin, 2010)):

Lemma 1.28. *(Uniform symmetrization) Let x_{ik}, $i = 1 \ldots m$ be independent random vectors in some Banach space where indexed by some set Ξ such that $k \in \Xi$. Also, assume that h_i, $i = 1 \ldots m$ denote independent Bernoulli random variables (with parameter 1/2) for $i = 1 \ldots m$. Then,*

$$\mathbb{E} \sup_{k \in \Xi} \left\| \sum_{i}^{m} (x_{ik} - \mathbb{E} x_{ik}) \right\| \leq 2\mathbb{E} \sup_{k \in \Xi} \left\| \sum_{i}^{m} h_i x_{ik} \right\|. \tag{1.50}$$

Now we apply the Uniform Rudelson's inequality on $\lambda_{\max}\left(\frac{1}{m}\sum_{i=1}^{m} \Gamma_U^T a_i a_i^T \Gamma_U\right)$ over all set U with $|U| \leq 6s$. We have:

$$\begin{aligned} R \triangleq \mathbb{E} \max_U \left\| \frac{1}{m} \sum_{i=1}^{m} \Gamma_U^T a_i a_i^T \Gamma_U - \Gamma_U^T \Gamma_U \right\| &\overset{r_1}{\leq} 2\mathbb{E} \max_U \left\| \frac{1}{m} \sum_{i=1}^{m} h_i \Gamma_U^T a_i a_i^T \Gamma_U \right\| \\ &\overset{r_2}{\leq} \frac{C_\vartheta l \sqrt{6s}}{\sqrt{m}} \mathbb{E} \max_U \left\| \frac{1}{m} \sum_{i=1}^{m} \Gamma_U^T a_i a_i^T \Gamma_U \right\|^{\frac{1}{2}}, \end{aligned} \tag{1.51}$$

where r_1 follows from Lemma 1.28 with h_i defined in this lemma and r_2 follows from (1.49). In addition $l = \log(6s)\sqrt{\log m}\sqrt{\log 2n}$. Then by application of a triangle inequality, we have:

$$\mathbb{E} \max_U \left\| \frac{1}{m} \sum_{i=1}^{m} \Gamma_U^T a_i a_i^T \Gamma_U \right\| \leq R + \max_U \left\| \Gamma_U^T \Gamma_U \right\|.$$

On the other hand by Cauchy-Schwarz inequality, we get:

$$\mathbb{E}\max_U \left\| \frac{1}{m}\sum_{i=1}^m \Gamma_U^T a_i a_i^T \Gamma_U \right\|^{\frac{1}{2}} \leq \left(\mathbb{E}\max_U \left\| \frac{1}{m}\sum_{i=1}^m \Gamma_U^T a_i a_i^T \Gamma_U \right\| \right)^{\frac{1}{2}}$$

By combining the above inequalities, we obtain:

$$R \leq \frac{C'_\vartheta l\sqrt{s}}{\sqrt{m}} \left(R + \max_U \left\| \Gamma_U^T \Gamma_U \right\| \right)^{\frac{1}{2}}, \tag{1.52}$$

where C'_ϑ depends only on ϑ. This inequality is a quadratic inequality in terms of R and is easy to solve. By noting $\beta = \max_U \left\| \Gamma_U^T \Gamma_U \right\|$, we can write (1.52) as $\frac{R}{\beta} \leq \frac{C'_\vartheta l\sqrt{s}}{\sqrt{m}} \frac{1}{\beta} \left(1 + \frac{R}{\beta}\right)^{\frac{1}{2}}$. Now we can consider two cases; either $\frac{R}{\beta} \leq 1$, or $\frac{R}{\beta} > 1$. As a result, we have:

$$R \leq \max\left(\delta_0 \left(\max_U \left\| \Gamma_U^T \Gamma_U \right\| \right)^{\frac{1}{2}}, \delta_0^2 \right), \tag{1.53}$$

where $\delta_0 = \frac{C'_\vartheta l\sqrt{s}}{\sqrt{m}}$. In addition, we can use the Gershgorin Circle Theorem (Horn and Johnson, 2012) to bound $\lambda_{\max}(\Gamma_U^T\Gamma_U) = \|\Gamma_U^T\Gamma_U\|$ and $\lambda_{\min}(\Gamma_U^T\Gamma_U)$. This follows since:

$$\Gamma_U^T\Gamma_U = \begin{bmatrix} I & \Phi^T\Psi \\ \\ \Psi^T\Phi & I \end{bmatrix}_{6s\times 6s},$$

and hence we have:

$$\left| \lambda_i(\Gamma_U^T\Gamma_U) - 1 \right| \leq (6s-1)\gamma, \quad i = 1\ldots 6s,$$

where γ denotes the mutual coherence of Γ. Hence, the following holds for all index set U:

$$1 - (6s-1)\gamma \leq \lambda_{\min}(\Gamma_U^T\Gamma_U) \leq \lambda_{\max}(\Gamma_U^T\Gamma_U) \leq 1 + (6s-1)\gamma, \tag{1.54}$$

provided that $\gamma \leq \frac{1}{6s-1}$ to have nontrivial lower bound.

Proof of Theorem 1.15. If we choose $m \geq \left(\frac{C''_\vartheta}{\delta^2} s\log(n)\log^2 s \log\left(\frac{1}{\delta^2} s\log(n)\log^2 s\right)(1+(6s-1)\gamma) \right)$ in (1.53), then we have $R \leq \delta$ for some $\delta \in (0,1)$ and $C''_\vartheta > 0$ which depends only on ϑ. If $s = o(1/\gamma)$, then we obtain the stated sample complexity in Theorem 1.15. □

1.7.2.2 Case (b): isotropic subgaussian rows of A

Now, suppose that the measurement matrix A has independent isotropic subgaussian rows. We show that under this assumption, one can obtain better sample complexity bounds compared to the previous case. We use the following argument (which is more or less standard; see (Rauhut et al., 2008; Mendelson et al., 2008; Candes et al., 2011)). Let $\Gamma = [\Phi \ \Psi]$, and let $B_U = A\Gamma_U$ for any fixed $|U| \leq 6s$, where $3s$ elements are chosen from each basis. According to the notation from Section 1.7.2, we have:

$$l_1 \min_U \lambda_{\min}\left(\frac{1}{m}B_U^T B_U\right) \leq \Lambda_{\min} \leq \Lambda_{\max} \leq l_2 \max_U \lambda_{\max}\left(\frac{1}{m}B_U^T B_U\right). \tag{1.55}$$

where l_1, l_2 are upper and lower bounds on the derivative of the link function. Therefore, all we need to do is to bound the maximum and minimum singular values of $\frac{1}{\sqrt{m}}B_U$. To do so, we use the fact that if the rows of A are m independent copies of an isotropic vector with bounded ψ_2 norm, then the following holds for any fixed vector $v \in \mathbb{R}^{2n}$:

$$\left|\frac{1}{m}\left\|Bv\right\|_2^2 - \left\|\Gamma v\right\|_2^2\right| \leq \max(\delta_0, \delta_0^2) \triangleq \varepsilon', \tag{1.56}$$

with high probability where $\delta_0 = C\sqrt{\frac{6s}{m}} + \frac{t}{\sqrt{m}}$ for some absolute constant $C > 0$ (Mendelson et al., 2008) and $\forall t > 0$. Now fix any set U as above. Then, one can show using a covering number argument (for example, Lemma 2.1 in (Rauhut et al., 2008)) with $\frac{1}{4}$-net $(\mathcal{N}_{\frac{1}{4}})$ of the unit sphere and applying the upper bound in (1.54) for any $v \in U$, we get:

$$\mathbb{P}\left(\left|\frac{1}{m}\|B_U v\|_2^2 - \|\Gamma_U v\|_2^2\right| \geq \frac{\varepsilon'}{2}\right) \leq 2(9)^{6s}\exp\left(-\frac{c}{(1+(6s-1)\gamma)^2}\delta_0^2 m\right)$$

where $c > 0$ is a constant. Taking a union bound over all possible subsets U with $|U| \leq 6s$ and choosing $t = C_1\left(1+(6s-1)\gamma\right)^2\sqrt{s\log\frac{en}{6s}} + u\sqrt{m}$ in δ_0 where $C_1 > 0$ (absolute constant) and $u > 0$ are arbitrary small constants, we obtain:

$$\mathbb{P}\left(\max_U \max_{v\in\mathcal{N}_{\frac{1}{4}}}\left|\frac{1}{m}\|Bv\|_2^2 - \|\Gamma_U v\|_2^2\right| \geq \frac{\varepsilon'}{2}\right) \leq 2\exp\left(-c_2 u^2 m\right),$$

where $c_2 > 0$ is an absolute constant. By plugging t in the expression of δ_0 and letting $u \leq \delta/2$ and m sufficiently large, we have $\delta_0 \leq \delta$ for some $\delta \in (0,1)$. As a result, from (1.56), we have:

$$\max_U \left\|\frac{1}{m}B_U^T B_U - \Gamma_U^T\Gamma_U\right\| \leq \delta, \tag{1.57}$$

with probability at least $1 - 2\exp\left(-c_2 u^2 m\right)$. Therefore, for sufficiently large m (that we specify below), the following holds with high probability:

$$\lambda_{\min}\left(\Gamma_U^T \Gamma_U\right) - \delta \leq \lambda_{\min}\left(\frac{1}{m} B_U^T B_U\right) \leq \lambda_{\max}\left(\frac{1}{m} B_U^T B_U\right) \leq \lambda_{\max}\left(\Gamma_U^T \Gamma_U\right) + \delta$$

We use (1.54) to bound $\lambda_{\max}(\Gamma_U^T \Gamma_U) = \|\Gamma_U^T \Gamma_U\|$ and $\lambda_{\min}(\Gamma_U^T \Gamma_U)$; as a result,

$$1 - (6s-1)\gamma - \delta \leq \lambda_{\min}\left(\frac{1}{m} B_U^T B_U\right) \leq \lambda_{\max}\left(\frac{1}{m} B_U^T B_U\right) \leq 1 + (6s-1)\gamma + \delta \tag{1.58}$$

Thus, we obtain the desired bound in (1.55). That is:

$$l_1\left(1 - (6s-1)\gamma - \delta\right) \leq \Lambda_{\min} \leq \Lambda_{\max} \leq l_2\left(1 + (6s-1)\gamma - \delta\right). \tag{1.59}$$

holds with high probability for some $0 < \delta < 1 - (6s-1)\gamma$.

Proof of Theorem 1.16. The probability of failure of the above statement can be vanishingly small if we set $m \geq \frac{C'}{\delta^2} s \log \frac{n}{s}$ for some $\delta \in (0,1)$ and absolute constant $C' > 0$. Note that we only obtain nontrivial upper and lower bounds on $\Lambda_{\min}, \Lambda_{\max}$ if $\gamma \leq \frac{1}{6s-1}$. Assuming constant δ and coherence γ inversely proportional to s, we obtain the required sample complexity of DHT as: $m = \mathcal{O}\left(s \log \frac{n}{s}\right)$. □

For both cases (a) and (b), RSC and RSS constants follow by setting $M_{6s} \leq l_2\left(1 + (6s-1)\gamma + \delta\right)$ and $m_{6s} \geq l_1\left(1 - (6s-1)\gamma - \delta\right)$. As we discussed in the begging of section 1.7.2, we require that $\frac{0.5}{M_{6s}} < \eta' < \frac{1.5}{m_{6s}}$ in order to establish linear convergence of *DHT*. Hence, for linear convergence, the step size must satisfy:

$$\frac{0.5}{l_2\left(1 + (6s-1)\gamma + \delta\right)} < \eta' < \frac{1.5}{l_1\left(1 - (6s-1)\gamma - \delta\right)}$$

for some $0 < \delta < 1 - (6s-1)\gamma$.

CHAPTER 2. DEMIXING STRUCTURED SUPERPOSITIONS IN HIGH DIMENSIONS WITH PERIODIC AND APERIODIC NONLINEAR FUNCTIONS

In chapter 1, the demixing problem of constituent components has been considered with focus on two issues. First, we assume that the constituent components have arbitrary sparse representations in some incoherent dictionaries. Then, in the second scenario in which the link function is assumed to be known, we consider the monotonic nonlinear link function. In this chapter, we target both of these issues. In particular, we study certain families of structured sparsity models in the constituent components, and propose a method which provably recovers the components given (nearly) $m = \mathcal{O}(s)$ samples where s denotes the sparsity level of the underlying components. This strictly improves upon previous nonlinear demixing techniques and asymptotically matches the best possible sample complexity. Regarding the second issue, we study the bigger class of nonlinear link functions and consider demixing problem from a limited number of nonlinear observations where this nonlinearity is due to a either periodic function or aperiodic function. For both of these directions, we provide a range of simulations to illustrate the performance of the proposed algorithms.

2.1 Introduction

2.1.1 Motivation

As we discussed in the previous chapter, the *demixing* problem involves disentangling two (or more) high-dimensional vectors from their linear superposition. In applications involving signal recovery, such superpositions can be used to model situations when there is some ambiguity in the components (e.g., the true components can be treated as “ground truth” + “outliers”) or when there is some existing prior knowledge that the true underlying vector is a superposition of two components. Similar to chapter 1, we focus on the sample-poor regime where the dimension far exceeds the number of measurements. Hence, our observation model is given by:

$$y = g(X\beta) + e = g(X(\Phi\theta_1 + \Psi\theta_2)) + e, \tag{2.1}$$

where g denotes a nonlinear *link* function and e denotes observation noise. This is akin to the *Generalized Linear Model* (GLM) as well as the *Single Index Model* (SIM) from statistical learning (Kakade et al., 2011). Reconstruction of high dimensional vectors in the GLM setting has been an intense focus of study in the statistical learning literature in recent years; applications of such recovery methods span scalable machine learning (Kakade et al., 2011), 1-bit compressive sensing (Boufounos and Baraniuk, 2008; Jacques et al., 2013), and imaging (Candes et al., 2015). In signal processing applications, observation models with periodic link functions have been proposed for scalar quantization (Boufounos, 2012) as well as computational imaging (Zhao et al., 2015).

2.1.2 Summary of Contributions

In this chapter, we focus on the case where the components θ_1, θ_2 obey certain *structured sparsity* assumptions. Structured sparsity models are useful in applications where the support patterns (i.e., the coordinates of the nonzero entries) belong to model-specific restricted families (for example, the support is assumed to be *group-sparse* (Huang and Zhang, 2010)). It is known that such assumptions can significantly reduce the required number of samples for estimating the underlying signal, compared to generic sparsity assumptions (Baraniuk et al., 2010; Hegde et al., 2015a). In addition, we consider two classes of link functions: *aperiodic* and *periodic* functions, and accordingly, two different demixing approaches.

In the aperiodic case, we follow the setup of second scenario (section 1.4.2) in chapter 1 where g is assumed to be monotonic; satisfies some type of *restricted strong convexity* (RSC) (Negahban et al., 2011); and some type of *restricted strong smoothness* (RSS) assumptions (Yang et al., 2015). For this case, we develop a non-convex iterative algorithm that stably estimates the components θ_1 and θ_2. In the periodic case, specifically, we let the designing matrix, X be *factorized* as $X = DB$, where $D \in \mathbb{R}^{m\times q}$, and $B \in \mathbb{R}^{q\times n}$ have some specific structures; please see Section 2.2

for details. Again, for this case, we demonstrate a novel two-stage algorithm that stably estimate the components θ_1 and θ_2. In particular, we provide a theoretical sample complexity analysis. Our theory demonstrates that stable recovery from nonlinear sinusoidal observations can be done with *essentially* the same number of samples as standard sparse recovery with linear observations. The only additional cost is a logarithmic factor overhead in the dimension of the embedding that depends on n and the Euclidean norm of the original data vector.

For both cases considered above, we show that under certain sufficiency conditions, the performance of our methods strictly improves upon previous nonlinear demixing techniques, and asymptotically matches (close to) the best possible sample-complexity. We also support the theories and the algorithms via a range of numerical experiments.

2.1.3 Prior Work

Demixing approaches in high dimensions with structured sparsity assumptions have appeared before in the literature (McCoy and Tropp, 2014; McCoy et al., 2014; Rao et al., 2014). However, our method differs from these earlier works in a few different aspects. The majority of these methods involve solving a convex relaxation problem; in contrast, our algorithm is manifestly *non-convex*. Despite this feature, for certain types of structured superposition models, our method provably recovers the components given (nearly) $m = \mathcal{O}(s)$ samples. Moreover, these earlier methods have not explicitly addressed the nonlinear observation model (with the exception of (Plan et al., 2014)). In this chapter, we leverage the structured sparsity assumptions to our advantage, and show that this type of structured sparsity priors significantly decreases the sample complexity (both for periodic and aperiodic nonlinearities) for estimating the signal components.

To study periodic nonlinearities, our approach relies upon ideas from several related problems in machine learning and signal processing. The technique of using random projections in conjunction with sinusoidal nonlinearities has emerged as a popular alternative for kernel-based inference (Vedaldi and Zisserman, 2012; Shi et al., 2009; Le et al., 2013; Lopez-Paz et al., 2014). However, these works do not explicitly consider recovering the original data from the nonlinear random projections. While there has been considerable recent interest in reconstructing from nonlinear measurements (features), the majority of the methods deal with the case where the nonlinearity is either monotonic (Kakade et al., 2011) or satisfies restricted regularity conditions (Bahmani et al., 2013a; Yang et al., 2015). None of the proposed methods in the literature can be directly applied for the specific problem of stable sparse reconstruction from random sinusoidal features, with one exception: the theoretical framework developed in (Plan et al., 2014) can indeed be adapted to our problem. However, the recovery algorithm only yields an estimate up to a scalar ambiguity, which can be problematic in applications. Moreover, even if scalar ambiguities can be tolerated, we show in our experimental results below that the sample complexity of this method is far too high in practice. In contrast, our proposed method yields stable and accurate results with only a mild factor increase in sample complexity when compared to standard sparse recovery.

2.2 Preliminaries

We first introduce some notations. Let $\|.\|_q$ denote the ℓ_q-norm of a vector. Denote the spectral norm of the matrix X as $\|X\|$ and the true parameter vector, $\theta = [\theta_1; \theta_2] \in \mathbb{R}^{2n}$ as the vector obtaining by stacking the true and unknown coefficient vectors, θ_1, θ_2. For simplicity of exposition, in this chapter we suppose that the components θ_1 and θ_2 exhibit *block* sparsity with sparsity s and block size b (Baraniuk et al., 2010). (Analogous approaches apply for other structured sparsity models.)

The problem (2.1) is inherently ill-posed. To resolve this issue, we need to assume that the coefficient vectors θ_1, θ_2 are somehow distinguishable from each other. This is characterized by a notion of incoherence of the components θ_1, θ_2 (Soltani and Hegde, 2017a).

Definition 2.1. *The bases Φ and Ψ are called ε-incoherent if* $\varepsilon = \sup_{\substack{\|u\|_0 \le s,\ \|v\|_0 \le s \\ \|u\|_2 = 1,\ \|v\|_2 = 1}} |\langle \Phi u, \Psi v \rangle|$.

For the analysis of aperiodic link functions, we need the following standard definition (Negahban et al., 2011):

Definition 2.2. *A function $f : \mathbb{R}^{2n} \to \mathbb{R}$ satisfies Structured Restricted Strong Convexity/Smoothness (SRSC/SRSS) if:*

$$m_{4s} \leq \|\nabla_{\xi}^2 f(t)\| \leq M_{4s}, \quad t \in \mathbb{R}^{2n},$$

where $\xi = supp(t_1) \cup supp(t_2)$, for all $t_i \in \mathbb{R}^{2n}$ such that t_i belongs to $(2s, b)$ block-sparse vectors for $i = 1, 2$, and m_{4s} and M_{4s} are (respectively) the SRSC and SRSS constants. Also, $\nabla_{\xi}^2 f(t)$ denotes a $4s \times 4s$ sub-matrix of the Hessian matrix $\nabla^2 f(t)$ comprised of rows/columns indexed by $\xi \subset [2n]$.

Furthermore, for aperiodic functions, we assume that the derivative of the link function is strictly bounded either within a positive interval, or within a negative interval. In addition, let β_j denotes the j^{th} entry of the signal $\beta \in \mathbb{R}^n$. Also, for $j \in \{1, 2, \ldots, q\}$, $\beta(j : q : (k-1)q + j) \in \mathbb{R}^k$ denotes the sub-vector of β, starting at index $j + qr$, where $r = 0, 1, \ldots, k-1$. Finally, $Y((j : q : (k-1)q, l)$ represents the sub-vector made by picking the l^{th} column of any matrix Y and choosing the entries of this column as stated.

For the analysis of periodic link functions, we let the design matrix X be *factorized* as $X = DB$, where $D \in \mathbb{R}^{m \times q}$, and $B \in \mathbb{R}^{q \times n}$. We assume that m is a multiple of q, and that D is a concatenation of k diagonal matrices of $q \times q$ such that the diagonal entries in the blocks of D are i.i.d. random variables distributed uniformly within an interval $[-T, T]$ for some $T > 0$. The choice of B is flexible and can be chosen such that it supports stable demixing. In particular, as (Soltani and Hegde, 2017a) has shown, B can be any random matrix with independent subgaussian rows. Overall, our low-dimensional observation model can be written as:

$$y = g(DB\beta) + e = g(DB(\Phi\theta_1 + \Psi\theta_2)) + e, \tag{2.2}$$

where g is either sinusoidal function, or any periodic function such that in each period, it behaves monotonically. Furthermore, $D = [D_1, \ldots, D_k]^T$ comprises k diagonal matrices D_i's, and $e \in \mathbb{R}^m$ denotes additive noise such that $e \sim \mathcal{N}(0, \sigma^2 I)$. The goal is to stably recover θ_1, θ_2 from the embedding y. The diagonal structure of the matrix D reduces the the final recovery of underlying components to first obtaining a good enough estimation of $B\beta$, and then using a linear demixing approach from a (possibly noisy) estimate of $B\beta$ will lead to the estimation of θ_1, θ_2.

2.3 Algorithms and Analysis

In this section, we describe our algorithm and theoretical result for both aperiodic and periodic link functions.

2.3.1 Aperiodic link functions

To solve the demixing problem in (2.1), we consider the minimization of a special loss function $F(t)$, following (Soltani and Hegde, 2017a):

$$\min_{t\in\mathbb{R}^{2n}} F(t) = \frac{1}{m}\sum_{i=1}^{m} \Theta(x_i^T\Gamma t) - y_i x_i^T \Gamma t \quad \text{s. t.} \;\; t\in\mathcal{D}, \tag{2.3}$$

where $\Theta'(x) = g(x)$, $\Gamma = [\Phi \;\; \Psi]$, x_i is the i^{th} row of the design matrix X and $\mathcal{D}$ denotes the set of length-$2p$ vectors formed by stacking a pair of (s, b) block-sparse vectors. The objective function in (2.3) is motivated by the single index model in statistics; for details, see (Soltani and Hegde, 2017a). To approximately solve (2.3), we propose an algorithm which we call it *Structured Demixing with Hard Thresholding* (STRUCT-DHT). The pseudocode of this algorithm is given in Algorithm 2.1.

At a high level, STRUCT-DHT tries to minimize loss function defined in (2.3) (tailored to g) between the observed samples y and the predicted responses $X\Gamma\widehat{t}$, where $\widehat{t} = [\widehat{\theta}_1;\; \widehat{\theta}_2]$ is the estimate of the parameter vector after N iterations. The algorithm proceeds by iteratively updating the current estimate of $\widehat{t}$ based on a gradient update rule followed by (myopic) *hard thresholding* of the residual onto the set of s-sparse vectors in the span of Φ and Ψ. Here, we consider a version of DHT (Soltani and Hegde, 2017a) which is applicable for the case that coefficient vectors θ_1 and θ_2 have block sparsity. For this setting, we use component-wise block-hard thresholding, $\mathcal{P}_{s;s;b}$ (Baraniuk et al., 2010). Specifically, $\mathcal{P}_{s;s;b}(\tilde{t}^k)$ projects the vector $\tilde{t}^k \in \mathbb{R}^{2n}$ onto the set of concatenated (s, b) block-sparse vectors by projecting the first and the second half of $\tilde{t}^k$ separately. Now, we provide the theorem supporting the convergence analysis and sample complexity (required number of observations for successful estimation of θ_1, θ_2) of STRUCT-DHT.

Algorithm 2.1 Structured Demixing with Hard Thresholding (STRUCT-DHT)

```
**Inputs:** Bases $\Phi$ and $\Psi$, design matrix $X$, link function $g$, observation $y$, sparsity $s$, block size $b$, step size $\eta'$.
**Outputs:** Estimates $\widehat{\beta} = \Phi\widehat{\theta}_1 + \Psi\widehat{\theta}_2$, $\widehat{\theta}_1$, $\widehat{\theta}_2$
**Initialization:**
$\left(\beta^0, \theta_1^0, \theta_2^0\right) \leftarrow$ RANDOM INITIALIZATION
$k \leftarrow 0$
**while** $k \leq N$ **do**
    $t^k \leftarrow [\theta_1^k; \theta_2^k]$    {Forming constituent vector}
    $t_1^k \leftarrow \frac{1}{m}\Phi^T X^T(g(X\beta^k) - y)$
    $t_2^k \leftarrow \frac{1}{m}\Psi^T X^T(g(X\beta^k) - y)$
    $\nabla F^k \leftarrow [t_1^k; t_2^k]$    {Forming gradient}
    $\tilde{t}^k = t^k - \eta' \nabla F^k$    {Gradient update}
    $[\theta_1^k; \theta_2^k] \leftarrow \mathcal{P}_{s;s;b}\left(\tilde{t}^k\right)$    {Projection}
    $\beta^k \leftarrow \Phi\theta_1^k + \Psi\theta_2^k$    {Estimating $\widehat{x}$}
    $k \leftarrow k + 1$
**end while**
**Return:** $\left(\widehat{\theta}_1, \widehat{\theta}_2\right) \leftarrow \left(\theta_1^N, \theta_2^N\right)$
```

Theorem 2.3. *Consider the observation model (2.1) with all the assumption and definitions mentioned in the section 2.2. Suppose that the corresponding objective function F satisfies the Structured SRSS/SRSC properties with constants M_{6s} and m_{6s} such that $1 \leq \frac{M_{6s}}{m_{6s}} \leq \frac{2}{\sqrt{3}}$. Choose a step size parameter η' with $\frac{0.5}{M_{6s}} < \eta' < \frac{1.5}{m_{6s}}$. Then,* DHT *outputs a sequence of estimates (θ_1^k, θ_1^k) $(t^{k+1} = [\theta_1^k; \theta_1^k])$ such that the estimation error of the underlying signal satisfies the following upper bound (in expectation) for any $k \geq 1$:*

$$\|t^{k+1} - \theta\|_2 \leq (2q)^k \|t^0 - \theta\|_2 + C\tau\sqrt{\frac{s}{m}}, \tag{2.4}$$

where $q = 2\sqrt{1 + \eta'^2 M_{6s}^2 - 2\eta' m_{6s}}$ and $C > 0$ is a constant that depends on the step size η' and the convergence rate q. Here, θ denotes the true stacked signal defined in section 2.2.

The inequality (2.4) indicates the linear convergence behavior of our proposed algorithm. Specifically, in the noiseless scenario to achieve κ-accuracy in estimating the parameter vector $\widehat{t} = [\widehat{\theta}_1;\ \widehat{\theta}_2]$, STRUCT-DHT only requires $\log\left(\frac{1}{\kappa}\right)$ iterations. We also have the following theorem regarding the sample complexity of Alg. 2.1:

Theorem 2.4. *If the rows of X are independent subgaussian random vectors (Vershynin, 2010), then the required number of samples for successful estimation of the components, n is given by $\mathcal{O}\left(\frac{s}{b}\log\frac{n}{s}\right)$. Furthermore, if $b = \Omega\left(\log\frac{n}{s}\right)$, then the sample complexity of our proposed algorithm is given by $m = \mathcal{O}(s)$, which is asymptotically optimal.*

The big-Oh constant hides dependencies on various parameters, including the coherence parameter ε, as well as the upper and the lower bounds on the derivative of the link function g.

2.3.2 Periodic Link Functions

In this section, we focus on the periodic link functions which are either sinusoidal (complex-exponential), or any periodic function such that it is monotonic within each period. We start with the sinusoidal (complex-exponential) link function and establishing recovery of an underlying signal which is arbitrary sparse, or is the superposition of two arbitrary sparse components. Then, we generalize this approach to other periodic link functions. Finally, we apply this framework to our main problem, demixing of two constituent components with structured sparsity.

2.3.2.1 Stable Recovery Of Sparse Vectors From Random Sinusoidal Feature Maps

Setup. Several popular techniques in statistical regression and classification rely on the *kernel trick*, which enables approximating complicated nonlinear functions given a sufficiently large number of training samples. However, in large-scale problems where the number as well as dimension of the training data points is high, the computational costs of kernel-based techniques can be severe.

One approach to alleviate these computational costs is via the use of *random sinusoidal feature maps*, pioneered by (Rahimi and Recht, 2007). The mechanism is conceptually simple. Suppose that the kernel of choice is the Gaussian (RBF) kernel. Prior to the inference stage, the approach suggests performing a *dimensionality reduction* of a given data vector by first multiplying by a random Gaussian matrix, followed by the application of a nonlinearity (specifically, a sinusoid or complex exponential). Mathematically, one can represent the above process as follows. Let $\mathcal{X} \subseteq \mathbb{R}^n$ be any set in the data space, and consider a data vector $x \in \mathcal{X}$. Construct a matrix $B \in \mathbb{R}^{q\times n}$

whose entries are independently from a normal distribution. Then, modulo scaling factors, the random sinusoidal feature map, $y = \mathcal{A}(x)$, can be represented as the composition of the linear map $A \in \mathbb{R}^{m \times n}$ with a (complex) sinusoidal nonlinearity (see section 2.3.2.1 for more details):

$$z_j = (Ax)_j, \quad y_j = \exp(i\, z_j), \quad j = 1 \ldots m\ . \tag{2.5}$$

One can alternately replace the complex exponential by a real-valued sinusoidal function with similar consequences. The authors of (Rahimi and Recht, 2007) show that collecting a sufficient number of such features leads to statistically robust performance of kernel-based learning methods. Subsequent works (Vedaldi and Zisserman, 2012; Shi et al., 2009; Le et al., 2013; Lopez-Paz et al., 2014) have progressively extended this approach to more general types of kernels, faster methods of dimensionality reduction, as well as sparsity assumptions on the training data (Chang et al., 2016).

Moving beyond inference, a natural question arises whether the original data vector x can at all be *reconstructed* from the nonlinear features y. Geometrically, we would like to understand if (and how) the random feature map can be efficiently inverted, given the observations y and knowledge of the dimensionality reduction operator B. This question is of both theoretical and practical interest; if the embedding is shown to be *reversible*, then it could have implications for the privacy-preserving properties of random feature maps.

We consider a slightly more general version of the model (2.5) where the features y can be corrupted by noise. Moreover, we start with the data vector x which is s-sparse, i.e., it contains no more than s non-zeros (We will consider the structured sparsity later.). For this model, we propose an efficient, two-stage algorithm for reconstructing signals from a small number of random sinusoidal features (in particular, far smaller than the dimension of the data). To the best of our knowledge, our method is the first to propose an algorithm that is specialized to reconstructing sparse vectors from random sinusoidal feature maps.

Techniques. There are two main challenges that we will need to overcome. The first challenge is due to the ill-posed nature of the problem; in the high-dimensional regime, the total number of samples can be far less than the native dimension of the data. The second (and more compelling) challenge arises due to the fact that the sinusoidal transfer function is highly non-invertible. The typical way to deal with this is to assume some upper bound on the magnitude of the entries of the linear projection (Ax). However, we observe that unless this upper bound is relatively small, we can only estimate z_j up to some unknown integer multiple of 2π. If the projected dimension of the linear embedding, q, is large, then the number of possible combinations becomes exponential in q.

We alleviate these challenges using a simple idea. The key is to *decouple* the problem of inverting the sinusoidal nonlinearity from the sparse recovery step by replacing the linear map A with a different (carefully designed) map that still possesses all the properties required for reliable kernel-based inference. Particularly, our new linear map A is the product of two matrices $D \cdot B$; the way to construct D and B is described in detail below. This technique has been inspired by some recent approaches for the related problem of *phase retrieval* (Iwen et al., 2015; Bahmani and Romberg, 2015). Using this decoupling technique, we can separately solve the inversion of the nonlinearity and the actual recovery of the sparse high-dimensional data in two stages.

For the first stage, we *individually* estimate the linear projection $z = Bx$ using classical signal processing techniques for line spectral estimation. In particular, we leverage the method of matched filtering from random samples (Eftekhari et al., 2013). In the absence of noise, any spectral estimation technique (such as MUSIC, root-MUSIC, and ESPRIT) can also be used; however, our Monte Carlo simulations show below that the randomized method of (Eftekhari et al., 2013) et al. is considerably more robust compared to these more classical methods. For the second stage, we can use any sparse recovery method of our choice. In our experiments, we have used the CoSaMP algorithm proposed in (Needell and Tropp, 2009b).

While conceptually simple, the generic nature of our algorithm is advantageous in many ways. From a theoretical perspective, mirroring the approach of (Iwen et al., 2015), we can combine existing results for robust line spectral estimation with robust sparse recovery and obtain guarantees on the sample complexity and robustness of our algorithm. Moreover, since the nonlinear inversion and the sparse recovery steps are decoupled, one could conceivably extend this decoupling approach to a variety of other signal models and recovery algorithms (such as structured-sparsity (Baraniuk et al., 2010; Hegde et al., 2015a), low-rank matrix models (Recht et al., 2010a), and demixing a pair of sparse incoherent vectors (Soltani and Hegde, 2017a)).

Algorithm. In this section, we present our algorithm for solving problem (2.5) and provide the theory for sample complexity of the proposed algorithm. We first establish some notation. Let the j^{th} entry of the signal $x \in \mathbb{R}^n$ be denoted as x_j. For any $j \in [q]$, $x(j : q : (k-1)q + j)$ denotes the sub-vector in $\mathbb{R}^k$ formed by the entries of x starting at index $j + qr$, where $r = 0, 1, \ldots, k-1$. Similarly, $A((j : q : (k-1)q, l)$ represents the sub-vector constructed by picking the l^{th} column of any matrix A and selecting the entries of this column with the same procedure as mentioned.

As the key component in our approach, instead of using a random Gaussian linear projection as proposed in (Rahimi and Recht, 2007), we construct a linear operator A that can be *factorized* as $A = DB$, where $D \in \mathbb{R}^{m\times q}$, and $B \in \mathbb{R}^{q\times n}$. We assume that m is a multiple of q, and that D is a concatenation of k diagonal matrices of $q \times q$ such that the diagonal entries in the blocks of D are i.i.d. random variables, drawn from a distribution that we specify later. The choice of B depends on the model assumed on the data vector x; if the data is s-sparse, then B can be any matrix that supports stable sparse recovery (more precisely, B satisfies the null-space property (Foucart and Rauhut,)). Overall, our low-dimensional feature map can be written as:

$$y = \exp(iDBx) + e = \exp\left(i \begin{bmatrix} D^1 \\ D^2 \\ \vdots \\ D^k \end{bmatrix} Bx \right) + e, \tag{2.6}$$

where D^i are diagonal matrices and $e \in \mathbb{R}^m$ denotes additive noise such that $e \sim \mathcal{N}(0, \sigma^2 I)$. The goal is to stably recover x from the embedding y.

By the block diagonal structure of the outer projection D, we can reduce the overall reconstruction problem to first obtaining a good enough estimate of each entry of Bx. We show below that this can be achieved using line spectral estimation methods. The output of the first stage is used as the input of the second stage, which involves estimating x from a (possibly noisy) estimate of Bx.

Line spectral estimation. Consider the observation model (2.6) and let $z = Bx \in \mathbb{R}^q$. Suppose we know *a priori* that the entries of z belong to some bounded set $\Omega \in R$. Fix $l \in [q]$, and let $t = D(l : q : (k-1)q + l, l), u = y(l : q : (k-1)q + l), h = e(l : q : (k-1)q + l)$ which are vectors in $\mathbb{R}^k$. We observe that:

$$u = \exp(iz_l t) + h\,.$$

In other words, u can be interpreted as a collection of time samples of a (single-tone, potentially off-grid) complex-valued signal with frequency $z_l \in \Omega$, measured at time locations t. Therefore, we can independently estimate z_l for $l = 1, \ldots, q$ by solving a least-squares problem (Eftekhari et al., 2013):

$$\widehat{z}_l = \arg\min_{v\in\Omega} \|u - \exp(i\,vt)\|_2^2 = \arg\max_{v\in\Omega} |\langle u, \psi_v\rangle|\,, \tag{2.7}$$

for all $l = 1, \ldots, q$, where $\psi_v \in \mathbb{R}^k$ denotes a *template vector* given by $\psi_v = \exp(jtv)$ for any $v \in \Omega$. In essence, the solution of the least-squares problem can be interpreted as a *matched filter*. Numerically, the optimization problem in (2.7) can be solved using a grid search over the set Ω, and the resolution of this grid search controls the running time of the algorithm; for fine enough resolution, the estimation of z_l is more accurate albeit with increased running time. This issue is also discussed in (Eftekhari et al., 2013) and more sophisticated spectral estimation techniques have appeared in (Mishali and Eldar, 2011; Tang et al., 2013; Chen and Chi, 2014). After obtaining all the estimates $\widehat{z}_l$'s, we stack them in a vector $\widehat{z}$.

Sparse recovery. We now propose to recover the high-dimensional signal $x \in \mathbb{R}^m$ in problem (2.6) using our estimated $\widehat{z} \in \mathbb{R}^q$. According to our model, we also have assumed that the matrix B supports stable sparse recovery, and the underlying signal x is s-sparse. Hence, we can use any generic sparse recovery algorithm of our choice to obtain an estimate of x. In our simulations below, we use the CoSaMP algorithm (Needell and Tropp, 2009b) due to its speed and ease of parameter tuning. Again, we stress that this stage depends on a model assumption on x, and other recovery algorithms (such as structured-sparse recovery (Baraniuk et al., 2010), low-rank matrix recovery (Recht et al., 2010a), or demixing a pair of sparse (but incoherent) vectors (Soltani and Hegde, 2017a)) can equally well be used. Our overall algorithm, *Matched Filtering+Sparse recovery* (MF-Sparse) is described in pseudocode form in Algorithm 2.2. We now provide a theoretical analysis of our proposed algorithm. Our result follows from a concatenation of the results of (Eftekhari et al., 2013) and (Needell and Tropp, 2009b).

Theorem 2.5 (Sample complexity of MF-Sparse)**.** *Assume that the nonzero entries of D are i.i.d. samples from a uniform distribution $[-T, T]$, and the entries of B are i.i.d. samples from $\mathcal{N}(0, 1/q)$. Also, assume $\|x\|_2 \leq R$ for some constant $R > 0$. Set $m = kq$ where $k = c_1 \log\left(\frac{Rq}{\varepsilon}\frac{1}{\delta}\right)(1 + \sigma^2)$ for some $\varepsilon > 0$ and $q = c_2\left(s \log \frac{n}{s}\right)$. Set $\omega = c_3 R$ and $\Omega = [-\omega, \omega]$. Then, MF-Sparse returns an estimate $\widehat{x}$, such that*

$$\|x - \widehat{x}\|_2 \leq C\varepsilon\,,$$

with probability exceeding $1 - \delta$. Here, c_1, c_2, c_3, C are constants.

In model (2.5), the features y_j are modeled in terms of complex exponentials. With only a slight modification in the line spectral estimation stage, we can recover x from real-valued random sine features. More precisely, these random features are given by:

$$y = \sin(DBx) + e \tag{2.8}$$

where e denotes the additive noise as defined before. If we follow an analogous approach for estimating x, then in lieu of the least squares estimator (2.7), we have the following estimator:

$$\begin{aligned} \widehat{z}_l &= \arg\min_{v \in \Omega} \|u - \sin(vt)\|_2^2 && (2.9) \\ &= \arg\max_{v \in \Omega} \left(2\,|\langle u, \psi_v\rangle| - \|\psi_v\|_2^2\right), && (2.10) \end{aligned}$$

Algorithm 2.2 MF-Sparse

```
**Inputs:** $y$, $D$, $B$, $\Omega$, $s$
**Output:** $\widehat{x}$
**Stage 1: Tone estimation:**
**for** $l = 1 : q$ **do**
    $t \leftarrow D(l : q : (k-1)q + l, l)$
    $u \leftarrow y(l : q : (k-1)q + l)$
    $\widehat{z}_l = \arg\max_{\omega \in \Omega} |\langle y, \psi_\omega \rangle|$
**end for**
$\widehat{z} \leftarrow [\widehat{z}_1, \widehat{z}_2 \ldots, \widehat{z}_q]^T$
**Stage 2: Sparse recovery**
$\widehat{x} \leftarrow$ CoSaMP$(\widehat{z}, B, s)$
```

for $l = 1, \ldots, q$ and u as defined above. Also, $\psi_v = \sin(tv)$ for any $v \in \Omega$. We only provide some numerical experiments for this estimator and leave a detailed theoretical analysis for future research.

2.3.3 Demixing with Periodic Nonlinearity

Now, we use the results of previous section about signal recovery from sinusoidal link function for the demixing of constituent components with structured sparsity. In this case, we use MF-Sparse algorithm as a core algorithm for estimating the underlying components albeit with two differences: first, we might have a preprocessing step before tone estimation depending on the periodic nonlinearity. More precisely, if we use a link function except sinusoidal, we first map the observation vector y to $\tilde{y}$ through a sinusoidal function and use this new observation vector $\widetilde{y}$ as the input to the second step, tone estimation. To give a explanation why this method works, we note that in each period, the link function is assumed to be monotonic; as a result, for each entry of $\widetilde{y}$, there is one and only one entry from y. Thus, we can use the method of recovery under sinusoidal nonlinearity to estimate the underlying components $\widehat{\theta}_1, \widehat{\theta}_2$. Second, for the third stage, we invoke STRUCT-DHT with identity link function $g(x) = x$ instead of any regular sparse recovery method. We call the resulting algorithm *MF-STRUCT-DHT* and is given in Algorithm 2.3.

Algorithm 2.3 MF-STRUCT-DHT

```
Inputs: $y$, $D$, $B$, $\Omega$, $s$, $b$,$\Phi$,$\Psi$,$\eta'$,$g$
Output: $\widehat{\theta}_1, \widehat{\theta}_2$
Stage 1: Mapping:
if $g(x) \neq \sin(x)$ then
    $\tilde{y} = \sin(y)$
    $y \leftarrow \tilde{y}$
end if
Stage 2: Tone estimation:
for $l = 1 : q$ do
    $t \leftarrow D(l : q : (k-1)q + l, l)$
    $u \leftarrow y(l : q : (k-1)q + l)$
    $\widehat{z}_l = \arg\max_{\omega \in \Omega} |\langle y, \psi_\omega \rangle|$
end for
$\widehat{z} \leftarrow [\widehat{z}_1, \widehat{z}_2 \ldots, \widehat{z}_q]^T$
Stage 2: Structured demixing recovery
$g(x) \leftarrow x$
$X \leftarrow B$
$\widehat{\theta}_1, \widehat{\theta}_2 \leftarrow$ STRUCT-DHT$(\widehat{z}, X, s, b, \Phi, \Psi, \eta', g)$
```

By combining Theorem 2.4 and Theorem 2.5, we obtain the sample complexity of the MF-STRUCT-DHT scheme to achieve κ-accuracy.

Theorem 2.6 (Sample complexity of MF-STRUCT-DHT)**.** *Consider the measurement model in (2.2) without any additive noise. Assume that the nonzero entries of block diagonal matrix D are i.i.d. random variables, distributing uniformly within the interval $[-T, T]$, and the rows of B are independent subgaussian random vectors (normalized by $\frac{1}{q}$). Moreover, assume $\|x\|_2 \leq R$ for some constant $R > 0$. If we set $m = kq$ where $k = c_1 \log\left(\frac{Rq}{\kappa}\frac{1}{\delta}\right)$ for some $\kappa > 0$, $q = \mathcal{O}\left(\frac{s}{b}\log\frac{n}{s}\right)$, $\omega = c_2 R$, and $\Omega = [-\omega, \omega]$, MF-STRUCT-DHT scheme provides an estimate $\widehat{\beta}$, such that $\|\beta - \widehat{\beta}\|_2 \leq \mathcal{O}(\kappa)$, with probability at least $1 - \delta$. Here, c_1, c_2 are constants. Furthermore, if b scales as $b = \Omega\left(\log\frac{n}{s}\right)$, then the sample complexity of MF-STRUCT-DHT scheme is given by $m = \mathcal{O}(s)$, which is asymptotically optimal.*

Note that big-Oh constant does not depend on the bounds on the derivative of the link function since it is a identity function. In addition, if the periodic link function g is set to the sinusoidal (complex-exponential), then the additive noise can be added to (2.2). In this case, the sample complexity is increased by a multiplicative factor equals to $1 + \sigma^2$ where σ^2 denotes the variance of the Gaussian noise; see (Soltani and Hegde, 2017d) for details.

2.4 Experimental Results

We compare our algorithm with existing algorithms in various scenarios. First, we start with the set of experiments related to the stable recovery of (superposition) sparse vectors from random sinusoidal feature map. Next, we conduct some experiments regarding the demixing of two components with structured sparsity with both periodic and aperiodic link functions.

For the first set of experiments, we assume that our random features are computed using a real sine link function. In other words, the features, $\{y_j\}$ is given by (2.8) for all $j = 1, \ldots, q$. In Fig. 2.1(a), we compare the probability of recovery of MF-Sparse with *Gradient Hard Thresholding* (GHT), a projected-gradient descent type algorithm whose variants have proposed in (Bahmani et al., 2013a; Yuan et al., 2014b; Jain et al., 2014). To recover x, GHT tries to minimize a specific loss function (typically, the squared loss) between the observed random feature vector, y, and $\sin(DBx)$ by iteratively updating the current estimate of x based on a gradient update rule, and then projecting it into the set of s-sparse vectors via hard thresholding.

The setup for the experiment illustrated in Fig. 2.1(a) is as follows. First, we generate a synthetic signal of length $n = 2^{14}$ with sparsity $s = 100$ such that the support is random and the values of the signal in the support are drawn from a standard normal distribution. Then, the ℓ_2-norm of x is adjusted via a global scaling to coincide with three different values; $\|x\|_2 = 1, \|x\|_2 = 15$, and $\|x\|_2 = 30$. We generate a matrix $B \in \mathbb{R}^{q\times n}$ where $q = 700$ and $n = 2^{14}$ such that the entries of it are i.i.d random variables with distribution $\mathcal{N}(0, \frac{1}{\sqrt{q}})$. All nonzero entries of D, d_l^r for $l = 1, \ldots, q$ and $r = 1, \ldots, k$ are assumed to be standard normal random variables for $k = 1, \ldots, 8$. Next, we generate $y \in \mathbb{R}^m$ as $y = \sin(DBx)$ where $m = kq$. (There is no noise considered in this experiment.) By running MF-Sparse and GHT, we obtain the estimate of x, denoted by $\widehat{x}$, and calculate the (normalized) estimation error defined as $\frac{\|\widehat{x}-x\|_2}{\|x\|_2}$. We repeat this process for 60 Monte Carlo trials,

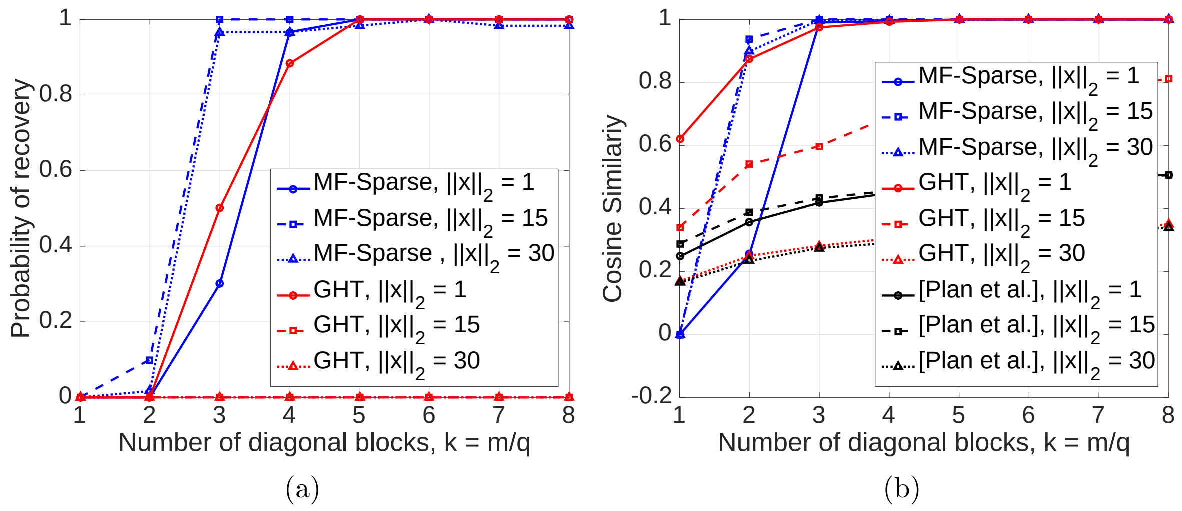


Figure 2.1: Comparison of the proposed algorithm with other algorithms. Parameters: $n = 2^{14}, q = 700$. (a) Probability of recovery in terms of normalized error. (b) Cosine similarity between recovered and true signal.

and define the empirical probability of successful recovery as the fraction of simulations in which the relative error is less than 0.05.

As we can see in Fig. 2.1(a), MF-Sparse can successfully recover x even with $k = 4$ number of blocks with probability close to 1 when the norm of x is small. In this regime, i.e., $\|x\|_2$ being small, both GHT and MF-Sparse display similar performance. However, GHT shows extremely poor performance when $\|x\|_2$ has moderate or large value. The reason is that when the norm of x is small, the entries of Bx are also small (i.e., the entries of DBx are close to the origin with high probability), and the sinusoidal nonlinearity can be assumed to be almost linear. Consequently, GHT can recover x in this regime. However, this assumption breaks down for larger values of $\|x\|_2$.

In Fig. 2.1(b), we repeat a similar experiment, but measure performance with a different recovery criterion. In this scenario, we measure the *cosine similarity* of the estimated vector $\widehat{x}$ with x, defined as $\frac{\widehat{x}^T x}{\|\widehat{x}\|_2 \|x\|_2}$. In addition to GHT, we also compare the performance of MF-Sparse with the single-step thresholding approach proposed by (Plan et al., 2014). The approach of (Plan et al., 2014) only recovers the vector modulo an (unknown) scaling factor, and does not need to possess

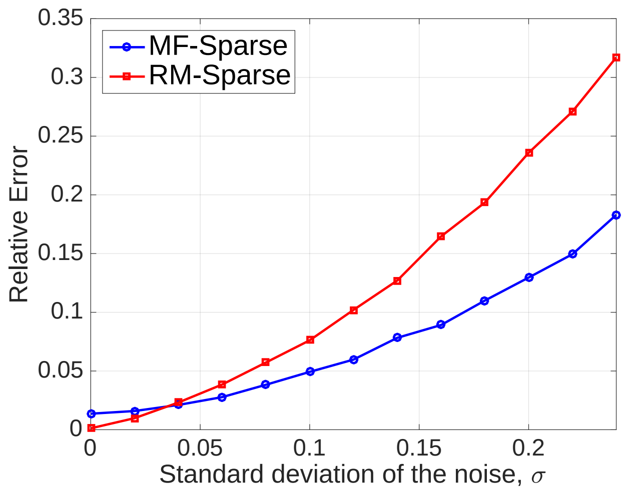


Figure 2.2: Comparison of matched filtering versus rootMUSIC in the first stage. Parameters: $n = 2^{14}, q = 800,\ k = 6$.

knowledge of the nonlinearity. As illustrated in Fig. 2.1(b), the approach of (Plan et al., 2014) has worse performance compared to two other methods. Also, GHT shows good performance only when $\|x\|_2 = 1$, as expected. In contrast, MF-Sparse shows the best performance.

Next, we consider the experiment illustrated in Fig. 2.2. In this experiment, we assume a complex sinusoid as the nonlinearity generating the feature map. In this scenario, we fix $k = 6$ (number of diagonal blocks in matrix D), $q = 800$, and add i.i.d. Gaussian noise to the embeddings with increasing variance. Our goal is to compare the matched filtering part of MF-Sparse with classical spectral estimation techniques. Indeed, a natural question is whether we can use other spectral estimation approaches instead of the matched filter, and whether there is any advantage in choosing the elements of D randomly. We attempt to illustrate the benefits of randomness through the experiment in Fig. 2.2. Here, we compare the relative error of MF-Sparse with an analogous algorithm that we call *RM-Sparse*, which uses the RootMUSIC spectral estimation technique (Schmidt, 1986). For RM-Sparse, in the line spectral estimation stage, we replace random diagonal blocks in matrix D with deterministic diagonal ones, and letting the diagonal entries of the r^{th} block being proportional to r. In other words, the time samples are chosen to lie on a uniform grid, as RootMUSIC expects. As we see in Fig. 2.2, while RM-Sparse does recover the original x for very

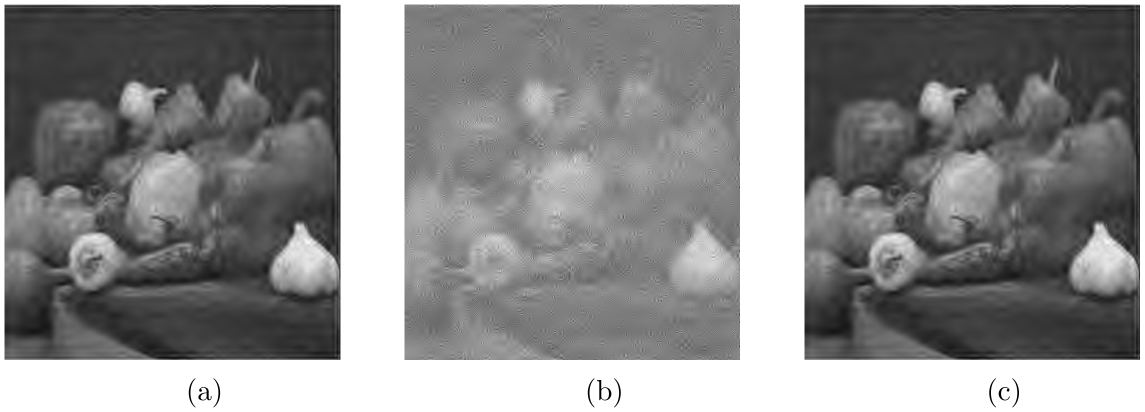


Figure 2.3: Successful reconstruction on a real 2-D image from random sinusoidal features. (a) Test image. (b) Reconstruction quality with $k = 2$ diagonal blocks. (c) Reconstruction with $k = 3$.

low values of noise variance, MF-Sparse outperforms RM-Sparse when the noise level increases; this verifies the robustness of MF-Sparse to noise in the observations.

In addition, we evaluate our proposed algorithm for a real 2D image. We begin with a 512×512 test image. First, we obtain its 2D Haar wavelet decomposition and sparsify it by retaining the $s = 2000$ largest coefficients. Then, we synthesize the test image based on these largest coefficients, and multiply it by a subsampled Fourier matrix with $q = 16000$ multiplied with a diagonal matrix with random ± 1 entries (Krahmer and Ward, 2011). Further, we multiply this result with a block diagonal matrix D with number of blocks $k = 2, 3$. Now, we apply the $\sin(\cdot)$ function element-wise to the resulting vector. Figure 2.3 shows the recovered image using MF-Sparse. As is visually evident, with $k = 2$, the quality of the reconstructed image is poor, but $k = 3$ already yields a solution close to the true image.

Finally, we present another experiment on a real 2D image. In this case, we assume that our signal x is the superposition of two constituent signals, i.e., $x = \Phi w + \Psi z$. Hence, the random feature map is given by $y = \sin(DB\,(\Phi w + \Psi z))$. In this setting, as illustrated in Fig. 2.4(a), the signal x is the mixture of galaxy and star images, where the constituent coefficient vectors w and z are s-sparse signals in the DCT basis (Φ) and the canonical basis (Ψ). In this experiment, we fix $s = 1000$ for both w and z. The matrices B and D are generated the same way as in the experiment

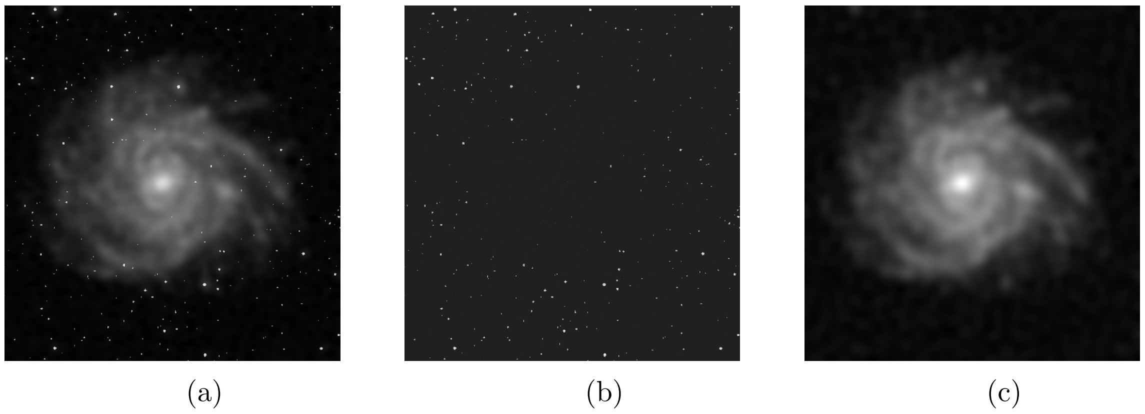


Figure 2.4: Successful demixing on a real 2-dimensional image from random sinusoidal features. Parameters: $n = 512 \times 512, s = 1000, m = k \times q = 48000, g(x) = \sin(x)$. Image credits: McCoy et al. (2014).

expressed in Fig. 2.3 with $q = 16000, n = 2^{18}$, and $k = 3$. As we can see in Figs. 2.4(b) and 2.4(c), running a variant of MF-Sparse along with the DHT algorithm (Soltani and Hegde, 2017a) for the second stage can successfully separate the galaxy from the stars, verifying the applicability of MF-Sparse for more generic structured inverse problems.

To show the efficacy of Struct-DHT for demixing components with structured sparsity for aperiodic link funciotns, we numerically compare Struct-DHT with ordinary DHT (which does *not* leverage structured sparsity), and also with an adaptation of a convex formulation described in (Yang et al., 2015) that we call *Demixing with Soft Thresholding* (DST). We first generate true components θ_1 and θ_2 with length $n = 2^{16}$ with nonzeros grouped in blocks with length $b = 16$ and total sparsity $s = 656$. The nonzero (active) blocks are randomly chosen from a uniform distribution over all possible blocks.

We construct a design (observation) matrix following the construction of (Krahmer and Ward, 2011). Finally, we use a (shifted) sigmoid link function given by $g(x) = \frac{1-e^{-x}}{1+e^{-x}}$ to generate the observations y. Fig 2.5 shows the the performance of the three algorithms with different number of samples averaged over 10 Monte Carlo trials. In Fig 2.5(a), we plot the probability of successful recovery, defined as the fraction of trials where the normalized error is less than 0.05. Fig 2.5(b)

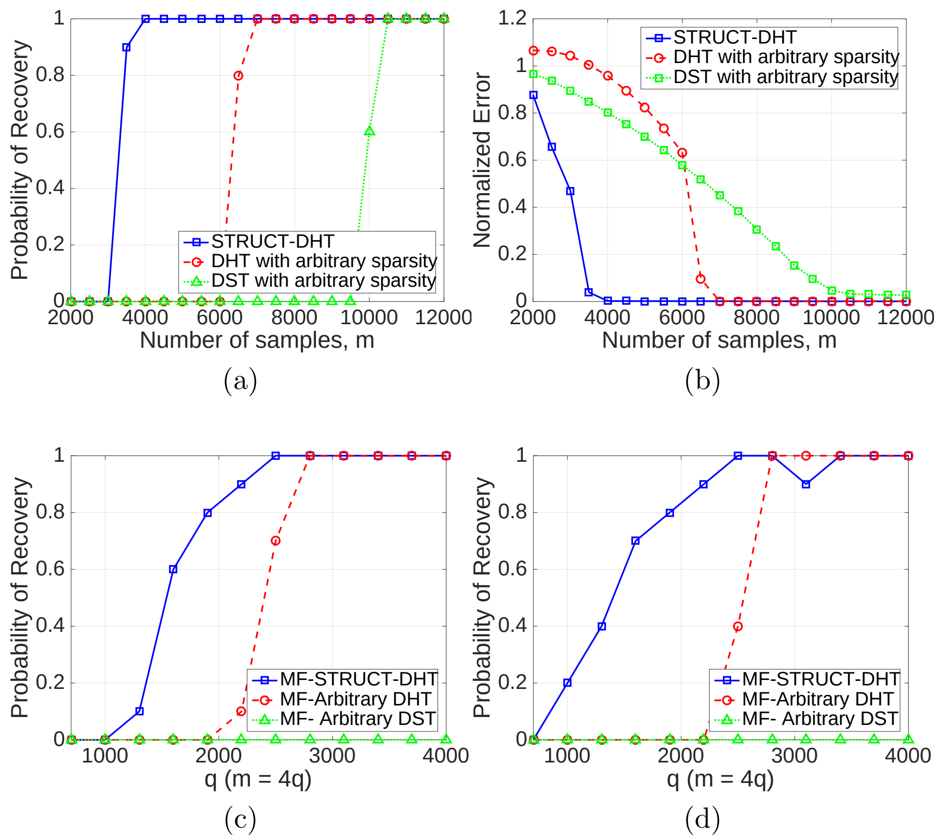


Figure 2.5: Comparison of DHT and MF with structured sparsity with other algorithms. (a) and (b) Probability of recovery in terms of normalized error and Normalized error between $\widehat{\beta} = \Phi\widehat{\theta}_1 + \Psi\widehat{\theta}_2$ and true β, respectively for $g(x) = \frac{1-e^{-x}}{1+e^{-x}}$. (c) and (d) Probability of recovery in terms of normalized error for $g(x) = \sin(x)$ and $g(x) = \mod(x)$, respectively.

shows the normalized estimation error for these algorithms. As we can observe, Struct-DHT shows much better sample complexity (the required number of samples for obtaining small relative error) as compared to DHT and DST.

We conduct a similar experiment for two periodic link functions: sinusoidal and sawtooth (*modulo*) functions with period 2π and amplitude 1. The parameters are as before, except we set $n = 2^{14}$, $s = 160$, and $k = 4$. We numerically compare MF-STRUCT-DHT scheme with the case where we do not consider the structured sparsity, and with a convex relaxation formulation (Yang et al., 2015). Figures 2.5(c) and (d) show the probability of success for the sinusoidal and sawtooth cases, respectively. Again, we get the same conclusion as in the aperiodic case: our proposed algorithm achieves far improved sample complexity over previous existing methods that solely rely on sparsity assumptions.

2.5 Conclusion

In this chapter, we addressed the problem of demixing from a set of limited nonlinear measurements in high dimensions. Specifically, we considered two nonlinearities: aperiodic and periodic link functions and the structured sparsity in the underlying signal components. For each of these nonlinearities, we proposed an algorithm and support them with sample complexity analysis. As a result of our proposed schemes, we showed that having structured sparsity assumption in the underlying components can significantly reduce the sample complexity compared to the case where we just have regular sparsity prior in these components. Finally, we verified our theoretical claims with some experimental results.

2.6 Appendix. Proof of Theoretical Results

In this section, we provide the proof of the theorems in chapter 2.

Proof sketch of Theorem 2.3. The proof follows the technique used to prove Theorem 1.14 in chapter 1. The main steps are as follows. Let $b' \in \mathbb{R}^{2n} = [b'_1; b'_2] = t^k - \eta' \nabla F(t^k)$, $b = t^k - \eta' \nabla_J F(t^k)$ where $J := J_k = \text{supp}(t^k) \cup \text{supp}(t^{k+1}) \cup \text{supp}(\theta)$ and $b'_1,\ b'_2 \in \mathbb{R}^n$ (Here, $\theta = [\theta_1; \theta_2]$ denotes the true signal). Also define $t^{k+1} = \mathcal{P}_{s;s}(b') = [\mathcal{P}_s(b'_1); \mathcal{P}_s(b'_2)]$. Now, by the triangle inequality, we have: $\|t^{k+1} - \theta\|_2 \leq \|t^{k+1} - b\|_2 + \|b - \theta\|_2$. The proof is completed by showing that $\|t^{k+1} - b\|_2 \leq 2\|b - \theta\|_2$. Finally, we use the Khintchine inequality (Vershynin, 2010) to bound the expectation of the ℓ_2-norm of the restricted gradient function, $\nabla F(\theta)$ (evaluated at the true stacked signal θ) with respect to the support set J). □

Proof sketch of Theorem 2.4. The proof is similar to the proof of Theorem 4.8 in (Soltani and Hegde, 2017a) where we had previously derived upper bounds on the sample complexity of demixing by proving that F satisfies RSC/RSS with reasonable parameters. Here, the steps are essentially the same as in (Soltani and Hegde, 2017a). The proof approach uses standard concentration techniques to show that the Euclidean norm of a sparse vector with fixed support is preserved with high probability under the action of the design matrix X. The proof follows by taking union bound over the set of *all* sparse vectors, the size of which is given by $\mathcal{O}\left((\frac{n}{s})^s\right)$. This increases the sample complexity by a log factor over the number of "free" parameters. The same strategy is applicable here, except that we need to compute union bound over the set of (s, b) block-sparse vectors. The size of this set is given by $\binom{\frac{n}{b}}{\frac{s}{b}} = \mathcal{O}\left((\frac{n}{s})^{\frac{s}{b}}\right)$ which is considerably smaller than the set of all sparse vectors. Now, if we choose $m = \mathcal{O}\left(\frac{s}{b}\log\frac{n}{s}\right)$, then the objective function in (2.3) satisfies SRSC/SRSS condition. Finally, if b scales as $b = \Omega\left(\log\frac{n}{s}\right)$, we obtain $m = \mathcal{O}(s)$ which is an asymptotic gain over $\mathcal{O}\left(s\log\frac{n}{s}\right)$. □

Proof of Theorem 2.5. Set $\varepsilon' > 0$. Based on Corollary 8 of (Eftekhari et al., 2013), by setting $|T| = \mathcal{O}(\frac{1}{\varepsilon'})$, we need $k = \mathcal{O}\left((1+\sigma^2)\log\left(\frac{|\Omega|}{\varepsilon'}\frac{1}{\delta}\right)\right)$ to obtain $|\widehat{z}_l - z_l| \leq \varepsilon'$ with probability at least $1-\delta$ for each $l = 1, \dots, q$. Here, $|\Omega|$ denotes the radius of the feasible range for z, and appears in the maximization problem (2.7). Observe that $|\Omega| = \mathcal{O}\left(\|Bx\|_\infty\right)$. If the entries of matrix B are chosen as stipulated in the theorem, then $\|Bx\|_\infty \leq \|Bx\|_2 \lesssim \mathcal{O}(R)$ which justifies the choice of $|\Omega|$ in the theorem. Thus, $k = \mathcal{O}\left((1+\sigma^2)\log\left(\frac{R}{\varepsilon'}\frac{1}{\delta}\right)\right)$. By a union bound, it follows that with probability at least $1 - q\delta$, we have $\|\widehat{z} - z\|_\infty \leq \varepsilon'$. Now, we can write $\widehat{z} = z + e' = Bx + e'$ where $\|e'\|_\infty \leq \varepsilon'$. Since we have used CoSaMP in the sparse recovery stage, if the choice of $q = \mathcal{O}\left(s\log\frac{n}{s}\right)$ enables us to obtain $\|\widehat{x} - x\|_2 \leq c\|e'\|_2 \leq c\sqrt{q}\|e'\|_\infty \leq c\sqrt{q}\varepsilon'$ (Needell and Tropp, 2009b). (In fact, a more general guarantee can be derived even for the case when x is not exactly s-sparse.) Now, let $\varepsilon' = \mathcal{O}\left(\frac{\varepsilon}{\sqrt{q}}\right)$ to obtain the final bound on the estimation error $\|\widehat{x} - x\|_2 \leq \mathcal{O}(\varepsilon)$ with $k = \mathcal{O}\left((1+\sigma^2)\log\left(\frac{Rq}{\varepsilon}\frac{1}{\delta}\right)\right)$. □

Proof of Theorem 2.6. The proof follows from a straightforward application of Theorem 2.1 in (Soltani and Hegde, 2017d). According to this result, one can estimate $\widehat{z}$ (the estimation of $z = B\beta$) up to υ-accuracy if T scales as $|T| = \mathcal{O}(\frac{1}{\upsilon})$. Under this choice for T, the required number of block diagonal matrices in D to achieve υ accuracy for estimating z is given by $k = \mathcal{O}\left(\log(\frac{\Omega}{\upsilon})\right)$ where

$|\Omega| = \mathcal{O}(R)$ (see Algorithm 2.3). Now by choosing the design matrix $B \in \mathbb{R}^{q\times n}$, final accuracy parameter κ as $\upsilon = \mathcal{O}(\frac{\kappa}{\sqrt{q}})$, and choosing $q = \mathcal{O}\left(\frac{s}{b}\log\frac{n}{s}\right)$ according to Theorem 2.4, the result follows. $\square$

CHAPTER 3. LEARNING GENERATIVE MODELS OF STRUCTURED SIGNALS FROM THEIR SUPERPOSITION USING GANS WITH APPLICATION TO DENOISING AND DEMIXING

[1]In chapters 1 and 2, we focused on the demixing problem under different setups. In particular, we showed that having some structure on the underlying constituent components are crucial for the success of demixing in high-dimension. As a result, we considered two important structures: arbitrary sparse and structured sparsity in chapters 1 and 2, respectively. However, many natural signals while they show some low-dimension structure, it is not known what exactly their structure is. For this reason, in many applications, the underlying structures are *hard-coded* and are given as prior knowledge. Although this approach has been successful in some specific domains, it is limited in practice.

In this chapter, we are going to remove the assumption of having prior knowledge on the underlying components through learning of the underlying structure as a part of the whole process. To do this, we use Generative Adversarial Networks (GANs). Recently, GANs have emerged as a popular alternative for modeling complex high dimensional distributions. In particular, we consider the problem of learning GANs under the observation setting when the samples from target distribution are given by the superposition of two structured components. We propose two novel frameworks: denoising-GAN and demixing-GAN. The denoising-GAN assumes access to clean samples from the second component and try to learn the other distribution, whereas demixing-GAN learns the distribution of the components at the same time. Through extensive numerical experiments, we demonstrate that proposed frameworks can generate clean samples from unknown distributions, and provide competitive performance in tasks such as denoising, demixing, and compressive sensing.

[1]The work in chapter 3 was accomplished when the author, Mohammadreza Soltani was doing his internship in Technicolor AI Lab from May 2018 to Dec 2018.

3.1 Introduction

In this chapter, we consider the classical problem of separating two structured signals observed under the following linear superposition model:

$$Y = X + N, \tag{3.1}$$

where $X \in \mathcal{X}$ and $N \in \mathcal{N}$ are the constituent signals, and $\mathcal{X}, \mathcal{N} \subseteq \mathbb{R}^p$ denote the two structured sets. As we mentioned in the previous chapters, in general the separation problem is inherently ill-posed; however, with enough structural assumption on $\mathcal{X}$ and $\mathcal{N}$, it has been established that separation is possible. Depending on the application one might be interested in estimating only X, which is referred to as *denoising*, or in recovering both X and N which is referred to as *demixing*. Both demixing and denoising arise in a variety of important practical applications in the areas of signal/image processing, computer vision, and statistics Chen et al. (2001); Elad et al. (2005); Bobin et al. (2007); Candès et al. (2011) (please see the previous chapter for more references).

Most of the existing techniques assume some prior knowledge on the structures of $\mathcal{X}$ and $\mathcal{N}$ in order to recover the desired component signal(s). Prior knowledge about the structure of $\mathcal{X}$ and $\mathcal{N}$ can only be obtained if one has access to the generative mechanism of the signals or has access to clean samples from the probability distribution defined over sets $\mathcal{X}$ and $\mathcal{N}$. In many practical settings, neither of these may be feasible. In this paper, we consider the problem of separating constituent signals from superposed observations when clean access to samples from the distribution is not available. In particular, we are given a set of superposed observations $\{Y_i = X_i + N_i\}_{i=1}^{K}$ where $X_i \in \mathcal{X}$ and $Y_i \in \mathcal{N}$ ($\mathcal{X}$ and $\mathcal{N}$ are not known in general) are i.i.d samples from their respective (unknowns) distributions. In this setup, we explore two questions: First, *How can one learn prior knowledge about the individual components from superposition samples?* Second, *Can we leverage the implicitly learned constituent distributions for tasks such as denoising and demixing?*

3.1.1 Setup and Our Technique

Motivated by the recent success of generative models in high dimensional statistical inference tasks such as compressed sensing in Bora et al. (2017, 2018), in this paper, we focus on Generative Adversarial Network (GAN) based generative models to implicitly learn the distributions, i.e., generate samples from their distributions. Most of the existing works on GANs typically assume access to clean samples from the underlying signal distribution. However, this assumption clearly breaks down in the superposition model considered in our setup, where the structured superposition makes training generative models very challenging.

In this context, we investigate the first question with varying degrees of assumption about the access to clean samples from the two signal sources. We first focus on the setting when we have access to samples only from the constituent signal class $\mathcal{N}$ and observations, Y_i's. In this regard, we propose the *denoising*-GAN framework. However, assuming access to samples from one of the constituent signal class can be restrictive and is often not feasible in real-world applications. Hence, we further relax this assumption and consider the more challenging demixing problem, where samples from the second constituent component are not available and solve it using what we call the *demixing*-GAN framework.

Finally, to answer the second question, we use our trained generator(s) from the proposed GAN frameworks for denoising and demixing tasks on unseen test samples (i.e., samples not used in the training process) by discovering the best hidden representation of the constituent components from the generative models. In addition to the denoising and demixing problems, we also consider a compressive sensing setting to test the trained generator(s). Below we explicitly list the contribution made in this paper:

1. Under the assumption that one has access to the samples from one of the constituent component, we extend the canonical GAN framework and propose *denoising*-GAN framework. This learns the prior from the training data that is heavily corrupted by additive structured component. We demonstrate its utility in denoising task via numerical experiments.

2. We extend the above denoising-GAN and propose *demixing*-GAN framework. This learns the prior for both the constituent components from their superpositions, without access to separate samples from any of the individual components. We demonstrate its utility in demixing task via numerical experiments.

3.2 Application and Prior Art

To overcome the inherent ambiguity issue in problem (3.1), many existing methods have assumed that the structures of sets (i.e., the structures can be low-rank matrices, or have sparse representation in some domain (McCoy and Tropp, 2014)) $\mathcal{X}$ and $\mathcal{N}$ are a prior known and also that the signals from $\mathcal{X}$ and $\mathcal{N}$ are "distinguishable" (Elad and Aharon, 2006; Soltani and Hegde, 2016, 2017a; Druce et al., 2016; Elyaderani et al., 2017; Jain et al., 2017). The assumption of having the prior knowledge is a big restriction in many real-world applications. Recently, there have been some attempts to automate this *hard-coding* approach. Among them, structured sparsity Hegde et al. (2015b), dictionary learning Elad and Aharon (2006), and in general manifold learning are the prominent ones. While these approaches have been successful to some extent, they still cannot fully address the need for the prior structure. Over the last decade, deep neural networks have been demonstrated to learn useful representations of real-world signals such as natural images, and thus have helped us understand the structure of the high dimensional signals, for e.g. using deep generative models (Ulyanov et al., 2017).

In this paper, we focus on Generative Adversarial Networks (GANs) Goodfellow et al. (2014) as the generative models for implicitly learning the distribution of constituent components. GANs have been established as a very successful tool for generating structured high-dimensional signals (Berthelot et al., 2017; Vondrick et al., 2016) as they do not directly learn a probability distribution; instead, they generate samples from the target distribution(s) (Goodfellow, 2016). In particular, if we assume that the structured signals are drawn from a distribution lying on a low-dimensional manifold, GANs generate points in the high-dimensional space that resemble those coming from the true underlying distribution.

Since their inception by Goodfellow et al. (2014), there has been a flurry of works on GANs (Zhu et al., 2017; Yeh et al., 2016; Subakan and Smaragdis, 2018) to name a few. In most of the existing works on GANs with few notable exceptions Wu et al. (2016); Bora et al. (2018); Kabkab et al. (2018); Hand et al. (2018); Zhu et al. (2016), it is implicitly assumed that one has access to clean samples of the desired signal. However, in many practical scenarios, the desired signal is often accompanied by unnecessary components. Recently, GANs have also been used for capturing of the structure of high-dimensional signals specifically for solving inverse problems such as sparse recovery, compressive sensing, and phase retrieval (Bora et al., 2017; Kabkab et al., 2018; Hand et al., 2018). Specifically, Bora et al. (2017) have shown that generative models provide a good prior to structured signals, for e.g., natural images, under compressive sensing settings over sparsity-based recovery methods. They rigorously analyze the statistical properties of a generative model based on compressed sensing and provide theoretical guarantees and experimental evidence to support their claims. However, they don't explicitly propose an optimization procedure to solve the recovery problem. They simply suggest using stochastic gradient-based methods in the low-dimensional latent space to recover the signal of interest. This has been addressed by Shah and Hegde (2018), where the authors propose using a projected gradient descent algorithm for solving the recovery problem directly in the ambient space (space of the desired signal). They provide theoretical guarantees for the convergence of their algorithm and also demonstrate improved empirical results over Bora et al. (2017).

While GANs have found many applications, most of them need direct access to the clean samples from the unknown distribution, which is not the case in many real applications such as medical imaging. AmbientGAN framework Bora et al. (2018) partially addresses this problem. In particular, they studied various measurement models and showed that their GAN can find samples of clean signals from corrupted observations. Although similar to our effort, there are several key differences between ours and AmbientGAN. Firstly, AmbientGAN assumes that the measurement model and parameters are known, which is a very strong and limiting assumption in real applications. One of our main contributions is addressing this limitation by studying the demixing problem. Second, for the noisy measurement settings, AmbientGAN assumes an arbitrary measurement noise and no corruption in the underlying component. We consider the corruption models in the signal domain rather than as a measurement one. This allows us to study the denoising problem from a highly structured corruption. Lastly, their approach just learns the distribution of the clean images; however, it has not been used for the task of image denoisng (i.e., how to denoise an unseen corrupted image). Our framework addresses this issue as well.

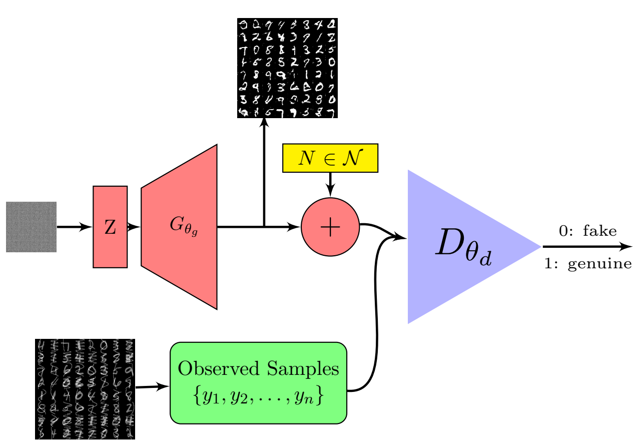
3.3 Background and the Proposed Idea

3.3.1 Background

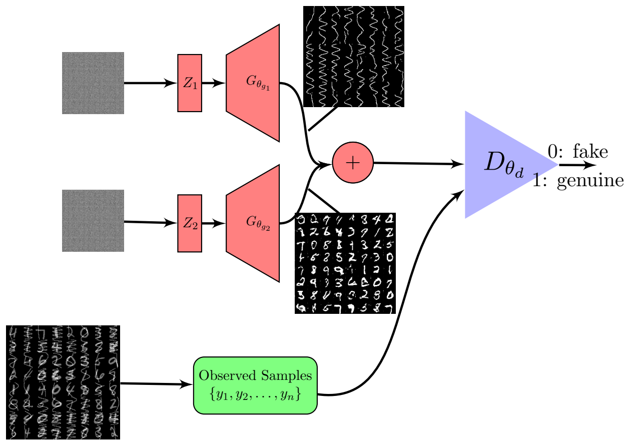
Generative Adversarial Networks (GANs) are one of the successful generative models in practice was first introduced by Goodfellow et al. (2014) for generating samples from an unknown target distribution. As opposed to the other approaches for density estimation such as *Variational Auto-Encoders (VAEs)* Kingma and Welling (2013), which try to learn the distribution itself, GANs are designed to generate samples from the target probability density function. This is done through a zero-sum game between two players, *generator*, G and *discriminator*, D in which the generator G plays the role of producing the fake samples and discriminator D plays the role of a cop to find the fake and genuine samples. Mathematically, this is accomplished through the following *min-max* optimization problem:

$$\min_{\theta_g} \max_{\theta_d} \quad \mathbb{E}_{x\sim\mathcal{D}_x}[log(D_{\theta_d}(x))]\mathbb{E}_{z\sim\mathcal{D}_z}[log(1 - D_{\theta_d}(G_{\theta_g}(z)))], \tag{3.2}$$

where θ_g and θ_d are the parameters of generator networks and discriminator network respectively, and $\mathcal{D}_x$ denotes the target probability distribution , and $\mathcal{D}_z$ represents the probability distribution of the hidden variables $z \in \mathbb{R}^h$, which is assumed either a uniform distribution in $[-1, 1]^h$, or standard normal. One can also use identity function instead of $log(.)$ function in the above expression. The resulting formulation is called WGAN (A. et al., 2017). It has been shown that if G and D have enough capacity, then solving optimization problem (3.2) by alternative stochastic gradient descent algorithm guarantees the distribution $\mathcal{D}_g$ at the output of the generator converges to $\mathcal{D}_x$. Having



(a)



(b)

Figure 3.1: The architecture of proposed GANs. (a) denoising-GAN. (b) demixing-GAN.

discussed the basic setup of GANs, next we present the proposed modifications to the basic GAN setup that allows for usage of GANs as a generative model for denoising and demixing structured signals.

3.3.2 denoising-GAN

Our idea is inspired by AmbientGAN due to Bora et al. (2018) in which they used a regular GAN architecture to solve some inverse problems such as inpainting, denoising from unstructured noise, and so on. In particular, assume that instead of clean samples, one has access to a corrupted version of the samples where the corruption model is captured by a known random function $f_N(n)$ where N is a random variable. For instance, the corrupted samples can be generated via just adding noise, i.e., $Y = X + N$. The idea is to feed discriminator with observed samples, y_i's (distributed as Y) together with the output of generator G which is corrupted by model $f_N(n)$. Our denoising-GAN framework is illustrated in Figure 3.1(a). This framework is similar to one proposed in Ambient GAN paper; however, the authors did not consider denoising task from structured superposition based corruption. In the experiment section, we show that denoising-GAN framework can generate clean samples even from structured corruption. Now we use the trained denoising-GAN framework for denoising of a new test corrupted image which has not been used in the training process. To this end, we use our assumption that the components have some structure and the representation of this structure is given by the last layer of the trained generator, i.e., $X \in G_{\widehat{\theta}_g}$[2]. This observation together with this fact that in GANs, the low-dimension random vector z is representing the hidden variables, leads us to this point: in order to denoise a new test image, we have to find a hidden representation, giving the smallest distance to the corrupted image in the space of $G_{\widehat{\theta}_g}$ (Shah and Hegde, 2018; Bora et al., 2017). In other words, we have to solve the following optimization problem:

$$\widehat{z} = \arg\min_{z} \|u - G_{\widehat{\theta}_g}(z)\|_2^2 + \lambda \|z\|_2^2, \tag{3.3}$$

where u denotes the corrupted test image. The solution of this optimization problem provides the (best) hidden representation for an unseen image. Thus, the clean image can be reconstructed by evaluating $G_{\widehat{\theta}_g}(\widehat{z})$. While optimization problem (3.3) is non-convex, we can still solve it by running gradient descent algorithm in order to get a stationary point[3].

3.3.3 Theoretical Insights for the denoising-GAN

Here, we revisit some theoretical arguments pioneered by (Bora et al., 2018). As authors have discussed in this paper in Lemma 5.1 and Theorem 5.2, so long as there is a bijection map from the probability distribution of observation space (Y-domain) to the probability distribution of signal space (X-domain), and by choosing optimal discriminator as $D = \frac{\mathcal{D}_y}{\mathcal{D}_y + \mathcal{D}_g}$, then it is guaranteed that the generator, G is optimal if and only if $\mathcal{D}_g = \mathcal{D}_y$. Here, $\mathcal{D}_g$ and $\mathcal{D}_y$ denote the probability distribution of generator and observation (corrupted signal), respectively. This can be proved by arguments given in the original GAN paper by Goodfellow et al. (2014). Authors in Bora et al. (2018) showed that the uniqueness map assumption is satisfied in some special cases. Since the probability distribution of the sum is given by convolution, from equation (3.1), we have: $\mathcal{D}_y = \mathcal{D}_x * \mathcal{D}_n$, where $\mathcal{D}_N$ denotes the distribution of the corruption part, N which we have access

[2]$G_{\widehat{\theta}_g}(.)$ denotes the trained generator network with parameter $\widehat{\theta}_g$.

[3]While we cannot guarantee the stationary point is a local minimum, but the empirical experiments show that gradient descent (implemented by backpropagation) can provide a good quality result.

to the samples from it, and $*$ denotes the convolution operator. If we take Fourier transform (or using the characteristic function, $\Phi(.)$) from this equation, we obtain: $\Phi_y = \Phi_x.\Phi_n$. As a result, $\mathcal{D}_x = \Phi_x^{-1}(\frac{\Phi_y}{\Phi_n})$. This means that the probability distribution of signal domain is determined uniquely (due to one-to-one relation between the probability distribution and the characteristic function). For $\mathcal{D}_x = \Phi_x^{-1}(\frac{\Phi_y}{\Phi_n})$ to be well defined a straightforward condition is that the Φ_n is non-zero almost everywhere. This condition is satisfied for many corruption distributions. For example, consider the case the true signal is distributed as Gaussian, and the corruption samples are drawn from uniform distribution between $[-1, 1]$ the Φ_n is non-zero almost everywhere. This is also satisfied for a n-dimensional random sparse vector with k non-zero entries such that there is non-zero probability of each set of k-entries and the joint-distribution of non-zero k-entries is Gaussian with full-rank covariance matrices. This covers many structured noises.

3.3.4 demixing-GAN

Now, we go through our main contribution, demixing. Figure 3.1(b) shows the GAN architecture, we are using for the purpose of separating or demixing of two structured signals form their superposition. As illustrated, we have used two generators and have fed them with two random noise vectors $z_1 \in \mathbb{R}^{h_1}$ and $z_2 \in \mathbb{R}^{h_2}$ according to a uniform distribution defined on a hyper-cube, where h_1, h_2 are less than the dimension of the input images. We also assume that they are independent of each other. Next, the output of generators are summed up and the result is fed to the discriminator along with the superposition samples, $y_i's$. In Figure 3.1(b), we just show the output of each generator after training for an experiment case in which the mixed image consists of 64 MNIST binary image LeCun and Cortes (2010) (for X part) and a second component constructed by random sinusoidal (for N part) (please see the experiment appendix for the details of this specific experiments). Somewhat surprisingly, this architecture based on two generators can produce samples from the distribution of each component after enough number of training iterations. We note that this approach is fully unsupervised as we only have access to the mixed samples and nothing from the samples of constituent components is known. As mentioned above, this is in sharp contrast with AmbientGAN and our previous structured denoising approach. As a result, the demixing-GAN framework can generate samples from the second components (for example random sinusoidal, which further can be used in the task of denoising where the corruption components are sampled from highly structured sinusoidal waves). Now similar to the denoising-GAN framework, we can use the trained generators in Figure 3.1(b), for demixing of the constituent components for a given test mixed image which has not been used in training. Similarly, we can solve the following optimization problem:

$$\widehat{z}_1, \widehat{z}_2 = \underset{z_1, z_2}{\arg\min} \|y - G_{\widehat{\theta}_{g_1}}(z_1) - G_{\widehat{\theta}_{g_2}}(z_2)\|_2^2 + \lambda_1 \|z_1\|_2^2 + +\lambda_2 \|z_2\|_2^2, \tag{3.4}$$

where u denotes the test mixed image.

Now, each component can be estimated by evaluating $G_{\widehat{\theta}_{g_1}}(\widehat{z}_1)$ and $G_{\widehat{\theta}_{g_2}}(\widehat{z}_2)$[4]. Similar to the previous case, while the optimization problem in (3.4) is non-convex, we can still solve it through block coordinate gradient descent algorithm, or in a alternative minimization fashion. We note that in both optimization problems (3.3) and (3.4), we did not project on the box sets on which z_1 and z_2 lie on. Instead we have used regularizer terms in the objective functions (which are not meant as projection step). We empirically have observed that imposing these regularizers can help to obtain good quality images in our experiment; plus, they may help that the gradient flow to be close in the region of interest by generators. This is also used in Bora et al. (2017). Finally, we have provided some theoretical intuitions for the demixing-GAN in the appendix.

3.4 Numerical Experiments

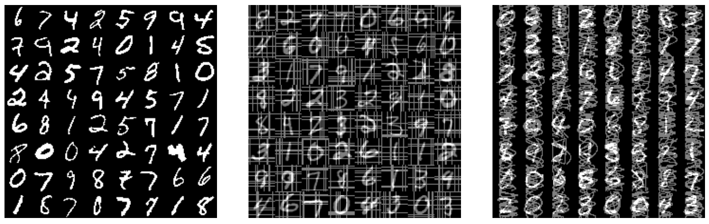
In this section, we present various experiments showing the efficacy of the proposed frameworks (depicted in Figure 3.1(a) and Figure 3.1(b)) in three different setups. First, we will focus on the denoising from structured corruption both in training and testing scenarios. Next, we focus on demixing signals from structured distributions. Finally, we explore the use of generative models from the proposed GAN frameworks in compressive sensing setup. In all the following experiments, we did our best for choosing all the hyper-parameters. We defer the details of experiments setup,

[4]$G_{\widehat{\theta}_{g_1}}(.)$ and $G_{\widehat{\theta}_{g_2}}(.)$ denote the first and second trained generator with parameter $\widehat{\theta}_{g_1}$ and $\widehat{\theta}_{g_2}$, respectively.

the complementary experiments on the compressive sensing scenario, and experiments on the other datasets (F-MNIST (Han et al., 2017), combination of F-MNIST and MNIST, SVHN Netzer et al. (2011), and Quick-Draw (Qui, a)) to the appendix.

3.4.1 Structured Corruption Models

For all the experiments in this section, we have used the network architectures for discriminator and generator(s) similar to the one proposed in DCGAN Radford et al. (2015). DCGAN is a CNN based GAN consists of convolutional layers followed by batch normalization (except the last layer of the generator and first layer of discriminator). We have also considered the binary MNIST dataset as the clean ground-truth component. For the corruption part, we have used two structured noise models similar to Chen and Srihari (2014). In the first one, we generate random vertical and horizontal lines and add them to the dataset. The second structured noise is constructed based on random sinusoidal waves in which the amplitude, frequency, and phase are random numbers. We define the level of corruption (lc) as the number of sinusoidal or lines added to the original image. We note that both of these corruption models are highly structured. The clean MNIST images along with these two corruption models have been shown in the left panel of Figure 3.2 and 3.3, respectively.



(a) (b) (c)

Figure 3.2: Left panel: (a). Clean binary MNIST image. (b). Corrupted image with random horizontal and vertical lines. (c). Corrupted image with random sinusoidal waves.

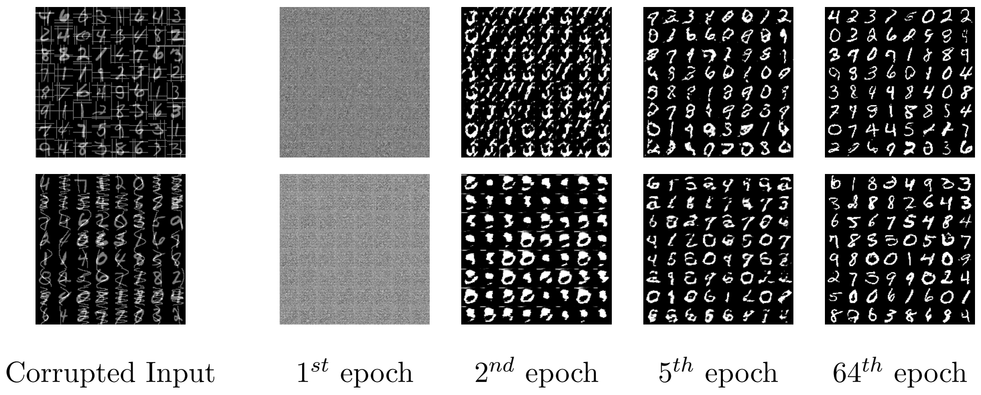


Figure 3.3: Evolution of output samples by the generator for fixed z. Top row is for random horizontal and vertical corruption. Bottom row is for random sinusoidal corruption.

3.4.2 Denoising from Structured Corruption – Training

In this section, we use GAN architecture illustrated in Figure 3.1(a) for removing of the structured noise. The setup of the experiment is as follows: we use 55000 images with size 28×28 corrupted by either of the above corruption models[5]. The resulting images, y_i's are fed to the discriminator. We also use hidden random vector $z \in \mathbb{R}^{100}$ drawn from a uniform distribution in $[-1, 1]^{100}$ for the input of the generator. During the training, we use 64 mini-batches along with the regular loss function in the GAN literature, stated in problem (3.2). The optimization algorithm is set to Adam optimizer, and we train discriminator and generator one time in each iteration. We set the number of epochs to 64. To show the evolution of the quality of output samples by the generator, we fix an input vector z and save the output of the generator at different times during the training process. The right panel of Figure 3.3 shows the denoising process for both corruption models. The top row denotes the denoising from random vertical and horizontal lines, while the bottom row corresponds to the random sine waves.

As we can see, the GAN used in estimating the clean images from structured noises is able to generate clean images after training of generator and discriminator. In the next section, we use our trained generator for denoising of new images which have not been used during the training.

[5]We set level of corruption 1 for sinusoidal and 2 for the vertical and horizontal lines in the training of the denoising-GAN.

3.4.3 Denoising from Structured Corruption – Testing

Now we test our framework with test corrupted images for both models of corruption introduced above. For reconstructing of the clean images, we solve the optimization problem in (3.3) to obtain solution $\widehat{z}$. Then we find the reconstructed clean images by evaluating $G_{\widehat{\theta}_g}(\widehat{z})$. In Figures 3.4, we have used the different level of corruptions for both of the corruption models. In the top right, we vary the level of corruption from 1 to 5 with random sines. The result of $G_{\widehat{\theta}_g}(\widehat{z})$ has been shown below of each level of corruption. In the top left, we have a similar experiment with various level of vertical and horizontal corruptions. Also, we show the denoised images in the below of the corrupted ones. As we can see, even with heavily corrupted images (level corruption equals to 5), GAN is able to remove the corruption from unseen images and reconstruct the clean images. In the bottom row of Figure 3.4, we evaluate the quality of reconstructed images compared to the corrupted ones through a classification task. That is, we use a pre-trained model for MNIST classifier which has test accuracy around %98[6]. We feed the MNIST classifier with both denoised (output of denoising-GAN) and corrupted images with a different level of corruptions. For the ground truth labels, we use the labels corresponding to the images before corruption. One interesting point is that when the level of corruption increases the denoised digits are sometimes tweaked compared to the ground truth. That is, in pixel-level, they might not close to the ground truth; however, semantically they are the same. We have also plot the reconstruction error per pixel (normalized by 16 images) for various lc's.

3.4.4 Demixing of Structured Signals – Training

In this section, we present the results of our experiments for demixing of the structured components. To do this, we use the proposed architecture in Figure 3.1(b). We first present our experiment with MNIST dataset, and then we show the similar set of experiments with Fashion-MNIST dataset (F-MNIST) (Han et al., 2017). This dataset includes 60000 training 28×28 gray-scale images with

[6]The architecture comprises of two initial convolutional layers along with max-pooling followed by a fully connected layer and a dropout on top of it. Relu is used for all the activation functions and a soft-max function is used in the last layer.

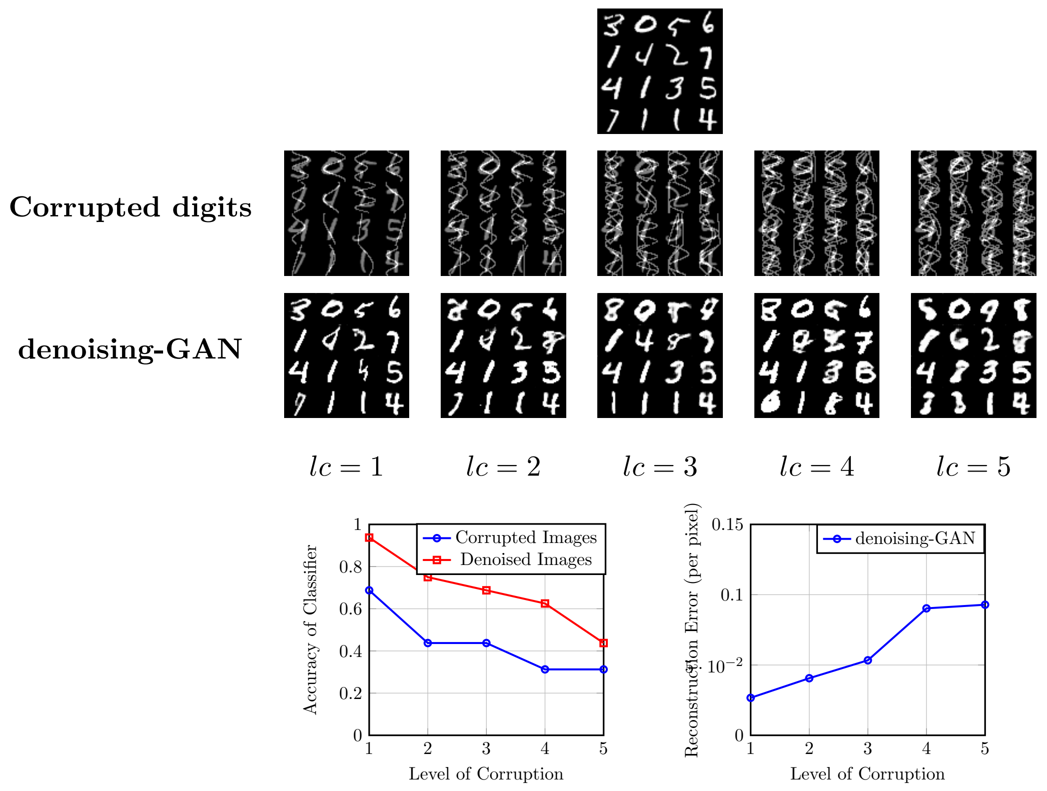


Figure 3.4: The Performance of the trained generator for various *levels of corruption (lc)* in denoising of unseen images. Top row: Ground-truth digits together with corrupted digits with random sines with a level of corruption from 1 to 5 and denoised digits using denoised-GAN. Bottom row: Classification accuracy of pre-trained MNIST classifier for both corrupted and denoised digits along with the reconstruction error per-pixel.

10 labels. The different labels denote objects, including T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot.

3.4.4.1 Experiments on MNIST Dataset

We start the experiments with considering four sets of constituent components. In the first two, similar to the denoising case, we use both random sinusoidal waves and random vertical and horizontal lines for the second constituent component. The difference here is that we are interested in generating of the samples from the second component as well. In Figure 3.6, we show the training evolution of two fixed random vectors, z_1 and z_2 in $\mathbb{R}^{100}$ in the output of two generators. In the

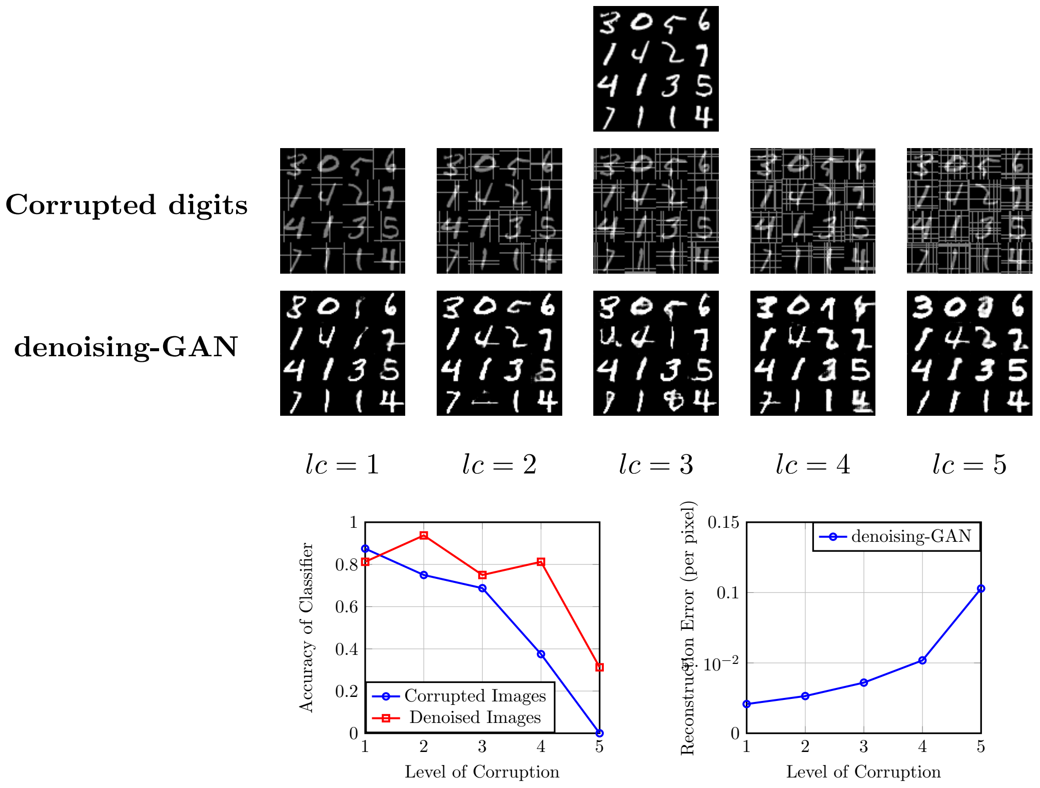


Figure 3.5: The Performance of trained generator for various *level of corruption (lc)* in denoising of unseen images. Top row: Ground-truth digits together with corrupted digits with vertical and horizontal lines with a level of corruption from 1 to 5 and denoised digits using denoised-GAN. Bottom row: Classification accuracy of pre-trained MNIST classifier for both corrupted and denoised digits along with the reconstruction error per-pixel.

top panel, we have added one random sinusoidal waves to the clean images. As we can see, our proposed GAN architecture can learn two distributions and generate samples from each of them. In the bottom panel, we repeat the same experiment with random vertical and horizontal lines as the second component (two random vertical and two random horizontal lines are added to the clean images). While there is some notion of mode collapse, still two generators can produce the samples from the distribution of the constituent components.

In the second scenario, our mixed images comprise of two MNIST digits from 0 to 9. In this case, we are interested in learning the distribution from which each of the digits is drawn. The Top panel in Figure 3.7 shows the evolution of two fixed random vectors, z_1 and z_2. As we can see, after 32 epoch, the output of the generators would be the samples of MNIST digits. Finally, in the last scenario, we generate the mixed images as the superposition of digits 1 and 2. In the training set of MNIST dataset, there are around 6000 samples from each digits of 1 and 2. We have used these digits to form the set of superposition images. The bottom panel of Figure 3.7 shows the output of two generators, which can learn the distribution of the two digits. The interesting point is that these experiments show that each GAN can learn the existing digit variety in MNIST training dataset, and we typically do not see mode collapse, which is a major problem in the training of GANs (Goodfellow, 2016).

3.4.4.2 Experiments on F-MNIST Dataset

In this section, we illustrate the performance of the proposed demixing-GAN for another F-MNIST dataset. This dataset includes 60000 training 28×28 gray-scale images with 10 labels. The different labels denote objects, including T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot. Similar to the experiment with MNIST dataset being illustrated in Figure 3.7, we train the demixing-GAN where we have used InfoGAN Chen et al. (2016) architecture for the generators. The architecture of the generators in InfoGAN is very similar to the DCGAN discussed above with the same initialization procedure. The dimension of input noise to the generators is set to 62. We have also used the same discriminator in DCGAN. Figure 3.8 shows the output of two generators, which can learn the distribution of dress and bag images during 21 epochs.

3.4.5 Demixing of Structured Signals – Testing

Similar to the test part of denoising, in this section, we test the performance of two trained generators in a demixing scenario for the mixed images, which have not been seen in the training time. Figure 3.9 shows our third experiment in which we have illustrated the demixing on three different input mixed images. Here, we have compared the performance of demixing-GAN with *Independent component analysis (ICA)* method (Hoyer et al., 1999). In the top and middle rows

Table 3.1: Numerical Evaluation of the results in Figure 3.9 according to the *Mean Square Error (MSE)* and *Peak Signal-to-Noise ratio (PSNR)* criteria between the corresponding components in the superposition model.

	MSE (1st Part)	MSE (2nd Part)	PSNR (1st Part)	PSNR (2nd Part)
First Row	0.04715	0.03444	13.26476	14.62877
Second Row	0.04430	0.03967	13.53605	14.01344
Third Row	0.05658	0.05120	12.47313	12.90715
Forth Row	0.08948	0.10203	10.48249	9.91264

Table 3.2: Numerical Evaluation of the results in Figure 3.10 according to the *Mean Square Error (MSE)* and *Peak Signal-to-Noise ratio (PSNR)* criteria between the corresponding components in the superposition model.

	MSE (1st Part)	MSE (2nd Part)	PSNR (1st Part)	PSNR (2nd Part)
First Row	0.16859	0.12596	7.73173	8.99763
Second Row	0.05292	0.03304	12.76368	14.80992
Third Row	0.13498	0.11758	8.69732	9.29655
Fourth Row	0.12959	0.08727	8.87432	10.59132
Fifth Row	0.18250	0.12221	7.38733	9.12906

of Figure 3.9, we consider the mixed images generated by adding a digit (drawn from MNIST test dataset) and a random sinusoidal. Then the goal is to separate (demix) these two from the given superimposed image. To do this, we use GAN trained for learning the distribution of digits and sinusoidal waves (the top panel of Figure 3.6) and solve the optimization problem in (3.4) through an alternative minimization fashion. As a result, we obtain $\widehat{z}_1$ and $\widehat{z}_2$. The corresponding constituent components is then obtained by evaluating $G_{\widehat{\theta}_{g_1}}(\widehat{z}_1)$ and $G_{\widehat{\theta}_{g_2}}(\widehat{z}_2)$. In Figure 3.9, the first two columns denote the ground-truth of the constituent components. The middle one is the mixed ground-truth, and the last two show the recovered components using demixing-GAN and ICA. In the last row, digits 1 and 2 drawn from the MNIST test dataset are added to each other and we apply the GAN trained for learning the distribution of digits 1 and 2 (bottom panel in Figure 3.7). As we can see, our proposed GAN can separate two digits; however, ICA method fails in demixing of two components. In addition, Table 3.1 has compared numerically the quality of recovered components with the corresponding ground-truth ones through *mean square error (MSE)* and *Peak Signal-to-Noise Ratio (PSNR)* criteria.

3.4.5.1 Demixing of F-MNIST – Testing

In this section, we evaluate the performance of trained demixing-GAN on the F-MNIST dataset. In Figure 3.10, we have illustrated an experiment similar to the setup in section 3.4.5. The first two columns in Figure 3.10 denote two objects from F-MNIST test dataset as the ground-truth components. The third column is the ground-truth mixed image, and the last two columns show the recovered constituent components. The first row uses the generator trained for only two objects for 20 epochs. The second row uses the generator trained for all 10 objects for 20 epochs. The third and fourth rows use the same generator trained for only two objects for 30 epochs. The last row shows the result of demixing with ICA method. We have implemented ICA using Scikit-learn module (Pedregosa et al., 2011). As we can see, ICA fails to separate the components (images of F-MNIST) from each other, while the proposed demixing-GAN can separate the mixed images from each other. However, the estimated image components are not exactly matched to the ground-truth ones (first two columns). This has been shown through numerical evaluation according to MSE and PSNR in Table 3.2.

Finally, as an attempt to understand the condition under which the demixing-GAN is failed, we empirically investigated the role of two aspects pf demixing-GAN. First, it seems that the hidden space (z-space) of the generators for characterizing the distribution of the constituent components play an essential role in the success/failure of the demixing-GAN. Second the incoherent hidden structures in the components in the generator space. We investigate these observations through some numerical experiments in section 3.6.8 in the appendix.

3.5 Conclusion

In this chapter, we considered the GANs framework for learning the structure of the constituent components in a superposition observation model. We empirically showed that it is possible to implicitly learn the underlying distribution of each component and use them in the downstream task such as denoising, demixing, and sparse recovery. We also investigate the conditions under which the proposed demixing framework fails through extensive experimental simulations and some theoretical insights.

3.6 Appendix. Overview

3.6.1 Appendix A. Some Theoretical Intuitions about The demixing-GAN

Recall that the superposition model is given by $Y = X + N$, and $\mathcal{D}_y$, $\mathcal{D}_x$ and $\mathcal{D}_n$ denote the distribution of Y, X, and N, respectively. Let $G_1(z_1) \triangleq G_{\theta_{g_1}}(z_1) \sim \mathcal{D}_{g_1}$ and $G_2(z_2) \triangleq G_{\theta_{g_2}}(z_2) \sim \mathcal{D}_{g_2}$. Also assume that $(z_1, z_2) \sim \mathcal{D}_{z_1,z_2}$ denotes the joint distribution of the hidden random vectors with marginal probability as $\mathcal{D}_{z_i}$ for $i = 1, 2$. We note that in demixing setting there are not samples from the component N as opposed to the denoising scenario. Now we have the following mini-max loss as (3.2):

$$\min_{G_1,G_2} \max_{D} \mathcal{L}(G_1, G_2, D) = \mathbb{E}_{u\sim\mathcal{D}_y} log(D(u)) + \mathbb{E}_{(z_1,z_2)\sim\mathcal{D}_{z_1,z_2}} log(1 - D(G_1(z_1) + G(z_2))). \tag{3.5}$$

Following the standard GAN framework, for the fixed G_1 and G_2, we have:

$$\mathcal{L}(G_1, G_2, D) = \int_u (\mathcal{D}_y(u) log(D(u)) + \mathcal{D}_G(u) log(1 - D(u)))du,$$

where $\mathcal{D}_G = \mathcal{D}_{g_1} * \mathcal{D}_{g_2}$. Hence, the optimal discriminator is given by $D^* = \frac{\mathcal{D}_x*\mathcal{D}_n}{\mathcal{D}_x*\mathcal{D}_n+\mathcal{D}_{g_1}*\mathcal{D}_{g_2}}$ since $Y = X + N$ and the fact that $\mathcal{D}_y$ and $\mathcal{D}_g$ are the pdf and defined in $[0, 1]$. This means that the global optimal of problem (3.5) is achieved iff $\mathcal{D}_x * \mathcal{D}_n = \mathcal{D}_{g_1} * \mathcal{D}_{g_2}$ ($*$ denotes the convolution operator). However, this condition is generally an ill-posed equation. That is, in general, $\mathcal{D}_x \neq \mathcal{D}_{g_1}$ and $\mathcal{D}_n \neq \mathcal{D}_{g_2}$. In the best case, we can have hope to uniquely determine the distributions up to a permutation (similar thing is also true for the ICA method). This is the point actually we need some notion of incoherence between two constituent structures, $\mathcal{X}$, and $\mathcal{N}$. So, the question is this under what incoherent condition, we can have a well-conditioned equation? According to our previous discussion in the denoising case, even if we somehow figure out the right incoherent condition, $\mathcal{D}_x$ is uniquely determined by $\mathcal{D}_n$ if the Fourier transform of $\mathcal{D}_n$ is non-zero. While we currently do not have a right answer for the above question, we conjecture that in addition to the incoherence issue in the signal domain, the hidden space (z-space) in both generators play an important role to make the demixing problem possible. We investigate this idea and the other things empirically at the end of the appendix.

3.6.2 Appendix B. Detail of Experiments

Here, we give some more details about the initialization of the generators we have used for the MNIST and F-MNIST experiments in sections 3.4.4 and 3.4.5. As we mentioned earlier, we have used the same architecture based on the DCGAN for both of the generators used in the demixing-GAN for MNIST dataset. In particular, the generators include three layers: the first two layers are fully connected layers with Relu activation function. We have not used any batch-normalization (as opposed to the original DCGAN). Also, the weights in these fully connected layers are initialized according to the random normal distribution with standard deviation equals to 0.02. We have also used zero initialization for the biases. The third layer includes a transposed convolution layer with filters size of which are set to 5 and stride to 2. We have initialized these filters according to the random normal distribution with standard deviation equals to 0.02. Relu nonlinearity has also been used after the transposed convolution operation. We have not used any max-pooling in this architecture. Similarly, the discriminator comprises two convolution layers with leaky-rely activation function followed by a fully connected layer without any max-pooling. The weights in these convolution layers have been initialized based on the truncated normal distribution with standard deviation equals to 0.02. The fully connected layer is initialized as before. Finally, we fed two generators with $i.i.d$ random noise vector with entries uniformly drawn from $[-1, 1]$. For the F-MNIST experiments, we have used a similar architecture to the InfoGAN generator (Chen et al., 2016). The architecture of the generators is similar to the previous case except that the transposed convolution in the third layer has 128 filters with size 4×4 and stride 2, while the previous DCGAN has 64 filters with size 5×5 and stride 2. The discriminators are the same. For the ICA experiment, we simulate the mixing process simalr to Anirudh et al. (2018) using mixing matrix with entries drawn from a truncated random normal distribution, i.e., $W_{ij} \sim \mathcal{N}(-0.5, 0.5)$ for $i, j = 1, 2$, allowing for negative weights. Hence, the final mixed observation is given by $Y = XW^T$ where X denotes the constituent component matrix (source matrix) with 2 columns (number of components) and with number of rows equal to the size of input mixed image (28×28).

3.6.3 Appendix C. Compressive Sensing

In this section, we present results in the compressed sensing setting. The goal here is to understand the quality of generative models that can be learned under the structured corruption in *denoising*-GAN and *demixing*-GAN architectures. To this end, we test the capability of the proposed GANs as the generative model for natural images. The experimental setup is similar to the reconstruction from Gaussian measurement experiment for MNIST data reported in Bora et al. (2017). In particular, we assume the following observation model by a random sensing matrix $A \in \mathbb{R}^{m \times p}$ with $m < p$ under structured corruption by N as follows:

$$Y = A(X + N), \tag{3.6}$$

where the entries of A are i.i.d Gaussian with zero mean and variance $\frac{1}{m}$. For a given generator model $G_{\widehat{\theta}}$, we solve the following inference problem:

$$\min_z \|Y - AG_{\widehat{\theta}}(z)\|_2^2 + \lambda \|z\|_2^2, \tag{3.7}$$

where the generator $G_{\widehat{\theta}}$ is taken from various generator networks. For this experiment, we select the signal X randomly from MNIST test dataset, and N from random sinusoidal corruption with various number of waves. We compare generator models learned from clean MNIST images, i.e., generator model obtained from *denoising*-GAN and *demixing*-GAN framework. Both the generator models learned from denoising-GAN and demixing-GAN were trained under wave corruption model. For demixing-GAN, we select appropriate GAN by manually looking at the output of generators and choosing the one that gives MNIST like images as output. We also compare our approach to LASSO as MNIST images are naturally sparse. As reported in Bora et al. (2017). for this experiment the regularization parameter λ was set as 0.1 as it gives the best performance on validation set. For solving the inference problem in (3.7), we have used ADAM optimizer with step size set to 0.01. Since the problem is non-convex, we have used 10 random initialization with 10000 iterations. We choose the one gives the best measurement error.

The results of this experiment are presented in Figure 3.11 where we report per-pixel reconstruction error on 25 images chosen randomly from the test dataset for two different corruption levels. The plot on the left panel is obtained when the signal is corrupted with 2 random waves, whereas the right plot corresponds to the corruption with 4 random waves. In the both figures, we observe that with corruption, the performance of LASSO significantly degrades. The generator from demixing-GAN improves the performance over LASSO. Generator obtained from denoising-GAN performs comparably to the vanilla GAN. This experiment establishes the quality of the generative models learned from GANs as the prior. The comparable performance of denoising GAN with vanilla GAN shows that meaningful prior can be learned even from heavily corrupted samples.

3.6.4 Appendix D. Experiments on F-MNIST Dataset

In this section, we present more experiment on F-MNIST dataset. In particular, we construct the mixed images by adding randomly the images in all classes in FMNIST training dataset. As mentioned before, there are 10 different categories in this dataset. Figure 3.12 shows the evolution of two fixed random vectors, z_1 and z_2 drawn from $[-1,1]^{62}$. As we can see, after 21 epoch, the output of the generators would be the samples of F-MNIST objects. We also generate mixed images as the superposition of two objects, dress and bag images. In the training set of the F-MNIST dataset, there are around 6000 dress and bag images. We have used these images to form the set of superposition images.

3.6.5 Appendix E. Experiments on Both MNIST and F- MNIST Datasets

In this section, we explore the performance of demixing-GAN when the superposed images comprise the sum of a digit 8 from MNIST dataset and dress from the F-MNIST dataset. The experiment for this setup has been illustrated in Figure 3.13. Since our goal is to separate dress from the digit 8, for the first generator, we have used the InfoGAN architecture being used in the experiment in section 3.6.4 and similarly the DCGAN architecture for the second generator as section 3.4.4. As a result, the input noise to the first generator is drawn uniformly from $[-1, 1]^{62}$ and uniformly from $[-1, 1]^{100}$ for the second generator. Figure 3.13 shows the evolution of output samples by two generators for fixed z_1 and z_2. As we can see, after 21 epoch, the first generator is able to generate dress samples and the second one outputs samples of digit 8.

3.6.5.1 Demixing both MNIST and F-MNIST – Testing

Similar to the previous Testing scenarios, in this section, we evaluate the performance of the demixing-GAN in comparison with ICA for separating a test image which is the superposition of a digit 8 drawn randomly from MNIST test dataset and dress object drawn randomly from F-MNIST test dataset. Figure 3.14 shows the performance of demixng-GAN and ICA method. As we can see, ICA totally fails to demix the two images from each other, whereas the demixing-GAN is able to separate digit 8 very well and to some extend the dress object from the input superposed image. MSE and PSNR values for the first component using ICA recovery method is given by 0.40364 and 3.94005, respectively. Also, MSE and PSNR for the first component using ICA recovery method is given by 0.15866 and 7.99536, respectively.

3.6.6 Appendix F. Experiment on Quick-Draw Dataset

In this section, we present our demixing framework in another dataset, Quick-Draw dataset (Qui, a) released by Google recently. The Quick Draw Dataset is a collection of 50 million drawings categorized in 345 classes, contributed by players of the game Quick, Draw! (Qui, b). For this section, we select 5 classes out of 345 possible categories, including airplane, animal migration, face, flower, and The Eiffel Tower. Also, for each class, we consider 16000 images of size 28×28. Figure 3.15 shows the experiment for all the classes. Similar to MNIST and F-MNIST, we construct the training set by adding all the images randomly from 5 categories. To run this experiment, we feed each generator with random vectors z_1 and z_2 drawn uniformly from $[-1, 1]^{64}$. As illustrated in the figure 3.15, after 31 epochs two generators can generate samples from the distribution of all five classes. From this experiment, it seems that Quick-Draw dataset is more complicated than MNIST or perhaps F-MNIST as it takes more time for demixing-GAN to be able to output resemble samples to the original Quick-Draw objects.

Next, we consider only two objects, face, and flower in the Quick Draw Dataset. As a result, the input mixed images are the superposition of different faces and flowers. Figure 3.16 shows the evolution of the random vectors z_1 and z_2 (drawn uniformly from $[-1, 1]^{64}$). As we can see, after 31 epochs, one generator can produce various kind of faces, while the other one generates different shapes of flowers.

Finally, we consider a more challenging scenario in which the constituent components in the mixed images are just airplane shapes. That is, we randomly select the airplane shapes from 16000 images in the training set, and add them together to construct the input mixed images. We have been noticed that in the 16000 images of the airplane shapes, in general, there are two structures. One is related to the airplanes having been drawn by the players in the game Quick, Draw more simply and somehow flat (they are mostly similar to an ellipse with or without wings), while the second one consists the more detailed shapes (they have the tail and maybe with different orientation).

Figure 3.17 depicts the performance of demixing-GAN for this setup. One surprising point is that while both components in the superposition are drawn from one class (e.g., airplane shapes), the demixing-GAN is still able to demix the hidden structure in the airplane distribution. Thus, we think that just having the same distribution for both of the constituent components is not necessarily a barrier for demixing performance. We guess that somehow different features of the shapes drawn from the same distribution makes demixing possible by forcing the incoherence between the components. As we can see, after 31 epochs, both generators can learn two mentioned structures, and regarding two structures, they can cluster the shape of airplanes in two types.

3.6.7 Appendix G. Experiment on SVHN Dataset

Now we present some experimental results with colorful images. Specifically, we use the demixing-GAN with SVHN dataset Netzer et al. (2011). The Street View House Numbers (SVHN) training dataset is a collection of almost 70000 images, containing images of digits from 1 to 10. SVHN dataset is significantly more challenging to learn its distribution as it is noisy, including images of various resolution and distracting digits (Chen et al., 2016). In our experiment, we use the character level representation of SVHN which are pre-processed 32×32 colorful images. Figure 3.18 illustrates a similar experiment which we mentioned before. The mixed images also comprise the superposition of two randomly chosen digits from the SVHN training dataset. In addition, the dimension of the hidden space (i.e., z-space) for selecting random vectors z_1 and z_2 is set to 100. As expected, it takes more time compared to the other datasets for demixing-GAN to explore the distribution of the digits in SVHN. The authors of (Chen et al., 2016) pointed out that InfoGAN can capture two components in SVHN: Lighting and Context, where the context represents the central digit in an image. In demixing-GAN, we can see that one generator tries to learn the samples of SVHN dataset, the context part, while the other one mostly captures the lightning of the digits.

3.6.8 Appendix H. Failure of The demixing-GAN

In this section, we empirically explore our observation about the failure of the demixing-GAN. As we discussed briefly in section 3.4.5, we focus on two spaces, hidden space (z-space) and signal or generator space (the output of generator) in discovering the failure of demixing-GAN.

Our first observation concerns the z-space. We observe that if the hidden vectors form z-space of two generators are aligned to each other, then the two generators cannot output the samples in the signal space, representing the distribution of the constituent components. To be more precise, in Figure 3.19, we consider separating digits 8 and 2 from their superpositions similar to the experiment in the bottom panel of Figure 3.7. However, here, we feed both generators with the same vector, i.e., $z_1 = z_2$ in each batch (this is considered as the extreme case where precisely the hidden variables equal to each other) and track the evolution of the output samples generated by both generators. As we can see even after 21 epochs, the generated samples by both generators are an unclear combination of both digits 2 and 8, and they are not separated clearly as opposed to the case when we feed the generators with $i.i.d$ random vectors. We also repeat the same experiment with two aligned vectors z_1 and z_2, i.e., $z_2 = 0.1z_1$, Figure 3.20 shows the evolution of the output samples generated by both generators for this setup. As shown in this experiment, two generators cannot learn the distribution of digits 8 and 2. While we do not currently have a mathematical argument for this observation, we conjecture that the hidden space (z-space) is one of the essential pieces in the demixing performance of the proposed demixing-GAN. We think that having (random) independent or close orthogonal vector z's for the input of each generator is a necessary condition for the success of learning of the constituent components distribution, and consequently demixing of them. Further investigation of this line of study is indeed an interesting research direction, and we defer it for future research.

In addition to the hidden space, here we design some experiments in the generator space that reveals the condition under which the demixing is failed. In particular, we consider the airplane images in Quick-Draw dataset. To construct the input mixed images, we consider randomly chosen images of the airplane from 16000 images as the first component. Then, the second component is constructed by rotating exactly the same one in the first components in a counterclockwise direction. We consider 5 different rotations, $0°$, $10°$, $30°$, $60°$, $90°$. Five samples of such images are depicted in Figure 3.21. This experiment is sort of similar to the one in Figure 3.17 in which we have seen that demixing-GAN can capture the internal structure in the airplane dataset by clustering it into two types.

Now we perform the demixing-GAN on these datasets. Figure 3.22 illustrated the the evolution of the generators for various rotation degrees. The top panel shows the case exactly both components are the same shape. Obviously, the demixing, in this case, is impossible as there is no hope to distinguish the components from each other. Going down in the figure3.21, we have different rotation settings. As we can see, once we move forward to the 90°, both generators can capture the samples from the airplane distribution; however, as not clear as the case in which we had added the airplane shapes randomly for the input mixed images. We conjecture that changing the orientation of one component can make it incoherent to some extent from the other component, and consequently makes the demixing possible. In other words, we see again when two images show some distinguishable structures (in this case, the first one has 0-oriented object and the other is the same one but rotated 90°), then the demixing-GAN can capture these structures.

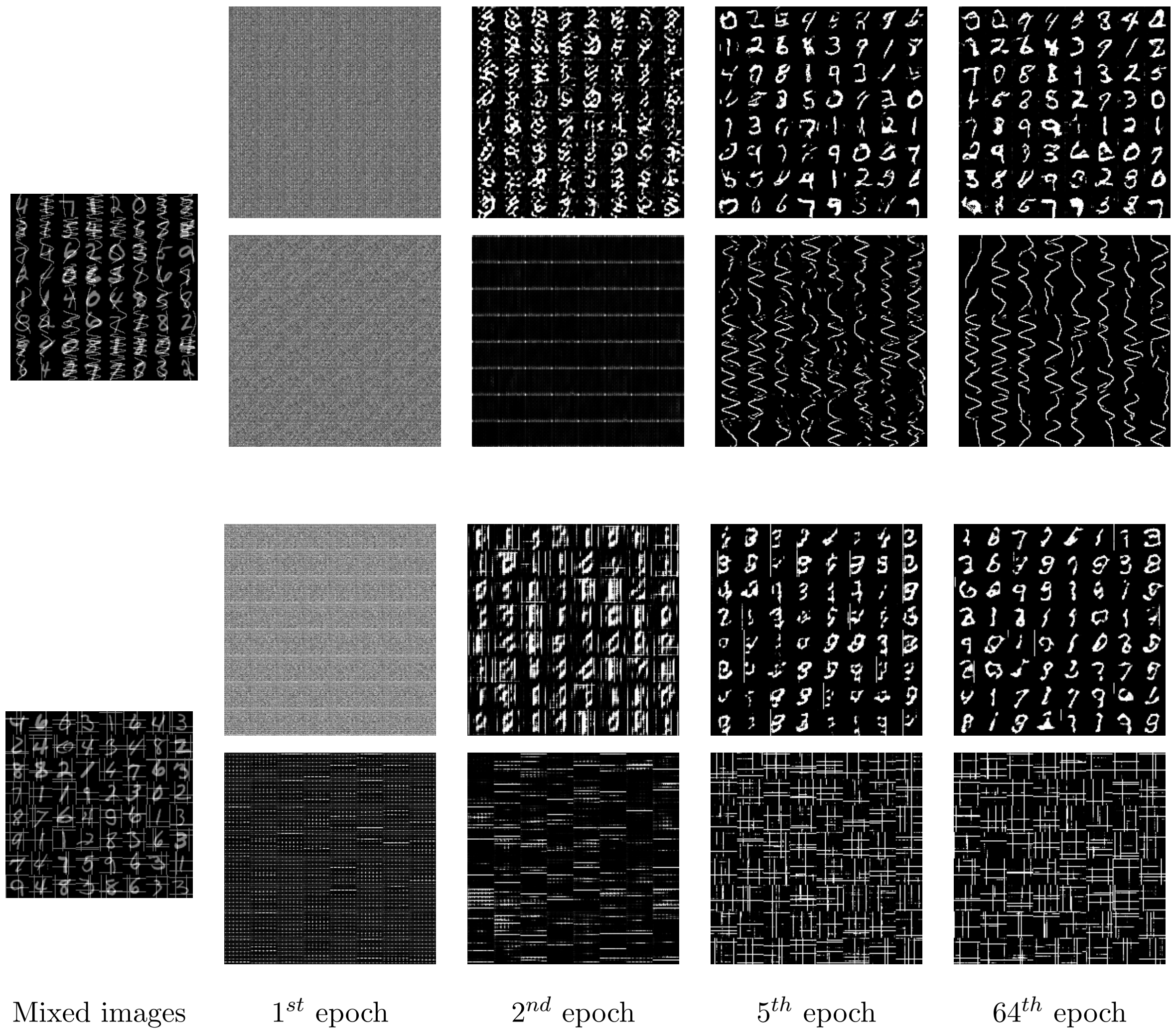


Figure 3.6: Evolution of output samples by two generators for fixed z_1 and z_2. The top panel shows the evolution of the two generators in different epochs where the mixed images comprise of digits and sinusoidal. The first generator is learning the distribution of MNIST digits, while the second one is learning the random sinusoidal waves. The bottom panel shows the same experiment with random horizontal and vertical lines as the second components in the mixed images.

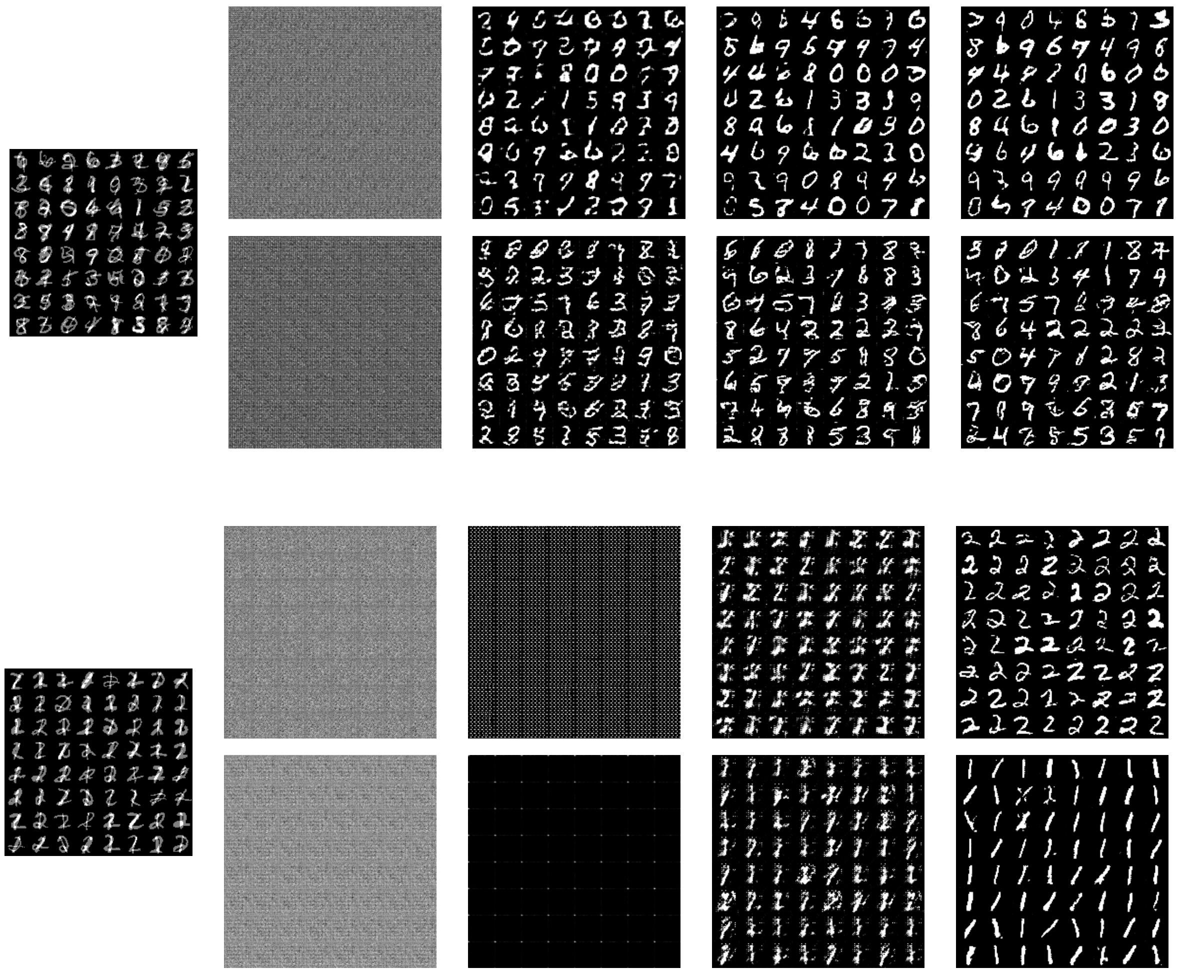


Figure 3.7: Evolution of output samples by two generators for fixed z_1 and z_2. The top panel shows that each generator is learning the distribution of one digit out of all 10 possible digits. The mixed images comprise two arbitrary digits between 0 to 9. The bottom panel panel shows a similar experiment where the mixed images comprise only digits 1 and 2.

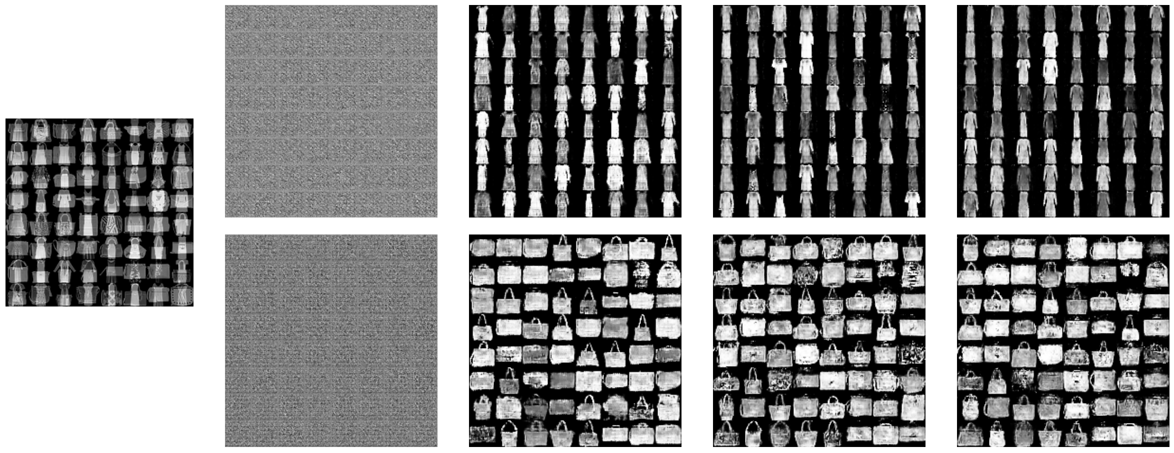


Figure 3.8: Evolution of output samples by two generators for fixed z_1 and z_2. The mixed images comprise only two objects, dress, and bag in training F-MNIST dataset. One generator produces the samples from dress distribution, while the other one outputs the samples from the bag distribution.

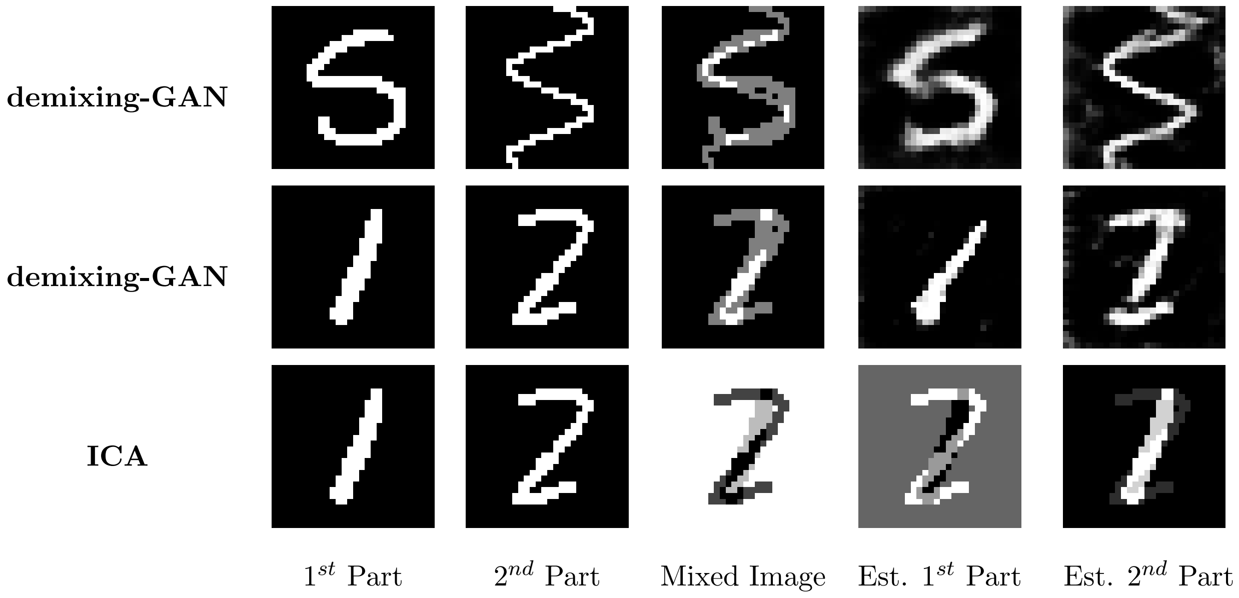


Figure 3.9: The performance of trained generators for demixing of two constituent components. The first two columns are the ground-truth components. The third column is the ground-truth mixed image and the last two columns denote the recovered components. The first row uses the same generator trained for only one digit (drawn from MNIST test dataset) and a random sinusoidal. The second row uses the generator trained only for digits 1 and 2. The last row shows the result of demixing with ICA method.

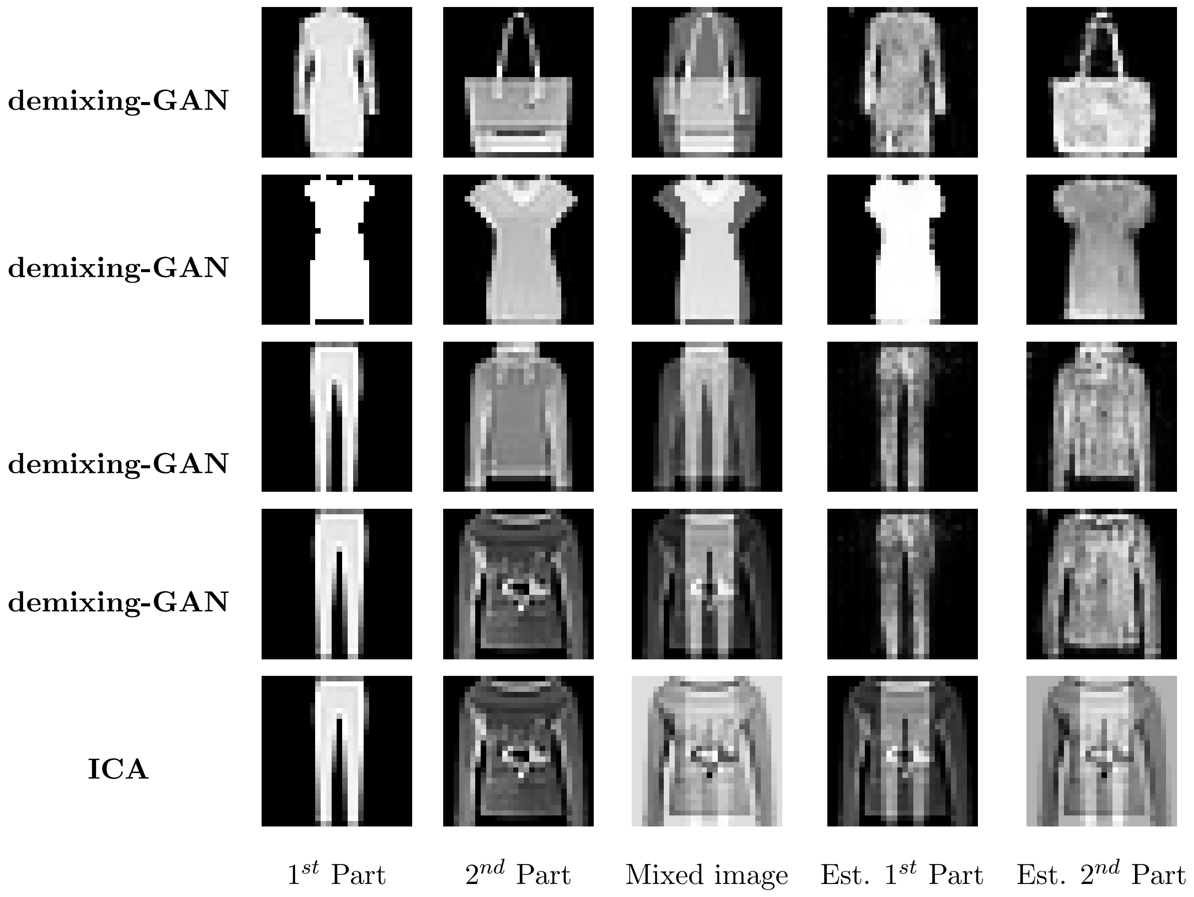


Figure 3.10: The performance of trained generators for demixing of two constituent components. The first two columns are the ground-truth components. The third column is the ground-truth mixed image and the last two columns denote the recovered components. The first row uses the generator trained for only two objects for 20 epochs. The second row uses the generator trained for all 10 objects for 20 epochs. The third and fourth rows use the same generator trained for only two objects for 30 epochs. The last row shows the result of demixing with ICA method.

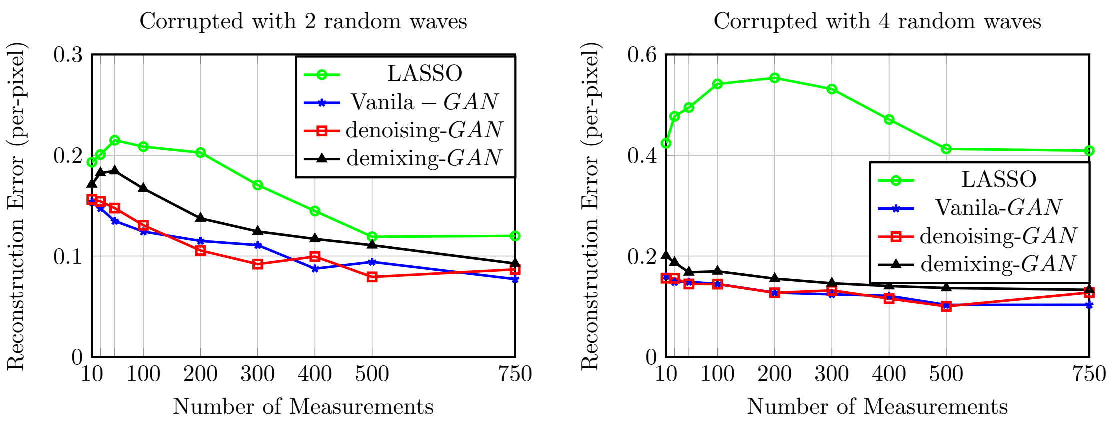


Figure 3.11: Performance of Different GANs in Compressive Sensing Experiments.

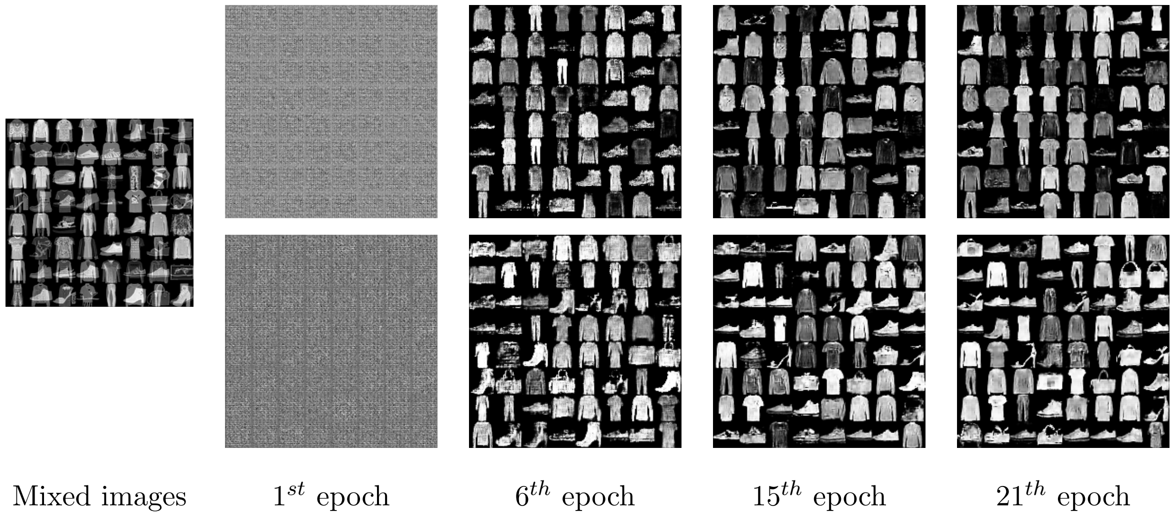


Figure 3.12: Evolution of output samples by two generators for fixed z_1 and z_2. The mixed images comprise two arbitrary objects drawn from 10 objects from training F-MNIST dataset. Each generator outputs the samples from the distribution of all 10 possible objects.

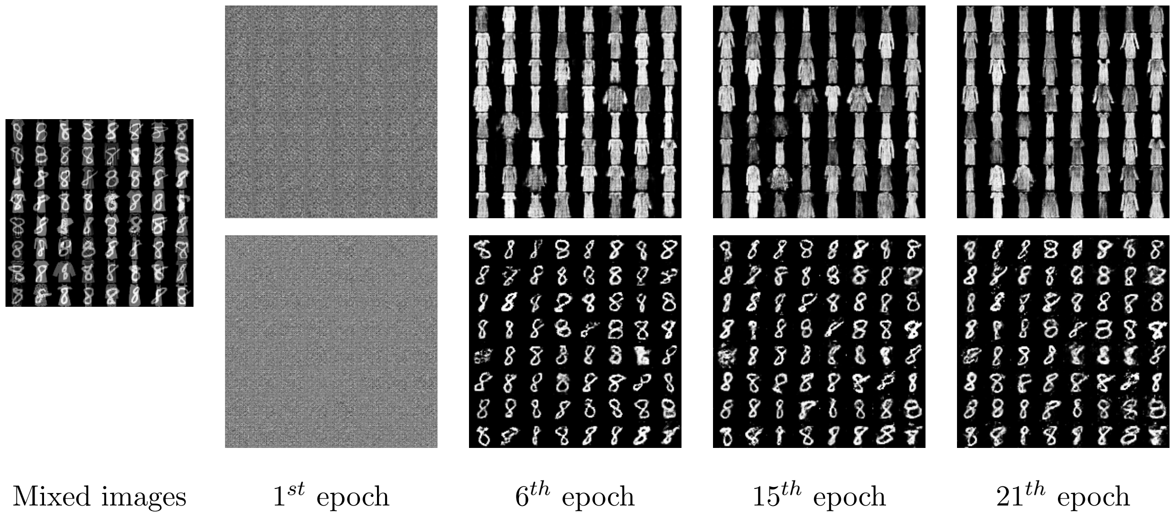


Figure 3.13: Evolution of output samples by two generators for fixed z_1 and z_2. The mixed images comprise only two objects, dress, and bag in training F-MNIST dataset. One generator produces the samples from digit 8 distribution, while the other one outputs the samples from the dress distribution.

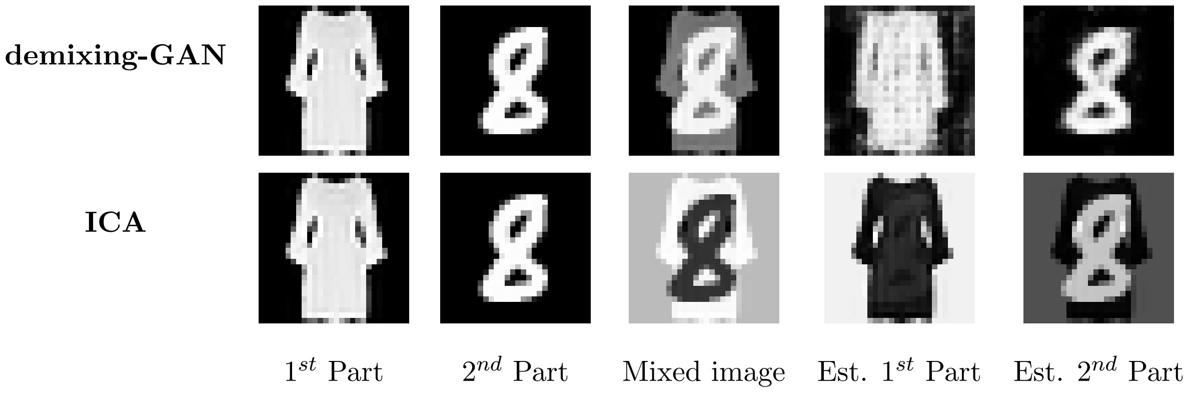


Figure 3.14: The performance of trained generators for demixing of two constituent components. The first two columns are the ground-truth components. The third column is the ground-truth mixed image and the last two columns denote the recovered components. The first row uses the generator trained through demixing-GAN. The second row shows the result of demixing with ICA method.

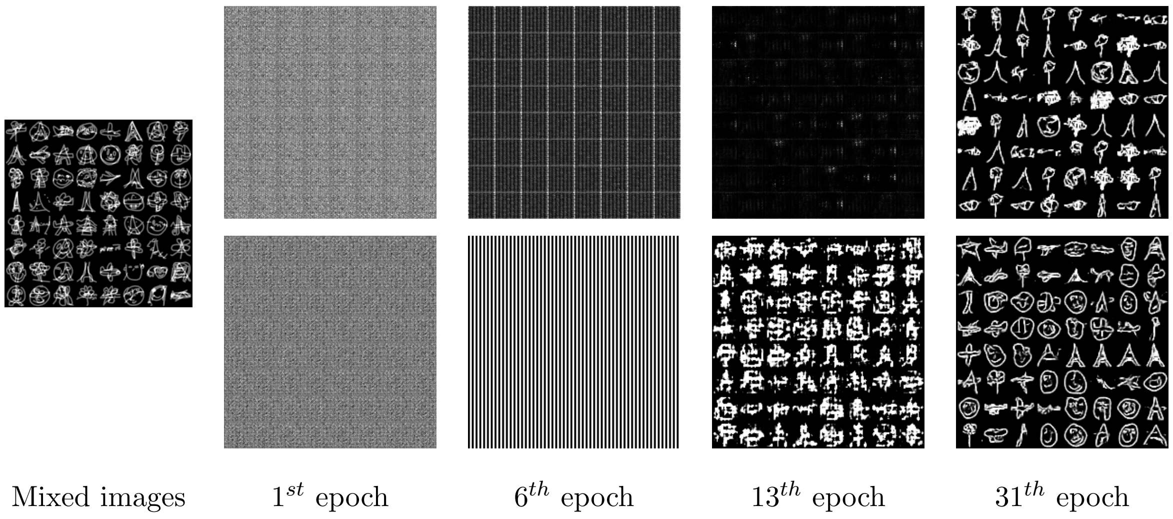


Figure 3.15: Evolution of output samples by two generators for fixed z_1 and z_2. The mixed images comprise two arbitrary objects drawn from 10 objects in training Quick-Draw dataset. Each generator outputs the samples from the distribution of all 5 objects.

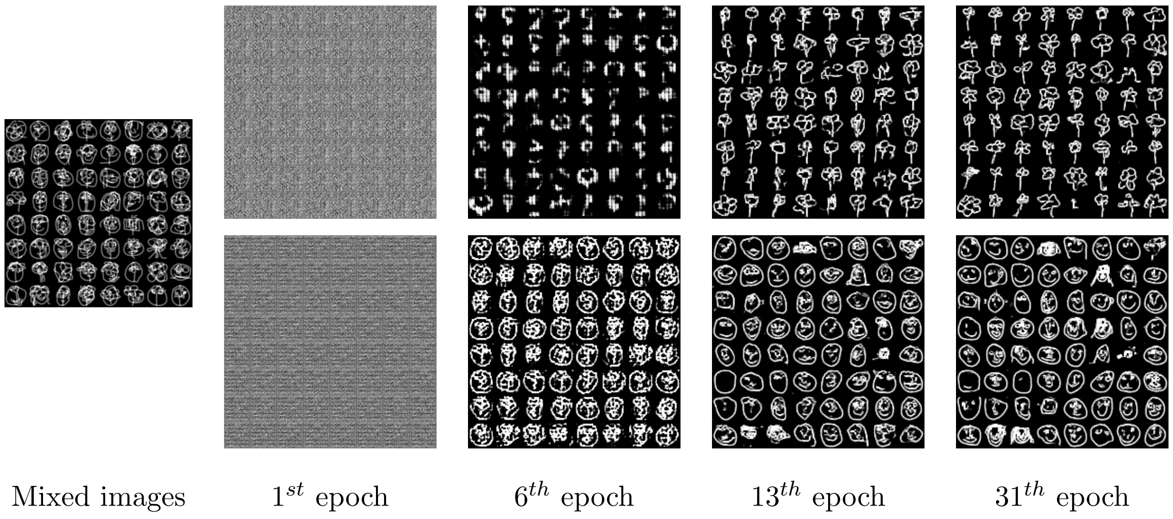


Figure 3.16: Evolution of output samples by two generators for fixed z_1 and z_2. The mixed images comprise only two objects, face, and flower in training Quick-Draw dataset. One generator produces the samples from dress distribution, while the other one outputs the samples from the bag distribution.

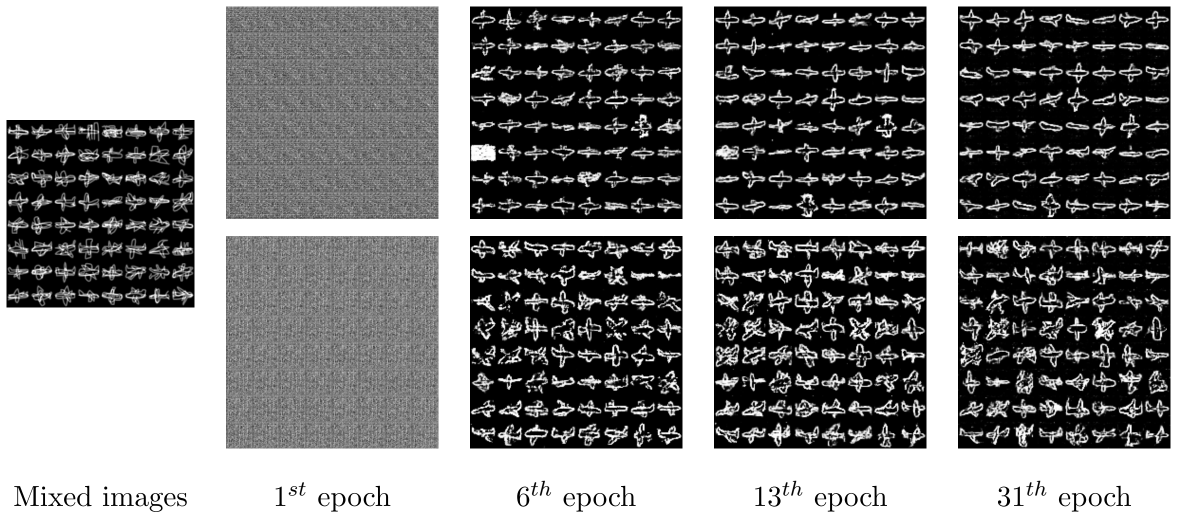


Figure 3.17: Evolution of output samples by two generators for fixed z_1 and z_2. The mixed images comprise only airplane object, randomly drawn from the training Quick-Draw dataset. The top generator produces mostly the samples from simpler and flat airplanes, while the bottom one outputs the samples from the more detailed airplane shapes.

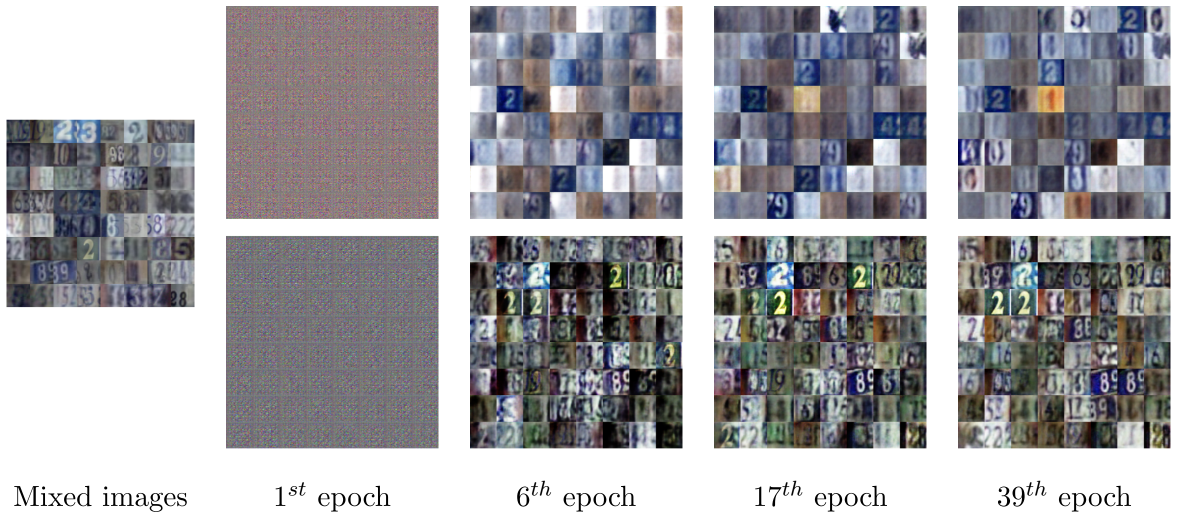


Figure 3.18: Evolution of output samples by two generators for fixed z_1 and z_2. The mixed images comprise two arbitrary digits drawn from 10 digits in SVHN training dataset. The top generator outputs the samples, representing the lightning condition of the digits, while the bottom one generates samples from the distribution of all digits.

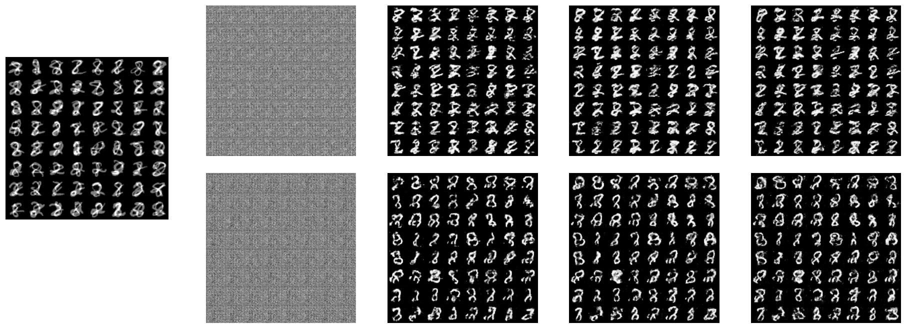


Figure 3.19: Failure of the demixing. Evolution of output samples by two generators for $z_1 = z_2$. The mixed images are the superposition of digits 2 and 8.

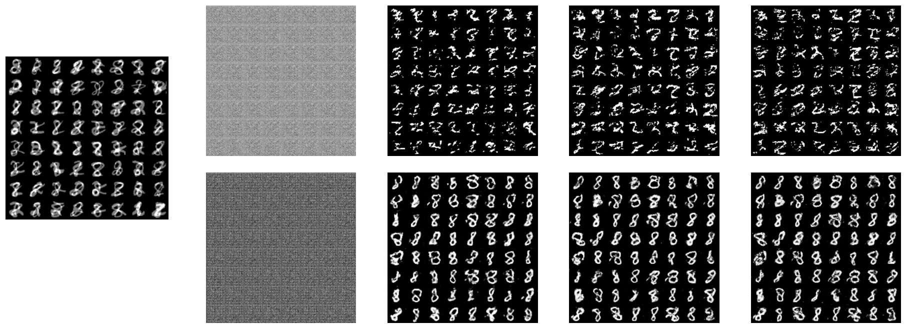


Figure 3.20: Failure of the demixing. Evolution of output samples by two generators for $z_1 = 0.1z_2$. The mixed images are the superposition of digits 2 and 8.

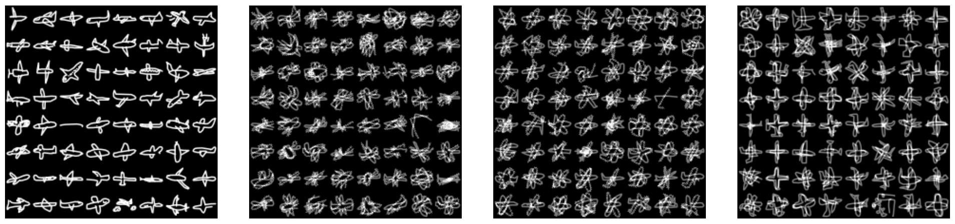


Figure 3.21: Mixed images of airplanes with different orientation, (a). Mixture of two 0° rotated images (b). Mixture of 0° and 30° rotated images rotated images (c). Mixture of 0° and 60° rotated images rotated images (d). Mixture of 0° and 90° rotated images rotated images

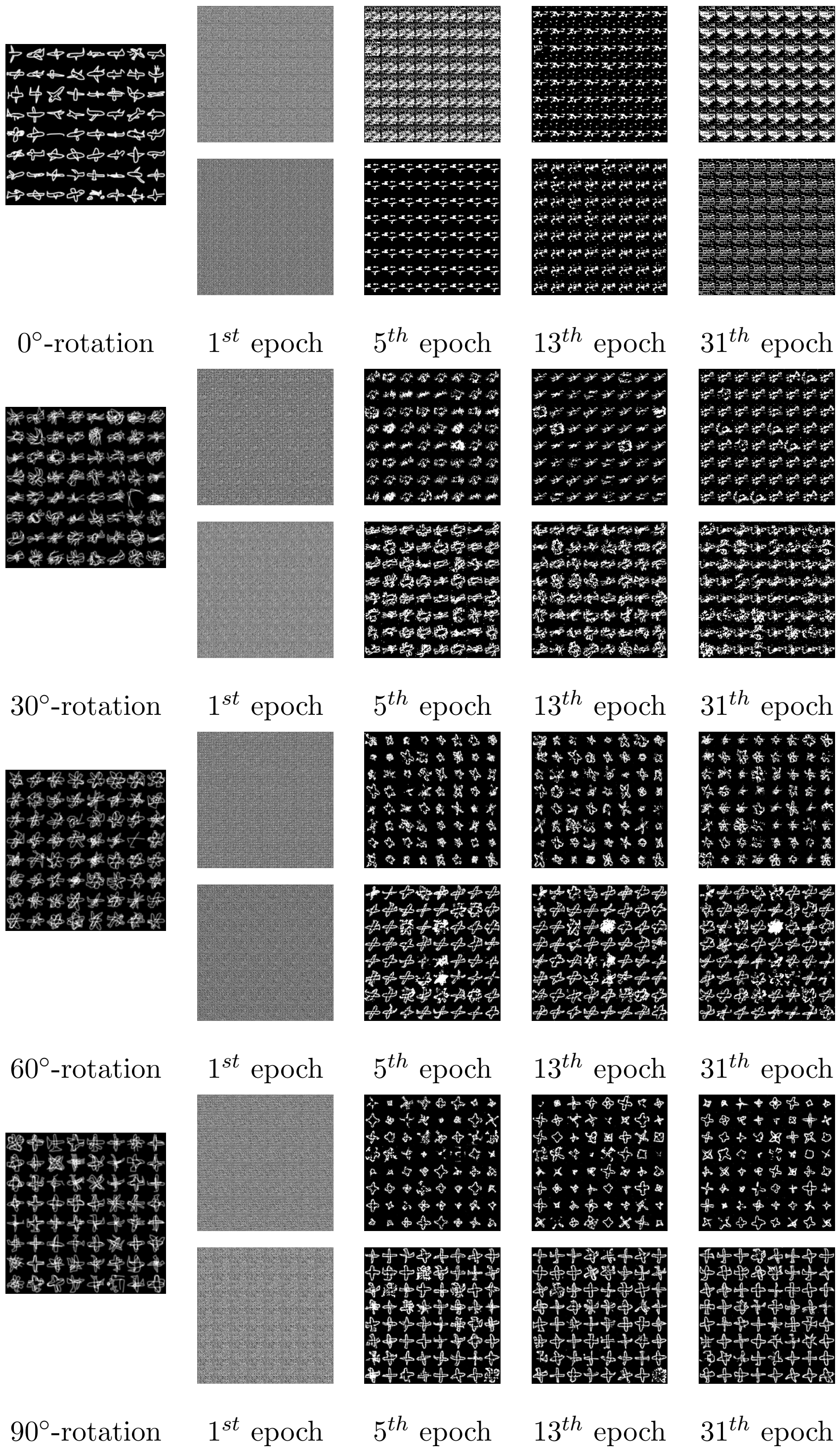


Figure 3.22: Failure of the demixing. Evolution of output samples by two generators. **Top:** Mixture of two 0° rotated images. **Second Top: Third Top:** Mixture of 0° and 30° rotated images rotated images. **Fourth Top:** Mixture of 0° and 60° rotated images rotated images. **Bottom:** Mixture of 0° and 90° rotated images rotated images.

CHAPTER 4. FAST LOW-RANK MATRIX ESTIMATION WITHOUT THE CONDITION NUMBER

In this chapter, we focus on another low-dimensional structure, low-rank matrices which is very common in many real scenarios. We specifically study the general problem of optimizing a convex function $F(L)$ over the set of $p \times p$ matrices, subject to rank constraints on L. However, existing first-order methods for solving such problems either are too slow to converge, or require multiple invocations of singular value decompositions. On the other hand, factorization-based non-convex algorithms, while being much faster, and has a provable guarantee, require stringent assumptions on the condition number of the optimum. In this chapter, we provide a novel algorithmic framework that achieves the best of both worlds: as fast as factorization methods, while requiring no dependency on the condition number. We instantiate our general framework for three important and practical applications; nonlinear affine rank minimization (NLARM), Logistic PC, and precision matrix estimation (PME) in probabilistic graphical model. We then derive explicit bounds on the sample complexity as well as the running time of our approach and show that it achieves the best possible bounds for both cases. We also provide an extensive range of experimental results for all of these applications to support our proposed algorithm.

4.1 Introduction

In this chapter, we consider the following optimization problem:

$$\min_{L} \quad F(L) \quad \text{s.t.} \quad \text{rank}(L) \leq r^*, \tag{4.1}$$

where $F(L) : \mathbb{R}^{p\times p} \to \mathbb{R}$ is a convex smooth function defined over matrices $L \in \mathbb{R}^{p\times p}$ with rank $r^* \ll p$.[1] This problem has recently received significant attention in machine learning, statis-

[1]For convenience, all our matrix variables will be of size $p \times p$, but our results extend seamlessly to rectangular matrices.

tics, and signal processing (Chen and Wainwright, 2015; Udell et al., 2016). Several applications abound, including affine rank minimization (Recht et al., 2010b; Tu et al., 2016; Jain et al., 2010), matrix completion (Candès and Recht, 2009), and collaborative filtering (Jain et al., 2013). Problem (5.4) also appears in the context of learning shallow polynomial neural networks (Livni et al., 2014; Soltani and Hegde, 2017c), and rigorous solutions to (5.4) sheds light on developing a non-asymptotic algorithmic understanding of training such networks. In most of the above applications, $F(L)$ is typically assumed to be a least-squares loss function. For instance, in machine learning, the squared loss between the pair of observed and predicted outputs would be a natural choice. But there are many cases in which other loss functions are used. For example, in neural network learning, the loss function is usually chosen according to the negative cross-entropy between the distribution of the fitting model and distribution of the training samples (Goodfellow et al., 2016). As another example, in graphical model, the goal is usually to estimate the covariance/precision matrix. In this case, Negative Log-Likelihood (NLL) function, defining on the distribution of output samples given the input ones is the common choice for $F(L)$. As an example in signal processing, one-bit matrix completion (Davenport et al., 2014) or related logistic PCA (Park et al., 2016a) problem are the ones either the observation model is a nonlinear of L^*, or $F(L)$ is modeled as the maximum likelihood function. In all these cases, we encounter with the mathematical optimization problem in the form of (5.4).

From the computational perspective, the traditional approach is to adopt first-order optimization for solving (5.4). Several different approaches (with theoretical guarantees) have been proposed in recent years. The first group of these methods are related to the convex methods in which the rank constraint is relaxed by the nuclear norm proxy (Fazel, 2002), resulting the overall convex problem which can be solved by off-the-shelf solvers. While these methods achieve the best sample complexity, i.e., the minimum required number of samples for achieving the small estimation error, they are computationally expensive and the overall running time can be slow if p is very large. To alleviate this issue, several non-convex methods have been proposed based on using non-convex regularizer instead of nuclear norm, factorizing the low-rank matrix, and singular value projection

(SVP). Non-convex regularizer methods (Quanming et al., 2017) can approximate the rank function better than nuclear norm, and have less computational complexity per iteration. The factorized methods (Chen and Wainwright, 2015; Bhojanapalli et al., 2016a; Tu et al., 2016) are computationally very appealing since they reduce the number of variables from p^2 to pr by writing L as $L = UV^T$ where $U, V \in \mathbb{R}^{p \times r}$ and $r \ll p$, and removing the rank constraint from problem (5.4). The singular value projection algorithms (Jain et al., 2010, 2014) use SVD as the projection step within the gradient descent framework in each iteration, and they are more robust to the spectral properties of the underlying matrix. However, all of these methods suffer from one or several of the following problems: their convergence rate is slow (they have sub-linear convergence), e.g. convex methods, or non-convex regularizer approaches; the computational cost per iteration is high, e.g., SVP-type algorithms; or they have stringent assumptions on the spectral properties (such as the condition number) of the solution to (5.4), e.g., factorized methods.

Our goal in this chapter is to propose an algorithm to alleviate the above problems simultaneously. **Specifically, we seek an algorithm that exhibits *linearly fast convergence*, *computationally efficient per iteration*, and at the same time, *robust to ill-conditioned problems.***

4.1.1 Our Contributions and Techniques

Here, we propose and analyze an algorithm for solving problems of the form (5.4) for objective functions F that satisfy the commonly-studied *Restricted Strongly Convex/Smooth* (RSC/RSS) conditions. We summarize our contribution in this chapter as follows:

Linear convergence. We propose a fast non-convex algorithm for solving of the optimization problem in (5.4). Specifically, we provide rigorous analysis to show that our proposed algorithm enjoy global linear convergence (no matter how it is initialized). Our algorithm enjoys fast per-iteration running time as well.

No spectral assumptions. We show that our proposed algorithm does not depend on stringent assumptions on the condition number (the ratio of the maximum to minimum nonzero singular value) of the solution to (5.4).

Provable guarantee on the optimal sample complexity regime Our algorithm is based on using the idea of approximate projection, and we show that its linear convergence is guaranteed in the optimal sample complexity regime as opposed to the previous work by (Becker et al., 2013).

No limitations on strong convexity/smoothness constants. In a departure from the majority of the matrix optimization literature, our algorithm succeeds under no particular assumptions on the *extent* to which the objective function F is strongly smooth/convex. (See below for details).

Instantiating in three applications. We instantiate our framework to three important and practical applications; Nonlinear Affine Rank Minimization (NLARM), Logistic PCA, and Precision Matrix Estimation (PME) in probabilistic graphical model. In addition, for NLARM, we show that by choosing an appropriate design operator $\mathcal{A}$ (defined later), we can reduce the computational complexity of calculating the gradient in each step significantly which makes the overall algorithm a promising approach for very large size nonlinear matrix sensing problem.

Putting together these ingredients, we get the first *condition-free*, almost-linear time algorithm for solving problems of the form (5.4).

Techniques. Our approach is an adaptation of the algorithm proposed in (Jain et al., 2014), which is a projected gradient-type algorithm. Specifically, they propose performing gradient descent, followed by thresholding the largest singular values of the matrix variable. The key idea of this work is that each gradient update is projected onto the space of matrices with rank r that is *larger* than r^*, the rank parameter in Problem (5.4). This trick can greatly alleviate situations where the objective function exhibits poor restricted strong convexity/smoothness properties; more generally, the overall algorithm can be applied to ill-posed problems. However, their algorithm requires performing a full exact singular value decomposition (SVD) after each gradient descent step.

This results in poor overall running time, as the per-iteration cost is *cubic* ($\mathcal{O}(p^3)$) in the matrix dimension.

Our method resolves this issue by replacing the exact SVD with a gap-independent *approximate* low-rank projection, while still retaining the idea of projecting onto a larger space. To establish soundness of our approach, we establish a property about (approximate) singular value thresholding that extends recent new results proved in (Shen and Li, 2016; Li et al., 2016). In particular, we prove a new structural result for approximate projection onto the space of rank-r matrices, showing that each projection step in our algorithm is *nearly non-expansive*, thus enjoying similar convergence guarantees as convex projected gradient descent. To be more precise, we know that for any matrix A and B, the best rank-r approximation of A satisfies the following:

$$\|H_r(A) - B\| \leq 2\|A - B\|,$$

where $H_r(A)$ returns the best rank approximation of A. This bound is very pessimistic and the upper bound is never achieved (Shen and Li, 2016; Becker et al., 2013). In this work, we prove a new structural result of the above hard-thresholding for approximate projection, achieving the small coefficient close to 1 if we use the projection onto a larger subspace. In particular, we prove:

$$\|\mathcal{T}(A) - B\|_F \leq \left(1 + \frac{2}{\sqrt{1-\epsilon}} \frac{\sqrt{r_1}}{\sqrt{r - r_1}}\right) \|A - B\|_F,$$

where $\text{rank}(B) = r_1$ and $\mathcal{T}$ implements the approximate projection onto the set of matrices with rank-r with ϵ accuracy. So by increasing r, we are recovering the non expansive property of the convex projection where the coefficient is exactly equals to 1.

Integrating the above result gives linear convergence of the proposed algorithm for a very broad class of objective functions $F(L)$. Since we use approximate low-rank projections, the running time of the projection step is (almost) linear in the size of the matrix if r^* is sub-linear in n.

4.1.2 Stylized Applications

We also instantiate our framework to three practical applications. First we consider a problem that we call it as *nonlinear affine rank minimization* (NLARM). Formally, we consider an observation model akin to the Generalized Linear Model (GLM) (Kakade et al., 2011):

$$y = g(\mathcal{A}(L^*)) + e,$$

where g denotes a nonlinear *link* function, $\mathcal{A}$ denotes a linear *measurement* (or observation) operator, which we formally define later, and $e \in \mathbb{R}^n$ denotes an additive noise vector. The goal is to reconstruct L^* from y, given that L^* is of rank at most r^*. For this application, we derive the sample complexity of our algorithm, calculate the running time, and analyze the statistical error rate. More specifically, we define an specific objective function tailored to g and verify that it is strongly convex/smooth; moreover, we show that $\widetilde{\mathcal{O}}(pr^*)$ samples is enough to estimate L^* up to the noise level, and this matches those of the best available methods. Our technique for deriving sample complexity is based the ϵ-net for the set of low-rank matrices, and designing sensing operator $\mathcal{A}$ based on Johnson-Lindenstrauss lemma. In addition, the running time to estimate L^* scales as $\widetilde{\mathcal{O}}(p^2r^*)$ which is nearly linear with the size of L^* and independent of all other spectral properties of L^* (such as its condition number). This marks a strict improvement over all other comparable existing methods.

Second we discuss about the Logistic PCA problem (Park et al., 2016a) in which we observe a binary matrix Y with entries belong to $\{0, 1\}$ such that the mean of each Y_{ij} is given by $P(Y_{ij} = 1|L_{ij}) = \sigma(L^*_{ij})$ where $\sigma(x) = \frac{1}{1+\exp(-x)}$. The goal is to estimate an underlying low-rank matrix L^* by trying to find the solution of following optimization problem:

$$F(L) = -\sum_{i,j} \big(Y_{ij} \log \sigma(L_{ij}) + (1 - Y_{ij}) \log(1 - \sigma(L_{ij}))\big).$$

Finally, we propose our third instantiation in which our goal is to estimate a precession matrix L^* based on the the samples $X_i \in \mathbb{R}^p$ for $i = 1, \dots, n$. In this setup, the objective function $F(L)$ is given by NLL of the distribution of output samples given the input ones. Our technique for verifying RSC/RSS conditions of NLL is according to a key observation which states $F(L)$ is globally strongly convex, and when restricted to any compact psd cone, it also satisfies strong smoothness condition. As a result of our analysis, we bound RSS/RSC constants of $F(L)$ which is a non-trivial task and considerably different from the the existing methods and consequently additional effort is required due to the several properties of the true precision matrix L^*. As a byproduct of this analysis, we show that with $n = \mathcal{O}(pr)$ independent samples, the proposed algorithm return an estimate up to constant error. Moreover, we show that the our algorithm provide the best empirical performance (in terms of estimation error) among all considered methods in the precession matrix estimation problem.

4.2 Prior Art

Optimization problems with rank constraints arise in several different applications; a few examples include robust PCA (Candès et al., 2011; Chandrasekaran et al., 2009; Netrapalli et al., 2014; Yi et al., 2016), covariance/precision matrix estimation using graphical models (Hsieh et al., 2014; Chandrasekaran et al., 2010), phase retrieval (Candes et al., 2015, 2013; Netrapalli et al., 2013), finding the square root of a PSD matrix (Jain et al., 2015), dimensionality reduction techniques (Johnson, 2014; Schein et al., 2003), video denoising (Ji et al., 2010), subspace clustering (Liu et al., 2013), face recognition (Yang et al., 2017) and many others. Beyond specific applications, solving (5.4) as efficiently as possible has attracted considerable interest in the optimization community. In general, most solution approaches can be categorized in four groups. In the first one, the non-convex rank constraint is relaxed into a *nuclear norm* penalty, which results in a convex problem and can be solved by off-the-shelf solvers such as SDP solvers (G. and B., 2014), singular value thresholding and its accelerated versions (Recht et al., 2010b; Cai et al., 2010; Goldstein et al., 2014), and active subspace selection methods (Hsieh and Olsen, 2014). While convex methods are well-known, their usage in the high dimensional regime is prohibitive (incurring cubic, or worse, running time).

The second group includes non-convex methods, replacing the rank constraint with a more tractable *non-convex* regularizer instead of the nuclear norm. To mention a few of them, smoothly clipped absolute deviation (SCAD) (Fan and Li, 2001), and iteratively re-weighted nuclear norm (IRNN) algorithm (Lu et al., 2016). While these approaches can reduce the computational cost per iteration, from p^3 to p^2r, they exhibit sub-linear convergence, and are quite slow in high dimensional regimes; see (Quanming et al., 2017) for details.

Methods in the third class try to solve the non-convex optimization problem (5.4) based on the factorization approach of (Burer and Monteiro, 2003). In these algorithms, the rank-r matrix L is factorized as $L = UV^T$, where $U, V \in \mathbb{R}^{p\times r}$. Using this idea removes the difficulties caused by the non-convex rank constraint; however, the objective function is not convex anymore. Nevertheless, under certain conditions, such methods succeed and have recently gained in popularity in the machine learning literature, and several papers have developed provable linear-convergence guarantees for both squared and non-squared loss functions (Tu et al., 2016; Bhojanapalli et al., 2016a; Park et al., 2016b; Chen and Wainwright, 2015; Zheng and Lafferty, 2015; Jain et al., 2013).

Such methods are currently among the fastest available. However, the major drawback is that they require a careful spectral initialization that usually involves one or multiple full singular value decomposition. More crucially, their convergence rate depends heavily on the condition number (i.e., the ratio of the largest to the smallest non-zero singular values) as well as other spectral properties of the optimum. As a consequence, if the problem is somehow poorly conditioned, their sample complexity and running time can blow up by a significant amount.

The last class of methods also includes non-convex methods. Unlike the factorized methods, they do not factorize the optimization variable, L, but instead use low-rank projections within classical gradient descent. This approach, also called singular value projection (SVP) or iterative hard thresholding, was introduced by (Jain et al., 2010) for matrix recovery from linear measurements, and was later modified for general M-estimation problems with well-behaved objective functions (Jain et al., 2014). These methods require multiple invocations of exact singular value decompositions (SVDs). While their computational complexity can be cubic in p, and consequently very slow in very large-scale problems, these methods do not depend on the condition number of optimum, and in this sense are more robust than factorized methods. A similar algorithm to SVP type algorithms was proposed by (Becker et al., 2013) for the squared loss case, which replaces the exact SVD with approximate one. However, their theoretical guarantees is very restrictive which overshadows the usage of any approximate SVD method instead of the exact one. That is, in the regime of optimal sample complexity, i.e., $n = \mathcal{O}(pr)$, their approximate projection should be applied onto a matrix with rank as the order of p in order to have convergence. This restriction overshadows the usage of any approximate SVD method instead of the exact one. Furthermore, we have to mention that while the idea of projecting on the larger set is theoretically backed up in (Jain et al., 2014), and also this chapter, the use of it within the factorized approach is just practically observed, and currently there is no theory for it (Bhojanapalli et al., 2016a).

In addition to the above algorithms, recently some stochastic gradient methods for low-rank matrix recovery have been proposed (Wang et al., 2017; Li et al., 2016). The goal of these methods are to reduce the cost of calculating the full gradient in each iteration which typically requires $\mathcal{O}(np^2)$ operations. For instance, (Wang et al., 2017) has combined the factorized method with SVRG algorithm (Johnson and Zhang, 2013), while the authors in (Li et al., 2016) have used the SVP algorithm along with SVRG or SAGA (Defazio et al., 2014) algorithms. However, these algorithms suffer from either heavy computational cost due to the initialization and projection step, or having stringent condition on the RSC/RSS conditions. Similar to the factorized method proposed in (Tu et al., 2016), the method in (Wang et al., 2017) requires some multiple SVDs for the initialization step, and its total running time depends the condition number of the ground truth matrix. In addition, to establish the linear convergence, one needs no limitations on the RSC/RSS conditions. On the other hand, the method in (Li et al., 2016) is robust to ill-condition problem and it uses the idea of projection on the set of matrices with larger rank than the true one. However, each iteration of it needs SVD and it may overshadow the benefit of it in alleviating the computation of the gradient.

Finally, we have to mention a non iterative algorithm for recovery of the low-rank matrices from a set of nonlinear measurements proposed by (Plan et al., 2017), While this approach does not need to know the nonlinearity of the link function, its recovery performance is limited, and we can only recover the the Frobenius/spectral norm of the solution of the optimization problem up to a scaling ambiguity. There are several other methods which we cannot mention al of then here. Please refer the recent survey (Davenport and Romberg, 2016) and references therein for a comprehensive discussion.

All the aforementioned algorithms suffer from one (or more) of the following issues: expensive computational complexity, slow convergence rate, and troublesome dependency on spectral properties of the optimum. In this chapter, we resolve these problems by a renewed analysis of approximate low-rank projection algorithms, and integrate this analysis to obtain a new algorithm for optimizing general convex loss functions with rank constraints.

4.3 Algorithm and Analysis

In this section, we propose our algorithm and provide the theoretical results to support it. Before that we introduce some notations and definitions.

4.3.1 Preliminaries

We denote the minimum and maximum eigenvalues of matrix $\bar{S}$ by S_p and S_1, respectively. We use $\|A\|_2$ and $\|A\|_F$ for spectral norm and Frobenius norm of a matrix A, respectively. We show the maximum and minimum eigenvalues of a matrix $A \in \mathbb{R}^{p\times p}$ as $\lambda_1(A), \lambda_p(A)$, respectively. In addition, for any subspace $W \subset \mathbb{R}^{p\times p}$, we denote $\mathcal{P}_W$ as the orthogonal projection operator onto it. Our analysis will rely on the following definition (Negahban et al., 2011; Jain et al., 2014):

Definition 4.1. *A function f satisfies the Restricted Strong Convexity (RSC) and Restricted Strongly Smoothness (RSS) conditions if for all $L_1, L_2 \in \mathbb{R}^{p\times p}$ such that rank$(L_1) \leq r$, rank$(L_2) \leq r$, we have:*

$$\frac{m_r}{2}\|L_2 - L_1\|_F^2 \leq f(L_2) - f(L_1) - \langle \nabla f(L_1), L_2 - L_1\rangle \leq \frac{M_r}{2}\|L_2 - L_1\|_F^2, \tag{4.2}$$

where m_r and M_r are called the RSC and RSS constants respectively.

Let $\mathbb{U}_r$ as the set of all rank-r matrix subspaces, i.e., subspaces of $\mathbb{R}^{p\times p}$ that are spanned by any r atoms of the form uv^T where $u, v \in \mathbb{R}^p$ are unit-norm vectors. We will exclusively focus on low-rank approximation algorithms that satisfy the following two properties:

Algorithm 4.1 MAPLE

```
Inputs: rank $r$, step size $\eta$, approximate tail projection $\mathcal{T}$
Outputs: Estimates $\widehat{L}$
Initialization: $L^0 \leftarrow 0$, $t \leftarrow 0$
while $t \leq T$ do
    $L^{t+1} = \mathcal{T}\left(L^t - \eta \nabla F(L^t)\right)$
    $t \leftarrow t + 1$
end while
Return: $\widehat{L} = L^T$
```

Definition 4.2 (Approximate tail projection)**.** *Let $\epsilon > 0$. Then, $\mathcal{T} : \mathbb{R}^{p\times p} \to \mathbb{U}_r$ is a approximate tail projection algorithm if for all $L \in \mathbb{R}^{p\times p}$, $\mathcal{T}$ returns a subspace $W = \mathcal{T}(L)$ that satisfies:*

$$\|L - \mathcal{P}_W L\|_F \leq (1+\epsilon)\|L - L_r\|_F,$$

where L_r is the optimal rank-r approximation of L in the Frobenius norm.

Definition 4.3 (Per-vector approximation guarantee)**.** *Let $A \in \mathbb{R}^{p\times p}$ and A_r denotes its best rank-r approximation. Suppose there is an algorithm that returns $B = ZZ^T A$ which is the projection of A onto the column space of matrix Z with orthonormal vectors $z_1, z_2, \ldots, z_r$, Then, this algorithm satisfies the per-vector approximation guarantee if*

$$|u_i^T AA^T u_i - z_i AA^T z_i| \leq \epsilon\sigma_{r+1}^2,$$

where $\epsilon > 0$ and u_i's are the eigenvectors of A.

Here, we focus on the randomized Block Krylov SVD (BKSVD) method for implementation of $\mathcal{T}$. This algorithm has been proposed by (Musco and Musco, 2015) which satisfies both of these properties with probability at least 99/100. However, one can alternately use a recent algorithm called LazySVD (Allen-Zhu and Li, 2016) with very similar properties. For constant approximation factors ϵ, the asymptotic running time of these algorithms is given by $\widetilde{\mathcal{O}}(p^2 r)$, *independent* of any spectral properties of the input matrix; however, BKSVD ensures slightly stronger per vector approximation guarantee.

As we discussed above, our goal is to solve the optimization problem (5.4). The traditional approach is to perform projected gradient descent:

$$L^{t+1} = P_r\left(L^t - \eta \nabla F(L^t)\right),$$

where P_r denotes an exact projection onto the space of rank-r matrices, and can be accomplished via SVD. However, for large p, this incurs cubic running time and can be very challenging. To alleviate this issue, one can instead attempt to replace the full SVD in each iteration with a tail-approximate low-rank projection; it is known that such projections can computed in $\mathcal{O}(p^2 \log p)$ time (Clarkson and Woodruff, 2017).

This is precisely our proposed algorithm, which we call *Matrix Approximation for Low-rank Estimation* (MAPLE), is described in pseudocode form as Algorithm 4.1. This algorithm is structurally very similar to (Jain et al., 2014; Becker et al., 2013). However, the proof of (Jain et al., 2014) requires exact low-rank projections, and (Becker et al., 2013) is specific to least-squares loss functions and with somewhat weak guarantees. In addition, very recent work of (Hegde et al., 2016), proposing approximate subspace-IHT algorithm which uses two step projection for general model of union subspaces, and is limited for squared loss function. Here we show that for the specific case of low-rank matrix recovery, one outer projection ($\mathcal{T}$ operator in Algorithm 4.1) is sufficient for estimating the solution of the optimization problem (5.4).

A key point is that our algorithm uses approximate low-rank projections with parameter r such that $r \geq r^*$. As we show in Theorem 4.5, the combination of using approximate projection, together with choosing a large enough rank parameter r, enables efficient solution of problems of the form (5.4) for *any (given) restricted convexity/smoothness constants* M, m. Specifically, this ability removes any upper bound assumptions on the ration $\frac{M}{m}$, which have appeared in several recent related works, such as (Bhojanapalli et al., 2016a). While the output matrix of MAPLE may have larger rank than r^*, one can easily post-process it with an final hard thresholding step in order to enforce the result to have exactly rank r^*.

In Algorithm 4.1, the choice of approximate low-rank projections is flexible, as long as the approximate tail and per-vector approximation guarantee are satisfied. We note that tail-approximate low-rank projection algorithms are widespread in the literature (Clarkson and Woodruff, 2013; Mahoney and Drineas, 2009; Rokhlin et al., 2009); however, per-vector approximation guarantee algorithms are less common. As will become clear in the proof of Theorem 4.5, the per-vector guarantee is crucial in our analysis.

In our implementation of MAPLE, we invoke the BKSVD method for low-rank approximation mentioned above[2]. Assuming BKSVD as the approximate low-rank projection of choice, we now prove a key structural result about the non-expansiveness of $\mathcal{T}$. This result, to the best of our knowledge, is novel and generalizes a recent result reported in (Shen and Li, 2016; Li et al., 2016). We defer the full proof of all theoretical results to the appendix.

Lemma 4.4. *For $r > (1+\frac{1}{1-\epsilon})r^*$ and for any matrices $L, L^* \in \mathbb{R}^{p\times p}$ with rank$(L^*) = r^*$, we have*

$$\|\mathcal{T}(L) - L^*\|_F^2 \leq \left(1 + \frac{2}{\sqrt{1-\epsilon}}\frac{\sqrt{r^*}}{\sqrt{r-r^*}}\right)\|L - L^*\|_F^2,$$

where $\mathcal{T} : \mathbb{R}^{p\times p} \to \mathbb{U}_r$ denotes the approximate tail projection defined in Definition 5.2 and $\epsilon > 0$ is the corresponding approximation ratio.

proof sketch. The proof follows the approach of (Li et al., 2016) where it is first given for sparse hard thresholding, and then is generalized to the low-rank case using Von Neumann's trace inequality. First define $\theta = [\sigma_1^2(L), \sigma_2^2(L)\dots,\sigma_r^2(L)]^T$. Also let $\theta^* = [\sigma_1^2(L^*), \sigma_2^2(L^*)\dots,\sigma_{r^*}^2(L^*)]^T$, and $\theta' = \mathcal{T}(\theta)$. Also, let *supp*$(\theta^*) = \mathcal{I}^*$, *supp*$(\theta) = \mathcal{I}$, *supp*$(\theta') = \mathcal{I}'$, and $\theta'' = \theta - \theta'$ with support I''. Now define new sets $\mathcal{I}^*\cap\mathcal{I}' = \mathcal{I}^{*1}$ and $\mathcal{I}^*\cap\mathcal{I}'' = \mathcal{I}^{*2}$ with restricted vectors to these sets as $\theta_{\mathcal{I}^{*1}} = \theta^{*1}$, $\theta_{\mathcal{I}^{*2}} = \theta^{*2}$, $\theta'_{\mathcal{I}^{*1}} = \theta^{1*}$, $\theta''_{\mathcal{I}^{*2}} = \theta^{2*}$ such that $|\mathcal{I}^{*2}| = r^{**}$, and $\theta_{\max} = \|\theta^{2*}\|_\infty$. The proof continues by upper bounding the ratio of $\frac{\|\theta'-\theta^*\|_2^2 - \|\theta-\theta^*\|_2^2}{\|\theta-\theta^*\|_2^2}$ in terms of r, r^*, r^{**} and by using the inequality $\widehat{\theta}_{\min} \geq (1-\epsilon)\theta_{\max}$ where $\widehat{\theta}$ denotes the vector of approximate eigenvalues returned back by $\mathcal{T}$. This inequality is resulted by invoking the per-vector guarantee property of $\mathcal{T}$. Now we can obtain the desired upper bound and get the final claim. □

[2]We note that since the BKSVD algorithm is randomized while the definitions of approximate tail projection and per-vector approximation guarantee are deterministic. Fortunately, the running time of BKSVD depends only logarithmically on the failure probability, and therefore an additional union bound argument is required to precisely prove algorithmic correctness of our method.

We now leverage the above lemma to provide our main theoretical result supporting the efficiency of MAPLE.

Theorem 4.5 (Linear convergence of MAPLE)**.** *Assume that the objective function $F(L)$ satisfies the RSC/RSS conditions with parameters M_{2r+r^*} and m_{2r+r^*}. Define $\nu = \sqrt{1 + \frac{2}{\sqrt{1-\epsilon}} \frac{\sqrt{r^*}}{\sqrt{r-r^*}}}$. Let J_t denotes the subspace formed by the span of the column spaces of the matrices L^t, L^{t+1}, and L^*, the solution of (5.4). In addition, assume that $r > \frac{C_1}{1-\epsilon}\left(\frac{M_{2r+r^*}}{m_{2r+r^*}}\right)^4 r^*$ for some $C_1 > 2$. Choose step size as η as $\frac{1-\sqrt{\alpha'}}{M_{2r+r^*}} \leq \eta \leq \frac{1+\sqrt{\alpha'}}{m_{2r+r^*}}$ where $\alpha' = \frac{\sqrt{\alpha-1}}{\sqrt{1-\epsilon}\sqrt{\alpha-1}+2}$ for some $\alpha = \Theta(r/r^*) > 1$. Then, MAPLE outputs a sequence of estimates L^t such that:*

$$\|L^{t+1} - L^*\|_F \leq \rho\|L^t - L^*\|_F + \nu\eta\|\mathcal{P}_{J_t}\nabla F(L^*)\|_F, \tag{4.3}$$

where $\rho = \nu\sqrt{1 + M_{2r+r^*}^2\eta^2 - 2m_{2r+r^*}\eta} < 1$.

In particular, Theorem 4.5 guarantees the linear convergence to L^* up to the gradient of L^*. We note that the contraction factor ρ is not affected by extent to which the objective function $F(L)$ is strongly smooth/convex. In other words, no matter how large the ratio $\frac{M}{m}$ is, its effect is balanced by ν through choosing large enough r. Also, the quality of the estimates in Theorem 4.5 is upper-bounded by the gradient term $\|\mathcal{P}_{J_t}\nabla F(L^*)\|_F$ in (4.3), within each iteration. In below, we instantiate the general optimization problem (5.4) to three problems of NLARM, logistic PCA, and PME. In NLARM and PME, L^* denotes the ground truth which we are looking for to estimate; as a result, the gradient term in (4.3) represents the statistical aspect of MAPLE. IN these problems, we give an upper bound on this term. Also, we show that the loss function $F(L)$ satisfies RSC/RSS conditions in these three instantiations, and consequently, derive the sample complexity and the running time of MAPLE.

4.3.2 Nonlinear Affine Rank Minimization (NLARM)

In this section, we formally state our first problem. Consider the nonlinear observation model $y = g(\mathcal{A}(L^*)) + e$, where $\mathcal{A}$ is a linear operator, $\mathcal{A} : \mathbb{R}^{p\times p} \to \mathbb{R}^n$ parametrized by n full rank matrices, $A_i \in \mathbb{R}^{p\times p}$ such that $(\mathcal{A}(L^*))_i = \langle A_i, L^*\rangle$ for $i = 1, \ldots, n$. Also, e denotes an additive

Table 4.1: Summary of our contributions, and comparison with existing methods for NLARM , κ denotes the condition number of L^*, and ϑ denotes the final optimization error. Also, SC and RT denote Sample Complexity and Running Time, respectively. Here we have presented for each algorithm the best known running time result.

Algorithm	SC	RT	Bounded $\frac{M}{m}$
Convex (Recht et al., 2010b)	$\widetilde{\mathcal{O}}(pr^*)$	$\mathcal{O}(\frac{p^3}{\sqrt{\vartheta}})$	Yes
Non-convex Reg (Quanming et al., 2017)	$\widetilde{\mathcal{O}}(pr^*)$	$\mathcal{O}(\frac{p^2 r^*}{\vartheta})$	Yes
Factorized (Bhojanapalli et al., 2016a)	$\widetilde{\mathcal{O}}(pr^*)$	$\mathcal{O}(p^2(r^* + \log p)\kappa^2 \log(\frac{1}{\vartheta}) + p^3)$	Yes
SVP (Jain et al., 2014)	$\widetilde{\mathcal{O}}(pr^*)$	$\mathcal{O}(p^3 \log(\frac{1}{\vartheta}))$	No
MAPLE	$\widetilde{\mathcal{O}}(\mathbf{pr}^*)$	$\mathcal{O}(\mathbf{p}^2\mathbf{r}^* \log \mathbf{p} \log(\frac{\mathbf{1}}{\vartheta}))$	No

subgaussian noise vector with i.i.d., zero-mean entries that is also assumed to be independent of $\mathcal{A}$ (see appendix for more details). If $g(x) = x$, we have the well-known matrix sensing problem for which a large number of algorithms have been proposed. The goal is to estimate the ground truth matrix $L^* \in \mathbb{R}^{p \times p}$ for more general nonlinear link functions.

In this chapter, we assume that link function $g(x)$ is a differentiable monotonic function, satisfying $0 < \mu_1 \leq g'(x) \leq \mu_2$ for all $x \in \mathcal{D}(g)$ (domain of g). This assumption is standard in statistical learning (Kakade et al., 2011) and in nonlinear sparse recovery (Negahban et al., 2011; Yang et al., 2015; Soltani and Hegde, 2017a). Also, as we will discuss below, this assumption will be helpful for verifying the RSC/RSS condition for the loss function that we define as follows. We estimate L^* by solving the optimization problem :

$$\begin{aligned} \min_{L} \quad & F(L) = \frac{1}{n}\sum_{i=1}^{n} \Omega(\langle A_i, L\rangle) - y_i\langle A_i, L\rangle \\ \text{s.t.} \quad & \operatorname{rank}(L) \leq r^*, \end{aligned} \tag{4.4}$$

where $\Omega : \mathbb{R} \to \mathbb{R}$ is chosen such that $\Omega'(x) = g(x)$. [3] Due assumption on the derivative of g, we see that $F(L)$ is a convex function (actually strongly convex), and can be considered as a special case of general problem in (5.4).

[3]The objective functioon $F(L)$ in (4.4) is standard; see (Soltani and Hegde, 2017a) for an in-depth discussion.

We assume that the design matrices A_i's are constructed as follows. Consider a partial Fourier or partial Hadamard matrix $X' \in \mathbb{R}^{n\times p^2}$ which is multiplied from the right by a diagonal matrix, D, whose diagonal entries are uniformly distributed over $\{-1,+1\}^{p^2}$. Call the resulting matrix $X = X'D$ where each row is denoted by $X_i^T \in \mathbb{R}^{p^2}$. If we reshape each of these rows as a matrix, we obtain "measurement" (or "design") matrices $A_i \in \mathbb{R}^{p\times p}$ for $i = 1,\ldots,m$. This particular choice of design matrices A_i's is because they support fast matrix-vector multiplication which takes $\mathcal{O}(p^2\log(p))$.

The following theorem gives the upper bound on the term, $\|\mathcal{P}_{J_t}\nabla F(L^*)\|_F$, that appears in Theorem 4.5. This can be viewed as a "statistical error" term, and is zero in the absence of noise.

Theorem 4.6. *Consider the observation model $y = g(\mathcal{A}L^*) + e$ described above. Let the number of samples scale as $n = \mathcal{O}(pr$ polylog $(p))$, then with high probability, for any given subspace $J \subset \mathbb{R}^{p\times p}$: we have for $t = 1,\ldots,T$:*

$$\|\mathcal{P}_J\nabla F(L^*)\|_F \leq \frac{1+\delta_{2r+r^*}}{\sqrt{n}}\|e\|_2, \tag{4.5}$$

where $0 < \delta_{2r+r^*} < 1$ denotes RIP constant of $\mathcal{A}$.

Corollary 4.7. *Consider all the assumptions and definitions stated in Theorem 4.5. If we initialize MAPLE with $L^0 = 0$, then after $T_{iter} = \mathcal{O}\left(\log\left(\frac{\|L^*\|_F}{\vartheta}\right)\right)$ iterations, we obtain:*

$$\|L^{T+1} - L^*\|_F \leq \vartheta + \frac{1}{\sqrt{n}}\frac{\nu\eta(1+\delta_{2r+r^*})}{1-\rho}\|e\|_2, \tag{4.6}$$

for some $\vartheta > 0$.

We now provide conditions under which the assumption of RSC/RSS in Theorem 4.5 are satisfied.

Theorem 4.8 (RSC/RSS conditions for MAPLE)**.** *Let the number of samples scaled as $n = \mathcal{O}(pr$ polylog $(p))$. Also, assume that $\frac{\mu_2^4(1+\omega)^4}{\mu_1^4(1-\omega)^4} \leq C_2(1-\epsilon)\frac{r}{r^*}$ for some $C_2, \omega > 0$ and $\epsilon > 0$ denotes the approximation ratio in algorithm 4.1. Then with high probability, the loss function $F(L)$ in (4.4) satisfies RSC/RSS conditions with constants $m_{2r+r^*} \geq \mu_1(1-\omega)$ and $M_{2r+r^*} \leq \mu_2(1+\omega)$ in each iteration.*

Sample complexity. By Corollary 4.7 and Theorem 4.8, the sample complexity of MAPLE algorithm is given by $n = \mathcal{O}(pr \text{ polylog } (p))$ in order to achieve a specified estimation error. This sample complexity is nearly as good as the optimal rate, $\mathcal{O}(pr)$. We note that the leading constant in the Oh- notation depends on ρ, η, RIP constant of the linear operator $\mathcal{A}$, and the magnitude of the additive noise. (Since we assume that this noise term is subgaussian, it is easy to show that $\|e\|_2$ scales as $\mathcal{O}(\sqrt{n})$ in expectation and with high probability).

Time complexity. Each iteration of MAPLE needs to compute the gradient, plus an approximate tail projection to produce a rank-r matrix. Computing the gradient involves one application of the linear operator $\mathcal{A}$ for calculating $\mathcal{A}(L)$, and one application of the adjoint operator, i.e., $\mathcal{A}^*(y - g(\mathcal{A}(L))$. Let T_{mult} and T'_{mult} denote the required time for these operations, respectively. On the other hand, approximate tail projection takes $\mathcal{O}\left(\frac{p^2 r \log p}{\sqrt{\varepsilon}}\right)$ operations for achieving the approximate ratio ϵ According to (Musco and Musco, 2015). Thanks to the linear convergence of MAPLE, the total number of iterations for achieving ϑ accuracy is given by $T_{iter} = \mathcal{O}\left(\log\left(\frac{\|L^*\|_F}{\vartheta}\right)\right)$. Let $\pi = \frac{M}{m}$; thus, the overal running time scales as $T = \mathcal{O}\left(\left(T_{mult} + T'_{mult} + \frac{p^2 r^* \pi^4 \log p}{\sqrt{\epsilon}}\right)\left(\log \frac{\|L^*\|_F}{\vartheta}\right)\right)$ by the choice of r according to Theorem 4.5. If we assume that the design matrices A_i's are implemented via a Fast Fourier Transform, computing $T_{mult} = T'_{mult}$ takes $\mathcal{O}(p^2 \log p)$ operations. As a result, $T = \mathcal{O}\left(\left(p^2 \log p + \frac{p^2 r^* \pi^4 \log p}{\sqrt{\epsilon}}\right)\left(\log \frac{\|L^*\|_F}{\vartheta}\right)\right)$.

In Table 4.1, for $g(x) = x$ and the linear operator $\mathcal{A}$ defined above, we summarize the sample complexity as well as (asymptotic) running time of several algorithms. In this table, we assume constant RSC/RSS ratio for all the algorithms. We find that all previous methods, while providing excellent sample complexity benefits, suffer from either cubic dependence on p, or inverse dependence on the estimation error ϑ, or quadratic dependence on the condition number κ of the optimum. In contrast, MAPLE enjoys (unconditional) $\widetilde{\mathcal{O}}(p^2 r^*)$ dependence, which is (nearly) linear in the size of the matrix for small enough r^*.

4.3.3 Logistic PCA

Principle component analysis (PCA) is a widely used statistical tool in various applications such as dimensionality reduction, denoting, and visualization, to name a few. While the regular PCA sometimes called linear PCA can be applied for any data type, its usage for binary or categorical observed data is not satisfactory, due to the fact that it tries to minimize a least square objective function. In other words, PCA is useful where the likelihood of underlying data is distributed as Gaussian. As a result, applying it to the binary case makes the result less interpretable (Jolliffe, 2002).

To alleviate this issue, one can assume that each row of the observed binary matrix (a sample data) follows the multivariate Bernoulli distribution such that its maximum variations can be captured by a low-dimensional manifold, and then use the logistic loss to find this low-dimensional representation of the observed data. This problem has been also studied in the context of collaborative filtering on binary data (Johnson, 2014), one-bit matrix completion (Davenport et al., 2014), and network sign prediction (Chiang et al., 2014).

Mathematically, consider an observed binary matrix $Y \in \mathbb{R}^{p\times p}$ with entries belong to set $\{0, 1\}$ such that the mean of each Y_{ij} is given by $p_{ij} = P(Y_{ij} = 1|L_{ij}) = \sigma(L^*_{ij})$ where $\sigma(z) = \frac{1}{1+\exp(-z)}$. The goal is to estimate an underlying low-rank matrix L^* such that $L^*_{ij} = \log(\frac{p_{ij}}{1-p_{ij}}) = \text{logit}(p_{ij})$ by minimizing the following regularized logistic loss:

$$\begin{aligned} \min_{L} \quad & F(L) = -\sum_{i,j} \big(Y_{ij}\log\sigma(L_{ij}) + (1-Y_{ij})\log(1-\sigma(L_{ij}))\big) + \lambda\|L\|_F^2 \\ \text{s.t.} \quad & \text{rank}(L) \le r^*, \end{aligned} \tag{4.7}$$

where $\lambda > 0$ is a tuning parameter. We note that the objective function in (4.7) without the regularizer term is only strongly smooth. By adding the Frobenius norm of the optimization variable, we make sure that it is also globally strongly convex (Hence, RSC/RSC conditions are automatically satisfied.). Here, we focus on finding the solution of (4.7), L^*. Hence, we do not have explicitly the notion of ground truth as previous application. For solving the optimization problem (4.7), different algorithms have been proposed in recent years which are either slow such as convex nuclear norm minimization, or they do not have theoretical guarantees. (Chiang et al., 2014; Johnson, 2014; Davenport et al., 2014). Very recently, a non-convex factorized algorithm which is supported by rigorous convergence analysis, and is comparable to our algorithm (MAPLE), proposed by (Park et al., 2016a). We will compare the performance of this algorithm with the proposed one in the experimental section. We also note that in this application, there is no notion of sample complexity as we observe all the entries of Y. In addition , running time of MAPLE for solving the above problem is given by $\widetilde{\mathcal{O}}(p^2r)$ as the dominating term is related to the projection step and gradient calculation takes $\mathcal{O}(p^2)$ time.

4.3.4 Precision Matrix Estimation (PME)

Gaussian graphical models are a popular tool for modeling the interaction of a collection of Gaussian random variables. In Gaussian graphical models, nodes represent random variables and edges model conditional (in)dependence among the variables (Wainwright and Jordan, 2008). Over the last decade, significant efforts have been directed towards algorithms for learning *sparse* graphical models.

Mathematically, let Σ^* denote the positive definite covariance matrix of p Gaussian random variables, and let $\Theta^* = (\Sigma^*)^{-1}$ be the corresponding precision matrix. Then, $\Theta^*_{ij} = 0$ implies that the i^{th} and j^{th} variables are conditionally independent given all other variables and the edge (i, j) does not exist in the underlying graph. The basic modeling assumption is that Θ^* is sparse, i.e., such graphs possess only a few edges. Such models have been fruitfully used in several applications including astrophysics (Padmanabhan et al., 2016), scene recognition (Souly and Shah, 2016), and genomic analysis (Yin and Li, 2013). Numerous algorithms for sparse graphical model learning – both statistically as well as computationally efficient – have been proposed in the machine learning literature (Friedman et al., 2008; Mazumder and Hastie, 2012; Banerjee et al., 2008; Hsieh et al., 2011). Unfortunately, sparsity is a simplistic first-order model and is not amenable to modeling more complex interactions. For instance, in certain scenarios, only some of the random variables are directly observed, and there could be relevant *latent* interactions to which we do not directly have access.

The existence of latent variables poses a significant challenge in graphical model learning since they can confound an otherwise sparse graphical model with a dense one. This scenario is illustrated in Figure 4.1. Here, nodes with solid circles denote the observed variables, and solid black edges are the "true" edges in the graphical model. One can see that the "true" graph is rather sparse. However, if there is even a single unobserved (hidden) variable denoted by the node with the broken red circle, then it will induce dense, apparent interactions between nodes that are otherwise disconnected; these are denoted by the dotted black lines. A flexible and elegant method to learn latent variables in graphical models was proposed by (Chandrasekaran et al., 2012). At its core, the method imposes a superposition structure in the observed precision matrix as the sum of *sparse* and *low-rank* matrices, i.e., $\Theta^* = S^* + L^*$. Here, Θ^*, S^*, L^* are $p \times p$ matrices where p is the number of variables. The matrix S^* specifies the conditional observed precision matrix given the latent variables, while L^* encodes the effect of marginalization over the latent variables. The rank of L^*, r^*, is equal to the number of latent variables and we assume that r is much smaller than p. The goal is to estimate precision matrix Θ^*. Here, we merely focus on the learning the low-rank part, and assume that the sparse part is a known prior[4].

To put formally the estimation of matrix Θ^* in our framework, suppose that we observe samples $x_1, x_2, \ldots, x_n \overset{i.i.d}{\sim} \mathcal{N}(0, \Sigma)$ where each $x_i \in \mathbb{R}^p$. Let $C = \frac{1}{n}\sum_{i=1}^n x_i x_i^T$ denote the sample covariance matrix, and $\Theta^* = (\Sigma^*)^{-1}$ denote the true precision matrix. Following the formulation of (Han et al., 2016), we want to solve the following minimization of NLL problem:

$$\begin{aligned} \min_{L} \quad & F(L) = -\log\ \det(\bar{S} + L) + \langle \bar{S} + L, C\rangle \\ \text{s.t.} \quad & \operatorname{rank}(L) \leq r^*,\ \ L \succeq 0. \end{aligned} \tag{4.8}$$

where $\Theta^* = \bar{S} + L^*$ such that $\bar{S}$ is a known positive diagonal matrix (in general, a positive definite matrix) imposed in the structure of precision matrix to make the above optimization problem

[4]For, instance, if the data obeys the spiked covariance model (Johnstone, 2001), the covariance matrix is expressed as the sum of a low-rank matrix and a diagonal matrix. Consequently, by the Woodbury matrix identity, the precision matrix is the sum of a diagonal matrix and a low-rank matrix; $\Theta^* = \bar{S} + L^*$. In addition, problem in (4.8) is similar to the latent variable in Gaussian graphical model proposed by (Chandrasekaran et al., 2010).

Table 4.2: Summary of our contributions, and comparison with existing methods. Here, $\gamma = \sqrt{\frac{\sigma_r}{\sigma_{r+1}}} - 1$ represents the spectral gap parameter in intermediate iterations. The overall running time of the ADMM approach is marked as poly(p) since the precise rate of convergence is unknown.

Algorithm	Running Time	Spectral dependency
SDP (Chandrasekaran et al., 2012)	poly(p)	Yes
ADMM(Ma et al., 2013)	poly(p)	Yes
QUICDIRTY(Yang and Ravikumar, 2013)	$\widetilde{\mathcal{O}}(p^3)$	Yes
SVP(Jain et al., 2014)	$\widetilde{\mathcal{O}}(p^3)$	No
Factorized(Bhojanapalli et al., 2016a)	$\widetilde{\mathcal{O}}(p^2 r/\gamma)$	Yes
MAPLE	$\widetilde{\mathcal{O}}(\mathbf{p^2 r})$	**No**

well-defined. We will exclusively function in the high-dimensional regime where $n \ll p^2$. As an instantiation of the general problem (5.4), our goal is to learn the low-rank matrix L^* with rank $r^* \ll p$, from samples x_i's. We provide a summary of the theoretical properties of our methods, and contrasts them with other existing methods for PME existing methods. Table 4.2 shows this comparison for PME methods.

While problem (4.8) has extra PSD cone constraint compared to the general problem (5.4), we can still use MAPLE. Please see Theorem 4.13.

To make a theoretical comparison with our proposed algorithm, we first consider the similar **exact projection approach** of (Li et al., 2016) (the approach of (Li et al., 2016) does not consider the PSD projection) as its analysis for establishing RSC/RSS is different from NLARM. In this setup, the algorithm starts with zero initialization and proceeds in each iteration as $L^{t+1} = \mathcal{P}_r^+\left(L^t - \eta'\nabla F(L^t)\right)$ where $\mathcal{P}_r^+(\cdot)$ for some $r > r^*$ denotes projection onto the space of rank-r matrices which is implemented through performing an exact eigenvalue decomposition (EVD) of the input and selecting the nonnegative eigenvalues and corresponding eigenvectors (Henrion and Malick, 2012). Please note that we may not impose a psd projection within every iteration. If an application requires a psd matrix as the output (i.e., if proper learning is desired), then we can simply post-process the final estimate $\widehat{L}$ by retaining the nonnegative eigenvalues (and

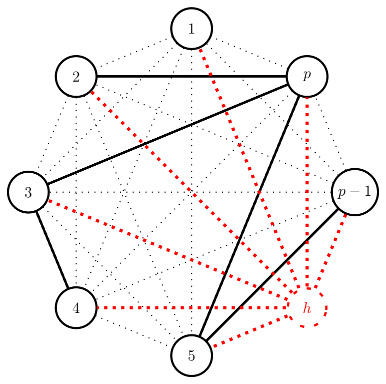


Figure 4.1: Illustration of effects of latent variable in graphical model learning. Solid edges represent "true" conditional dependence, while dotted edges represent apparent dependence due to the presence of the latent variable h.

corresponding eigenvectors) through an exact EVD. The following theorem shows an upper bound on the estimation error of the low-rank matrix at each iteration through exact projection.

Theorem 4.9 (Linear convergence with exact projection approach)**.** *Assume that the objective function $F(L)$ satisfies the RSC/RSS conditions with corresponding constants as M_{2r+r^*} and m_{2r+r^*}. Define $\nu' = \sqrt{1 + \frac{2\sqrt{r^*}}{\sqrt{r-r^*}}}$. Let J_t denotes the subspace formed by the span of the column spaces of the matrices L^t, L^{t+1}, and L^*. In addition, assume that $r > C_1' \left(\frac{M_{3r}}{m_{3r}}\right)^4 r^*$ for some $C_1' > 0$. Choose step size η' as $\frac{1-\sqrt{\beta'}}{M_{2r+r^*}} \leq \eta' \leq \frac{1+\sqrt{\beta'}}{m_{2r+r^*}}$ where $\beta' = \frac{\sqrt{\beta}-1}{\sqrt{\beta}-1+2}$ for some $\beta > 1$. Then, ESVP outputs a sequence of estimates L^t such that:*

$$\|L^{t+1} - L^*\|_F \leq \rho' \|L^t - L^*\|_F + \nu'\eta' \|\mathcal{P}_{J_t} \nabla F(L^*)\|_F, \tag{4.9}$$

where $\rho' = \nu' \sqrt{1 + M_{3r}^2 \eta'^2 - 2m_{3r}\eta'}$.

The quality of the estimates in Theorems 4.9 is upper-bounded by the gradient terms $\|\mathcal{P}_{J_t} \nabla F(L^*)\|_F$ in (4.9) within each iteration. The following theorem establishes these bounds:

Theorem 4.10. *Under the assumptions of Theorem 4.9, for any fixed t we have:*

$$\|\mathcal{P}_{J_t} \nabla F(L^*)\|_F \leq c_2 \sqrt{\frac{rp}{n}}, \tag{4.10}$$

with probability at least $1 - 2\exp(-p)$ *where* $c_2 > 0$ *are absolute constant.*

Next, we verify the RSS/RSC conditions of the objective function defined in (4.8), justifying the assumptions made in Theorem 4.9.

Theorem 4.11 (RSC/RSS conditions for exact projection approach)**.** *Let the number of samples scaled as* $n = \mathcal{O}\left(\frac{1}{\delta^2}\left(\frac{\eta'}{1-\rho'}\right)^2 rp\right)$ *for some small constant* $\delta > 0$ *and* ρ' *defined as Theorem 4.9. Also, assume that*

$$S_p \le S_1 \le C_2''(\frac{r}{r^*})^{\frac{1}{8}} S_p - \left(1+\sqrt{r^*}\right)\|L^*\|_2 - \delta.$$

Then, the loss function $F(L)$ *in* (4.8) *satisfies RSC/RSS conditions with constants* $m_{2r+r^*} \ge \frac{1}{(S_1+\left(1+\sqrt{r}\right)\|L^*\|_2+\delta)^2}$ *and* $M_{2r+r^*} \le \frac{1}{S_p^2}$ *that satisfy the assumptions of Theorem 4.9 in each iteration.*

The above theorem states that convergence of our method is guaranteed when the eigenvalues of $\bar{S}$ are roughly of the same magnitude, and large when compared to the spectral norm of L^*. We believe that this is merely a sufficient condition arising from our proof technique, and our numerical evidence shows that the algorithm succeeds for more general $\bar{S}$ and L^*.

Time complexity. Each iteration of exact projection appraoch needs a full EVD, which requires cubic running time (computing gradient needs only needs $\mathcal{O}(pr + r^3)$). Since the total number of iterations is logarithmic, the overall running time scales as $\widetilde{\mathcal{O}}(p^3)$.

Hence, we can use MAPLE (without posing psd constraint) to reduce the cubic time complexity of exact projection. Now we provide conditions under which the assumption of RSC/RSS in Theorem 4.5 are satisfied.

Theorem 4.12 (RSC/RSS conditions for MAPLE)**.** *Let* $n = \mathcal{O}\left(\frac{1}{\delta'^2}\left(\frac{\nu\eta}{1-\rho}\right)^2 rp\right)$ *for some small constant* $\delta' > 0$*, with* ρ *as defined in theorem 4.5. Also, assume the followings for some* $C_4, C_3'' > 0$*:*

$$\|L^*\|_2 \le \frac{1}{1+\sqrt{r^*}}\left(\frac{S_p}{1+C_4\left((1-\epsilon)(\frac{r^*}{r})\right)^{\frac{1}{8}}} - \frac{S_1(C_4((1-\epsilon)(\frac{r^*}{r}))^{\frac{1}{8}})}{1+C_4\left((1-\epsilon)(\frac{r^*}{r})\right)^{\frac{1}{8}}} - \frac{c_2\nu\eta}{1-\rho}\sqrt{\frac{rp}{n}}\right) \tag{4.11}$$

$$S_p \le S_1 \le \frac{C_3''}{(1-\epsilon)^{\frac{1}{8}}}(\frac{r}{r^*})^{\frac{1}{8}}(S_p - a') - \left(1+\sqrt{r^*}\right)\|L^*\|_2 - \delta', \tag{4.12}$$

where $0 < a' \leq \left(1+\sqrt{r^*}\right)\|L^*\|_2 + \delta'$ *for some* $\delta' > 0$*. Then, the loss function* $F(L)$ *in* (4.8) *satisfies RSC/RSS conditions with constants* $m_{2r+r^*} \geq \frac{1}{(S_1+(1+\sqrt{r})\|L^*\|_2+\delta')^2}$ *and* $M_{2r+r^*} \leq \frac{1}{(S_p-a')^2}$ *that satisfy the assumptions of Theorem 4.5 in each iteration.*

Theorem 4.12 specifies a family of true precision matrices $\Theta^* = \bar{S} + L^*$ that can be provably estimated using our approach with an optimal number of samples. Note that since we do not perform psd projection within MAPLE, it is possible that some of the eigenvalues of L^t are negative. Next, we show that with high probability, the absolute value of the minimum eigenvalue of L^t is small.

Theorem 4.13. *Under the assumptions in Theorem 4.12 on* L^**, using MAPLE to generate a rank* r *matrix* L^t *for all* $t = 1,\ldots,T$ *guarentees with high probability the minimum eigenvalue of* L^t *satisfies:* $\lambda_p(L^t) \geq -a'$ *where* $0 < a' \leq \left(1+\sqrt{r^*}\right)\|L^*\|_2 + \frac{c_2\nu\eta}{1-\rho}\sqrt{\frac{rp}{n}}$*.*

Time complexity. Each iteration of MAPLE needs a tail approximate projection on the set of rank r matrices. According to (Musco and Musco, 2015), these operations takes $k' = \mathcal{O}\left(\frac{p^2 r \log p}{\sqrt{\varepsilon}}\right)$ for error ε (computing gradient needs only needs $\mathcal{O}(pr + r^3)$). Since the total number of iterations is once again logarithmic, the overall running time scales as $\widetilde{\mathcal{O}}(p^2 r)$.

Sample complexity. Using the upper bounds in (4.10) and Theorems 4.11 and 4.12, the sample complexity of MAPLE, and also exact projection approach scales as $n = \mathcal{O}(pr)$ to achieve constant estimation error. This matches the number of degrees of freedom of a $p \times p$ matrix with rank r.

4.4 Experimental Results

We provide a range of numerical experiments supporting our proposed algorithm and comparing with existing approaches. For NLARM and logistic PCA frameworks, we compare our algorithms with factorized gradient descent (Bhojanapalli et al., 2016a) as well as projected gradient descent (i.e., the SVP algorithm of (Jain et al., 2014)). In our results below, FGD denotes factorized gradient descent algorithm, and SVD refers to SVP type algorithm where exact SVD has been used

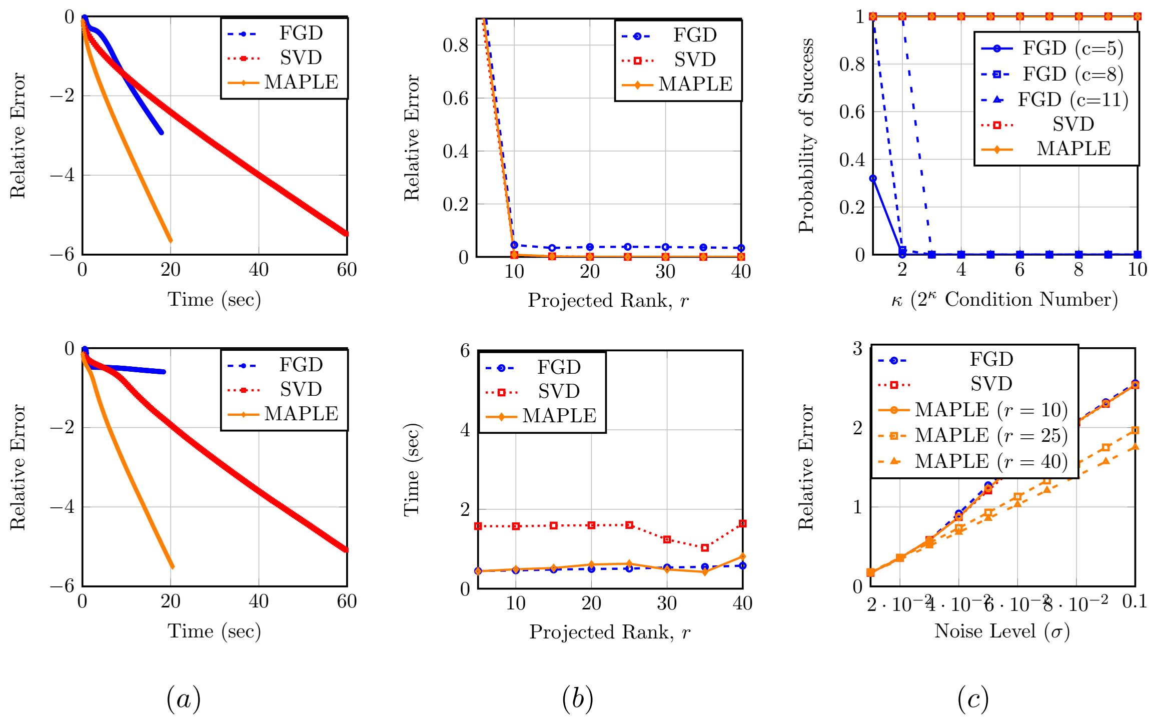


Figure 4.2: Comparisons of algorithms with $g(x) = 2x + sin(x)$. (a) Average of the relative error in estimating L^*. Parameters: $p = 1000$, $r^* = r = 50$, and $n = 4pr$. **Top:** $\kappa(L^*) = 1.1$. **Bottom:** $\kappa(L^*) = 20$. (b) Parameters: $p = 300$, $\kappa(L^*) = 1.1$, $r^* = 10$, and $n = 4pr$. **Top:** Average of the relative error. **Bottom:** Average running time. (c) **Top:** Probability of success. Parameters: $p = 300$, $r^* = r = 10$. (c) **Bottom:** Average of the relative error with different noise level. Parameters: $p = 300$, $\kappa = 2$, and $n = 7pr$.

for the projection step.[5] For PME application, our comparisons is with the regularized maximum likelihood approach of (Chandrasekaran et al., 2012), which we use CVX (G. and B., 2014), and with a modification of the ADMM-type method proposed by (Ma et al., 2013). We manually tuned step-sizes and regularization parameters in the different algorithms to achieve the best possible performance. First we provide the experiments for our fist instantiation, NLARM.

4.4.1 Nonlinear Affine Rank Minimization (NLARM)

Synthetic data. We report results for all algorithms in Figure 4.2. The link function is set to $g(x) = 2x + sin(x)$; this function satisfies the derivative conditions discussed above. We construct

[5]we have to mention that we have also used more well-known (but spectrum-dependent) Lanczos method for the projection step, and have obtained the same performance as SVD. Hence, we remove it from all the plots.

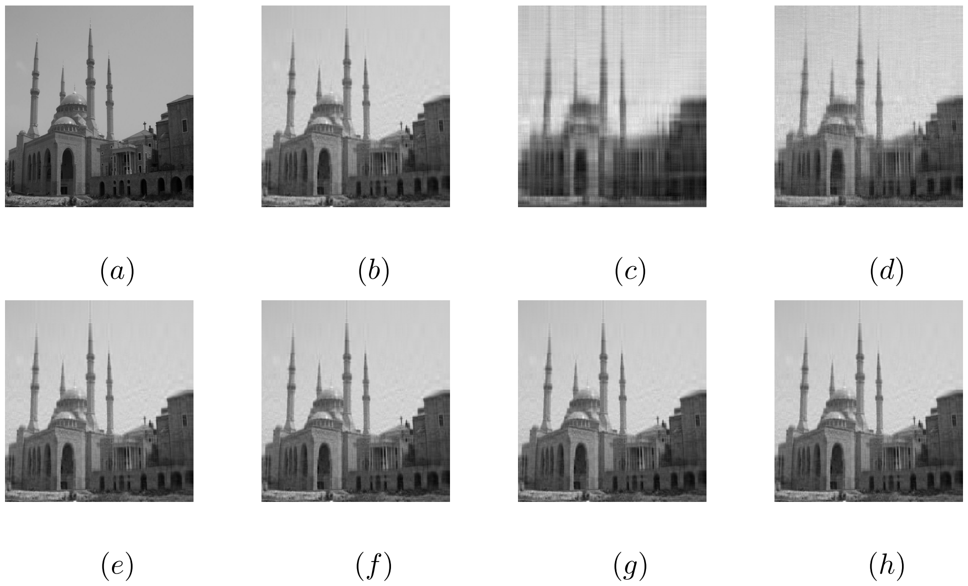


Figure 4.3: Comparison of algorithms for real 2D image with $g(x) = \frac{1-e^{-x}}{1+e^{-x}}$. (a) True 512×512 image. (b) Truncated true image with 30 top singular values. Reconstructed image using (c) FGD, (d) FGD with longer time, (e) SVD, (f) MAPLE ($r = 30$), (g) MAPLE ($r = 40$), (h) MAPLE ($r = 50$).

the ground truth low-rank matrix L^* with rank r^* by generating a random matrix $U \in \mathbb{R}^{p \times r^*}$ with entries drawn from the standard normal distribution. We ortho-normalize the columns of U, and set $L^* = UDU^T$ where $D \in \mathbb{R}^{r^* \times r^*}$ is a diagonal matrix with $D_{11} = \kappa(L^*)$, and $D_{jj} = 1$ for $j \neq 1$. After this, we apply a linear operator $\mathcal{A}$ on L^*, i.e., $\mathcal{A}(L^*)_i = \langle A_i, L^* \rangle$ where the choice of A_i has been discussed above. Finally, we obtain the measurements $y = g(\mathcal{A}(L^*))$. When reporting noise robustness, we add a Gaussian noise vector $e \in \mathbb{R}^m$ to $g(\mathcal{A}(L^*))$.

In Panel (a), the running time of the four algorithms are compared. For this experiment, we have chosen $p = 1000$, and the rank of the underlying matrix L^* to be 50. We also set the projected rank as $r = 50$. The number of measurements is set to $n = 4pr$. We consider a well-conditioned matrix L^* with $\kappa(L^*) = 1.1$ for top plot and $\kappa = 20$ for the bottom one. Then we measure the relative error in estimating of L^* in Frobenius norm in log scale versus the CPU time takes for 200 iterations for all of the algorithms. We run the algorithms for 15 Monte Carlo trials. As we can see,

Table 4.3: Numerical results for the real data experiment illustrating in Figure 4.3. T denotes the number of iterations.

Algorithm	Relative Error	Running Time	Projected Rank
FGD ($T = 300$)	0.0879	4.9816	30
FGD ($T = 1000$)	0.0602	15.9472	30
SVD ($T = 300$)	$4.4682e - 04$	19.4700	30
MAPLE ($T = 300$)	$9.7925e - 05$	4.2375	30
MAPLE ($T = 300$)	$9.9541e - 05$	5.5571	40
MAPLE ($T = 300$)	$1.3286e - 04$	7.1306	50

when κ is small, FGD has comparable running time with MAPLE (top plot); on the other hand, when we have ill-posed L^*, FGD takes much longer to achieve the same relative error.

Next, we show the performance of the algorithms when the projected rank is changed. The parameters are as $p = 300$, $\kappa(L^*) = 1.1$, $r^* = 10$, and $n = 4pr$. We set the number of Monte Carlo trials to 50. In the top plot in Panel (b), we have plotted the relative error as before versus the various r values by averaging over the trials. As we can see, projecting onto the larger space is an effective and practical strategy to achieve small relative error when we do not know the true rank. Furthermore, the bottom plot of Panel (b) shows the the average running time for either achieving relative error less than 10^{-4}, or 100 iterations versus the projected rank. These results suggest that both FGD and MAPLE have the comparable running when we increase the projected rank, while the other SVP algorithms have much longer running time.

Next, we consider the effect of increasing condition number of the underlying low-rank matrix L^* on the performance of the different algorithms. To do this, we set $p = 300$, and $r^* = r = 10$. The number of measurements is set to cpr where $c = 5, 8, 11$ for FGD and 5 for others. Then we run all the algorithms 50 times with different condition numbers ranging from $\kappa = 1$ (well-posed) to i.e., $\kappa = 1024$ (highly ill-posed). We define the probability of success as the number of times that the relative error is less than 0.001. As illustrated in the top plot of panel (c), all SVP-type algorithms are always able to estimate L^* even for large condition number, i.e., $\kappa = 1024$, whereas

FGD fails. In our opinion, this feature is a key benefit of MAPLE over the current fastest existing methods for low-rank estimation (based on factorization approaches).

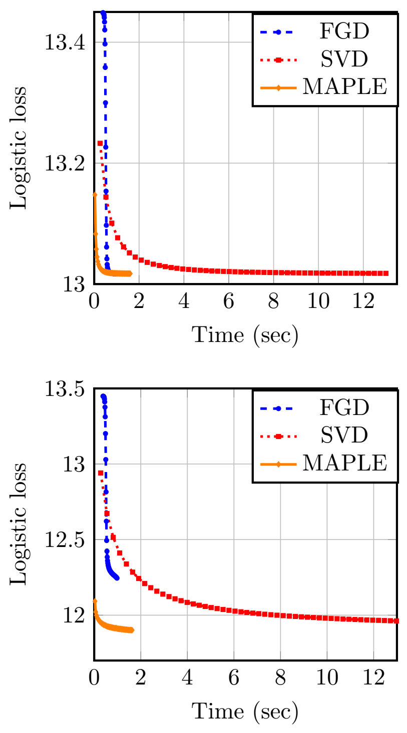
Finally, we consider the noisy scenario in which the observation y is corrupted by different Gaussian noise level. The parameters are set as $p = 300$, $r = 10, 25, 40$ for MAPLE and 10 for the others, $r^* = 10$, $n = 7pr$, and $\kappa = 2$. The bottom plot in Panel (c) shows the averaged over 50 trials of the relative error in L^* versus the various standard deviations. From this plot, we see that MAPLE with $r = 40$ is most robust, indicating that projection onto the larger subspace is beneficial when noise is present.

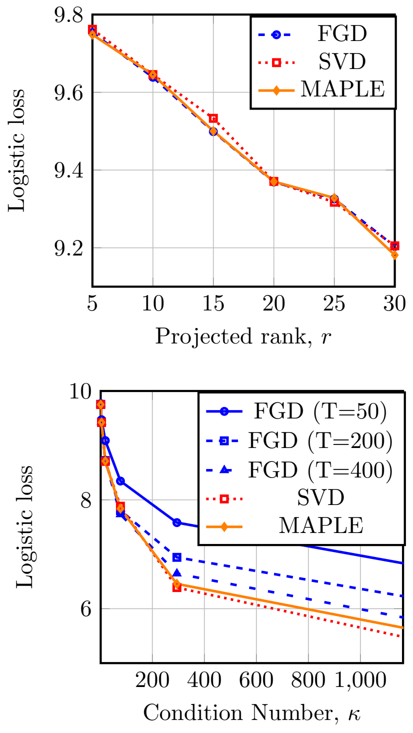
Real data. We also run MAPLE on a real 2D 512×512 image, assumed to be an approximately low-rank matrix. The choice of $\mathcal{A}$ is as before, but for the link function, we choose the sigmoid $g(x) = \frac{1-e^{-x}}{1+e^{-x}}$. Figure 4.3 visualizes the reconstructed image by different algorithms. In Figure 4.3, (a) is the true image and (b) is the same image truncated to its $r^* = 30$ largest singular values. The result of FGD is shown in (c) and (d) where for (d) we let algorithm run for many more iterations. Reconstruction by SVD is shown in (e). Finally, (f), (g), and (h) illustrate the reconstructed image by using MAPLE with various rank parameters. The numerical reconstruction error is given in Table 4.3. MAPLE is the fastest method among all methods, even when performing rank-r projection with r larger than r^*.

4.4.2 Logistic PCA

In this section, we provide some representative experimental results for our second application, logistic PCA. We report results for all algorithms in Figure 4.4. We construct the ground truth low-rank matrix L^* with rank r^* similar to NLARM case.

In panel (a), the running time of all algorithms are compared. For this experiment, we have chosen $p = 1000$, and the rank of the underlying matrix L^* to be 5. We also set the projected rank as $r = 5$. We consider a well-conditioned matrix L^* with $\kappa(L^*) = 1.1531$ for top plot and $\kappa = 21.4180$ for the bottom one. Then we measure the evolution of the logistic loss defined in (4.7) without any regularizer versus the CPU time takes for 50 iterations for all of the algorithms. We





(a) (b)

Figure 4.4: Comparisons of the algorithms for the average of the logarithm of the logistic loss. (a) Parameters: $p = 1000$, $r^* = r = 5$. **Top:** $\kappa(L^*) = 1.1699$. **Bottom:** $\kappa(L^*) = 21.4712$. (b) Parameter: $p = 200$. **Top:** Effect of extending the projected space. **Bottom:** Effect of increasing the condition number for $r^* = r = 5$. T denotes the number of iterations.

run the algorithms for 20 Monte Carlo trials, and illustrate the average result. As we can see, when κ is small, FGD has comparable running time with MAPLE (top plot); on the other hand, when we have ill-posed L^*, FGD takes longer to achieve the same performance.

In panel (b), top plot, we consider the effect of increasing the dimension of the projected space. In this experiment, we set $p = 200$, consider the well-posed case where $\kappa(L^*) = 1.4064$, and use 20 Monte Carlo trials. As we can see all the algorithm show the same trend which verifies that projecting onto the larger space is an effective and practical strategy to achieve small relative error when we do not know the true rank (This is expected according to the theory of MAPLE, while it is not theoretically justified by factorized method).

Finally, bottom plot in panel (b) shows the effect of increasing condition number of L^*. In this experiment, $p = 200$, $r^* = r = 5$, and the number of trials equals to 20. We first let all algorithms run for 50 iterations, and also consider FGD for more number of iterations, $T = 200$ and $T = 400$. As it is illustrated, both MAPLE and SVD algorithms are more robust to the large condition number than FGD with 50 number of iterations. But if we let FGD run longer, it shows the same performance as SVPs which again verifies the dependency of the running time of factorized method to the condition number.

4.4.3 Precision Matrix Estimation (PME)

We now represents some simulation results for our last application. First we start with synthetic data. **Synthetic data.** We use a diagonal matrix with positive values for the (known) sparse part, $\bar{S}$. For a given number of observed variables p, we set $r = 5\%$ as the number of latent variables. We then follow the method proposed in (Ma et al., 2013) for generating the sparse and low-rank components $\bar{S}$ and L^*. For simplicity, we impose the sparse component to be psd by forcing it to be positive diagonal matrix. All reported results on synthetic data are the average of 5 independent Monte-Carlo trials. Our observations comprise n samples, $x_1, x_2, \ldots, x_n \overset{i.i.d}{\sim} \mathcal{N}(0, (\bar{S} + L^*)^{-1})$. In our experiments, we used a full SVD as projection step for exact projection procedure (Due to numerical stability, we use SVD rather than EVD), FGD, ADMM method and convex approach based on nuclear norm minimization. We used CVX to run convex approach (it uses SDP method to solve nuclear norm minimization); alternatively, one can use other convex approaches which might be faster than convex.

Panel (a) and (b) in Figure 4.5 illustrate the comparison of algorithms for PME in terms of the relative error of the estimated L in Frobenius norm versus the "oversampling" ratio n/p. In this experiment, we fixed $p = 100$ in (a) and $p = 1000$ in (b) and vary n. In addition, for both of these results, condition number is given by $\kappa(L^*) = 2.4349$ and $\kappa(L^*) = 2.9666$, respectively. We observe that MAPLE, FGD, and exact procedure estimate the low-rank matrix even for the regime where n is very small, whereas both ADMM and CVX does not produce very meaningful results.

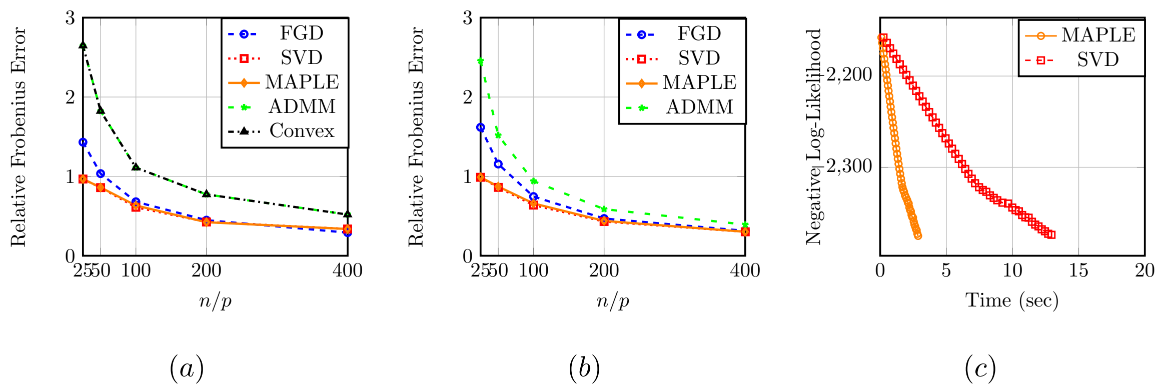


Figure 4.5: Comparison of algorithms both in synthetic and real data. (a) relative error of L in Frobenius norm with $p = 100$, and $r = r^* = 5$. (b) relative error of L in Frobenius norm with $p = 1000$, and $r = r^* = 50$. (c) NLL versus time in Rosetta data set with $p = 1000$.

We also report the results of several more experiments on synthetic data. In the first experiment, we set $p = 100$, $n = 400p$, and $r = r^* = 5$. Table 4.4 lists several metrics that we use for algorithm comparison. From Table 4.4, we see that MAPLE, FGD, and exact procedure produce better estimates of L compared to ADMM and convex method. As anticipated, the total running time of convex approach is much larger than other algorithms. Finally, the estimated objective function for first three algorithms is very close to the optimal (true) objective function compared to ADMM and CVX.

We increase the dimension to $p = 1000$ and reported the same metrics in Table 4.5 similar to Table 4.4. We did not report convex results as it takes long time to be completed. Again, we get the same conclusions as Table 4.4. Important point here is that in this specific application, FGD has better running time compared to MAPLE for both well-condition and ill-condition problem. Here, we did not report the running time for the ill-posed case; however, we observed that FGD is not affected by condition number of ground-truth. We conjecture that FGD delivers a solution for problem (4.8) such that its convergence is independent of the condition number of ground-truth similar to (Bhojanapalli et al., 2016b) where authors showed that for linear matrix sensing problem, there is no dependency on the condition number if they use FGD method. Proving of this

Table 4.4: Comparison of different algorithms for $p = 100$ and $n = 400p$. NLL stands for negative log-likelihood.

Alg	Estimated NLL	True NLL	Relative error	Total time
FGD	$-9.486278e+01$	$-9.485018e+01$	$2.914218e-01$	$2.150596e-02$
SVD	$9.485558e+01$	$-9.485018e+01$	$3.371867e-01$	$6.552529e-01$
MAPLE	$-9.485558e+01$	$-9.485018e+01$	$3.371742e-01$	$3.092728e-01$
ADMM	$-9.708976e+01$	$-9.485018e+01$	$5.192783e-01$	$1.475124e+00$
Convex	$-9.491779e+01$	$-9.485018e+01$	$5.192783e-01$	$7.482316e+02$

conjecture can be interesting future direction. Also, Tables 4.6 and 4.7 show the same experiment discussed in Tables 4.4 and 4.5, but for small number of samples, $n = 50p$.

Real data Here, we just evaluate our methods through the *Rosetta* gene expression data set (Hughes et al., 2000). This data set includes 301 samples with 6316 variables. We run the ADMM algorithm by (Ma et al., 2013) with $p = 1000$ variables which have highest variances, and obtained an estimate of the positive definite component $\bar{S}$. Then we used $\bar{S}$ as the input for MAPLE and exact projection procedure. The target rank for all three algorithms is set to be the same as that returned by ADMM. In Figure 4.5 plot (c), we illustrate the NLL for these algorithms versus wall-clock time (in seconds) over 50 iterations. We observe that all the algorithms demonstrate linear convergence, as predicted in the theory. Among the these algorithms, MAPLE obtains the quickest rate of decrease of the objective function.

Table 4.5: Comparison of different algorithms for $p = 1000$ and $n = 400p$.

Alg	Estimated NLL	True NLL	Relative error	Total time
FGD	$-2.638684e+03$	$-2.638559e+03$	$3.144617e-01$	$1.301985e+01$
SVD	$-2.638674e+03$	$-2.638559e+03$	$3.019913e-01$	$1.584453e+02$
MAPLE	$-2.638675e+03$	$-2.638559e+03$	$3.020130e-01$	$2.565310e+01$
ADMM	$-2.638920e+03$	$-2.638559e+03$	$3.921407e-01$	$3.375073e+02$

Table 4.6: Comparison of different algorithms for $p = 100$ and $n = 50p$. NLL stands for negative log-likelihood.

Alg	Estimated NLL	True NLL	Relative error	Total time
FGD	$-9.483037e+01$	$-9.470944e+01$	$1.034812e+00$	$2.294928e-02$
SVD	$-9.477855e+01$	$-9.470944e+01$	$8.586494e-01$	$1.026811e+00$
MAPLE	$-9.478611e+01$	$-9.470944e+01$	$8.606593e-01$	$4.854349e-01$
ADMM	$-9.356307e+01$	$-9.470944e+01$	$1.823421e+00$	$3.001534e+00$
Convex	$-9.528296e+01$	$-9.470944e+01$	$1.864212e+00$	$7.046295e+02$

Table 4.7: Comparison of different algorithms for $p = 1000$ and $n = 50p$.

Alg	Estimated NLL	True NLL	Relative error	Total time
FGD	$-2.639646e+03$	$-2.638491e+03$	$1.155856e+00$	$1.335701e+01$
SVD	$-2.638804e+03$	$-2.638491e+03$	$8.610451e-01$	$1.567543e+02$
MAPLE	$-2.638878e+03$	$-2.638491e+03$	$8.722342e-01$	$2.606750e+01$
ADMM	$-2.643757e+03$	$-2.638491e+03$	$1.517834e+00$	$4.019458e+02$

4.5 Conclusion

In this chapter, we focused on the low-rank matrix structure for our underlying signal. In particular, we formulated a general convex optimization problem subject to the rank-constraint. Then, we proposed an efficient algorithm which is as fas as the factorized based method and unconditional as the projection gradient algorithms. We evaluated the general framework for different problems, including Nonlinear Affine Rank Minimization, Logistic PCA, and Precision Matrix Estimation in probabilistic graphical models. We also provide statistical and computational analysis for these problems.

4.6 Appendix. Overview

We provide full proofs of all theorems discussed in this chapter.

$\mathcal{M}(\mathbb{U}_r)$ denotes the set of vectors associated with $\mathbb{U}_r$, the set of all rank-r matrix subspaces. We show the maximum and minimum eigenvalues of a matrix $A \in \mathbb{R}^{p\times p}$ as $\lambda_{\min}(A), \lambda_{\max}(A)$, respectively. Furthermore $\sigma_i(A)$ denotes the i^{th} largest singular value of matrix A. We need the following equivalent definitions of restricted strongly convex and restricted strong smoothness conditions.

Definition 4.14. *A function f satisfies the Restricted Strong Convexity (RSC) and Restricted Strong Smoothness (RSS) conditions if one of the following equivalent definitions is satisfied for all $L_1, L_2, L \in \mathbb{R}^{p\times p}$ such that rank$(L_1) \le r$, rank$(L_2) \le r$, rank$(L) \le r$:*

$$\frac{m_r}{2}\|L_2 - L_1\|_F^2 \le f(L_2) - f(L_1) - \langle \nabla f(L_1), L_2 - L_1\rangle \le \frac{M_r}{2}\|L_2 - L_1\|_F^2, \tag{4.13}$$

$$m_r\|L_2 - L_1\|_F^2 \le \langle \mathcal{P}_U\left(\nabla f(L_2) - \nabla f(L_1)\right), L_2 - L_1\rangle \le M_r\|L_2 - L_1\|_F^2, \tag{4.14}$$

$$m_r \le \|\mathcal{P}_U \nabla^2 f(L)\|_2 \le M_r, \tag{4.15}$$

$$m_r\|L_2 - L_1\|_F \le \|\mathcal{P}_U\left(\nabla f(L_2) - \nabla f(L_1)\right)\|_F \le M_r\|L_2 - L_1\|_F, \tag{4.16}$$

where U is the span of the union of column spaces of the matrices L_1 and L_2. Here, m_r and M_r are the RSC and RSS constants, respectively.

4.6.1 Appendix A. Proof of Theorems in Section 4.3.1

Before proving the main theorems, we restate the following hard-thresholding result from lemma 3.18 in (Li et al., 2016):

Lemma 4.15. *For $r > r^*$ and for any matrix $L \in \mathbb{R}^{p\times p}$, we have*

$$\|H_r(L) - L^*\|_F^2 \le \left(1 + \frac{2\sqrt{r^*}}{\sqrt{r - r^*}}\right)\|L - L^*\|_F^2, \tag{4.17}$$

where rank$(L^*) = r^*$, and $H_r(.) : \mathbb{R}^{p\times p} \to \mathbb{U}_r$ denotes the singular value thresholding operator, which keeps the largest r singular values and sets the others to zero.

For proving theorem 4.5, we cannot use directly lemma 4.15 since $\mathcal{T}$ operator returns an approximation of the top r singular vectors, and using exact projection in the proof of lemma 4.15 is necessary (Li et al., 2016). However, we can modify the proof of lemma 4.15 to make it applicable through the approximate projection approach. Hence, we can prove Lemma 5.12:

Proof of Lemma (5.12). The proof is similar to the procedure described in (Li et al., 2016) with some modification based on the per-vector guarantee property of approximate projection. In this work, the proof in is given first for sparse hard thresholding, and then is generalized to the low-rank case using Von Neumann's trace inequality, i.e., for two matrices $A, B \in \mathbb{R}^{p\times p}$ and corresponding singular values $\sigma_i(A)$ and $\sigma_i(B)$, respectively, we have:

$$\langle A, B\rangle = \Sigma_{k=1}^{\min\{\mathrm{rank}(A),\mathrm{rank}(B)\}}\sigma_k(A)\sigma_k(B). \tag{4.18}$$

First define $\theta = [\sigma_1^2(L), \sigma_2^2(L)\dots, \sigma_r^2(L)]^T$. Let $\theta^* = [\sigma_1^2(L^*), \sigma_2^2(L^*)\dots, \sigma_r^2(L^*)]^T$, and $\theta' = \mathcal{T}(\theta)$. Also, let $supp(\theta^*) = \mathcal{I}^*$, $supp(\theta) = \mathcal{I}$, $supp(\theta') = \mathcal{I}'$, and $\theta'' = \theta - \theta'$ with support I''. It follows that

$$\|\theta' - \theta^*\|_2^2 - \|\theta - \theta^*\|_2^2 \leq 2\langle\theta'', \theta^*\rangle - \|\theta''\|_2^2.$$

Now define new sets $\mathcal{I}^* \cap \mathcal{I}' = \mathcal{I}^{*1}$ and $\mathcal{I}^* \cap \mathcal{I}'' = \mathcal{I}^{*2}$ with restricted vectors to these sets as $\theta_{\mathcal{I}^{*1}} = \theta^{*1}$, $\theta_{\mathcal{I}^{*2}} = \theta^{*2}$, $\theta'_{\mathcal{I}^{*1}} = \theta^{1*}$, and $\theta''_{\mathcal{I}^{*2}} = \theta^{2*}$ such that $|\mathcal{I}^{*2}| = r^{**}$. Hence, $\|\theta^{2*}\|_2 = \beta\theta_{\max}$ where $\beta \in [\sqrt{r^{**}}]$ and $\theta_{\max} = \|\theta^{2*}\|_\infty$. By these definitions, we have:

$$\|\theta' - \theta^*\|_2^2 - \|\theta - \theta^*\|_2^2 \leq 2\|\theta^{2*}\|_2\|\theta^{*2}\|_2 - \|\theta^{2*}\|_2^2.$$

The proof continues to discuss in three cases as:

1. if $\|\theta^{2*}\|_2 \leq \theta_{\max}$, then $\beta = 1$.
2. if $\theta_{\max} \leq \|\theta^{2*}\|_2 < \sqrt{r^{**}}\theta_{\max}$, then $\beta = \frac{\|\theta^{2*}\|_2}{\theta_{\max}}$.
3. if $\|\theta^{2*}\|_2 \geq \sqrt{r^{**}}\theta_{\max}$, then $\beta = \sqrt{r^{**}}$.

In each case, the ratio of $\frac{\|\theta'-\theta^*\|_2^2 - \|\theta-\theta^*\|_2^2}{\|\theta-\theta^*\|_2^2}$ is upper bounded in terms of r, r^*, r^{**} and by using the inequality $|\theta_{\min}| \geq |\theta_{\max}|$ where $|\theta_{\min}|$ is defined as the smallest entry of θ^{1*}. This inequality holds due to the exact hard thresholding. However, it does not necessary hold when approximate projection is used. To resolve this problem, we note that in our framework the approximate tail projection is implemented via any randomized SVD method which supports the so-called per-vector guarantee. Recall from our discussion in section 4.3.1, the per vector guarantee condition means:

$$|u_i^T LL^T u_i - z_i LL^T z_i| \leq \epsilon\sigma_{r+1}^2 \leq \epsilon\sigma_i^2, \qquad i \in [r].$$

In our implementation, we use randomized block Krylov method (BK-SVD) which supports this condition. In our notations, this condition implies, $|\theta_{\min} - \widehat{\theta}_{\min}| \leq \epsilon\theta_{\min}$ where $\widehat{\theta} = [\widehat{\sigma}_1^2(L), \widehat{\sigma}_2^2(L) \ldots, \widehat{\sigma}_r^2(L)]^T$. By combing with $|\theta_{\min}| \geq |\theta_{\max}|$, we thus have $\widehat{\theta}_{\min} \geq (1-\epsilon)\theta_{\max}$. Now by this modification, we can continue the proof with the procedure described in (Li et al., 2016). Let $I := \frac{\|\theta' - \theta^*\|_2^2 - \|\theta - \theta^*\|_2^2}{\|\theta - \theta^*\|_2^2}$. For each case, we have::

- case 1: $I \leq \frac{\theta_{\max}^2}{(r - r^* + r^{**})(1-\epsilon)\theta_{\min}^2 - \theta_{\max}^2} \leq \frac{1}{(r-r^*+r^{**})(1-\epsilon)-1}$.
- case 2: $I \leq \frac{r^{**}\theta_{\max}^2}{(r-r^*+r^{**})(1-\epsilon)\theta_{\min}^2} \leq \frac{r^{**}}{(r-r^*+r^{**})(1-\epsilon)}$.
- case 3: $I \leq \frac{2\gamma\sqrt{r^{**}}\theta_{\max}^2 - r^{**}\theta_{\max}^2}{(r-r^*+r^{**})(1-\epsilon)\theta_{\min}^2 + r^{**}\theta_{\max}^2 + \gamma^2\theta_{\max}^2 - 2\gamma\sqrt{r^{**}}\theta_{\max}^2}$
  $\overset{e_1}{\leq} \frac{2\sqrt{r^{**}}}{2\sqrt{(r-r^*)(1-\epsilon)+r^{**}(\frac{5}{4}-\epsilon)} - \sqrt{r^{**}}}$, for some $\gamma \geq \sqrt{r^{**}}$.

In all the above cases, we have used the fact that $\widehat{\theta}_{\min} \geq (1-\epsilon)\theta_{\max}$. In addition, e_1 in case 3 holds due to maximizing the R.H.S. with respect to γ. After taking derivative w.r.t. γ, setting to zero, and solving the resulted quadratic equation, we obtain that:

$$\gamma = \max\left\{\sqrt{r^{**}}, \frac{\sqrt{r^{**}}}{2} + \sqrt{(r-r^*)(1-\epsilon) + r^{**}(\frac{5}{4} - \epsilon)}\right\}$$

Now if we plug in the value of γ in the R.H.S of case 3, we obtain the claimed bound. Putting the three above bounds all together, we have:

$$\frac{\|\theta' - \theta^*\|_2^2 - \|\theta - \theta^*\|_2^2}{\|\theta - \theta^*\|_2^2} \leq \max\left\{\frac{1}{(r - r^* + r^{**})(1-\epsilon) - 1}, \frac{r^{**}}{(r - r^* + r^{**})(1-\epsilon)}, \frac{2\sqrt{r^{**}}}{2\sqrt{(r-r^*)(1-\epsilon) + r^{**}(\frac{5}{4}-\epsilon)} - \sqrt{r^{**}}}\right\}$$

Hence,

$$\frac{\|\theta' - \theta^*\|_2^2 - \|\theta - \theta^*\|_2^2}{\|\theta - \theta^*\|_2^2} \overset{e_1}{\leq} \frac{2\sqrt{r^{**}}}{2\sqrt{(r - r^*)(1-\epsilon) + r^{**}(\frac{5}{4} - \epsilon)} - \sqrt{r^{**}}}$$
$$\overset{e_2}{\leq} \frac{2\sqrt{r^*}}{2\sqrt{(r - r^*)(1-\epsilon)} - \sqrt{r^*}}$$
$$\overset{e_3}{\leq} \frac{2}{\sqrt{1-\epsilon}} \frac{\sqrt{r^*}}{\sqrt{r - r^*}},$$

where e_1 follows by choosing r sufficiently large and the fact that ϵ can be chosen arbitrary small (this inceases the running time of the approximate projection by $\log(\frac{1}{\epsilon})$ factor), e_2 holds due to $r^{**} \leq r^*$, and finally e_3 holds by the assumption on r in the lemma. This completes the proof. □

Proof of Theorem 4.5. Let V^t, V^{t+1}, and V^* denote the bases for the column space of L^t, L^{t+1}, and L^*, respectively. Assume $\nu = \sqrt{1 + \frac{2}{\sqrt{1-\epsilon}} \frac{\sqrt{r^*}}{\sqrt{r-r^*}}}$. Also, by the definition of the tail projection, we have $L^t \in \mathcal{M}(\mathbb{U}_r)$, and by definition of set J in the theorem, $V^t \cup V^{t+1} \cup V^* \subseteq J_t := J$ such that $rank(J_t) \leq 2r + r^* \leq 3r$. Define $b = L^t - \eta \mathcal{P}_J \nabla F(L^t)$. We have:

$$\|L^{t+1} - L^*\|_F \overset{e_1}{\leq} \nu \|b - L^*\|_F$$
$$\leq \nu \|L^t - L^* - \eta \mathcal{P}_J \nabla F(L^t)\|_F$$
$$\overset{e_2}{\leq} \nu \|L^t - L^* - \eta \mathcal{P}_J \left(\nabla F(L^t) - \nabla F(L^*)\right)\|_F + \nu\eta \|\mathcal{P}_J \nabla F(L^*)\|_F$$
$$\overset{e_3}{\leq} \nu \sqrt{1 + M_{2r+r^*}^2 \eta^2 - 2m_{2r+r^*}\eta} \|L^t - L^*\|_F + \nu\eta \|\mathcal{P}_J \nabla F(L^*)\|_F \tag{4.19}$$

where e_1 holds due to applying lemma 4.15. Moreover, e_2 holds by applying triangle inequality and e_3 is obtained by combining the lower bound in (4.14) and upper bound in (4.16), i.e.,

$$\|L^t - L^* - \eta' \left(\nabla_J F(L^t) - \nabla_J F(L^*)\right)\|_2^2 \leq (1 + \eta'^2 M_{2r+r^*}^2 - 2\eta' m_{2r+r^*})\|L^t - L^*\|_2^2.$$

In order that (4.19) implies convergence, we require that

$$\rho = \left(\sqrt{1 + \frac{2}{\sqrt{1-\epsilon}} \frac{\sqrt{r^*}}{\sqrt{r - r^*}}}\right) \sqrt{1 + M_{2r+r^*}^2 \eta^2 - 2m_{2r+r^*}\eta} < 1$$

. By solving this quadratic inequality with respect to η, we obtain:

$$\left(\frac{M_{2r+r^*}}{m_{2r+r^*}}\right)^2 \leq 1 + \frac{\sqrt{r - r^*}\sqrt{1-\epsilon}}{2\sqrt{r^*}},$$

As a result, we obtain the the condition $r \geq \frac{C_1}{1-\epsilon}\left(\frac{M_{2r+r^*}}{m_{2r+r^*}}\right)^4 r^*$ for some $C_1 > 0$. Furthermore, since $r = \alpha r^*$ for some $\alpha > 1$, we conclude the condition on step size η as $\frac{1-\sqrt{\alpha'}}{M_{2r+r^*}} \leq \eta \leq \frac{1+\sqrt{\alpha'}}{m_{2r+r^*}}$ where $\alpha' = \frac{\sqrt{\alpha-1}}{\sqrt{1-\epsilon}\sqrt{\alpha-1}+2}$. This completes the proof of Theorem 4.5.

□

4.6.2 Appendix B. Proof of Theorems in Section 4.3.2

We first prove the statistical error rate, staing in Theorem 4.6.

proof of Theorem 4.6. Let $b_i = vec(A_i) \in \mathbb{R}^{p^2}$ denotes the i^{th} row of matrix $X \in \mathbb{R}^{m\times p^2}$, defining in the section 4.3.2 for $i = 1, \ldots, n$. Since X is constructed by uniform randomly chosen m rows of a $p^2 \times p^2$ DFT matrix multiplied by a diagonal matrix whose diagonal entries are uniformly distributed over $\{-1,+1\}^{P^2}$, $\frac{1}{\sqrt{n}}X$ satisfies the rank-r RIP condition with probability at least $1-\exp(-cn\varpi^2)$ ($c > 0$ is a constant) provided that $m = \mathcal{O}(\frac{1}{\varpi^2}pr\textbf{polylog}(\mathrm{p}))$ (Candes and Plan, 2011). On the other hand, if a matrix B satisfies the rank-r RIP condition, then (Lee and Bresler, 2010)

$$\left\|\mathcal{P}_U\frac{1}{\sqrt{n}}B^*a\right\|_2 \leq (1+\delta_r)\|a\|_2, \quad for\ all\ a \in \mathbb{R}^n, \tag{4.20}$$

where U denotes the set of rank-r matrices, and δ_r is the RIP constant. As a result, for all $t = 1, \ldots, T$ we have:

$$\left\|\frac{1}{n}\mathcal{P}_{J_t}\nabla F(L^*)\right\|_F = \left\|\frac{1}{n}\mathcal{A}^*e\right\|_F = \frac{1}{\sqrt{n}}\left\|\mathcal{P}_{J_t}\frac{1}{\sqrt{n}}X^*e\right\|_2 \leq \frac{1+\delta_{2r+r^*}}{\sqrt{n}}\|e\|_2,$$

where the last inequality holds due to (4.20) ($\frac{1}{\sqrt{n}}X$ has RIP constant δ_r, and from our definition, $rank(J_t) \leq 2r + r^*$), and the fact that $e \in \mathbb{R}^n$. □

proof of corollary 4.7. Consider upper bound in (4.19). By using induction, zero initialization, and Theorem 4.6, we obtain ϑ accuracy after $T_{iter} = \mathcal{O}\left(\log\left(\frac{\|L^*\|_F}{\vartheta}\right)\right)$ iterations. In other words, after T_{iter} iterations, we obtain:

$$\|L^{T+1} - L^*\|_F \leq \vartheta + \frac{1}{\sqrt{n}}\frac{\nu\eta(1+\delta_{2r+r^*})}{1-\rho}\|e\|_2.$$

□

The above results shows the linear convergence of APRM if there is no additive noise. We now prove that the objective function defined in problem (4.4) satisfies the RSC/RSS conditions in each iteration.

proof of Theorem 4.8. Let $L = L^t$ for all $t = 1, \dots, T$. We follow the approach in (Soltani and Hegde, 2017a). hence, we use the the Hessian based definition of RSC/RSC, stating in equation (4.15) in definition 4.14. We note that the Hessian of $F(L)$ is given by:

$$\nabla^2 F(L) = \frac{1}{n}\sum_{i=1}^{n} A_i g'(\langle A_i, L\rangle) A_i^T,$$

According to our assumption on the link function, we know $0 < \mu_1 \leq g'(x) \leq \mu_2$ for all $x \in \mathcal{D}(g)$. As a result $\lambda_{\min}(\nabla^2 F(L)) \geq 0$ due to the positive semidefinite of $A_i A_i^T$ for all $i = 1, \dots, n$. Now let $\Lambda_{\max} = \max_U \lambda_{\max}(\mathcal{P}_U \nabla^2 F(L))$ and $\Lambda_{\min} = \min_U \lambda_{\min}(\mathcal{P}_U \nabla^2 F(L))$. Moreover, let W be any set of rank-$2r$ matrices such that $U \subseteq W$. We have:

$$\mu_1 \min_W \lambda_{\min}\left(\mathcal{P}_W\left(\frac{1}{n}\sum_{i=1}^{n} A_i A_i^T\right)\right) \leq \Lambda_{\min}$$
$$\leq \Lambda_{\max} \leq \mu_2 \max_W \lambda_{\max}\left(\mathcal{P}_W\left(\frac{1}{n}\sum_{i=1}^{n} A_i A_i^T\right)\right), \tag{4.21}$$

Now, we need to bound the upper bound and the lower bound in the above inequality. To do this, we are using the assumption on the design matrices A_i's, stating in the theorem. According to this, we can write, $\mathcal{P}_W\left(\frac{1}{n}\sum_{i=1}^{n} A_i A_i^T\right) = \mathcal{P}_W\left(\frac{1}{n}X^T X\right)$. We follow the approach of (Hegde et al., 2016). Now fix any set W as defined above. Recall that $X = X'D$, where X' is a partial Fourier or partial Hadamard matrix. Thus, by (Haviv and Regev, 2017), X' satisfies RIP condition with constant 4υ over the set of of s-sparse vectors with high probability when $m = \mathcal{O}\left(\frac{1}{\upsilon^2} s \log^2(\frac{s}{\upsilon}) \log(p)\right)$. Also from (Krahmer and Ward, 2011), X is a $(1 \pm \xi)-$Johnson-Lindenstrauss embedding (with $4\upsilon < \xi$) for set W with probability at least $1 - \varsigma$ provided that $s > \mathcal{O}(\frac{V}{\varsigma})$, where V is the number of vectors in W. In other words, the Euclidean distance between any two vectors (matrix) $\beta_1, \beta_2 \in W \in \mathbb{R}^{p\times p}$ is preserved up to a $\pm\xi$ by application of X. As a result, with high probability

$$1 - \xi \leq \lambda_{\min}\left(\mathcal{P}_W\left(\frac{1}{n}\sum_{i=1}^{n} A_i A_i^T\right)\right) \leq \lambda_{\max}\left(\mathcal{P}_W\left(\frac{1}{n}\sum_{i=1}^{n} A_i A_i^T\right)\right) \leq 1 + \xi.$$

Now it remains to argue the final bound in (4.21). By (Candes and Plan, 2011), we know that the set of $p \times p$ rank-r matrices can be discretized by a ζ-cover S_r such that $|S_r| = (\frac{9}{\zeta})^{(2p+1)r}$. In addition, They show that if a matrix X satisfies JL embedding by constant ξ, then X satisfies the rank-r RIP with constant $\omega = \mathcal{O}(\xi)$. As a result, by taking union bound (taking maximum over all set W in (4.21)), we establish RSC/RSS constants such that $M_{2r+r^*} \leq \mu_2(1+\omega)$ and $m_{2r+r^*} \geq \mu_1(1-\omega)$ provided that $s = \mathcal{O}(pr)$ and $V = |S_r|$ which implies $m = \mathcal{O}(pr\textbf{polylog}(\text{p}))$. Now, In order to satisfy the assumptions in Theorem 4.5, we need to have $\frac{M^2_{2r+r^*}}{m^4_{2r+r^*}} \leq C_2(1-\epsilon)\frac{r}{r^*}$ for some $C_2 > 0$ and ϵ defined in lemma 5.12. Thus, we have $\frac{\mu_2^4(1+\omega)^4}{\mu_1^4(1-\omega)^4} \leq C_2(1-\epsilon)\frac{r}{r^*}$ which justifies the assumption in Theorem 4.8. □

4.6.3 Appendix C. Proof of Theorems in Section 4.3.4

Proof of Theorem 4.9. Let V^t, V^{t+1}, and V^* denote the bases for the column space of L^t, L^{t+1}, and L^*, respectively. Assume $\nu' = \sqrt{1+\frac{2\sqrt{r^*}}{\sqrt{r-r^*}}}$. By definition of set J in the theorem, $V^t \cup V^{t+1} \cup V^* \subseteq J_t := J$ and $\text{rank}(J_t) \leq 2r + r^*$. Define $b = L^t - \eta' \mathcal{P}_J \nabla F(L^t)$. We have:

$$\begin{aligned}
\|L^{t+1} - L^*\|_F &\overset{e_1}{\leq} \nu' \|b - L^*\|_F \\
&\leq \nu \|L^t - L^* - \eta' \mathcal{P}_J \nabla F(L^t)\|_F \\
&\overset{e_2}{\leq} \nu' \|L^t - L^* - \eta' \mathcal{P}_J \left(\nabla F(L^t) - \nabla F(L^*)\right)\|_F + \nu'\eta' \|\mathcal{P}_J \nabla F(L^*)\|_F \\
&\overset{e_3}{\leq} \nu' \sqrt{1 + M^2_{2r+r^*}\eta'^2 - 2m_{2r+r^*}\eta'} \|L^t - L^*\|_F + \nu'\eta' \|\mathcal{P}_J \nabla F(L^*)\|_F, \qquad (4.22)
\end{aligned}$$

where e_1 holds due to applying lemma 4.15. Moreover, e_2 holds by applying triangle inequality and e_3 is obtained by combining the lower bound in (4.14) and upper bound in (4.16), i.e.,

$$\|L^t - L^* - \eta' \left(\nabla_J F(L^t) - \nabla_J F(L^*)\right)\|_2^2 \leq (1 + {\eta'}^2 M^2_{2r+r^*} - 2\eta' m_{2r+r^*})\|L^t - L^*\|_2^2.$$

In order that (4.22) implies convergence, we require that

$$\rho' = \sqrt{1 + 2\frac{\sqrt{r^*}}{\sqrt{r-r^*}}} \sqrt{1 + M^2_{2r+r^*}\eta'^2 - 2m_{2r+r^*}\eta'} < 1$$

. By solving this quadratic inequality with respect to η, we obtain:

$$\left(\frac{M_{2r+r^*}}{m_{2r+r^*}}\right)^2 \leq 1 + \frac{\sqrt{r-r^*}}{2\sqrt{r^*}},$$

As a result, we obtain the the condition $r \geq C_1' \left(\frac{M_{2r+r^*}}{m_{2r+r^*}}\right)^4 r^*$ for some $C_1' > 0$. Furthermore, since $r = \alpha r^*$ for some $\beta > 1$, we conclude the condition on step size η' as $\frac{1-\sqrt{\beta'}}{M_{2r+r^*}} \leq \eta' \leq \frac{1+\sqrt{\beta'}}{m_{2r+r^*}}$ where $\beta' = \frac{\sqrt{\beta-1}}{\sqrt{\beta-1}+2}$ for some $\beta > 1$. If we initialize at $L^0 = 0$, then we obtain ϑ accuracy after $T = \mathcal{O}\left(\log\left(\frac{\|L^*\|_F}{\vartheta}\right)\right)$ iterations. □

Proof of Theorem 4.10. The proof of this theorem is a direct application of the Lemma 5.4 in (Chandrasekaran et al., 2009) and we restate it for completeness:

Lemma 4.16. *Let C denote the sample covariance matrix, then with probability at least $1 - 2\exp(-p)$ we have $\|C - (S^* + L^*)^{-1}\|_2 \leq c_1\sqrt{\frac{p}{n}}$ where $c_1 > 0$ is a constant.*

By noting that $\nabla F(L^*) = C - (S^* + L^*)^{-1}$ and $\text{rank}(J_t) \leq 2r + r^* \leq 3r$, we can bound the term on the right hand side in Theorem 4.9 as:

$$\|\mathcal{P}_{J_t}\nabla F(L^*)\|_F \leq \sqrt{3r}\|\nabla F(L^*)\|_2 \leq c_2\sqrt{\frac{rp}{n}}.$$

□

The key observation is that the objective function in (4.8) is globally strongly convex, and when restricted to any compact psd cone, it also satisfies the smoothness condition. As a result, it satisfies RSC/RSS conditions. Our strategy to prove Theorems 4.11 and 4.12 is to establish upper and lower bounds on the spectrum of the sequence of estimates L^t independent of t. We use the following lemma.

Lemma 4.17. *(Yuan et al., 2014a; Boyd and Vandenberghe, 2004) The Hessian of the objective function $F(L)$ is given by $\nabla^2 F(L) = \Theta^{-1} \otimes \Theta^{-1}$ where $\otimes$ denotes the Kronecker product and $\Theta = \bar{S} + L$. In addition if $\alpha I \preceq \Theta \preceq \beta I$ for some α and β, then $\frac{1}{\beta^2} I \preceq \nabla^2 F(L) \preceq \frac{1}{\alpha^2} I$.*

Lemma 4.18 (Weyl type inequality)**.** *For any two matrices $A, B \in \mathbb{R}^{p\times p}$, we have:*

$$\max_{1\leq i\leq p} |\sigma_i(A+B) - \sigma_i(A)| \leq \|B\|_2.$$

If we establish an universal upper bound and lower bound on $\lambda_1(\Theta^t)$ and $\lambda_p(\Theta^t)$ for all $t = 1\dots T$, then we can bound the RSC constant as $m_{2r+r^*} \geq \frac{1}{\lambda_1(\Theta^t)^2}$ and the RSS-constant as $M_{2r+r^*} \leq \frac{1}{\lambda_p(\Theta^t)^2}$ using Lemma 4.17 and the definition of RSS/RSC.

Proof of Theorem 4.11. Recall that by Theorem 4.9, we have $\|L^t - L^*\|_F \leq \rho'\|L^{t-1} - L^*\|_F + \nu'\eta'\|\mathcal{P}_{J_t}\nabla F(L^*)\|_{F},$. By Theorem 4.10, the second term on the right hand side can be bounded by $\mathcal{O}(\sqrt{\frac{rp}{n}})$ with high probability. Therefore, recursively applying this inequality to L^t (and initializing with zero), we obtain:

$$\|L^t - L^*\|_F \leq (\rho')^t\|L^*\|_F + \frac{c_2\nu'\eta'}{1-\rho'}\sqrt{\frac{rp}{n}}. \tag{4.23}$$

Since $\rho' < 1$, then $(\rho')^t < 1$. On the other hand $\|L^*\|_F \leq \sqrt{r^*}\|L^*\|_2$. Hence, $\rho^t\|L^*\|_F \leq \sqrt{r^*}\|L^*\|_2$. Also, by the Weyl inequality, we have:

$$\|L^t\|_2 - \|L^*\|_2 \leq \|L^t - L^*\|_2 \leq \|L^t - L^*\|_F. \tag{4.24}$$

Combining (4.23) and (4.25) and using the fact that $\lambda_1(L^t) \leq \sigma_1(L^t)$,

$$\begin{aligned}\lambda_1(L^t) &\leq \|L^*\|_2 + \|L^t - L^*\|_F \\ &\leq \|L^*\|_2 + \sqrt{r^*}\|L^*\|_2 + \frac{c_2\nu'\eta'}{1-\rho'}\sqrt{\frac{rp}{n}}.\end{aligned}$$

Hence for all t,

$$\lambda_1(\Theta^t) = S_1 + \lambda_1(L^t) \leq S_1 + \left(1+\sqrt{r^*}\right)\|L^*\|_2 + \frac{c_2\nu'\eta'}{1-\rho'}\sqrt{\frac{rp}{n}}. \tag{4.25}$$

For the lower bound, we trivially have for all t:

$$\lambda_p(\Theta^t) = \lambda_p(\bar{S} + L^t) \geq S_p. \tag{4.26}$$

If we select $n = \mathcal{O}\left(\frac{1}{\delta^2}\left(\frac{\nu'\eta'}{1-\rho'}\right)^2 rp\right)$ for some small constant $\delta > 0$, then (4.25) becomes:

$$\lambda_1(\Theta^t) \leq S_1 + \left(1+\sqrt{r}\right)\|L^*\|_2 + \delta.$$

As mentioned above, we set $m_{2r+r^*} \geq \frac{1}{\lambda_1^2(\Theta^t)}$ and $M_{2r+r^*} \leq \frac{1}{\lambda_p^2(\Theta^t)}$ which implies $\frac{M_{2r+r^*}}{m_{2r+r^*}} \leq \frac{\lambda_1^2(\Theta^t)}{\lambda_p^2(\Theta^t)}$. In order to satisfy the assumption on the RSC/RSS in theorem 4.9, i.e., $\frac{M^4_{2r+r^*}}{m^4_{2r+r^*}} \leq C_2'\frac{r}{r^*}$ for some

$C_2' > 0$, we need to establish a regime such that $\frac{\lambda_1^8(\Theta^t)}{\lambda_p^8(\Theta^t)} \leq C_2' \frac{r}{r^*}$. As a result, to satisfy this condition, we need to have the following condition, verifying the assumption in the theorem.

$$S_p \leq S_1 \leq C_3' (\frac{r}{r^*})^{\frac{1}{8}} S_p - \left(1 + \sqrt{r^*}\right) \|L^*\|_2 - \delta. \tag{4.27}$$

for some constant $C_3' > 0$. □

Proof of Theorem 4.12. The proof is similar to the proof of theorem 4.11. Recall that by theorem 4.5, we have

$$\|L^{t+1} - L^*\|_F \leq \rho \|L^t - L^*\|_F + \nu\eta \|\mathcal{P}_{J_t} \nabla F(L^*)\|_F,$$

As before, the second term on the right hand side is bounded by $\mathcal{O}(\sqrt{\frac{rp}{n}})$ with high probability by Theorem 4.10. As above, recursively applying this inequality to L^t and using zero initialization, we obtain:

$$\|L^t - L^*\|_F \leq \rho^t \|L^*\|_F + \frac{c_2 \nu\eta}{1-\rho} \sqrt{\frac{rp}{n}}.$$

Since $\rho < 1$, then $\rho^t < 1$. Now similar to the exact algorithm, $\|L^*\|_F \leq \sqrt{r^*} \|L^*\|\|_2$ and $\rho_1^t \|L^*\|_F \leq \sqrt{r^*} \|L^*\|_2$. , Hence with high probability,

$$\begin{aligned} \lambda_1(L^t) &\leq \|L^*\|_2 + \|L^t - L^*\|_F \\ &\leq \left(1 + \sqrt{r^*}\right) \|L^*\|_2 + \frac{c_2 \nu\eta}{1-\rho} \sqrt{\frac{rp}{n}}, \end{aligned} \tag{4.28}$$

Hence, for all t:

$$\lambda_1(\Theta^t) = S_1 + \lambda_1(L^t) \leq S_1 + \left(1 + \sqrt{r^*}\right) \|L^*\|_2 + \frac{c_2 \nu\eta}{1-\rho} \sqrt{\frac{rp}{n}}, \tag{4.29}$$

Also, we trivially have:

$$\lambda_p(\Theta^t) = \lambda_p(\bar{S} + L^t) \geq S_p - a', \ \forall t. \tag{4.30}$$

By selecting $n = \mathcal{O}\left(\frac{1}{\delta'^2} \left(\frac{\nu\eta}{1-\rho}\right)^2 rp\right)$ for some small constant $\delta' > 0$, (4.29) can be written as follows:

$$\lambda_1(\Theta^t) \leq S_1 + \left(1 + \sqrt{r^*}\right) \|L^*\|_2 + \delta',$$

In order to satisfy the assumptions in Theorem 4.5, i.e., $\frac{M^4_{2r+r^*}}{m^4_{2r+r^*}} \leq \frac{C''_2}{1-\epsilon}\frac{r}{r^*}$, we need to guarantee that $\frac{\lambda_1^8(\Theta^t)}{\lambda_p^8(\Theta^t)} \leq \frac{C''_2}{1-\epsilon}\frac{r}{r^*}$. As a result, to satisfy this inequality, we need to have the following condition on S_1 and S_p:

$$S_p \leq S_1 \leq \frac{C''_3}{(1-\epsilon)^{\frac{1}{8}}}(\frac{r}{r^*})^{\frac{1}{8}}(S_p - a') - \left(1+\sqrt{r^*}\right)\|L^*\|_2 - \delta'. \tag{4.31}$$

for some $C''_3 > 0$. Also, we can choose RSC/RSS constant as previous case. □

Proof of Theorem 4.13. Recall from (4.28) that with very high probability,

$$\|L^t\|_2 \leq \left(1+\sqrt{r^*}\right)\|L^*\|_2 + \frac{c_2\nu\eta}{1-\rho}\sqrt{\frac{rp}{n}}.$$

Also, we always have: $\lambda_p(L^t) \geq -\|L^t\|_2$. As a result:

$$\lambda_p(L^t) \geq -\left(1+\sqrt{r^*}\right)\|L^*\|_2 - \frac{c_2\nu\eta}{1-\rho}\sqrt{\frac{rp}{n}}. \tag{4.32}$$

Now if the inequality $\left(1+\sqrt{r^*}\right)\|L^*\|_2 + \frac{c_2\nu\eta}{1-\rho}\sqrt{\frac{rp}{n}} < S_p$ is satisfied, then we can select $0 < a' \leq \left(1+\sqrt{r^*}\right)\|L^*\|_2 + \frac{c_2\nu\eta}{1-\rho}\sqrt{\frac{rp}{n}}$. The former inequality is satisfied by the assumption of Theorem 4.12 on $\|L^*\|_2$, i.e.,

$$\|L^*\|_2 \leq \frac{1}{1+\sqrt{r^*}}\left(\frac{S_p}{1+C_4\left((1-\epsilon)(\frac{r^*}{r})\right)^{\frac{1}{8}}} - \frac{S_1(C_4((1-\epsilon)(\frac{r^*}{r}))^{\frac{1}{8}})}{1+C_4\left((1-\epsilon)(\frac{r^*}{r})\right)^{\frac{1}{8}}} - \frac{c_2\nu\eta}{1-\rho}\sqrt{\frac{rp}{n}}\right).$$

□

CHAPTER 5. FAST AND PROVABLE ALGORITHMS FOR LEARNING TWO-LAYER POLYNOMIAL NEURAL NETWORKS

In this chapter, we study the problem of (provably) learning the weights of a two-layer neural network with quadratic activations. In particular, we focus on the under-parametrized regime where the number of neurons in the hidden layer is (much) smaller than the dimension of the input. Our approach uses a lifting trick, which enables us to borrow algorithmic ideas from low-rank matrix estimation. In this context, we propose three novel, non-convex training algorithms. We support our algorithms with rigorous theoretical analysis, and show that the proposed algorithms enjoy linear convergence, fast running time per iteration, and near-optimal sample complexity. Finally, we complement our theoretical results with several numerical experiments.

5.1 Introduction

The re-emergence of neural networks (spurred by the advent of deep learning) has had a remarkable impact on various sub-domains of artificial intelligence (AI) object recognition in images, natural language processing, and automated drug discovery, among many others. However, despite the successful *empirical* performance of neural networks for these AI tasks, *provable* methods for learning neural networks remain relatively mysterious. Indeed, training a network of even moderate size requires solving a very large-scale, highly non-convex optimization problem.

In this chapter, we (provably) resolve several algorithmic challenges that arise in the context of a special class of (shallow) neural networks by making connections to the better-studied problem of *low-rank matrix estimation*. Our hope is that a rigorous understanding of the fundamental limits of training shallow networks can be used as building blocks to obtain theoretical insights for more complex networks.

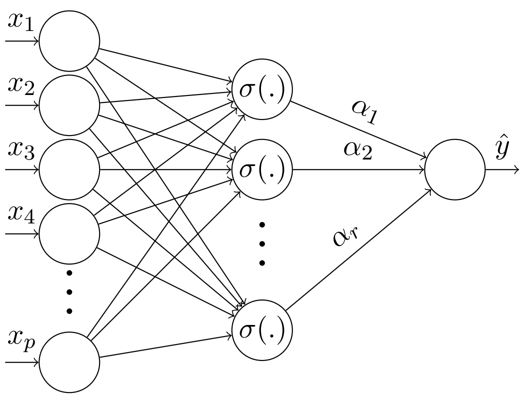


Figure 5.1: Two-layer polynomial neural network.

5.1.1 Setup

Consider a shallow (two-layer) neural network architecture, as illustrated in Figure 5.1. This network comprises p input nodes, a single hidden layer with r neurons with activation function $\sigma(z)$, first layer weights $\{w_j\}_{j=1}^r \subset \mathbb{R}^p$, and an output layer comprising of a single node and weights $\{\alpha_j\}_{j=1}^r \subset \mathbb{R}$. If $\sigma(z) = z^2$, then the above network is called a *polynomial neural network* (Livni et al., 2014). More precisely, the input-output relationship between an input, $x \in \mathbb{R}^p$, and the corresponding output, $y \in \mathbb{R}$, is given by:

$$\hat{y} = \sum_{j=1}^{r} \alpha_j \sigma(w_j^T x) = \sum_{j=1}^{r} \alpha_j \langle w_j, x \rangle^2.$$

In this chapter, our focus is in the so-called "under-parameterized" regime where $r \ll p$. Our goal is to learn this network, given a set of training input-output pairs $\{(x_i, y_i)\}_{i=1}^m$. We do so by finding a set of weights $\{\alpha_j, w_j\}_{j=1}^r$ that minimize the following *empirical risk*:

$$\min_{W \in \mathbb{R}^{r \times p}, \alpha \in \mathbb{R}^r} \quad F(W, \alpha) = \frac{1}{2m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2, \tag{5.1}$$

where the rows of W and the entries of α indicate the first-and second-layer weights, respectively. Numerous recent papers have explored (provable) algorithms to learn the weights of such a network under distributional assumptions on the input data (Livni et al., 2014; Lin and Ye, 2016; Janzamin et al., 2015; Tian, 2016; Zhong et al., 2016; Soltanolkotabi et al., 2017; Li and Yuan, 2017)[1].

[1]While quadratic activation functions are rarely used in practice, stacking multiple such two-layer blocks can be used to simulate networks with higher-order polynomial and sigmoidal activations (Livni et al., 2014).

Clearly, the empirical risk defined in (5.1) is extremely nonconvex (involving fourth-powers of the entries of w_j, coupled with the squares of α_j). However, this can be circumvented using a clever *lifting* trick: if we define the matrix variable $L_* = \sum_{j=1}^{r} \alpha_j w_j w_j^T$, then the input-output relationship becomes:

$$\hat{y}_i = x_i^T L_* x_i = \langle x_i x_i^T, L_* \rangle, \tag{5.2}$$

where $x_i \in \mathbb{R}^p$ denotes the i^{th} training sample. Moreover, the variable L_* is a rank-r matrix of size $p \times p$ Therefore, (5.1) can be viewed as an instance of learning a fixed (but unknown) rank-r symmetric matrix $L_* \in \mathbb{R}^{p \times p}$ with $r \ll p$, from a small number of *rank-one* linear observations given by $A_i = x_i x_i^T$. While still non-convex, low-rank matrix estimation problems such as (5.2) are much better understood. Two specific instances in statistical learning include:

Matrix sensing and matrix completion. Reconstructing low-rank matrices from (noisy) linear measurements of the form $y_i = \langle X_i, L_* \rangle$ impact several applications in control and system identification (Fazel, 2002), collaborative filtering (Candès and Recht, 2009; Recht et al., 2010b), and imaging. The problem (5.2) specializes the matrix sensing problem to the case where the measurement vectors X_i are constrained to be themselves rank-one.

Covariance sketching. Estimating a high-dimensional covariance matrix, given a stream of independent samples $\{s_t\}_{t=1}^{\infty}$, $s_t \in \mathbb{R}^p$, involves maintaining the empirical estimate $Q = E[s_t s_t^T]$, which can require *quadratic* ($\mathcal{O}(p^2)$) space complexity. Alternatively, one can record a sequence of $m \ll p^2$ *linear sketches* of each sample: $z_i = x_i^T s_t$ for $i = 1, \ldots, m$. At the conclusion of the stream, sketches corresponding to a given vector x_i are squared and aggregated to form a measurement: $y_i = E[z_i^2] = E[(x_i^T s_t)^2] = x_i^T Q x_i$, which is nothing but a linear sketch of Q of the form (5.2). Again, several matrix recovery methods that "invert" such sketches exist; see (Cai and Zhang, 2015; Chen et al., 2015; Dasarathy et al., 2015).

5.1.2 Our Contributions

Here, we make concrete algorithmic progress on solving low-rank matrix estimation problems of the form (5.2). In the context of learning polynomial neural networks, once we have estimated a rank-r symmetric matrix L_*, we can always produce weights $\{\alpha_j, w_j\}$ by an eigendecomposition of L_*. In general, a range of algorithms for solving (5.2) (or variants thereof) exist in the literature, and can be broadly classified into two categories: (i) *convex* approaches, all of which involve enforcing the rank-r assumption in terms of a convex penalty term, such as the nuclear norm (Fazel, 2002; Recht et al., 2010b; Cai and Zhang, 2015; Chen et al., 2015; Cai et al., 2010); (ii) *nonconvex* approaches based on either alternating minimization (Zhong et al., 2015; Lin and Ye, 2016) or greedy approximation (Livni et al., 2014; Shalev-Shwartz et al., 2011).

Both types of approaches suffer from severe computational difficulties, particularly when the data dimension p is large. Even the most computationally efficient convex approaches require multiple invocations of singular value decomposition (SVD) of a (potentially) large $p \times p$ matrix, which can incur *cubic* ($\mathcal{O}(p^3)$) running time. Moreover, even the best available non-convex approaches require a very accurate initialization, and also require that the underlying matrix L_* is well-conditioned; if this is not the case, the running time of all available methods again inflates to $\mathcal{O}(p^3)$, or worse.

In this chapter, we take a different approach, and show how to leverage recent results in low-rank approximation to our advantage (Musco and Musco, 2015; Hegde et al., 2016). Our algorithm is also non-convex; however, unlike all earlier works, our method does not require any full SVD calculations. Specifically, we demonstrate that a careful concatenation of *randomized, approximate* SVD methods, coupled with appropriately defined gradient steps, leads to efficient and accurate matrix estimation.

To our knowledge, this work constitutes the first *nearly-linear* time method for low-rank matrix recovery from rank-one observations. Consequently, in the context of learning two-layer polynomial networks, our method is the first to exhibit *nearly-linear* running time, is *nearly sample-optimal* for fixed target rank r, and is *unconditional* (i.e., it makes no assumptions on the condition number of L_* or the weight matrix W). Numerical experiments reveal that our methods yield a very attractive tradeoff between sample complexity and running time for efficient matrix recovery.

Table 5.1: Summary of our contributions and comparison with existing algorithms. Here, $\beta = \frac{\sigma_1}{\sigma_r}$ denotes the condition number of L_*.

Algorithm	Sample complexity (m)	Total Running Time
Convex	$\mathcal{O}(pr)$	$\mathcal{O}\left(\frac{p^3}{\sqrt{\epsilon}}\right)$
GECO	N/A	$\mathcal{O}\left(\frac{p^2 \log(p)\text{poly}(r)}{\epsilon}\right)$
AltMin-LRROM	$\mathcal{O}\left(pr^4 \log^2(p)\beta^2 \log(\frac{1}{\epsilon})\right)$	$\mathcal{O}\left(mpr \log(\frac{1}{\epsilon}) + p^3\right)$
gFM	$\mathcal{O}(pr^3\beta^2 \log(\frac{1}{\epsilon}))$	$\mathcal{O}\left(mpr \log(\frac{1}{\epsilon}) + p^3\right)$
EP-ROM	$\mathcal{O}\left(pr^2 \log^4(p) \log(\frac{1}{\epsilon})\right)$	$\mathcal{O}\left(mp^2 \log(\frac{1}{\epsilon})\right)$
AP-ROM	$\mathcal{O}\left(pr^3 \log^4(p) \log(\frac{1}{\epsilon})\right)$	$\mathcal{O}\left(mpr \log(p) \log(\frac{1}{\epsilon}\right)$
Algorithm 5.3	$\mathcal{O}\left(pr\right)$	$\mathcal{O}\left(mpr \log(p) \log(\frac{1}{\epsilon}\right)$

5.1.3 Techniques

At a high level, our method can be viewed as a variant of the seminal algorithms proposed in (Jain et al., 2010) and (Jain et al., 2014), which essentially perform projected (or proximal) gradient descent with respect to the space of rank-r matrices. However, since computing SVD in high dimensions can be a bottleneck, we cannot use this approach directly. To this end, we use the *approximation*-based matrix recovery framework proposed in (Hegde et al., 2016). This work demonstrates how to carefully integrate approximate SVD methods into singular value projection (SVP)-based matrix recovery algorithms; in particular, algorithms that satisfy certain "head" and "tail" projection properties (explained below in Section 5.3) are sufficient to guarantee robust and fast convergence. Crucially, this framework removes the need to compute *even a single SVD*, as opposed to factorized methods which necessarily require one or multiple SVDs, together with stringent condition number assumptions.

However, a direct application of (projected) gradient descent does not succeed for matrix estimation problems obeying (5.2); two major obstacles arise:

Obstacle 1. It is well-known the measurement operator that maps L_* to y does not satisfy the so-called Restricted Isometry Property over rank-r matrices (Cai and Zhang, 2015; Chen et al., 2015; Zhong et al., 2015); therefore, all statistical and algorithmic correctness arguments of (Hegde et al., 2016) no longer apply.

Obstacle 2. The algebraic structure of the rank-one observations in (5.2) inflates the running time of computing even a simple gradient update to $\mathcal{O}(p^3)$ (irrespective of the algorithmic cost of rank-r projection, whether done using exact or approximate SVDs).

We resolve **Obstacle 1** by studying the concentration properties of certain linear operators of the form of rank-one projections, leveraging an approach first proposed in (Zhong et al., 2015). We show that a non-trivial "bias correction" step, coupled with projected descent-type methods, within each iteration is sufficient to achieve fast (linear) convergence. To be more precise, define the operator $\mathcal{A}$ such that $(\mathcal{A}(L_*))_i = x_i^T L_* x_i$ for $i = 1, \ldots, m$, where x_i is a standard normal random vector. A simple calculation shows that at any given iteration t, if L_t is the current estimate of the underlying matrix variable, then we have:

$$\mathbb{E}\mathcal{A}^*\mathcal{A}(L_t - L_*) = 2(L_t - L_*) + Tr(L_t - L_*)I,$$

where the operator $Tr(\cdot)$ denotes the trace of a matrix and the expectation is taken with respect to the randomness in the x_i's. The left hand side of this equation (roughly) corresponds to the expected value of the gradient in each iteration, and it is clear that while the gradient points in the "correct" direction $L_t - L_*$, it is additionally biased by the extra $Tr(.)$ term. Motivated by this, we develop a new descent scheme by carefully accounting for this bias. Interestingly, the sample complexity of our approach only increases by a mild factor (specifically, an extra factor r together with polylogarithmic terms) when compared to the best available techniques. Moreover, this scheme exhibits *linear* convergence in theory, and shows very competitive performance in experimental simulations.

We resolve **Obstacle 2** by carefully exploiting the rank-one structure of the observations. In particular, we develop a modification of the randomized block-Krylov SVD (or BK-SVD) algorithm of (Musco and Musco, 2015) to work for the case of certain "implicitly defined" matrices; specifically, we design a randomized SVD routine where the input is a linear operator that is constructed using vector-valued components. This modification, coupled with the tail-and head-projection arguments developed in (Hegde et al., 2016), enables us to achieve a fast per-iteration computational complexity. In particular, our algorithm strictly improves over the (worst-case) per-iteration running time of all existing algorithms; see Table 5.1.

While the above approach produces fast running time (up to poly-logarithmic factors), its theoretical success depends on the idea of fresh samples in each iteration. Our next algorithm removes this restriction by using the ℓ_1-loss function instead of the squared loss used in (5.1). However, the ℓ_1-loss is non-differentiable, nor does it satisfy our previous concentration property of the gradient. This motivates us to use the so-called RIP(ℓ_1, ℓ_2) (Foucart and Rauhut, 2013; Chen et al., 2015; Cai and Zhang, 2015). For this, we propose a projected sub-gradient algorithm which does not require use of fresh samples within each iteration, and enjoys linear convergence with an optimal sample complexity.

5.2 Prior Art

Due to space constraints, we only provide here a brief (and incomplete) review of related work, and describe how our method differs from earlier techniques.

Problems involving low-rank matrix estimation have received significant attention from the machine learning community over the last few years; see (Davenport and Romberg, 2016) for a recent survey. In early works for matrix recovery, the observation operator $\mathcal{A}$ is assumed to be parametrized by m independent *full-rank* $p \times p$ matrices that satisfy certain restricted isometry conditions (Recht et al., 2010b; Liu, 2011). In this setup, it has been established that $m = \mathcal{O}(pr)$ observations are sufficient to recover an unknown rank-r matrix L_* in (5.3) (Candes and Plan, 2011), and this scaling of sample complexity is statistically optimal.

In the context of provable methods for learning neural networks, two-layer networks have received special attention. For instance, (Livni et al., 2014) has considered a two-layer network with quadratic activation function (identical to the model proposed above), and proposed a greedy, *improper* learning algorithm: in each iteration, the algorithms adds one hidden neuron to the network until the risk falls below a threshold. While this algorithm is guaranteed to converge, its convergence rate is *sublinear.* Recently, in (Zhong et al., 2016), the authors have proposed a linearly convergent algorithm for learning two-layer networks for several classes of activation functions. They also derived an upper bound on the sample complexity of network learning which is linear in p, and depends polynomially on r and other spectral properties of the ground-truth (planted) weights. However, their theory does not provide convergence guarantees for quadratic functions; our work here closes this gap. Note that our focus here is not the estimation of the weights $\{\alpha_j, w_j\}$ themselves, but rather, any network that gives the same input-relationship.. As a result, our guarantees are stated in terms of the low-rank matrix L_*. Furthermore, unlike their algorithm, our sample complexity does not depend on the spectral properties of ground-truth weights.

Other works have also studied similar two-layer setups, including (Janzamin et al., 2015; Tian, 2016; Soltanolkotabi et al., 2017; Li and Yuan, 2017). In contrast with these results, our framework does not assume the over-parameterized setting where the number of hidden neurons r is greater than p. In addition, we explicitly derive a sample complexity that is linear in p, as well as demonstrate linear time convergence. Also, observe that if we let L_* to be rank-1, then Problem (5.2) is known as *generalized phase retrieval* for which several excellent algorithms are known (Candes et al., 2013, 2015; Netrapalli et al., 2013). However, our problem is more challenging as it allows L_* to have arbitrary rank-r.

We now briefly contrast our method with other algorithmic techniques for low-rank matrix estimation. Broadly, two classes of such techniques exist. The first class of matrix estimation techniques can be categorized as approaches based on convex relaxation (Chen et al., 2015; Cai and Zhang, 2015; Kueng et al., 2017; Candes et al., 2013). For instance, the authors in (Chen et al., 2015; Cai and Zhang, 2015) demonstrate that the observation operator $\mathcal{A}$ satisfies a specialized mixed-norm isometry condition called the *RIP*-ℓ_2/ℓ_1. Further, they show that the sample complexity of matrix recovery using rank-one projections matches the optimal rate $\mathcal{O}(pr)$. However, these methods advocate using either semidefinite programming (SDP) or proximal sub-gradient algorithms (Boyd and Vandenberghe, 2004; Goldstein et al., 2014, 2015), both of which are too slow for very high-dimensional problems.

The second class of techniques can be categorized as non-convex approaches, which are all based on a factorization-based approach initially advocated by (Burer and Monteiro, 2003). Here, the underlying low-rank matrix variable is factorized as $L_* = UV^T$, where $U, V \in \mathbb{R}^{p\times r}$ (Zheng and Lafferty, 2015; Tu et al., 2016). In the Altmin-LRROM method proposed by (Zhong et al., 2015), U and V are updated in alternative fashion. However, the setup in (Zhong et al., 2015) is different from this chapter, as it uses an asymmetric observation model, in which observation y_i is given by $y_i = x_i^T L_* z_i$ with x_i and z_i being independent random vectors. Our goal is to analyze the more challenging case where the observation operator $\mathcal{A}$ is symmetric and defined according (5.3). In a subsequent work (called the generalized factorization machine) by (Lin et al., 2017), U and V are updated based on the construction of certain sequences of moment estimators.

Both the approaches of (Zhong et al., 2015) and (Lin et al., 2017) require a spectral initialization which involves running a rank-r SVD on a given $p\times p$ matrix, and therefore the running time heavily depends on the condition number (i.e., the ratio of the maximum and the minimum nonzero singular values) of L_*. To our knowledge, only three works in the matrix recovery literature require no full SVDs (Bhojanapalli et al., 2016b; Hegde et al., 2016; Ge et al., 2016). However, both (Bhojanapalli et al., 2016b) and (Hegde et al., 2016) assume that the restricted isometry property is satisfied, which is not applicable in our setting. Moreover, (Ge et al., 2016) makes stringent assumptions on the condition number, as well as the coherence, of the unknown matrix.

Finally, we mention that a matrix estimation scheme using approximate SVDs (based on Frank-Wolfe type greedy approximation) has been proposed for learning polynomial neural networks (Shalev-Shwartz et al., 2011; Livni et al., 2014). Moreover, this approach has been shown to compare favorably to typical neural network learning methods (such as stochastic gradient descent). However, the rate of convergence is sub-linear, and they provide no sample-complexity guarantees. Indeed, the main motivating factor of our approach was to accelerate the running time of such greedy approximation techniques. We complete this line of work by providing (a) rigorous statistical analysis that precisely establishes upper bounds on the number of samples required for learning such networks, and (b) an algorithm that provably exhibits *linear* convergence, as well as *nearly-linear* per iteration running time.

5.3 Main Results

5.3.1 Preliminaries

Let us first introduce some notation. Throughout this chapter, $\|\cdot\|_F$ and $\|\cdot\|_2$ denote the matrix Frobenius and spectral norm, respectively, and $\mathrm{Tr}(\cdot)$ denotes matrix trace. The phrase "with high probability" indicates an event whose failure rate is exponentially small. We assume that the training data samples (x, y) obey a generative model (5.2) written as:

$$y = \sum_{j=1}^{r} \alpha_j^* \sigma(\langle w_j^*, x\rangle) = x^T L_* x + e \tag{5.3}$$

where $L_* \in \mathbb{R}^{p\times p}$ is the "ground-truth" matrix (with rank equal to r). Define $\mathcal{A} : \mathbb{R}^{p\times p} \to \mathbb{R}^m$ such that:

$$\mathcal{A}(L_*) = [x_1^T L_* x_1, x_2^T L_* x_2, \dots, x_m^T L_* x_m]^T,$$

and each $x_i \overset{i.i.d}{\sim} \mathcal{N}(0, I)$ is a normal random vector in $\mathbb{R}^p$ for $i = 1, \dots, m$. The adjoint operator of $\mathcal{A}$ is defined as $\mathcal{A}^*(y) = \sum_{i=1}^m y_i x_i x_i^T$. Here, $e \in \mathbb{R}^m$ denotes an additive noise term; throughout the chapter (for the purpose of analysis) we assume that e is zero-mean, subgaussian with i.i.d entries, and independent of x_i's. The goal is to learn the rank-r matrix parameter L^* from as few samples as possible.

In our analysis, we require the operators $\mathcal{A}$ and $\mathcal{A}^*$ to satisfy the following regularity condition with respect to the set of low-rank matrices. We call this the *Conditional Unbiased Restricted Isometry Property*, abbreviated as *CU-RIP*(ρ):

Definition 5.1. *Consider fixed rank-r matrices L_1 and L_2. Then, $\mathcal{A}$ is said to satisfy CU-RIP(ρ) if there exists $0 < \rho < 1$ such that*

$$\left\| L_1 - L_2 - \frac{1}{2m}\mathcal{A}^*\mathcal{A}(L_1 - L_2) - \frac{1}{2m}\mathbf{1}^T\left(\mathcal{A}(L_1) - \mathcal{A}(L_2)\right) I \right\|_2 \le \rho \left\| L_1 - L_2 \right\|_2.$$

Algorithm 5.1 EP-ROM

```
**Inputs:** $y$, number of iterations $K$, independent data samples $\{x_1^t, x_2^t \ldots, x_m^t\}$ for $t = 1, \ldots, K$, rank $r$
**Outputs:** Estimates $\widehat{L}$
**Initialization:** $L_0 \leftarrow 0$, $t \leftarrow 0$
**Calculate:** $\bar{y} = \frac{1}{m}\sum_{i=1}^m y_i$
**while** $t \leq K$ **do**
    $L_{t+1} = \mathcal{P}_r\left(L_t - \frac{1}{2m}\sum_{i=1}^m \left((x_i^t)^T L_t x_i^t - y_i\right) x_i^t (x_i^t)^T - (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t) - \frac{1}{2}\bar{y})I\right)$
    $t \leftarrow t + 1$
**end while**
**Return:** $\widehat{L} = L_K$
```

Let $\mathbb{U}_r$ denote the set of all rank-r matrix subspaces, i.e., subspaces of $\mathbb{R}^{p\times p}$ which are spanned by any r atoms of the form uv^T where $u, v \in \mathbb{R}^p$ are unit ℓ_2-norm vectors. We use the idea of *head* and *tail* approximate projections with respect to $\mathbb{U}_r$ first proposed in (Hegde et al., 2015b), and instantiated in the context of low-rank approximation in (Hegde et al., 2016).

Definition 5.2 (Approximate tail projection). *$\mathcal{T} : \mathbb{R}^{p\times p} \to \mathbb{U}_r$ is a ε-approximate tail projection algorithm if for all $L \in \mathbb{R}^{p\times p}$, $\mathcal{T}$ returns a subspace $W = \mathcal{T}(L)$ that satisfies: $\|L - \mathcal{P}_W L\|_F \leq (1+\varepsilon)\|L - L_r\|_F$, where L_r is the optimal rank-r approximation of L.*

Definition 5.3 (Approximate head projection). *$\mathcal{H} : \mathbb{R}^{p\times p} \to \mathbb{U}_r$ is a ε-approximate head projection if for all $L \in \mathbb{R}^{p\times p}$, the returned subspace $V = \mathcal{H}(L)$ satisfies: $\|\mathcal{P}_V L\|_F \geq (1-\varepsilon)\|L_r\|_F$, where L_r is the optimal rank-r approximation of L.*

We also need the following mixed RIP definition due to (Foucart and Rauhut, 2013; Chen et al., 2015; Cai and Zhang, 2015) for our third algorithm, proposed in Section 5.3.4.

Definition 5.4 (RIP(ℓ_1, ℓ_2) for low-rank matrices). *A linear operator $\mathcal{B}$ satisfies RIP(ℓ_1, ℓ_2) if for any two rank-r matrices L_1 and L_2, there exists constants $0 < \alpha < \beta$ such the following holds: $\alpha\|L_1 - L_2\|_F \leq \frac{1}{m}\|\mathcal{B}(L_1 - L_2)\|_1 \leq \beta\|L_1 - L_2\|_F$.*

5.3.2 Algorithms and Theoretical Results

We now propose methods to estimate L_* given knowledge of $\{x_i, y_i\}_{i=1}^m$. Our first method is somewhat computationally inefficient, but achieves very good sample complexity and serves to illustrate the overall algorithmic approach. Consider the non-convex, constrained risk minimization problem:

$$\begin{aligned} \min_{L \in \mathbb{R}^{p\times p}} \quad & F(L) = \frac{1}{2m}\sum_{i=1}^{m}\left(y_i - x_i^T L x_i\right)^2 \\ \text{s.t.} \quad & \operatorname{rank}(L) \le r. \end{aligned} \tag{5.4}$$

To solve this problem, we first propose an algorithm that we call *Exact Projections for Rank-One Matrix* recovery, or *EP-ROM*, described in pseudocode form in Algorithm 5.1[2].

We now analyze this algorithm. First, we provide a theoretical result which establishes statistical and optimization convergence rates of EP-ROM. More precisely, we derive an upper bound on the estimation error (measured using the spectral norm) of recovering L_*. We defer all the proofs to the appendix.

Theorem 5.5 (Linear convergence of EP-ROM). *Consider the sequence of iterates (L_t) obtained in EP-ROM. Assume that in each iteration the linear operator $\mathcal{A}$ satisfies CU-RIP(ρ) for some $0 < \rho < \frac{1}{2}$, then EP-ROM outputs a sequence of estimates L_t such that:*

$$\|L_{t+1} - L_*\|_2 \le q\big\|L_t - L_*\big\|_2 + \frac{1}{2m}(|\mathbf{1}^T e| + \big\|\mathcal{A}^* e\big\|_2), \tag{5.5}$$

where $0 < q < 1$.

The contraction factor, q, in Equation (5.5) can be arbitrary small if we choose m sufficiently large, and we elaborate it in Theorem (5.7). The second and third term in (5.5) represent the *statistical error rate*. In the next Theorem, we show that these error terms can be suitably bounded. Furthermore, Theorem 5.5 implies (via induction) that EP-ROM exhibits *linear* convergence; please see Corollary 5.8.

Theorem 5.6 (Bounding the statistical error of EP-ROM). *Consider the generative model (5.3) with zero-mean subgaussian noise $e \in \mathbb{R}^m$ with i.i.d. entries (and independent of the x_i's) such*

[2]In Alg 5.1, $\mathcal{P}_r$ denotes the projection operator onto the set of rank-r matrices.

that $\tau = \max_{1 \le j \le m} \|e_j\|_{\psi_2}$ *(Here,* $\|\cdot\|_{\psi_2}$ *denotes the* ψ_2*-norm; see Definition 5.14 in the appendix). Then, with probability at least* $1-\gamma$*, we have:*

$$\frac{1}{m}|\mathbf{1}^T e| + \left\|\frac{1}{m}\mathcal{A}^* e\right\|_2 \le C''_\tau \sqrt{\frac{p \log^2 p}{m}} \log(\frac{p}{\gamma}). \tag{5.6}$$

where $C''_\tau > 0$ *is constant which depends on* τ*.*

To establish linear convergence of EP-ROM, we assume that the CU-RIP holds at *each* iteration. The following theorem certifies this assumption.

Theorem 5.7 (Verifying CU-RIP)**.** *At any iteration* t *of EP-ROM, with probability at least* $1-\xi$*, CU-RIP(*ρ*) is satisfied with* $0 < \rho < \frac{1}{2}$ *provided that* $m = \mathcal{O}\left(\frac{1}{\delta^2} pr^2 \log^3 p \log(\frac{p}{\xi})\right)$ *for some* $\delta > 0$*.*

Integrating the above results, together with the assumption of availability of a batch of m independent samples in each iteration, we obtain the following corollary formally establishing linear convergence. We acknowledge that this assumption of "fresh samples" is somewhat unrealistic and is an artifact of our proof techniques; nonetheless, it is a standard mechanism for proofs for non-convex low-rank matrix estimation (Hardt, 2014; Zhong et al., 2016)

Corollary 5.8. *After* K *iterations, with high probability the output of EP-ROM satisfies:*

$$\|L_K - L_*\|_2 \le q^K \|L_*\|_2 + \frac{C''_\tau}{1-q}\sqrt{\frac{p \log^3 p}{m}}. \tag{5.7}$$

where C''_τ is given in (5.6). Thus, under all the assumptions in theorem 5.5, to achieve ϵ-accuracy for estimation of L_* in terms of the spectral norm, EP-ROM needs $K = \mathcal{O}(\log(\frac{\|L_*\|_2}{\epsilon}))$ iterations. Based on Theorems 5.6, 5.7, and Corollary 5.8, the sample complexity of EP-ROM scales as $m = \mathcal{O}\left(pr^2 \log^4 p \log(\frac{1}{\epsilon})\right)$.

While EP-ROM exhibits linear convergence, the *per-iteration* complexity is still high since it requires projection onto the space of rank-r matrices, which necessitates the application of SVD. In the absence of any spectral assumptions on the input to the SVD, the per-iteration running time of EP-ROM can be *cubic*, which can be prohibitive. Overall, we obtain a running time of $\widetilde{\mathcal{O}}(p^3 r^2)$ in order to achieve ϵ-accuracy (please see Section 5.6.3 in the appendix for a longer discussion).

Algorithm 5.2 AP-ROM

```
Inputs: $y$, number of iterations $K$, independent data samples $\{x_1^t, x_2^t \ldots, x_m^t\}$ for $t = 1, \ldots, K$, rank $r$
Outputs: Estimates $\widehat{L}$
Initialization: $L_0 \leftarrow 0$, $t \leftarrow 0$
Calculate: $\bar{y} = \frac{1}{m}\sum_{i=1}^{m} y_i$
while $t \leq K$ do
    $L_{t+1} = \mathcal{T}\left(L_t - \mathcal{H}\left(\frac{1}{2m}\sum_{i=1}^{2m}\left((x_i^t)^T L_t x_i^t - y_i\right)x_i^t(x_i^t)^T - (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t) - \frac{1}{2}\bar{y})I\right)\right)$
    $t \leftarrow t + 1$
end while
Return: $\widehat{L} = L_K$
```

To reduce the running time, one can instead replace a standard SVD routine with approximation heuristics such as Lanczos iterations (Lanczos, 1950); however, these may not result in algorithms with provable convergence guarantees. Instead, following (Hegde et al., 2016), we can use a pair of *inaccurate* rank-r projections (in particular, tail-and head-approximate projection operators). Based on this idea, we propose our second algorithm that we call *Approximate Projection for Rank One Matrix* recovery, or *AP-ROM*. We display the pseudocode of AP-ROM in Algorithm 5.2.

The specific choice of approximate SVD algorithms that simulate the operators $\mathcal{T}(.)$ and $\mathcal{H}(.)$ is flexible. We note that tail-approximate projections have been widely studied in the numerical linear algebra literature (Clarkson and Woodruff, 2013; Mahoney and Drineas, 2009; Rokhlin et al., 2009); however, head-approximate projection methods are less well-known. In our method, we use the randomized Block Krylov SVD (BK-SVD) method proposed by (Musco and Musco, 2015), which has been shown to satisfy both types of approximation guarantees (Hegde et al., 2016). One can alternatively use LazySVD, recently proposed by (Allen-Zhu and Li, 2016), which also satisfies both guarantees. The nice feature of these methods is that their running time is ***independent of the spectral gap*** of the matrix. We leverage this property to show asymptotic improvements over other fast SVD methods (such as the power method).

We briefly discuss the BK-SVD algorithm. In particular, BK-SVD takes an input matrix with size $p \times p$ with rank r and returns a r-dimensional subspace which approximates the top right r singular vectors of the input. Mathematically, if $A \in \mathbb{R}^{p\times p}$ is the input, A_r is the best rank-r

approximation to it, and Z is a basis matrix that spans the subspace returned by BK-SVD, then the projection of A into Z, $B = ZZ^T A$ satisfies the following relations:

$$\|A - B\|_F \leq (1+\varepsilon)\|A - A_r\|_F,$$

$$|u_i^T AA^T u_i - z_i AA^T z_i| \leq \varepsilon \sigma_{r+1}^2,$$

where $\varepsilon > 0$ is defined as the tail and head projection approximate constant, and u_i denotes the i^{th} right eigenvector of A. In Appendix-B of (Hegde et al., 2016), it has been shown that the per-vector guarantee can be used to prove the approximate head projection property, i.e., $\|B\|_F \geq (1-\varepsilon)\|A_r\|_F$.

We now establish that AP-ROM also exhibits linear convergence, while obeying similar statistical properties as EP-ROM. We have the following results:

Theorem 5.9 (Convergence of AP-ROM)**.** *Consider the sequence of iterates (L_t) obtained in AP-ROM. Assume that in each iteration t, $\mathcal{A}$ satisfies CU-RIP(ρ'), then AP-ROM outputs a sequence of estimates L_t such that:*

$$\|L_{t+1} - L_*\|_F \leq q_1'\|L_t - L_*\|_F + q_2'\left(|\mathbf{1}^T e| + \left\|\mathcal{A}^* e\right\|_2\right), \tag{5.8}$$

where $q_1' = (2+\varepsilon)(\rho' + \sqrt{1-\phi^2})$, $q_2' = \frac{\sqrt{r}}{2m}\left(2 - \varepsilon + \frac{\phi(2-\varepsilon)(2+\varepsilon)}{\sqrt{1-\phi^2}}\right)$, and $\phi = (1-\varepsilon)(1-\rho') - \rho'$.

Similar to Theorem 5.7, we can show that CU-RIP is satisfied in each iteration of AP-ROM with probability at least $1-\xi$, provided that $m = \mathcal{O}\left(\frac{1}{\delta^2} pr^3 \log^3 p \log(\frac{p}{\xi})\right)$. Hence, we require a factor-r increase compared to before. Overall, we have the following result:

Corollary 5.10. *The output of AP-ROM satisfies the following after K iterations with high probability:*

$$\|L_K - L_*\|_F \leq (q_1')^K \|L_*\|_F + \frac{C_\tau'' q_2'}{1-q_1'}\sqrt{\frac{pr\log^3 p}{m}}. \tag{5.9}$$

where q_1' and q_2' have been defined in Theorem 5.8.

Hence, under the assumptions in Theorem 5.9, in order to achieve ϵ-accuracy in the estimation of L_* in terms of Frobenius norm, AP-ROM requires $K = \mathcal{O}(\log(\frac{\|L_*\|_2}{\epsilon}))$ iterations. From Theorem 5.8

and Corollary 5.10, we observe that the sample-complexity of AP-ROM (i.e., the number of samples m to achieve a given accuracy) slightly increases as $m = \mathcal{O}\left(pr^3 \log^4 p \log(\frac{1}{\epsilon})\right)$.

5.3.3 Improving Running Time

The above analysis of AP-ROM shows that instead of using exact rank-r projections (as in EP-ROM), one can use instead tail and head approximate projection which is implemented by the BK-SVD method of (Musco and Musco, 2015). The running time for this method is given by $\widetilde{\mathcal{O}}(p^2 r)$ if $r \ll p$. While the running time of the projection step is gap-independent, the calculation of the *gradient* (i.e., the input to the head projection method $\mathcal{H}$) is itself the major bottleneck. In essence, this is related to the calculation of the adjoint operator, $\mathcal{A}^*(d) = \sum_{i=1}^m d^{(i)} x_i x_i^T$, which requires $\mathcal{O}(p^2)$ operations for each sample. Coupled with the sample-complexity of $m = \Omega(pr^3)$, this means that the running time per-iteration scaled as $\Omega(p^3 r^3)$, which overshadows any gains achieved during the projection step (please see Section 5.6.3 in the appendix for more discussion).

To address this challenge, we propose a modified version of BK-SVD for head approximate projection which uses the special rank-one structures involved in the calculation of the gradients. We call this method *Modified BK-SVD*, or MBK-SVD. The basic idea is to *implicitly* evaluate each Krylov-subspace iteration within BK-SVD, and avoid any explicit calculation of the adjoint operator $\mathcal{A}^*$ applied to the current estimate. Due to space constraints, the pseudocode as well as the running time analysis of MBK-SVD is deferred to the appendix. We have:

Theorem 5.11. *AP-ROM (with modified BK-SVD) runs in time* $K = \mathcal{O}\left(p^2 r^4 \log^2(\frac{1}{\epsilon}) \text{polylog}(p)\right)$.

5.3.4 Achieving Optimal Sample Complexity

In the previous section, we saw that both our proposed algorithms result in suboptimal sample complexity by logarithmic factors, primarily because their analysis requires a set of fresh samples in each iteration. In this section, we propose a third algorithm that removes this assumption and achieves asymptotically optimal sample complexity, i.e., $m = \mathcal{O}(pr)$. Referring back to Table 5.1, we observe that convex methods (Chen et al., 2015; Cai and Zhang, 2015) exhibits the same sample

Algorithm 5.3

```
Initialization: $L_0 \leftarrow 0$, $t \leftarrow 0$
while $t \leq K$ do
    $L_{t+1} = \mathcal{T}_{\kappa r}\left(L_t - \frac{\eta_t}{m}\mathcal{B}^*\mathrm{sgn}(\mathcal{B}(L_t) - y)\right)$
    $t \leftarrow t + 1$
end while
Return: $\widehat{L} = L_K$
```

complexity, but are very slow. However, we show that our new algorithm enjoys a fast running time.

To overcome the issue of fresh samples, our key intuition is to replace squared loss with the absolute deviation loss function (i.e., ℓ_1-loss), and the CU-RIP with the RIP(ℓ_1, ℓ_2). For simplicity, in this section we ignore noise, while noting that our analysis seamlessly carries over to the noisy case with a somewhat more tedious (but straightforward) extension.

We introduce a (slightly) different observation model: $y' = \mathcal{B}(L_*)$, where $\mathcal{B} : \mathbb{R}^{p\times p} \to \mathbb{R}^m$ denotes a linear operator such that $\mathcal{B}(L)_i = \mathcal{A}(L)_{2i} - \mathcal{A}(L)_{2i-1}$, with $\mathcal{A}(.)_i$ denotes the i^{th} entry of vector $\mathcal{A}(.)$ as defined in observation model (5.3). It is easy to see that $\mathcal{B}$ can be implemented by doubling the number of training samples. The reason why $\mathcal{B}$ is constructed in this way is inspired by (Chen et al., 2015), where the authors have shown that $\mathcal{B}$ satisfies RIP(ℓ_1, ℓ_2) if the number of samples $m = \mathcal{O}(pr)$. Based on the above model, we consider the following risk minimization problem:

$$\min_{L\in\mathbb{R}^{p\times p}} \quad F(L) = \frac{1}{m}\|y - \mathcal{B}(L)\|_1 \quad \text{s.t.} \quad \mathrm{rank}(L) \leq r. \tag{5.10}$$

To solve this problem, we propose an approximate projected *sub*-gradient algorithm, displayed in pseudocode as Algorithm 5.3.

Compared to the previous algorithms, Alg. 5.3 has three major differences. First, it has only one approximation operator – an approximate tail projection $\mathcal{T}$ – which projects its argument onto a *larger* set of matrices with rank κr for $\kappa > 1$. To implement this operator, we use the modified BK-SVD algorithm similar to the discussion above. The idea of projecting onto a larger space was first proposed by (Jain et al., 2014), and subsequently has been extended for approximate tail projections by (Soltani and Hegde, 2017b). In that work, we showed that this idea essentially removes the need for an inner "head" projection. Second, the objective function in (5.10) is not differentiable; hence, we have to use a sub-gradients which, for our case, is given by $\partial F(L) = \frac{1}{m}\mathcal{B}^*\text{sgn}(\mathcal{B}(L_t) - y)$. Third, Algorithm 5.3 requires a time-varying step-size η_t specified below (Here, $\mathcal{B}^*$ denotes the adjoint operator of $\mathcal{B}$).

We now prove that with sufficiently many samples, Algorithm 5.3 converges linearly, and at termination, provides an accurate estimate of the true low-rank matrix. This proof complements the theoretical analysis of (Chen et al., 2015; Cai and Zhang, 2015) with a simple and easily implementable algorithm with provably fast convergence guarantees. For establishing the proof, we need the following Lemma, proved in (Soltani and Hegde, 2017b) and adapted to our notation.

Lemma 5.12. *Let $\kappa > (1 + \frac{1}{1-\varepsilon})$. For any matrices $L, L_* \in \mathbb{R}^{p\times p}$ with rank$(L_*) = r$, we have*

$$\|\mathcal{T}_{\kappa r}(L) - L_*\|_F^2 \leq \left(1 + \frac{2}{\sqrt{1-\varepsilon}\sqrt{\kappa - 1}}\right)\|L - L_*\|_F^2,$$

where $\mathcal{T} : \mathbb{R}^{p\times p} \to \mathbb{U}_{\kappa r}$ denotes the approximate tail projection defined in Definition 5.2 and $\varepsilon > 0$ is the corresponding approximation ratio.

This lemma says that the near-contraction factor $\nu = 1 + \frac{2}{\sqrt{1-\varepsilon}\sqrt{\kappa-1}}$ can be made arbitrary close to 1, provided that we increase the parameter κ accordingly. We now establish the linear convergence of Algorithm 5.3:

Theorem 5.13. *Suppose that the linear map $\mathcal{B}$ is constructed such that it satisfies RIP(ℓ_1, ℓ_2) property with constants α and β in each iteration. Set $\kappa > 1 + \max\left\{\frac{4\left((\frac{\alpha^2}{\beta^2}) - 1\right)^2}{1-\varepsilon}, \frac{1}{1-\varepsilon}\right\}$. Choose step size as $\eta_t = \frac{\|\mathcal{B}(L_t) - y\|_1}{\beta^2}$. Then, Algorithm 5.3 produces a sequence of estimates L_t for $t = 1, 2\ldots$ such that*

$$\|L_{t+1} - L_*\|_2 \leq \lambda\|L_t - L_*\|_2. \tag{5.11}$$

where $\lambda = \sqrt{\nu(1 - \frac{\alpha^2}{\beta^2})}$.

The RIP assumption for $\mathcal{B}$ is justified by Proposition 1 in (Chen et al., 2015), and the fact that in each iteration, the input matrix of $\mathcal{B}$ is a matrix with rank at most equals to $2\kappa r + r^*$ (see the proof). In addition, the running time of Algorithm 5.3 scales as $\mathcal{O}(p^2r^2\log(\text{p})\log(\frac{1}{\varepsilon}))$ as a result of implementing $\mathcal{T}$ with MBK-SVD.

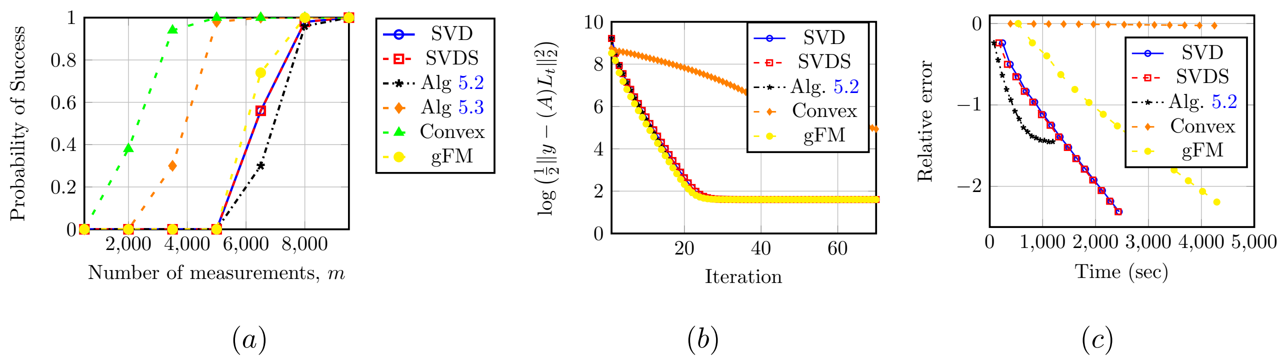


Figure 5.2: Comparison of algorithms. (a) Phase transition plot with $p = 100$. (b) Evolution of the objective function versus number of iterations with $p = 100$, $m = 8500$, and noise level $\sigma = 0.1$. (c) Running time of the algorithm with $p = 1000$ and $m = 75000$.

5.4 Experimental Results

We illustrate some experiments to support our proposed algorithms. We compare EP-ROM and AP-ROM with convex (nuclear norm) minimization as well as the gFM algorithm of (Lin and Ye, 2016). To solve the nuclear norm minimization, we use FASTA (Goldstein et al., 2014, 2015) which efficiently implements an accelerated proximal sub-gradient method. For AP-ROM, we consider our proposed modified BK-SVD method (MBK-SVD). In addition, SVD and SVDS denote the projection step being used in EP-ROM. In all the experiments, we generate a low-rank matrix, $L_* = UU^T$, such that $\mathbb{U} \in \mathbb{R}^{p\times r}$ with $r = 5$ where the entries of U is randomly chosen according to the standard normal distribution.

Figures 5.2(a) and 5.2(b) show the phase transition of successful recovery as well as the evolution of the objective function, $\frac{1}{2}\|y - \mathcal{A}(L_t)\|_2^2$ versus the iteration count t for five algorithms. In figure 5.2(a), we have used 50 Monte Carlo trials and the phase transition plot is generated based on the empirical probability of success; here, success is when the relative error between $\hat{L}$ (the estimate of L_*) and the ground truth L_* (measured in terms of spectral norm) is less than 0.05. For solving convex nuclear norm minimization using FASTA, we set the Lagrangian parameter, μ i.e., $\mu\|L\|_* + \frac{1}{2}\|y - \mathcal{A}L\|_F^2$ via a grid search. In Figure 5.2(a), there is no additive noise. As we can see in this Figure, the phase transition for the convex method and Alg 5.3 has comparable phase transition as predicted by theory, and they are slightly better than those for non-convex algorithms, which is consistent with known theoretical results. However, the convex method is *improper*, i.e., the rank of $\hat{L}$ is much higher than the target rank. In Figure 5.2(b) we consider an additive standard normal noise with standard deviation equal to 0.1, and average over 10 Monte Carlo trials. As illustrated in this plot, all non-convex algorithm have much better performance in decreasing the objective function compared to convex method.

Finally, in Figure 5.2(c), we compare the algorithms in the high-dimensional regime where $p = 1000$, $m = 75000$, and $r = 5$ in terms of running time. We let all the algorithms run 15 iterations, and then compute the CPU time in seconds for each of them. The y-axis denotes the logarithm of relative error in spectral norm and we report averages over 10 Monte Carlo trials. As we can see, convex methods are the slowest (as expected); the non-convex methods are comparable to each other, while MBK-SVD is the fastest. This plot verifies that our modified head approximate projection routine is faster than other non-convex methods, which makes it a promising approach for high-dimensional matrix recovery applications.

5.5 Conclusion

It seems plausible that the matrix-based techniques of this chapter can be extended to learn networks with similar polynomial-like activation functions (such as the squared ReLU). Moreover, similar algorithms can be plausibly used to train multi-layer networks using a greedy (layer-by-layer) learning strategy. Finally, it will be interesting to integrate our methods with practical approaches such as stochastic gradient descent (SGD).

5.6 Appendix. Overview

Below, the expression $C + D$ for sets C and D refers to the *Minkowski* sum of two sets, defined as $C + D = \{c + d \mid c \in C,\ d \in D\}$ for given sets C and D. For any given set $\mathcal{U} \in \mathbb{R}^{p\times p}$ we denote the orthogonal projection onto $\mathcal{U}$ by $\mathcal{P}_{\mathcal{U}}$. Also, $\mathcal{M}(\mathbb{U}_r)$ denotes the set of vectors associated with

$\mathbb{U}_r$, the set of all rank-r matrix subspaces. In addition, we denote the $\{p,q\}^{th}$ entry of matrix B and p^{th} entry of vector x as $B^{(pq)}$ and $x^{(p)}$, respectively.

Proof of Theorem 5.5. Here we show that the error in the estimate of L_* decreases in one iteration. Define b as follows:

$$\begin{aligned} b &= L_t - \frac{1}{2m}\sum_{i=1}^{2m}\left(x_i L_t x_i^T - y_i\right) x_i x_i^T - (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t) - \frac{1}{2}\bar{y})I \\ &= L_t - \frac{1}{2m}\mathcal{A}^*(\mathcal{A}(L_t) - y) + (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t) - \frac{1}{2}\bar{y})I. \end{aligned}$$

Thus, we have:

$$\begin{aligned} \|L_{t+1} - L_*\|_2 &\leq \|L_{t+1} - b\|_2 + \|b - L_*\|_2 \overset{a_1}{\leq} 2\|b - L_*\|_2 \\ &\leq 2\|L_t - L_* - \frac{1}{2m}\mathcal{A}^*(\mathcal{A}(L_t) - y) + (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t) - \frac{1}{2}\bar{y})I\|_2 \\ &\overset{a_2}{\leq} 2\|L_t - L_* - \frac{1}{2m}\mathcal{A}^*\mathcal{A}(L_t - L_*) + (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t) - \frac{1}{2}\bar{y})I\|_2 + \frac{1}{m}\|\mathcal{P}_J\mathcal{A}^*e\|_2 \\ &\overset{a_3}{\leq} 2\rho\|L_t - L_*\|_2 + \frac{1}{m}(|\mathbf{1}^Te| + \|\mathcal{A}^*e\|_2), \end{aligned} \tag{5.12}$$

above, a_1 holds since L_{t+1} is generated by projecting onto the set of matrices with rank r, and by definition of J, L_{t+1} also has the minimum Euclidean distance to b over all matrices with rank r; a_2 holds by the definition of y from (5.3) and the triangle inequality; finally, a_3 holds by the CU-RIP assumption in the theorem. By Letting $0 < q = 2\rho < 1$, the proof is completed. □

Proof of Corollary 5.8. First, we note that by our assumption on η in Theorem 5.5, $q_1 < 1$. Since EP-ROM uses fresh samples in each iteration, $L_t - L_*$ is independent of the sensing vectors, x_i's for all t. On the other hand, from Theorem 5.7, the CU-RIP holds with probability $1-\xi$. As a result, by a union bound over the K iterations of the algorithm, the CU-RIP holds after K iterations with probability at least $1 - K\xi$. By recursively applying inequality (5.5) (with zero initialization) and applying Theorem 5.6, we obtain the claimed result. □

Proof of Theorem 5.9. Assume that $V := V_t = \mathcal{H}\left(\mathcal{A}^*(\mathcal{A}(L_t) - y) - Tr(L_t - \bar{y})I\right)$ and $Y \in \mathcal{M}(\mathbb{U}_{2r})$ such that $L_t - L_* \in Y$ and. Also, define

$$\begin{aligned} b' &= L_t - \mathcal{H}\left(\frac{1}{m}\sum_{i=1}^{2m}\left(x_i L_t x_i^T - y_i\right) x_i x_i^T - (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t) - \frac{1}{2}\bar{y})I\right) \\ &= L_t - \frac{1}{2m}\mathcal{H}\left(\mathcal{A}^*(\mathcal{A}(L_t) - y) - (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t) - \frac{1}{2}\bar{y})I\right). \end{aligned}$$

Furthermore, by definition of approximate tail projection, $L_t \in \mathcal{M}(\mathbb{U}_r)$. Now, we have:

$$\begin{aligned} \|L_{t+1} - L_*\|_F &= \|L_* - \mathcal{T}(b')\|_F \\ &\leq \|L^* - b'\|_F + \|b' - \mathcal{T}(b')\|_F \\ &\overset{a_1}{\leq} (2+\varepsilon)\|b' - L_*\|_F \\ &= (2+\varepsilon)\left\|L_t - L_* - \mathcal{H}\left(\frac{1}{2m}\mathcal{A}^*(\mathcal{A}(L_t) - y) - (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t) - \frac{1}{2}\bar{y})I\right)\right\|_F \\ &\overset{a_2}{=} (2+\varepsilon)\left\|L_t - L_* - \mathcal{P}_V\left(\frac{1}{2m}\mathcal{A}^*(\mathcal{A}(L_t) - y) - (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t) - \frac{1}{2}\bar{y})I\right)\right\|_F, \end{aligned}$$

where a_1 is implied by the triangle inequality and the definition of approximate tail projection, and inequality a_2 holds by the definition of approximate head projection. Next, we have:

$$\begin{aligned} &\|L_{t+1} - L_*\|_F \\ &\overset{a_3}{\leq} (2+\varepsilon)\left\|\mathcal{P}_V(L_t - L_*) + \mathcal{P}_{V^\perp}(L_t - L_*) - \mathcal{P}_V\left(\frac{1}{2m}\mathcal{A}^*(\mathcal{A}(L_t) - y) - (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t) - \frac{1}{2}\bar{y})I\right)\right\|_F \\ &\overset{a_4}{\leq} (2+\varepsilon)\left\|\mathcal{P}_V(L_t - L_*) - \mathcal{P}_V\left(\frac{1}{2m}\mathcal{A}^*\mathcal{A}(L_t - L_*) - (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t) - \frac{1}{2}\bar{y})I\right)\right\|_F \\ &\qquad\qquad + (2+\varepsilon)\left\|\mathcal{P}_{V^\perp}(L_t - L_*)\right\|_F + \frac{2+\varepsilon}{2m}\left\|\mathcal{P}_V\mathcal{A}^* e\right\|_F \\ &\overset{a_5}{\leq} (2+\varepsilon)\left\|\mathcal{P}_{V+Y}\left(L_t - L_* - (\frac{1}{2m}\mathcal{A}^*\mathcal{A}(L_t - L_*) - \frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t - L_*)I)\right)\right\|_F \\ &\qquad + (2+\varepsilon)\left\|\mathcal{P}_{V^\perp}(L_t - L_*)\right\|_F + \frac{2+\varepsilon}{2m}\left(|\mathbf{1}^T e| + \left\|\mathcal{P}_V\mathcal{A}^* e\right\|_F\right), \end{aligned} \tag{5.13}$$

where a_3 follows by decomposing the residual $L_t - L_*$ on the two subspaces V and $V^\perp$, and a_4 is due to the triangle inequality, the fact that $L_t - L_* \in Y$, and $V \subseteq V + Y$.

Now, we need to bound the three terms in (5.13). The third and fourth terms can be bounded by using Theorem 5.6 which we will use in Corollary 5.10. For the first term, we have:

$$
\begin{aligned}
&(2+\varepsilon)\left\|\mathcal{P}_{V+Y}\left(L_t - L_* - (\frac{1}{2m}\mathcal{A}^*\mathcal{A}(L_t-L_*) - \frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t-L_*)I)\right)\right\|_F \\
&\qquad\qquad \overset{a_1}{\leq} (2+\varepsilon)\left\|L_t - L_* - (\frac{1}{2m}\mathcal{A}^*\mathcal{A}(L_t-L_*) - \frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t-L_*)I)\right\|_F \\
&\qquad\qquad \overset{a_2}{\leq} (2+\varepsilon)\rho'\left\|L_t - L_*\right\|_F, \qquad (5.14)
\end{aligned}
$$

above, a_1 holds by the properties of the Frobenius and spectral norm, and a_2 is due to the CU-RIP assumption in the theorem similar to (5.12). To bound the second term in (5.13), $(1+c_{\mathcal{T}})\left\|\mathcal{P}_{V^\perp}(L_t - L_*)\right\|_F$, we give upper and lower bounds for $\left\|\mathcal{P}_V\left(\frac{1}{2m}\mathcal{A}^*(\mathcal{A}(L_t)-y) - (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t)-\bar{y})I\right)\right\|_F$ as follows:

$$
\begin{aligned}
&\left\|\mathcal{P}_V\left(\frac{1}{2m}\mathcal{A}^*(\mathcal{A}(L_t)-y) - (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t)-\bar{y})I\right)\right\|_F \\
&\quad \overset{a_1}{\geq} (1-\varepsilon)\left\|\mathcal{P}_Y\left(\frac{1}{2m}\mathcal{A}^*(\mathcal{A}(L_t)-y) - (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t)-\bar{y})I\right)\right\|_F \\
&\quad \overset{a_2}{\geq} (1-\varepsilon)\left\|\mathcal{P}_Y\left(\frac{1}{2m}\mathcal{A}^*\mathcal{A}(L_t-L_*) - \frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t-L_*)I\right)\right\|_F - \frac{1-\varepsilon}{2m}|\mathbf{1}^Te| - \frac{1-\varepsilon}{2m}\left\|\mathcal{P}_V\mathcal{A}^*e\right\|_F \\
&\quad \overset{a_3}{\geq} (1-\varepsilon)(1-\rho')\left\|L_t - L_*\right\|_F - \frac{1-\varepsilon}{2m}\left(|\mathbf{1}^Te| + \left\|\mathcal{P}_V\mathcal{A}^*e\right\|_F\right), \qquad (5.15)
\end{aligned}
$$

above, a_1 holds by the definition of approximate head projection, a_2 is followed by triangle inequality, a_3 is due to Corollary 5.19, and finally a_4 holds due to the fact that $\operatorname{rank}(L_t - L_*) \leq 2r$. For the upper bound, we have:

$$
\begin{aligned}
&\left\|\mathcal{P}_V\left(\frac{1}{2m}\mathcal{A}^*(\mathcal{A}(L_t)-y) - (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t)-\bar{y})I\right)\right\|_F \\
&\quad \overset{a_1}{\leq} \left\|\mathcal{P}_{V+Y}\left(\frac{1}{2m}\mathcal{A}^*\mathcal{A}(L_t-L_*) - \frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t-L_*)I\right) - \mathcal{P}_{V+Y}(L_t-L_*)\right\|_F \\
&\qquad\qquad + \left\|\mathcal{P}_V(L_t-L_*)\right\|_F + \frac{1}{2m}\left(|\mathbf{1}^Te| + \left\|\mathcal{P}_V\mathcal{A}^*e\right\|_F\right) \\
&\quad \overset{a_2}{\leq} \left\|L_t - L_* - \frac{1}{2m}\mathcal{A}^*(\mathcal{A}(L_t)-y) + \frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t-L_*)I\right\|_F + \left\|\mathcal{P}_V(L_t-L_*)\right\|_F + \frac{1}{2m}\left(|\mathbf{1}^Te| + \left\|\mathcal{P}_V\mathcal{A}^*e+\right\|_F\right) \\
&\quad \overset{a_3}{\leq} \rho'\left\|L_t - L_*\right\|_F + \left\|\mathcal{P}_V(L_t-L_*)\right\|_F + \frac{1}{2m}\left(|\mathbf{1}^Te| + \left\|\mathcal{P}_V\mathcal{A}^*e\right\|_F\right), \qquad (5.16)
\end{aligned}
$$

above, a_1 holds by triangle inequality and the fact that projection onto the extended subspace $V+Y$ $(V \subseteq V+Y)$ does not decrease the Frobenius norm, a_2 is due to the inequality $\|AB\|_F \leq \|A\|_2\|B\|_F$, and finally a_3 is followed by CU-RIP assumption and the fact that $\operatorname{rank}(L_t - L_*) \leq 2r$. Putting together (5.15) and (5.16), we obtain:

$$\left\|\mathcal{P}_V(L_t - L_*)\right\|_F \geq \left((1-\varepsilon)(1-\rho') - \rho'\right)\left\|L_t - L_*\right\|_F - \frac{2-\varepsilon}{2m}\left(|\mathbf{1}^T e| + \left\|\mathcal{P}_V\mathcal{A}^* e\right\|_F\right). \tag{5.17}$$

By the Pythagoras theorem, we know $\left\|\mathcal{P}_V(L_t - L_*)\right\|_F^2 + \left\|\mathcal{P}_{V^\perp}(L_t - L_*)\right\|_F^2 = \left\|L_t - L_*\right\|_F^2$, and hence we can bound the second term in (5.13). To use this fact, we apply (14) in (Hegde et al., 2016) which results:

$$(2+\varepsilon)\left\|\mathcal{P}_{V^\perp}(L_t - L_*)\right\|_F \leq (2+\varepsilon)\sqrt{1-\phi^2}\left\|L_t - L_*\right\|_F + \frac{\phi(2-\varepsilon)(2+\varepsilon)}{2m\sqrt{1-\phi^2}}\left(|\mathbf{1}^T e| + \left\|\mathcal{P}_V\mathcal{A}^* e\right\|_F\right), \tag{5.18}$$

where $\phi = (1-\varepsilon)(1-\rho') - \rho'$. Putting all the bounds in (5.14), and (5.18) altogether, we obtain:

$$\begin{aligned}\|L_{t+1} - L_*\|_F &\leq \left((2+\varepsilon)\rho' + (2+\varepsilon)\sqrt{1-\phi^2}\right)\left\|L_t - L_*\right\|_F \\ &\quad + \frac{\sqrt{r}}{2m}\left(2-\varepsilon + \frac{\phi(2-\varepsilon)(2+\varepsilon)}{\sqrt{1-\phi^2}}\right)\left(|\mathbf{1}^T e| + \left\|\mathcal{A}^* e\right\|_2\right) \\ &= q_1'\left\|L_t - L_*\right\|_F + q_2'\left(|\mathbf{1}^T e| + \left\|\mathcal{A}^* e\right\|_2\right).\end{aligned} \tag{5.19}$$

We choose $q_1' = (2+\varepsilon)(\rho' + \sqrt{1-\phi^2})$, and $q_2' = \frac{\sqrt{r}}{2m}\left(2-\varepsilon + \frac{\phi(2-\varepsilon)(2+\varepsilon)}{\sqrt{1-\phi^2}}\right)$. Now in order to have convergence, we have to make sure that $0 < \phi < 1$ and $q_1' < 1$. These conditions are achieved if we let choose m sufficiently large such that $\rho' < \frac{1}{2+\varepsilon} - \sqrt{1-\phi^2}$. The completes the proof.

□

Proof of Corollary 5.10. The proof is similar to Corollary 5.8, and follows by using CU-RIP over K iterations which is guaranteed to be held by using fresh samples in each iteration. Finally, by using induction, zero initialization, and Theorem 5.6, we obtain the claimed result in the corollary. □

5.6.1 Appendix A. Supporting Lemmas and Theorems

For proving the lemmas in Section 5.6.2, we include some definitions and well-known Bernstein type inequalities for random variables and matrices. We restate these inequalities for completeness. Please see (Vershynin, 2010; Tropp, 2015) for more details.

Definition 5.14. *(Subgaussian and Subexponential random variables.) A random variable X is called subgaussian if it satisfies the following:*

$$\mathbb{E}\exp\left(\frac{cX^2}{\|X\|_{\psi_2}^2}\right) \leq 2,$$

where $\|X\|_{\psi_2}$ denotes the ψ_2-norm which is defined as follows:

$$\|X\|_{\psi_2} = \sup_{p\geq 1}\frac{1}{\sqrt{p}}(\mathbb{E}|X|^p)^{\frac{1}{p}}.$$

Furthermore, a random variable X is subexponential if it satisfies the following relation:

$$\mathbb{E}\exp\left(\frac{cX}{\|X\|_{\psi_1}}\right) \leq 2,$$

where $\|X\|_{\psi_1}$ denotes the ψ_1-norm, defined as follows:

$$\|X\|_{\psi_1} = \sup_{p\geq 1}\frac{1}{p}(\mathbb{E}|X|^p)^{\frac{1}{p}}.$$

In the above expressions, $c > 0$ is an absolute constant.

We note that the product of two standard normal random variables which is a χ^2 random variable satisfies the subexponential random variable definition with ψ_1-norm equals to 2.

Lemma 5.15. *(Bernstein-type inequality for random variables). Let $X_1, X_2, \ldots, X_n$ be independent sub-exponential random variables with zero-mean. Also, assume that $K = \max_i \|X_i\|_{\psi_1}$. Then, for any vector $a \in \mathbb{R}^n$ and every $t \geq 0$, we have:*

$$\mathbb{P}(|\Sigma_i a_i X_i| \geq t) \leq 2\exp\left(-c\min\left\{\frac{t^2}{K^2\|a\|_2^2}, \frac{t}{K\|a\|_\infty}\right\}\right).$$

where $c > 0$ is an absolute constant.

Lemma 5.16. *(Bernstein-type inequality for symmetric random matrices). Consider a sequence of symmetric and random independent identical distributed matrices $\{S_i\}_{i=1}^m$ with dimension $p \times p$. Also, assume that $\|S_i - \mathbb{E}S_i\|_2 \leq R$ for $i = 1, \ldots, m$. Then for all $t \geq 0$,*

$$\mathbb{P}\left(\left\|\frac{1}{m}\sum_{i=1}^m S_i - \mathbb{E}S_i\right\|_2 \geq t\right) \leq 2p\exp\left(\frac{-mt^2}{\sigma + Rt/3}\right),$$

where $\sigma = \|\mathbb{E}\,(S - \mathbb{E}S)^2\,\|_2$ and S is a independent copy of S_i's.

5.6.2 Appendix B. Verification Of CU-RIP(ρ)

Before verifying CU-RIP, we need the following lemmas. In the first lemma, we show that $\bar{y} = \frac{1}{m}\sum_{i=1}^{m} y_i$ is concentrated around its mean with high probability.

Lemma 5.17 (Concentration of $\bar{y}$). *Let $\mathcal{A} : \mathbb{R}^{p\times p} \to \mathbb{R}^m$ be a linear operator defined as (5.3) and $L \in \mathbb{R}^{p\times p}$ be some symmetric matrix. Then with probability at least $1-\xi_1$, we have for some constant $C > 0$:*

$$|\frac{1}{m}\mathbf{1}^T\mathcal{A}(L) - Tr(L)| \le C\sqrt{\frac{1}{m}\log(\frac{p}{\xi_1})}\|L\|_2. \tag{5.20}$$

Proof. In all the following expressions, $c_l > 0$ for $l = 1, \ldots, 4$ are absolute constants. We start by noting that:

$$\mathbb{E}\mathcal{A}(L) = \mathbb{E}Tr(x_i x_i^T L) = Tr(L)$$

where we have used the fact that $x_i \overset{i.i.d}{\sim} \mathcal{N}(0, I)$. We have for all $t > 0$:

$$\begin{aligned}
\mathbb{P}\left(|\frac{1}{m}\mathbf{1}^T\mathcal{A}(L) - Tr(L)| \ge t\right) &= \mathbb{P}\left(\Big|\frac{1}{m}\sum_{i=1}^{m}\langle x_i x_i^T, L\rangle - Tr(L)\Big| \ge t\right) \\
&= \mathbb{P}\left(\Big|\frac{1}{m}\sum_{i=1}^{m}\sum_{u,v}(x_i^u x_i^v L^{uv}) - Tr(L)\Big| \ge t\right) \\
&= \mathbb{P}\left(\Big|\sum_{u}\frac{1}{m}\sum_{i=1}^{m}((x_i^u)^2 L^{uu} - L^{uu}) + \sum_{u\neq v}\frac{1}{m}\sum_{i=1}^{m}(x_i^u x_i^v L^{uv})\Big| \ge t\right).
\end{aligned} \tag{5.21}$$

Now we bound two probabilities. First, $\forall\, t_1 \ge 0$:

$$\mathbb{P}\left(\Big|\sum_{u}\frac{1}{m}\sum_{i=1}^{m}((x_i^u)^2 L^{uu} - L^{uu})\Big| \ge t_1\right) \overset{a_1}{\le} p\exp\left(-c_1\frac{mt_1^2}{\|L\|_2^2}\right),$$

where a_1 is due to the union bound over p diagonal variables and by the fact that $(x_i^u)^2$ is a χ^2 random variable with mean 1 and $\|\chi^2\|_{\psi_1} = 2$; as a result, we can use the scalar version of Bernstein inequality in (5.15). Now by choosing $t_1 \ge c_2\|L\|_2\sqrt{\frac{\log(\frac{p}{\xi_1'})}{m}}$, with probability at least $1-\xi_1'$, we have:

$$\Big|\sum_{u}\frac{1}{m}\sum_{i=1}^{m}((x_i^u)^2 L^{uu} - L^{uu})\Big| \le \sqrt{\frac{c_2}{m}\log(\frac{p}{\xi_1'})}\|L\|_2. \tag{5.22}$$

Second, let $k = \max_{u \neq v} (L^{uv})^2$. Thus, $\forall t_2 \geq 0$,

$$\mathbb{P}\left(\Big| \sum_{u \neq v} \frac{1}{m} \sum_{i=1}^{m} (x_i^u x_i^v L^{uv}) \Big| \geq t_2 \right) \overset{a_2}{\leq} p^2 \exp\left(-c_2 \frac{m t_2^2}{k^2} \right),$$

where a_2 holds by a union bound over $p^2 - p$ off-diagonal variables, and the fact that $x_i^u x_i^v$ is a zero mean subexponential random variable. Hence, we can again use the scalar version of Bernstein inequality in (5.15). By choosing $t_2 \geq \sqrt{\frac{c_3}{m} \log(\frac{p}{\xi_1''})}$, with probability at least $1 - \xi_1''$, we have:

$$\Big| \sum_{u \neq v} \frac{1}{m} \sum_{i=1}^{m} (x_i^u x_i^v L^{uv}) \Big| \leq \sqrt{\frac{c_3}{m} \log(\frac{p}{\xi_1''})}. \tag{5.23}$$

Now from (5.21), (5.22), and (5.23) and by choosing $t = t_1 + t_2$ with probability at least $1 - \xi_1$ where $\xi_1 = \xi_1' + \xi_1''$, we obtain:

$$\mathbb{P}\left(\Big| \frac{1}{m} \mathbf{1}^T \mathcal{A}(L) - Tr(L) \Big| \geq t \right) \leq \sqrt{\frac{c_4}{m} \log(\frac{p}{\xi_1'})} \|L\|_2.$$

which proves the stated claim. $\square$

In the next lemma, we show that $\nabla F(M) = \frac{1}{m} \mathcal{A}^* \mathcal{A}(M)$ is concentrated around its mean (in terms of spectral norm) with high probability.

Lemma 5.18 (Concentration of $\frac{1}{m} \mathcal{A}^* \mathcal{A}(M)$). *Let $M \in \mathbb{R}^{p \times p}$ be a fixed matrix with rank r and let $S_i = x_i x_i^T (M) x_i x_i^T$ for $i = 1, \ldots, m$. Consider the linear operator $\mathcal{A}$ in model (5.3) independent of M. Then with probability at least $1 - \xi_2$, we have:*

$$\Big\| \frac{1}{m} \sum_{i=1}^{m} S_i - \mathbb{E} S_i \Big\|_2 \leq C' \sqrt{\frac{p r^2 \log^3 p}{m}} \log(\frac{p}{\xi_2}) \|M\|_2. \tag{5.24}$$

where $C' > 0$ is a constant.

Proof. In all the following expressions, $C_l > 0$ for $l = 1, \ldots, 11$ are absolute constants. First we note that by some calculations, one can show that

$$\mathbb{E}\left(\frac{1}{m} \mathcal{A}^* \mathcal{A}(M) \right) = \mathbb{E} S_i = 2(M) + Tr(M) I.$$

Our technique to establish the concentration of $\mathcal{A}^* \mathcal{A}(L_t - L_*)$ is based on the matrix Bernstein inequality. As stated in lemma (5.16), there should be a spectral bound on the summands,

$S_i = x_i x_i^T(M) x_i x_i^T$ for $i = 1, \dots, m$. Since the entries of a_i are Gaussian, the spectral norm is not absolutely bounded; hence, we cannot directly use the matrix Bernstein inequality. Inspired by (Zhong et al., 2015), we will use a *truncation* trick to make sure that the spectral norm of summands are bounded. Define the random variable $\widetilde{x_i}^{(j)}$ as follows:

$$\widetilde{x_i}^{(j)} = \begin{cases} x_i^{(j)}, & |x_i^{(j)}| \leq C_1\sqrt{\log mp} \\ 0, & otherwise, \end{cases} \tag{5.25}$$

where $x_i^{(j)}$ is the j^{th} entry of the random vector x_i. By this definition, we immediately have the following properties:

- $\mathbb{P}\left(x_i^{(j)} = \widetilde{x_i}^{(j)}\right) \geq 1 - \frac{1}{(mp)^{C_2}}$,
- $\mathbb{E}\left(\widetilde{x_i}^{(j)}\widetilde{x_i}^{(k)}\right) = 0$, for $j \neq k$,
- $\mathbb{E}\widetilde{x_i}^{(j)} = 0$ for $j = 1, \dots, p$,
- $\mathbb{E}\left(\widetilde{x_i}^{(j)}\right)^2 \leq \mathbb{E}\left(x_i^{(j)}\right)^2 = 1,$ for $j = 1, \dots, p$,

Let $\widetilde{S_i} = \widetilde{x_i}\widetilde{x_i}^T M \widetilde{x_i}\widetilde{x_i}^T$ for $i = 1, \dots, m$. We need to bound parameters R and σ in the matrix Bernstein inequality. Denote the SVD of M by $M = U_M \Sigma V_M^T$. Since x_i is a normal random vector, it is rotationally invariant. As a result, w.l.o.g., we can assume that $U_M = [e_1, e_2, \dots, e_r]$ and $V_M = [e_1, e_2, \dots, e_r]$ as long as the random vector x_i is independent of M. Here, e_j denotes the j^{th} canonical basis vector in $\mathbb{R}^p$. To make sure this happens, we use m fresh samples of x_i's in each iteration of the algorithm.

Now, we have for each i:

$$\begin{aligned}
\|\widetilde{x_i}\widetilde{x_i}^T M \widetilde{x_i}\widetilde{x_i}^T\|_2 &= \|\widetilde{x_i}\widetilde{x_i}^T U_M \Sigma V_M^T \widetilde{x_i}\widetilde{x_i}^T\|_2 \\
&\leq |\widetilde{x_i}^T U_M \Sigma V_M^T \widetilde{x_i}| \|\widetilde{x_i}\widetilde{x_i}^T\|_2 \\
&\leq \|U_M^T \widetilde{x_i}\|_2 \|V_M^T \widetilde{x_i}\|_2 \|\widetilde{x_i}\|_2^2 \|M\|_2 \\
&\overset{a_1}{\leq} pr\|\widetilde{x_i}\|_\infty^4 \|M\|_2 \\
&\overset{a_2}{\leq} C_3 pr \log^2(mp)\|M\|_2,
\end{aligned}$$

above, a_1 holds due to rotational invariance discussed above, and the relation between ℓ_2 and ℓ_∞ norms. Also, a_2 is due to applying bound in (5.25). Now, we can calculate R:

$$\|\widetilde{S}_i - \mathbb{E}\widetilde{S}_i\|_2 \leq \|\widetilde{S}_i\|_2 + \mathbb{E}\|\widetilde{S}_i\|_2 \leq 2\|\widetilde{S}_i\|_2 \leq C_4 pr \log^2(mp)\|M\|_2 = R,$$

where we have used both the triangle inequality and Jensen's inequality in the first inequality above. For σ, we define $\widetilde{S}$ as the truncated version of S, independent copy of S_i's. Hence:

$$\begin{aligned}
\sigma &= \left\|\mathbb{E}\widetilde{S}^2 - (\mathbb{E}\widetilde{S})^2\right\|_2 \\
&\overset{a_1}{\leq} \|\mathbb{E}\widetilde{S}^2\|_2 = \left\|\mathbb{E}\left(\widetilde{x}\widetilde{x}^T M\widetilde{x}\widetilde{x}^T\widetilde{x}\widetilde{x}^T M\widetilde{x}\widetilde{x}^T\right)\right\|_2 \\
&= \left\|\mathbb{E}\left(\|\widetilde{x}\|_2^2 \left(\widetilde{x}^T M\widetilde{x}\right)^2 \widetilde{x}\widetilde{x}^T\right)\right\|_2 \\
&\overset{a_1}{\leq} C_5 pr^2 \log^3(pm)\|M\|_2^2 \left\|\mathbb{E}\left(\widetilde{x}\widetilde{x}^T\right)\right\|_2 \\
&\overset{a_2}{\leq} C_5 pr^2 \log^3(pm)\|M\|_2^2,
\end{aligned}$$

where a_1 is followed as $(\mathbb{E}\widetilde{S})^2$ is a positive semidefinite matrix. In addition, a_2 holds due to the upper bound on $\left(\widetilde{x}^T M\widetilde{x}\right)^2 \|\widetilde{x}\|_2^2$:

$$\begin{aligned}
\left(\widetilde{x}^T M\widetilde{x}\right)^2 \|\widetilde{x}\|_2^2 &= \left(\widetilde{x}^T U_M \Sigma V_M^T \widetilde{x}\right)^2 \|\widetilde{x}\|_2^2 \\
&\leq \|U_M^T\widetilde{x}\|_2^2 \|V_M^T\widetilde{x}\|_2^2 \|M\|_2^2 \|\widetilde{x}\|_2^2 \\
&\leq pr^2 \|\widetilde{x}\|_\infty^6 \|M\|_2 \\
&\leq C_6 pr^2 \ \log^3(mp)\|M\|_2,
\end{aligned}$$

where we have again used the same argument of rotational invariance. Finally, a_2 holds due to the fact that $\mathbb{E}\left(\widetilde{x}_i\widetilde{x}_i^T\right) \preceq I$. Now, we can use the matrix Bernstein inequality for bounding $\left\|\frac{1}{m}\sum_{i=1}^m \widetilde{S}_i - \mathbb{E}\widetilde{S}_i\right\|_2$:

$$\begin{aligned}
\mathbb{P}\left(\left\|\frac{1}{m}\sum_{i=1}^m (\widetilde{S}_i - \mathbb{E}\widetilde{S}_i)\right\|_2 \geq t\right) &\leq 2p\exp\left(\frac{-mt^2}{\sigma + Rt/3}\right) \\
&\leq 2p\exp\left(\frac{-mt^2}{C_5 pr^2\log^3(pm)\|M\|_2^2 + C_4 pr\log^2(mp)\|M\|_2 t/3}\right) \\
&\overset{a_1}{\leq} 2p\exp\left(\frac{-mt^2}{C_7 pr^2\log^3(pm)\|M\|_2^2}\right), \qquad (5.26)
\end{aligned}$$

where a_1 holds by choosing constant C_7 to be sufficiently large. Now choose $t \geq \|M\|_2\sqrt{C_8\frac{pr^2\log^3(pm)}{m}\log(\frac{p}{\xi_2'})}$. Thus with probability at least $1-\xi_2'$, we have:

$$\Big\|\frac{1}{m}\sum_{i=1}^m(\widetilde{S}_i - \mathbb{E}\widetilde{S}_i)\Big\|_2 \leq \sqrt{C_8\frac{pr^2\log^3(pm)}{m}\log(\frac{p}{\xi_2'})}\|M\|_2,$$

This bound shows that by taking $m = \mathcal{O}(\frac{1}{\theta^2}pr^2\log^3 p\log(\frac{p}{\xi_2'}))$ for some $\theta > 0$, we can bound the LHS of the above inequality. Actually, this choice of m determines the sample complexity of EP-ROM and we will return back to this issue later. Recall that $\widetilde{S}_i$ includes the truncated random variables, i.e, $\widetilde{S}_i = \widetilde{x}_i\widetilde{x}_i^T M\widetilde{x}_i\widetilde{x}_i^T$. Also, $\mathbb{P}\left(x_i^{(j)} = \widetilde{x}_i^{(j)}\right) \geq 1 - \frac{1}{(mp)^{C_2}} \geq 1 - \frac{1}{(p)^{C_9}}$. Hence, we need to extend our result to the original x_i. By definition of $\widetilde{x}_i$ in (5.25) and choosing constant C_9 sufficiently large ($C_9 > 1$), we have $\mathbb{P}\left(\|S_i - \widetilde{S}_i\|_2 = 0\right) = \mathbb{P}\left(\|x_ix_i^T - \widetilde{x}_i\widetilde{x}_i^T\|_2 = 0\right) \geq 1 - \frac{1}{(p)^{C_{10}}}$. Here we have used the union bound over p^2 variables. Since we have m random matrices S_i, we need to take another union bound. As a result, with probability $1-\xi_2$ where $\xi_2 = \frac{1}{(p)^{C_{11}}}$, we have:

$$\Big\|\frac{1}{m}\sum_{i=1}^m(S_i - \mathbb{E}S_i)\Big\|_2 \leq \sqrt{C_8\frac{pr^2\log^3 p}{m}\log(\frac{p}{\xi_2})}\|M\|_2. \tag{5.27}$$

□

Proof of Theorem 5.7. Let L_t be the estimation of the algorithm in iteration t, and L_* denotes the ground truth matrix. Then for constants $C, C'C'' > 0$,

$$\begin{aligned}
&\big\|L_t - L_* - \frac{1}{2m}\mathcal{A}^*\mathcal{A}(L_t - L_*) + (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t) - \frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_*))I\big\|_2 \\
&\quad\overset{a_1}{\leq} \big\|\frac{1}{2m}\mathcal{A}^*\mathcal{A}(L_t - L_*) - (L_t - L_*) - \frac{1}{2}Tr(L_t - L_*)I - \frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t - L_*)I + \frac{1}{2}Tr(L_t - L_*)I\big\|_2 \\
&\quad\overset{a_2}{\leq} \big\|\frac{1}{2m}\mathcal{A}^*\mathcal{A}(L_t - L_*) - (L_t - L_*) - \frac{1}{2}Tr(L_t - L_*)I\big\|_2 + \big\|\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t - L_*)I - \frac{1}{2}Tr(L_t - L_*)I\big\|_2 \\
&\quad\overset{a_3}{\leq} \left(C'\sqrt{\frac{pr^2\log^3 p}{m}\log(\frac{p}{\xi_2})}\right)\big\|L_t - L_*\big\|_2 + C\sqrt{\frac{1}{m}\log(\frac{p}{\xi_1})}\big\|L_t - L_*\big\|_2 \\
&\quad\overset{a_4}{\leq} C''\delta\big\|L_t - L_*\big\|_2 = \rho\big\|L_t - L_*\big\|_2,
\end{aligned} \tag{5.28}$$

where a_1 is followed by adding and subtracting of $Tr(L_t - L_*)I$, inequality a_2 follows from triangle inequality, a_3 holds with probability $1-\xi_1-\xi_2 = 1-\xi$ by invoking Lemma 5.17, and Lemma 5.18 (by fixed matrix $L_t - L_*$ with rank $2r$), and finally a_4 is followed by choosing $m = \mathcal{O}\left(\frac{1}{\delta^2} pr^2 \log^3 p \log(\frac{p}{\xi})\right)$ for some $\delta > 0$. By choosing δ sufficiently small such that $0 < \rho < \frac{1}{2}$, the proof is completed. □

We also note that CU-RIP condition is also satisfied if we use the Frobenius norm instead of the spectral norm (in deriving inequality (5.28)) by increasing m by a factor r. In other words,

$$\left\|L_t - L_* - \frac{1}{2m}\mathcal{A}^*\mathcal{A}(L_t - L_*) + (\frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_t) - \frac{1}{2m}\mathbf{1}^T\mathcal{A}(L_*))I\right\|_F \leq \rho' \left\|L_t - L_*\right\|_F$$

with probability at least $1 - \xi$ provided that $m = \mathcal{O}\left(\frac{1}{\delta^2} pr^3 \log^3 p \log(\frac{p}{\xi})\right)$. Here $0 < \rho' < 1$.

Corollary 5.19. *From Theorem 5.7 we have the following conclusions:*

1. *Let U be the bases for the column space of fixed matrices L_1 and L_2 such that rank$(L_i) \leq r$ for $i = 1, 2$ and $\mathcal{P}_U$ is the projection onto it. Also consider all the assumptions of Theorem 5.7. Then*
$$\left\|L_1 - L_2 - \frac{1}{2m}\mathcal{P}_U\mathcal{A}^*\mathcal{A}(L_1 - L_2) + \mathcal{P}_U\frac{1}{2}Tr(L_1 - \bar{y})I\right\|_2 \leq \rho\|L_1 - L_2\|_2.$$

2. $\left\|\frac{1}{2m}\mathcal{A}^*\mathcal{A}(L_1 - L_2) - \frac{1}{2}Tr(L_1 - \bar{y})I\right\|_2 \geq (1 - \rho)\|L_1 - L_2\|_2.$

Proof. The first result holds by the fact that $L_1 - L_2$ lies in subspace U. The second result is directly follows from Theorem 5.7. □

Proof of Theorem 5.6. The proof is very similar to the proof of Lemma 5.18 and we only give a brief sketch. The idea is again to use the matrix Bernstein inequality; to do this, we have to use the truncation trick both on the random vector x_i and the noise vector e. We introduce $\widetilde{x_i}$ as (5.25) and similarly $\widetilde{e}$ as follows ($j = 1, \ldots, m$):

$$\widetilde{e}^{(j)} = \begin{cases} e^{(j)}, & |e^{(j)}| \leq c_1'\sqrt{\log m} \\ 0, & \textit{otherwise}, \end{cases} \tag{5.29}$$

In the following expressions, $c_l' > 0$ for $l = 1, 4$ are absolute constants and $c_l' > 0$ for $l = 2, 3, 5, 6, 7$ are some constants which depend on τ. Let $W_i = \widetilde{e_i}\widetilde{x_i}\widetilde{x_i}^T$ for $i = 1, \ldots, m$ and $W = \widetilde{e_r}\widetilde{x}\widetilde{x}^T$ be a independent copy of W_i's (i.e, $\widetilde{e_r}$ and $\widetilde{x}$ are independent copies of e_i and x_i, respectively). Hence,

$\mathbb{E}\frac{\mathcal{A}^* e}{m} = \frac{1}{m}\sum_{i=1}^m \mathbb{E}\widetilde{e_i}\widetilde{x_i}\widetilde{x_i}^T = \mathbb{E}S_i = 0$ and $\mathbb{P}(\widetilde{e_i} = e_i) \geq 1 - \frac{1}{m^{c_2'}}$ by assumptions on e. Now, parameters R and σ in the matrix Bernstein inequality can be calculated as follows:

$$\sigma = \|\mathbb{E}WW^T\|_2 = \left\|\mathbb{E}\widetilde{e_r}^2\mathbb{E}(\|\widetilde{x}\|_2^2\widetilde{x}\widetilde{x}^T)\right\|_2 \leq c_3' p\log(m)\log(mp),$$
$$R = \|\widetilde{e_r}\widetilde{x}\widetilde{x}^T\|_2 \leq c_4' p\sqrt{\log m}\log(mp),$$

As a result, for all $t_3 \geq 0$, we have

$$\mathbb{P}\left(\left\|\frac{1}{m}\sum_{i=1}^m W_i\right\|_2 \geq t_3\right) \leq 2p\exp\left(\frac{mt_3^2}{\sigma + Rt_3/3}\right) \leq 2p\exp\left(\frac{mt_3^2}{c_5' p\log(m)\log(mp)}\right),$$

where the last inequality holds by sufficiently large c_5'. Now, similar to Lemma 5.18 by choosing $t_3 \geq \sqrt{c_6' \frac{p\log^2 p}{m}\log(\frac{p}{\xi_3})}$ and the union bound, we obtain with probability at least $1-\xi_3$:

$$\left\|\frac{1}{m}\mathcal{A}^* e\right\|_2 \leq \sqrt{c_6' \frac{p\log^2 p}{m}\log(\frac{p}{\xi_3})}.$$

On the other hand, since e_i's are subgaussian random variables, by simple application of the Hoeffding inequality (Vershynin, 2010), we have, with probability at least $1-\xi_4$:

$$|\frac{1}{m}\mathbf{1}^T e| \leq \sqrt{\frac{c_7'}{m}\log(\frac{1}{\xi_4})}$$

Combining the above results together and letting $\gamma = \xi_3 + \xi_4$, we obtained the claim bound in the theorem. □

5.6.3 Appendix C. Running Time Analysis

Running time of EP-ROM. Each iteration of EP-ROM involves evaluation of the gradient at current estimation and an exact projection on the set of rank r matrices. Recall that the unbiased gradient of the objective function is given by:

$$\nabla F(L_t) + (\frac{1}{m}\mathbf{1}^T\mathcal{A}(L_t) - \bar{y})I = \frac{1}{m}\sum_{i=1}^m \left(x_i^T L_t x_i - y_i\right) x_i x_i^T + (\frac{1}{m}\mathbf{1}^T\mathcal{A}(L_t) - \bar{y})I.$$

The inner term $\left(x_i^T L_t x_i - y_i\right)$ can be computed only once per iteration and stored in a temporary vector $d \in \mathbb{R}^m$. Since in each iteration, we have access to the factors of $L_t = U_t V_t^T$ such

Algorithm 5.4 MBK-SVD

```
**Inputs:** $y$, measurement operator, $\mathcal{A} = \{x_1x_1^T, x_2x_2^T \dots, x_mx_m^T\}$, rank $r$, block size $b = r + 5$, $\varepsilon \in (0, 1)$
**Outputs:** matrix $Z \in \mathbb{R}^{p\times r}$
1: Set $q = \Theta(\frac{\log p}{\sqrt{\varepsilon}})$ and $G \sim \mathcal{N}(0, 1)^{p\times b}$
2: Calculate $\bar{y} = \frac{1}{m}\sum_{i=1}^{m} y_i$ and $d = x_i^T L_t x_i - y_i$
3: Allocate Krylov subspace, $K_r \in \mathbb{R}^{p\times q}$.
4: $I \leftarrow \mathcal{B}(\mathcal{A}, G, d, \bar{y})$, $G \leftarrow I$, $K_r[:, 1 : b] \leftarrow I$
**for** $i = 2 : q$ **do**
    $I \leftarrow \mathcal{B}(\mathcal{A}, G, d, \bar{y})$
    $J \leftarrow \mathcal{B}(\mathcal{A}, I, d, \bar{y})$
    $K_r[:, (i - 1)b + 1 : ib] \leftarrow J$
    $G \leftarrow J$
**end for**
5: Orthonormalize the columns of $K_r$ to find $Q \in \mathbb{R}^{p\times qb}$.
6: Compute $M \leftarrow \mathcal{B}(\mathcal{A}, Q, d, \bar{y})$, $M \leftarrow M^T$
7: Compute top $r$ singular vectors of $M$ and call it $\overline{U_k}$.
**Return**: $Z = Q\overline{U_k}$
```

that $U_t, V_t \in \mathbb{R}^{p\times r}$, the calculation of d takes $\mathcal{O}(pr)$ operations. Then we can calculate $dx_ix_i^T$ in $\mathcal{O}(p^2)$ operations. In addition, computing unbiasing term, $(\frac{1}{m}\mathbf{1}^T\mathcal{A}(L_t) - \bar{y})I$, takes $\mathcal{O}(m)$ operations. As a result, calculating the whole unbiased gradient takes $\mathcal{O}(mp^2)$ times which simplifies to $\mathcal{O}(p^3r^2\log^4(p)\log(\frac{1}{\epsilon}))$ due to the choice of m. On the other hand, exact projection on the set of rank r matrices takes $\mathcal{O}(p^3)$ time, since the SVD of even a rank-1 $p \times p$ matrix (without spectral assumptions) needs $\mathcal{O}(p^3)$ operations. As a result, the total running time for EP-ROM to achieve ϵ accuracy is given by $K = \mathcal{O}(p^3r^2\log^4(p)\log^2(\frac{1}{\epsilon}))$ due to the linear convergence of EP-ROM.

We note that even if we use the Lanczos method for the projection step, the required running time equals $\widetilde{\mathcal{O}}(\frac{p^2r}{\sqrt{\delta-1}})$ where δ denotes the gap between the r^{th} and $(r+1)^{th}$ largest singular values. Hence, the gradient calculation is the computationally dominating step and the total running time is as before.

Running time of AP-ROM. As discussed before, we use MBK-SVD as head approximate projection in AP-ROM. The pseudocode for MBK-SVD is given in Algorithm 5.4.

Algorithm 5.5 Operator $\mathcal{B}(\mathcal{A}, G, d, \bar{y})$

```
**Inputs:** $\mathcal{A}, G, d, \bar{y}$
**Outputs:** $W_3 = \left(\frac{1}{2m}\sum_{i=1}^{m} d_i x_i x_i^T - \frac{1}{2m}(\mathbf{1}^T d)I\right) G$
for $j = 1 : m$ do
  $W_1 \leftarrow x_j^T G$
  $W_2 \leftarrow d^{(i)} x_j W_1$
  $W_3 \leftarrow W_2 - d_j G$
end for
**Return:** $W_3 \leftarrow \frac{1}{2m} W_3$
```

In Algorithm 5.4, K_r denotes a Krylov subspace, and the parameter b determines the size of each block inside K_r which can be any value greater than r. Also, ε represents the desired accuracy in calculating of the projection.

Now let $\Delta = \frac{1}{2m}\sum_{i=1}^{m}\left(x_i^T L_t x_i - y_i\right) x_i x_i^T - \frac{1}{2m}(\mathbf{1}^T d)I$. In MBK-SVD, the computation of vector d takes $\mathcal{O}(pr)$ operations as before. In addition, instead of multiplying unbiased gradient by a random matrix, each sensing vector, x_i is multiplied by a matrix G which needs $\mathcal{O}(pr)$ operations. To be more precise, the Krylov subspace is formed by q iterations. Each iteration needs to compute the product of $(\Delta^2)^k.\Delta.G$ for $k = 0, \dots, q$ and this is done through operator $\mathcal{B}$. The code for this operator is given in Algorithm 5.5. To run this algorithm, we need $\mathcal{O}(mpr)$ operations; there are m iterations and each of them takes $m = \widetilde{\mathcal{O}}(pr^2 \log(\frac{1}{\epsilon}))$ time ($\widetilde{\mathcal{O}}$ hides dependency on polylog(p)). As a result, MBK-SVD requires $\mathcal{O}(qmpr)$ operations which implies that the total running time of MBK-SVD is scaled as $\mathcal{O}\left(p^2 r^4 \log^4(p) \log(\frac{1}{\epsilon}) \frac{\log(p)}{\sqrt{\varepsilon}}\right)$ by the choice of m and q.

Proof of Theorem 5.11. As we discussed before, AP-ROM uses two tail and head approximate projections. For implementing the head approximation step, we use MBK-SVD with rank set to $2r$ to obtain the approximation of right singular vectors. Let $U_{\mathcal{H}}$ be the returned $2r$-dimensional subspace by MBK-SVD. Now we have to form $U_t V_t^T - U_{\mathcal{H}} U_{\mathcal{H}}^T \Delta$ which is a matrix with rank at most $3r$. Here, U_t, V_t are factors of L_t. To efficiently compute this expression, we again use operator $\mathcal{B}$ by calculating $U_{\mathcal{H}}^T \Delta = (\mathcal{B}(\mathcal{A}, U_{\mathcal{H}}, d, \bar{y}))^T$ in $\mathcal{O}(pr)$ operations. Now to apply the approximate tail projection, we can use either the Lanczos algorithm (SVDs) or ordinary BK-SVD, both of which require $\mathcal{O}(p^2 r)$ operations. After calculating the r-dimensional subspace returned by tail operator,

$U_{\mathcal{T}}$, we can project $U_t V_t^T - U_{\mathcal{H}} U_{\mathcal{H}}^T \Delta$ onto it which needs another $\mathcal{O}(p^2 r)$ operations. As a result, the total running time for AP-ROM to achieve ϵ accuracy is scaled as $K = \mathcal{O}\left(p^2 r^4 \log^5(p) \log^2(\frac{1}{\epsilon})\right)$ due to the linear convergence of AP-ROM. □

Proof of Theorem 5.13. Let V^t, V^{t+1}, and V^* denote the bases for the column space of L_t, L_{t+1}, and L_*, respectively. Assume $\nu = 1 + \frac{2}{\sqrt{1-\varepsilon}\sqrt{\kappa-1}}$. Also, let $V^t \cup V^{t+1} \cup V^* \subseteq \Omega_t := \Omega$. Hence, Ω_t is the set of matrices with $rank \leq 2\kappa r + r^*$. Define $b = L_t - \eta \mathcal{P}_\Omega \partial F(L_t)$, $\alpha = \alpha_{2\kappa r + r^*}$, and $\beta = \beta_{2\kappa r + r^*}$. Thus:

$$\begin{aligned}
&\|L_{t+1} - L_*\|_F^2 \overset{a_1}{\leq} \nu \|b - L_*\|_F^2 = \nu \|L_t - L_* - \eta_t \mathcal{P}_\Omega \partial F(L_t)\|_F^2 \\
&= \nu \|L_t - L_*\|_F^2 - 2\eta_t \nu \langle L_t - L_*, \mathcal{P}_\Omega \frac{1}{m} \mathcal{B}^* \mathrm{sgn}(\mathcal{B}(L_t) - y)\rangle + \nu \eta_t^2 \|\mathcal{P}_\Omega \partial F(L_t)\|_F^2 \\
&= \nu \|L_t - L_*\|_F^2 - 2\frac{\eta_t \nu}{m} \langle \mathcal{B}(L_t - L_*), \mathrm{sgn}(\mathcal{B}(L_t - L_*))\rangle + \nu \eta_t^2 \|\mathcal{P}_\Omega \partial F(L_t)\|_F^2 \\
&= \nu \|L_t - L_*\|_F^2 - 2\frac{\eta_t \nu}{m} \|\mathcal{B}(L_t - L_*)\|_1 + \nu \eta_t^2 \|\mathcal{P}_\Omega \partial F(L_t)\|_F^2,
\end{aligned}$$

where a_1 holds by applying lemma 5.12, and due to the fact that L_{t+1} is the best low-rank approximation to b, it also happens to be the best low-rank approximation to b. Now we can bound the third term, $\|\mathcal{P}_\Omega \partial F(L_t)\|_F^2$ as follows:

$$\begin{aligned}
&\|\mathcal{P}_\Omega \partial F(L_t)\|_F^2 = \frac{1}{m} \|\mathcal{P}_\Omega \mathcal{B}^* \mathrm{sgn}(\mathcal{B}(L_t) - y)\|_F^2 = \frac{1}{m} \langle \mathcal{P}_\Omega \mathcal{B}^* \mathrm{sgn}(\mathcal{B}(L_t) - y), \mathcal{P}_\Omega \mathcal{B}^* \mathrm{sgn}(\mathcal{B}(L_t) - y)\rangle \\
&= \frac{1}{m} \langle \mathrm{sgn}(\mathcal{B}(L_t) - y), \mathcal{B}\mathcal{P}_\Omega \mathcal{B}^* \mathrm{sgn}(\mathcal{B}(L_t) - y)\rangle \overset{a_1}{\leq} \frac{1}{m} \|\mathcal{B}\mathcal{P}_\Omega \mathcal{B}^* \mathrm{sgn}(\mathcal{B}(L_t) - y)\|_1 \overset{a_2}{\leq} \beta \|\mathcal{P}_\Omega \partial F(L_t)\|_F,
\end{aligned}$$

where a_1 holds by Hölder's inequality, and a_2 is due to applying RIP(ℓ_1, ℓ_2) property. Hence, we obtain, $\|\mathcal{P}_\Omega \partial F(L_t)\|_F \leq \beta$. Now we have the error bound as $\|L_{t+1} - L_*\|_F^2 \leq \nu \|L_t - L_*\|_F^2 - 2\frac{\eta_t \nu}{m} \|\mathcal{B}(L_t - L_*)\|_1 + \nu \eta^2 \beta^2$, Now let $\eta_t = \frac{\|\mathcal{B}(L_t - L_*)\|_1}{\beta^2}$. By using the RIP$(\ell_1, \ell_2)$ property, we have

$$\|L_{t+1} - L_*\|_F^2 \leq \nu (1 - \frac{\alpha^2}{\beta^2}) \|L_t - L_*\|_F^2. \tag{5.30}$$

In order to have linear convergence, we need to have $\sqrt{\nu(1 - \frac{\alpha^2}{\beta^2})} < 1$. If we simplify this condition together with the condition on κ, stated in Lemma 5.12, we obtain: $\kappa > 1 + \max\left\{\frac{4\left((\frac{\alpha^2}{\beta^2}) - 1\right)^2}{1-\varepsilon}, \frac{1}{1-\varepsilon}\right\}$. This completes the proof. □

CHAPTER 6. FUTURE DIRECTIONS

In this chapter, we provide some possible future directions based on the previous results. In general, we will consider four different directions as follows:

1. Extending the nonlinear demixing problem to more complex models such as the superposition of low-rank and sparse matrices

2. Analyzing nonlinear regression problem with other link functions such as general periodic and noninvertible functions

3. Justifying theoretically the success of the GAN approach introduced in chapter 3 in the demixing problem

4. Extending our initial result in the two-layer neural network for more general cases

Regarding the first direction, we are working on the problem where the constituent signals have a more complex structure such as the mixture of low-rank and sparse matrices. Specifically, we want to know if a non-convex algorithm can be proposed for demixing of the constituent signals, and how efficient this algorithm is in terms of sample complexity, time complexity, and convergence rate.

Regarding the second direction, we want to know what happens if the link function is a general periodic function. For instance, what if the link function is modeled as a universal quantization function, or even some aperiodic yet non-invertible one. Can we still propose an algorithm to recover the underlying signal? If so, how efficient is this algorithm? And what are its statistical and computational limitations?

Regarding the third direction, since our GAN approach is based on the empirical observation, the natural question comes to mind is that if we can justify it mathematically? For instance, we know from chapter 1 and 2, having incoherent constituent components are crucial for the success of demixing problem. How can the proposed GAN approach encode the incoherence notion?, or under what condition(s), this approach is unable to learn the structure of the components? Other important questions here are related to the convergence analysis of the denoising and demixing optimization problems introduced in problems 3.3 and 3.4, respectively. Finally, can we learn some other models such as the low-rank and sparsity structures using the proposed GAN approach?

Regarding the fourth direction, there are many research challenges which we can address. For instance, if we still focus on the shallow network with two layers and with quadratic activation function, but use another loss function such as cross-entropy (negative likelihood) function, can we still use our proposed algorithms or similar ones for learning the weights of the network? What if we change the activation function to Relu which has practical importance. In this case, can we propose a new algorithm for training the network? Another possible extension is about going to more deeper networks either by quadratic, or other types of activation functions. In this case, providing efficient algorithms for training the network, or estimating the weights of the net is a challenging research question.

The above works are some of the possible future directions which I am working on, and my goal is to address all or part of them as my future work.

REFERENCES

The quick, draw! dataset. https://github.com/googlecreativelab/quickdraw-dataset.

The quick, draw! game. https://quickdraw.withgoogle.com/.

A., M., C., S., and B., L. (2017). Wasserstein generative adversarial networks. In *Proc. Int. Conf. Machine Learning*, pages 214–223.

Allen-Zhu, Z. and Li, Y. (2016). Lazysvd: Even faster svd decomposition yet without agonizing pain. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 974–982.

Anirudh, R., Thiagarajan, J., Kailkhura, B., and Bremer, T. (2018). An unsupervised approach to solving inverse problems using generative adversarial networks. *arXiv preprint arXiv:1805.07281*.

Bahmani, S., Boufounos, P., and Raj, B. (2013a). Greedy sparsity-constrained optimization. *JMLR*.

Bahmani, S., Raj, B., and Boufounos, P. (2013b). Greedy sparsity-constrained optimization. *J. Machine Learning Research*, 14(1):807–841.

Bahmani, S. and Romberg, J. (2015). Efficient compressive phase retrieval with constrained sensing vectors. In Adv. Neural Inf. Proc. Sys. (NIPS), pages 523–531.

Banerjee, O., Ghaoui, L., and d'Aspremont, A. (2008). Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *J. Machine Learning Research*, 9(Mar):485–516.

Baraniuk, R., Cevher, V., Duarte, M., and Hegde, C. (2010). Model-based compressive sensing. *IEEE Trans. Inform. Theory*, 56(4):1982–2001.

Beck, A. and Eldar, Y. (2013). Sparsity constrained nonlinear optimization: Optimality conditions and algorithms. *SIAM Journal on Optimization*, 23(3):1480–1509.

Becker, S., Cevher, V., and Kyrillidis, A. (2013). Randomized low-memory singular value projection. In *Proc. Sampling Theory and Appl. (SampTA)*, number EPFL-CONF-184017.

Berthelot, D., Schumm, T., and Metz, L. (2017). Began: boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*.

Bhojanapalli, S., Kyrillidis, A., and Sanghavi, S. (2016a). Dropping convexity for faster semi-definite optimization. In *29th Ann. Conf. Learning Theory*, pages 530–582.

Bhojanapalli, S., Neyshabur, B., and Srebro, N. (2016b). Global optimality of local search for low rank matrix recovery. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 3873–3881.

Blumensath, T. and Davies, M. (2009). Iterative hard thresholding for compressed sensing. *Appl. Comput. Harmon. Anal*, 27(3):265–274.

Bobin, J., Starck, J., Fadili, J., Moudden, Y., and Donoho, D. (2007). Morphological component analysis: An adaptive thresholding strategy. *IEEE Trans. Image Proc.*, 16(11):2675–2681.

Bora, A., Jalal, A., Price, E., and Dimakis, A. (2017). Compressed sensing using generative models. *Proc. Int. Conf. Machine Learning*.

Bora, A., Price, E., and Dimakis, A. (2018). Ambientgan: Generative models from lossy measurements. In *Int. Conf. on Learning Rep. (ICLR)*.

Boufounos, P. (2012). Universal rate-efficient scalar quantization. *IEEE Trans. Inform. Theory*, 58(3):1861–1872.

Boufounos, P. and Baraniuk, R. (2008). 1-bit compressive sensing. In *Int. Conf. Info. Sciences and Systems (CISS)*, pages 16–21. IEEE.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Burer, S. and Monteiro, R. (2003). A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357.

Cai, J., Candès, E., and Shen, Z. (2010). A singular value thresholding algorithm for matrix completion. *SIAM J. Optimization*, 20(4):1956–1982.

Cai, T. and Zhang, A. (2015). Rop: Matrix recovery via rank-one projections. *Ann. Stat.*, 43(1):102–138.

Candès, E. (2006). Compressive sampling. In *Proc. Int. Congress of Math.*, Madrid, Spain.

Candes, E., Eldar, Y., Neeell, D., and Paige, R. (2011). Compressed sensing with coherent and redundant dictionaries. *Appl. Comput. Harmonic Analysis*, 31(1):59–73.

Candès, E., Li, X., Ma, Y., and Wright, J. (2011). Robust principal component analysis? *Journal of the ACM*, 58(3):11.

Candes, E., Li, X., and Soltanolkotabi, M. (2015). Phase retrieval via wirtinger flow: Theory and algorithms. *IEEE Trans. Inform. Theory*, 61(4):1985–2007.

Candes, E. and Plan, Y. (2011). Tight oracle inequalities for low-rank matrix recovery from a minimal number of noisy random measurements. *IEEE Trans. Inform. Theory*, 57(4):2342–2359.

Candès, E. and Recht, B. (2009). Exact matrix completion via convex optimization. *Found. Comput. Math.*, 9(6).

Candes, E. and Romberg, J. (2007). Sparsity and incoherence in compressive sampling. *Inverse problems*, 23(3):969.

Candès, E., Romberg, J., and Tao, T. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inform. Theory*, 52(2):489–509.

Candes, E., Strohmer, T., and Voroninski, V. (2013). Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming. *Comm. Pure Appl. Math.*, 66(8):1241–1274.

Chandrasekaran, V., Parrilo, P., and Willsky, A. (2012). Latent variable graphical model selection via convex optimization. *The Annals of Statistics*, 40(4):1935–1967.

Chandrasekaran, V., Parrilo, P., and Willsky, A. S. (2010). Latent variable graphical model selection via convex optimization. In *Proc. Allerton Conf. on Comm., Contr., and Comp.*, pages 1610–1613.

Chandrasekaran, V., Sanghavi, S., Parrilo, P., and Willsky, A. S. (2009). Sparse and low-rank matrix decompositions. In *Proc. Allerton Conf. on Comm., Contr., and Comp.*, pages 962–967.

Chandrasekaran, V., Sanghavi, S., Parrilo, P., and Willsky, A. S. (2011). Rank-sparsity incoherence for matrix decomposition. *SIAM J. Opt.*, 21(2):572–596.

Chang, J., Sankaranarayanan, A., and Vijayakumar, B. V. K. (2016). Random features for sparse signal classification. In *IEEE Conf. Comp. Vision and Pattern Recog.*

Chen, G. and Srihari, S. (2014). Removing structural noise in handwriting images using deep learning. In *Proc. Indian Conf.e on Comp. Vision Graph. Image Proc.* ACM.

Chen, S., Donoho, D., and Saunders, M. (1998). Atomic decomposition by basis pursuit. *SIAM J. Sci. Comp.*, 20(1):33–61.

Chen, S., Donoho, D., and Saunders, M. (2001). Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Adv. Neural Inf. Proc. Sys. (NIPS)*.

Chen, Y. and Chi, Y. (2014). Robust spectral compressed sensing via structured matrix completion. *IEEE Trans. Inform. Theory*, 60(10).

Chen, Y., Chi, Y., and Goldsmith, A. (2015). Exact and stable covariance estimation from quadratic sampling via convex programming. *IEEE Trans. Inform. Theory*, 61(7):4034–4059.

Chen, Y. and Wainwright, M. (2015). Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees. *arXiv preprint arXiv:1509.03025*.

Cheraghchi, M., Guruswami, V., and Velingker, A. (2013). Restricted isometry of Fourier matrices and list decodability of random linear codes. *SIAM J. Comp.*, 42(5):1888–1914.

Chiang, K., Hsieh, C., Natarajan, N., Dhillon, I., and Tewari, A. (2014). Prediction and clustering in signed networks: a local to global perspective. *J. Machine Learning Research*, 15(1):1177–1213.

Clarkson, K. and Woodruff, D. (2017). Low-rank psd approximation in input-sparsity time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2061–2072. SIAM.

Clarkson, K. L. and Woodruff, D. (2013). Low rank approximation and regression in input sparsity time. In *Proc. ACM Symp. Theory of Comput.*, pages 81–90. ACM.

Coifman, R., Geshwind, F., and Meyer, Y. (2001). Noiselets. *Appl. Comput. Harmonic Analysis*, 10(1):27–44.

Dasarathy, G., Shah, P., Bhaskar, B., and Nowak, R. (2015). Sketching sparse matrices, covariances, and graphs via tensor products. *IEEE Trans. Inform. Theory*, 61(3):1373–1388.

Davenport, M., Plan, Y., van den Berg, E., and Wootters, M. (2014). 1-bit matrix completion. *Information and Inference*, 3(3):189–223.

Davenport, M. A. and Romberg, J. (2016). An overview of low-rank matrix recovery from incomplete observations. *IEEE J. Select. Top. Sig. Proc.*, 10(4):608–622.

Defazio, A., Bach, F., and Lacoste-Julien, S. (2014). Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 1646–1654.

Donoho, D. (2006). Compressed sensing. *IEEE Trans. Inform. Theory*, 52(4):1289–1306.

Donoho, D., Elad, M., and Temlyakov, V. (2006). Stable recovery of sparse overcomplete representations in the presence of noise. IEEE Trans. Inform. Theory, 52(1):6–18.

Druce, J., Gonella, S., Kadkhodaie, M., Jain, S., and Haupt, J. (2016). Defect triangulation via demixing algorithms based on dictionaries with different morphological complexity. In *Proc. European Workshop on Struc. Health Monitoring (IWSHM)*.

Eftekhari, A., Romberg, J., and Wakin, M. (2013). Matched filtering from limited frequency samples. *IEEE Trans. Inform. Theory*, 59(6):3475–3496.

Elad, M. and Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. Image Proc.*, 15(12):3736–3745.

Elad, M., Starck, J., Querre, P., and Donoho, D. (2005). Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). *Appl. Comput. Harmonic Analysis*, 19(3):340–358.

Elyaderani, M., Jain, S., Druce, J., Gonella, S., and Haupt, J. (2017). Group-level support recovery guarantees for group lasso estimator. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP)*, pages 4366–4370.

Fan, J. and Li, R. (2001). Variable selection via non-concave penalized likelihood and its oracle properties. *J. Amer. Statist. Assoc.*, 96(456):1348–1360.

Fazel, M. (2002). *Matrix rank minimization with applications*. PhD thesis, PhD thesis, Stanford University.

Foucart, S. and Rauhut, H. *A mathematical introduction to compressive sensing*, volume 1. Springer.

Foucart, S. and Rauhut, H. (2013). A mathematical introduction to compressive sensing. *Boston: Birkhuser*, 1(3).

Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

G., M. and B., S. (2014). CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx.

Ganti, R., Balzano, L., and Willett, R. (2015a). Matrix completion under monotonic single index models. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 1864–1872.

Ganti, R., Rao, N., Willett, R., and Nowak, R. (2015b). Learning single index models in high dimensions. *arXiv preprint arXiv:1506.08910*.

Ge, R., Lee, J., and Ma, T. (2016). Matrix completion has no spurious local minimum. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 2973–2981.

Goldstein, T., Studer, C., and Baraniuk, R. (2014). A field guide to forward-backward splitting with a FASTA implementation. *arXiv eprint*, abs/1411.3406.

Goldstein, T., Studer, C., and Baraniuk, R. (2015). FASTA: A generalized implementation of forward-backward splitting. http://arxiv.org/abs/1501.04979.

Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 2672–2680.

Han, L., Zhang, Y., and Zhang, T. (2016). Fast component pursuit for large-scale inverse covariance estimation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1585–1594. ACM.

Han, X., Kashif, R., and Roland, V. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint cs.LG/1708.07747*.

Hand, P., Leong, O., and Voroninski, V. (2018). Phase retrieval under a generative prior. *arXiv preprint arXiv:1807.04261*.

Hardt, M. (2014). Understanding alternating minimization for matrix completion. In *Proc. IEEE Symp. Found. Comp. Science (FOCS)*, pages 651–660. IEEE.

Haviv, I. and Regev, O. (2017). The restricted isometry property of subsampled fourier matrices. In *Geom. Aspec. Func. Anal.*, pages 163–179. Springer.

Hegde, C. and Baraniuk, R. (2012a). Signal recovery on incoherent manifolds. *IEEE Trans. Inform. Theory*, 58(12):7204–7214.

Hegde, C. and Baraniuk, R. (2012b). SPIN : Iterative signal recovery on incoherent manifolds. In *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*.

Hegde, C., Indyk, P., and Schmidt, L. (2015a). Approximation algorithms for model-based compressive sensing. *IEEE Trans. Inform. Theory*, 61(9):5129–5147.

Hegde, C., Indyk, P., and Schmidt, L. (2015b). Approximation algorithms for model-based compressive sensing. *IEEE Trans. Inform. Theory*, 61(9):5129–5147.

Hegde, C., Indyk, P., and Schmidt, L. (2016). Fast recovery from a union of subspaces. In *Adv. Neural Inf. Proc. Sys. (NIPS)*.

Henrion, D. and Malick, J. (2012). Projection methods in conic optimization. In *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 565–600. Springer.

Horn, R. A. and Johnson, C. (2012). *Matrix analysis*. Cambridge university press.

Hoyer, P. et al. (1999). Independent component analysis in image denoising. *Master degree dissertation, Helsinki University of Technology*.

Hsieh, C., Dhillon, I., Ravikumar, P., Becker, S., and Olsen, P. (2014). Quic & dirty: A quadratic approximation approach for dirty statistical models. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 2006–2014.

Hsieh, C., Dhillon, I., Ravikumar, P., and Sustik, M. (2011). Sparse inverse covariance matrix estimation using quadratic approximation. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 2330–2338.

Hsieh, C. and Olsen, P. (2014). Nuclear norm minimization via active subspace selection. In *Proc. Int. Conf. Machine Learning*, pages 575–583.

Huang, J. and Zhang, T. (2010). The benefit of group sparsity. *The Annals of Statistics*, 38(4):1978–2004.

Hughes, T., Marton, M., Jones, A., Roberts, C., Stoughton, R., Armour, C., Bennett, H., Coffey, E., Dai, H., He, Y., et al. (2000). Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–126.

Iwen, M., Viswanathan, A., and Wang, Y. (2015). Robust sparse phase retrieval made easy. *Appl. Comput. Harmon. Anal.*

Jacques, L., Laska, J., Boufounos, P., and Baraniuk, R. (2013). Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors. *IEEE Trans. Inform. Theory*, 59(4):2082–2102.

Jain, P., Jin, C., Kakade, S., and Netrapalli, P. (2015). Computing matrix squareroot via non convex local search. *arXiv preprint arXiv:1507.05854*.

Jain, P., Meka, R., and Dhillon, I. (2010). Guaranteed rank minimization via singular value projection. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 937–945.

Jain, P., Netrapalli, P., and Sanghavi, S. (2013). Low-rank matrix completion using alternating minimization. In *Proc. ACM Symp. Theory of Comput.*, pages 665–674. ACM.

Jain, P., Tewari, A., and Kar, P. (2014). On iterative hard thresholding methods for high-dimensional m-estimation. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 685–693.

Jain, S., Oswal, U., Xu, K., Eriksson, B., and Haupt, J. (2017). A compressed sensing based decomposition of electrodermal activity signals. *IEEE Trans. Biom. Eng.*, 64(9):2142–2151.

Janzamin, M., Sedghi, H., and Anandkumar, A. (2015). Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *arXiv preprint arXiv:1506.08473*.

Ji, H., Liu, C., Shen, Z., and Xu, Y. (2010). Robust video denoising using low rank matrix completion. In *cvpr*, pages 1791–1798. IEEE.

Johnson, C. (2014). Logistic matrix factorization for implicit feedback data. *Adv. Neural Inf. Proc. Sys. (NIPS)*, 27.

Johnson, R. and Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 315–323.

Johnstone, I. (2001). On the distribution of the largest eigenvalue in principal components analysis. *Annals of statistics*, pages 295–327.

Jolliffe, I. (2002). Principal component analysis and factor analysis. In *Principal component analysis*. Springer.

Kabkab, M., Samangouei, P., and Chellappa, R. (2018). Task-aware compressed sensing with generative adversarial networks. *Proc. Assoc. Adv. Artificial Intelligence (AAAI)*.

Kakade, S., Kanade, V., Shamir, O., and Kalai, A. (2011). Efficient learning of generalized linear and single index models with isotonic regression. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 927–935.

Kalai, A. and Sastry, R. (2009). The isotron algorithm: High-dimensional isotonic regression. In *COLT*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Krahmer, F. and Ward, R. (2011). New and improved johnson–lindenstrauss embeddings via the restricted isometry property. *SIAM J. Math. Anal.*, 43(3):1269–1281.

Kueng, R., Rauhut, H., and Terstiege, U. (2017). Low rank matrix recovery from rank one measurements. *Appl. Comput. Harmon. Anal.*, 42(1):88–116.

Lanczos, C. (1950). An iteration method for the solution of the eigenvalue problem of linear differential and integral operators1. *Journal of Research of the National Bureau of Standards*, 45(4).

Le, Q., Sarlos, T., and Smola, A. (2013). Fastfood-approximating kernel expansions in loglinear time. In *Proc. Int. Conf. Machine Learning*.

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

Ledoux, M. and Talagrand, M. (2013). *Probability in Banach Spaces: isoperimetry and processes*. Springer Science & Business Media.

Lee, K. and Bresler, Y. (2010). Admira: Atomic decomposition for minimum rank approximation. *IEEE Trans. Inform. Theory*, 56(9):4402–4416.

Li, X., Zhao, T., Arora, R., Liu, H., and Haupt, J. (2016). Nonconvex sparse learning via stochastic optimization with progressive variance reduction. *arXiv preprint arXiv:1605.02711*.

Li, Y. and Yuan, Y. (2017). Convergence analysis of two-layer neural networks with relu activation. *In Adv. Neural Inf. Proc. Sys. (NIPS)*.

Lin, M., Qiu, S., Hong, B., and Ye, J. (2017). The second order linear model. *arXiv preprint arXiv:1703.00598*.

Lin, M. and Ye, J. (2016). A non-convex one-pass framework for generalized factorization machine and rank-one matrix sensing. In *In Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 1633–1641.

Liu, G., Lin, Z., Yan, S., Sun, J., Yu, Y., and Ma, Y. (2013). Robust recovery of subspace structures by low-rank representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1):171–184.

Liu, Y. (2011). Universal low-rank matrix recovery from pauli measurements. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 1638–1646.

Livni, R., Shalev-Shwartz, S., and Shamir, O. (2014). On the computational efficiency of training neural networks. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 855–863.

Lopez-Paz, D., Sra, S., Smola, A., Ghahramani, Z., and Schölkopf, B. (2014). Randomized nonlinear component analysis. In *Proc. Int. Conf. Machine Learning*.

Lu, C., Tang, J., Yan, S., and Lin, Z. (2016). Nonconvex nonsmooth low rank minimization via iteratively reweighted nuclear norm. *IEEE Trans. Image Proc.*, 25(2):829–839.

Ma, S., Xue, L., and Zou, H. (2013). Alternating direction methods for latent variable gaussian graphical model selection. *Neural computation*, 25(8):2172–2198.

Mahoney, M. and Drineas, P. (2009). Cur matrix decompositions for improved data analysis. *Proc. Natl. Acad. Sci.*, 106(3):697–702.

Mazumder, R. and Hastie, T. (2012). The graphical lasso: New insights and alternatives. *Electronic journal of statistics*, 6:2125.

McCoy, M., Cevher, V., Dinh, Q., Asaei, A., and Baldassarre, L. (2014). Convexity in source separation: Models, geometry, and algorithms. *IEEE Sig. Proc. Mag.*, 31(3):87–95.

McCoy, M. and Tropp, J. (2014). Sharp recovery bounds for convex demixing, with applications. *Foundations of Comp. Math.*, 14(3):503–567.

McLeod, R. (1965). Mean value theorems for vector valued functions. *Proceedings of the Edinburgh Mathematical Society (Series 2)*, 14(03):197–209.

Mendelson, S., Pajor, A., and Tomczak-Jaegermann, N. (2008). Uniform uncertainty principle for Bernoulli and subgaussian ensembles. *Constructive Approximation*, 28(3):277–289.

Mishali, M. and Eldar, Y. (2010). From theory to practice: Sub-Nyquist sampling of sparse wideband analog signals. *IEEE J. Select. Top. Signal Processing*, 4(2):375–391.

Mishali, M. and Eldar, Y. (2011). Wideband spectrum sensing at sub-nyquist rates [applications corner]. *IEEE Sig. Proc. Mag.*, 28(4):102–135.

Musco, C. and Musco, C. (2015). Randomized block krylov methods for stronger and faster approximate singular value decomposition. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 1396–1404.

Needell, D. and Tropp, J. (2009a). Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Appl. Comput. Harmon. Anal*, 26(3):301–321.

Needell, D. and Tropp, J. (2009b). CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Appl. Comput. Harmon. Anal.*, 26(3):301–321.

Negahban, S., Yu, B., Wainwright, M., and Ravikumar, P. (2011). A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. In *Adv. Neural Inf. Proc. Sys. (NIPS)*.

Netrapalli, P., Jain, P., and Sanghavi, S. (2013). Phase retrieval using alternating minimization. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 2796–2804.

Netrapalli, P., Niranjan, U., Sanghavi, S., Anandkumar, A., and Jain, P. (2014). Non-convex robust pca. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 1107–1115.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5.

Padmanabhan, N., White, M., Zhou, H., and O'Connell, R. (2016). Estimating sparse precision matrices. *Monthly Notices of the Royal Astronomical Society*, 460(2):1567–1576.

Park, D., Kyrillidis, A., Caramanis, C., and Sanghavi, S. (2016a). Finding low-rank solutions via non-convex matrix factorization, efficiently and provably. *arXiv preprint arXiv:1606.03168*.

Park, D., Kyrillidis, A., Caramanis, C., and Sanghavi, S. (2016b). Non-square matrix sensing without spurious local minima via the burer-monteiro approach. *stat*, 1050:12.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *J. Machine Learning Research*, 12:2825–2830.

Peng, Y., Ganesh, A., Wright, J., Xu, W., and Ma, Y. (2012). Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images. *IEEE Trans. Pattern Anal. Machine Intell.*, 34(11):2233–2246.

Plan, Y. and Vershynin, R. (2013a). One-bit compressed sensing by linear programming. *Comm. Pure and Applied Math.*, 66(8):1275–1297.

Plan, Y. and Vershynin, R. (2013b). Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach. *IEEE Trans. Inform. Theory*, 59(1):482–494.

Plan, Y. and Vershynin, R. (2016). The generalized LASSO with nonlinear observations. *IEEE Trans. Inform. Theory*, 62(3):1528–1537.

Plan, Y., Vershynin, R., and Yudovina, E. (2014). High-dimensional estimation with geometric constraints. *arXiv preprint arXiv:1404.3749*.

Plan, Y., Vershynin, R., and Yudovina, E. (2017). High-dimensional estimation with geometric constraints. *Inform. and Infer: A Journal of the IMA*, 6(1):1–40.

Quanming, Y., Kwok, J., Wang, T., and Liu, T. (2017). Large-scale low-rank matrix learning with non-convex regularizers. *arXiv preprint arXiv:1708.00146*.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 1177–1184.

Rao, N., Shah, P., and Wright, S. (2014). Forward-backward greedy algorithms for signal demixing. In *Proc. Asilomar Conf. Sig. Sys. Comput.*, pages 437–441.

Raskutti, G., Wainwright, M. J., and Yu, B. (2010). Restricted eigenvalue properties for correlated gaussian designs. *J. Machine Learning Research*, 11(Aug):2241–2259.

Rauhut, H., Schnass, K., and Vandergheynst, P. (2008). Compressed sensing and redundant dictionaries. *IEEE Trans. Inform. Theory*, 54(5):2210–2219.

Recht, B., Fazel, M., and Parrilo, P. (2010a). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501.

Recht, B., Fazel, M., and Parrilo, P. (2010b). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501.

Rokhlin, V., Szlam, A., and Tygert, M. (2009). A randomized algorithm for principal component analysis. *SIAM J. Matrix Anal. Applications*, 31(3):1100–1124.

Rudelson, M. and Vershynin, R. (2008). On sparse reconstruction from fourier and gaussian measurements. *Communications on Pure and Applied Mathematics*, 61(8):1025–1045.

Schein, A., Saul, L., and Ungar, L. (2003). A generalized linear model for principal component analysis of binary data. In *AISTATS*, volume 3, page 10.

Schmidt, R. (1986). Multiple emitter location and signal parameter estimation. *IEEE Trans. Antennas and Prop.*, 34(3):276–280.

Shah, V. and Hegde, C. (2018). Solving linear inverse problems using gan priors: An algorithm with provable guarantees. *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP)*.

Shalev-Shwartz, S., Gonen, A., and Shamir, O. (2011). Large-scale convex minimization with a low-rank constraint. In *Proc. Int. Conf. Machine Learning*, pages 329–336.

Shen, J. and Li, P. (2016). A tight bound of hard thresholding. *arXiv preprint arXiv:1605.01656*.

Shi, Q., Petterson, J., Dror, G., Langford, J., Smola, A., and Vishwanathan, S. (2009). Hash kernels for structured data. *J. Machine Learning Research*, 10(Nov):2615–2637.

Soltani, M. and Hegde, C. (2016). Iterative thresholding for demixing structured superpositions in high dimensions. *NIPS Workshop on Learning. High Dimen. Structure (LHDS).*

Soltani, M. and Hegde, C. (2017a). Fast algorithms for demixing sparse signals from nonlinear observations. *IEEE Trans. Sig. Proc.*, 65(16):4209–4222.

Soltani, M. and Hegde, C. (2017b). Fast low-rank matrix estimation without the condition number. *arXiv preprint arXiv:1712.03281.*

Soltani, M. and Hegde, C. (2017c). Improved algorithms for matrix recovery from rank-one projections. *arXiv preprint arXiv:1705.07469.*

Soltani, M. and Hegde, C. (2017d). Stable recovery from random sinusoidal feature maps. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP).*

Soltanolkotabi, M., J., A., and Lee, J. (2017). Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *arXiv preprint arXiv:1707.04926.*

Souly, N. and Shah, M. (2016). Scene labeling using sparse precision matrix. In *IEEE Conf. Comp. Vision and Pattern Recog*, pages 3650–3658.

Studer, C., Kuppinger, P., Pope, G., and Bölcskei, H. (2012). Recovery of sparsely corrupted signals. *IEEE Trans. Inform. Theory*, 58(5):3115–3130.

Subakan, Y. . and Smaragdis, P. (2018). Generative adversarial source separation. In *Proc. Int. Conf. Acoust., Speech, and Signal Processing (ICASSP)*, pages 26–30.

Tang, G., Bhaskar, B., Shah, P., and Recht, B. (2013). Compressed sensing off the grid. *IEEE Trans. Inform. Theory*, 59(11):7465–7490.

Thrampoulidis, C., Abbasi, E., and Hassibi, B. (2015). LASSO with non-linear measurements is equivalent to one with linear measurements. In *Proc. Adv. Neural Inf. Proc. Sys (NIPS).*

Tian, Y. (2016). Symmetry-breaking convergence analysis of certain two-layered neural networks with relu nonlinearity. *In Submitted to ICLR 2017.*

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. Royal Statist. Soc B*, 58(1):267–288.

Tropp, J. (2008). On the conditioning of random subdictionaries. *Appl. Comput. Harmon. Anal.*, 25(1):1–24.

Tropp, J. (2015). An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230.

Tu, S., Boczar, R., Simchowitz, M., Soltanolkotabi, M., and Recht, B. (2016). Low-rank solutions of linear matrix equations via procrustes flow. In *Proc. Int. Conf. Machine Learning*, pages 964–973.

Udell, M., Horn, C., Zadeh, R., and Boyd, S. (2016). Generalized low rank models. *Foundations and Trends® in Machine Learning*, 9(1):1–118.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2017). Deep image prior. *arXiv preprint arXiv:1711.10925*.

van den Berg, E. and Friedlander, M. (2007). SPGL1: A solver for large-scale sparse reconstruction. http://www.cs.ubc.ca/labs/scl/spgl1.

van den Berg, E. and Friedlander, M. (2008). Probing the pareto frontier for basis pursuit solutions. *SIAM J. Sci. Comp.*, 31(2):890–912.

Vedaldi, A. and Zisserman, A. (2012). Efficient additive kernels via explicit feature maps. *IEEE Trans. Pattern Anal. Machine Intell.*, 34(3):480–492.

Vershynin, R. (2010). Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*.

Vondrick, C., Pirsiavash, H., and Torralba, A. (2016). Generating videos with scene dynamics. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 613–621.

Wainwright, M. and Jordan, M. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305.

Wakin, M., Laska, J., Duarte, M., Baron, D., Sarvotham, S., Takhar, D., Kelly, K., and Baraniuk, R. (2006). An architecture for compressive imaging. In *Proc. IEEE Int. Conf. Image Processing (ICIP)*, Atlanta, GA.

Wang, L., Zhang, X., and Gu, Q. (2017). A universal variance reduction-based catalyst for non-convex low-rank matrix recovery. *arXiv preprint arXiv:1701.02301*.

Wu, J., Zhang, C., Xue, T., Freeman, B., and Tenenbaum, J. (2016). Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 82–90.

Yang, E. and Ravikumar, P. (2013). Dirty statistical models. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 611–619.

Yang, J., Luo, L., Qian, J., Tai, Y., Zhang, F., and Xu, Y. (2017). Nuclear norm based matrix regression with applications to face recognition with occlusion and illumination changes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(1):156–171.

Yang, M., Ahuja, N., and Kriegman, D. (2000). Face recognition using kernel eigenfaces. In *Proc. IEEE Int. Conf. Image Processing (ICIP)*, Vancouver, BC.

Yang, Z., Wang, Z., Liu, H., Eldar, Y., and Zhang, T. (2015). Sparse nonlinear regression: Parameter estimation and asymptotic inference. *J. Machine Learning Research*.

Yeh, R., Chen, C., Lim, T., Hasegawa-Johnson, M., and Do, M. (2016). Semantic image inpainting with perceptual and contextual losses. arxiv preprint. *arXiv preprint arXiv:1607.07539*, 2.

Yi, X., Park, D., Chen, Y., and Caramanis, C. (2016). Fast algorithms for robust pca via gradient descent. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 4152–4160.

Yi, X., Wang, Z., Caramanis, C., and Liu, H. (2015). Optimal linear estimation under unknown nonlinear transform. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 1549–1557.

Yin, J. and Li, H. (2013). Adjusting for high-dimensional covariates in sparse precision matrix estimation by ℓ_1-penalization. *Journal of multivariate analysis*, 116:365–381.

Yuan, X., Li, P., and Zhang, T. (2014a). Gradient hard thresholding pursuit for sparsity-constrained optimization. In *Proc. Int. Conf. Machine Learning*, pages 127–135.

Yuan, X., Li, P., and Zhang, T. (2014b). Gradient hard thresholding pursuit for sparsity-constrained optimization. In Proc. Int. Conf. Machine Learning, pages 127–135.

Zhao, H., Shi, B., Fernandez-Cull, C., Yeung, S.-K., and Raskar, R. (2015). Unbounded high dynamic range photography using a modulo camera. In *Proc. Int. Conf. Comp. Photography*.

Zheng, Q. and Lafferty, J. (2015). A convergent gradient descent algorithm for rank minimization and semidefinite programming from random linear measurements. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pages 109–117.

Zhong, K., Jain, P., and Dhillon, I. (2015). Efficient matrix sensing using rank-1 gaussian measurements. In *Int. Conf on Algorithmic Learning Theory*, pages 3–18. Springer.

Zhong, K., Song, Z., Jain, P., Bartlett, P. L., and Dhillon, I. (2016). Recovery guarantees for one-hidden-layer neural networks.

Zhu, J., Krähenbühl, P., Shechtman, E., and Efros, A. (2016). Generative visual manipulation on the natural image manifold. In *Proc. European Conf. Comp. Vision (ECCV)*, pages 597–613.

Zhu, J., Park, T., Isola, P., and Efros, A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. *Proc. Int. Conf. Comp. Vision (ICCV)*.